UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema de realidad aumentada para realizar una práctica de circuitos eléctricos

TESIS

Que para obtener el título de

Ingeniería Mecatrónica

P R E S E N T A N

Roberto Ángel García García

Zaa Ribe Jazmín Ramírez Grajeda

DIRECTOR DE TESIS

M.I. Yukihiro Minami Koyama

Ciudad Universitaria, Cd. Mx., 2021

"Aunque os esforcéis diligentemente en vuestro propio camino, día tras día, si vuestro corazón no está de acuerdo con él, aunque pensáis que estáis en el buen camino, desde el punto de vista de la justicia y de la verdad, no es un auténtico camino. Si no seguís un auténtico camino hasta el final, una pequeña maldad al principio, se convierte en una gran perversión."

---Miyamoto Musashi

Agradecimientos

Investigación realizada gracias al Programa UNAM-DGAPA-PAPIME PE11218 "Diseño de prácticas de laboratorio para fortalecer el aprendizaje de conceptos matemáticos en Ciencias Básicas".

Agradecemos a los profesores de la DCB que nos facilitaron la aplicación de las encuestas a sus alumnos, al maestro Yukihiro Minami Koyama y a los miembros del Taller de Robótica Abierta por su apoyo.

Zaa Ribe Jazmín Ramírez Grajeda

Agradezco al lector de este trabajo por ser, un miembro de mi familia que siempre me apoyó, una amiga que nunca dudó en escucharme, un amigo listo para aconsejarme, un profesor que me alentó, una persona que se ha interesado por el título y ha abierto las páginas de este escrito.

Roberto Ángel García García

A mis padres, por su comprensión, paciencia y sacrificio durante estos años.

A mis hermanos, por su gran e incondicional apoyo.

A mis amigos, por su invaluable compañía.

A mi compañera de equipo, por su perseverancia y excelente trabajo.

CONTENIDO

1 INTRODUCCIÓN

Con el progreso de la sociedad, constantemente se presentan nuevas necesidades en el sector educativo, derivadas de situaciones como la creciente matrícula de estudiantes, un problema que se presenta en todos los niveles educativos, en particular en la educación superior. En medio de este panorama, el uso de las nuevas tecnologías de realidad, ha permitido solventar la demanda de los recursos educativos necesarios, a través de la creación de herramientas como contenido multimedia, al mismo tiempo que permite la exploración de diversas modalidades de la labor docente.

En el presente trabajo se hace uso de la tecnología de realidad aumentada, en el desarrollo de una aplicación para dispositivos móviles, como una herramienta útil para los alumnos en la adquisición de conocimientos y habilidades propias de una práctica de laboratorio de la materia de Ecuaciones Diferenciales. El diseño de esta herramienta toma como punto de partida una problemática real y que busca solucionar definiendo las necesidades de su público usuario, además de apoyarse en los resultados que ha tenido la realidad aumentada en su aplicación docente en la educación superior.

A continuación, se muestra un resumen por capítulos del contenido del presente escrito.

Capítulo 1 Introducción. En este capítulo se presentan los antecedentes necesarios para el planteamiento del proyecto, la problemática, y se definen los objetivos generales y particulares del proyecto.

Capítulo 2 Marco teórico. Se muestra un estudio del estado de la técnica enfocado a la Realidad Aumentada como recurso educativo en la educación superior. Se

expone además la metodología de diseño de *software* en la que se basó el desarrollo del proyecto.

Capítulo 3 Exploración de aplicación RACE: Realidad aumentada en circuitos eléctricos. Se realiza la planificación del proyecto, estableciendo sus necesidades, requerimientos y alcances.

Capítulo 4 Gestación. En este capítulo se realiza la caracterización de los usuarios objetivo; así como la selección de las herramientas utilizadas.

Capítulo 5 Construcción. Se detalla el proceso de diseño y desarrollo de las partes que componen el producto final, siguiendo el proceso indicado en la metodología de diseño seleccionada.

Capítulo 6 Estabilización. Se describen las pruebas de uso del sistema con el fin de verificar el cumplimiento de las necesidades y requerimientos planteados.

Capítulo 7 Análisis de resultados. Se presentan y analizan los resultados de las pruebas de facilidad de uso realizadas a los usuarios objetivo.

Capítulo 8 Conclusiones y trabajo a futuro. Se examinan los resultados obtenidos respecto a los objetivos planteados.

Apéndices: Se muestra contenido complementario de diversos capítulos, así como información estadística a detalle obtenida durante la gestación del proyecto.

Anexo: Se muestra contenido adicional utilizado como referencia durante el desarrollo del proyecto, pero no creado durante el desarrollo del mismo.

1.1 Planteamiento del problema

Desde el año 2015, la División de Ciencias Básicas, DCB, perteneciente a la Facultad de Ingeniería (FI) de la UNAM, ha contemplado en su plan de desarrollo el aumento en la matrícula de estudiantes de ingeniería a nivel nacional, considerando la importancia de equipar, ampliar y actualizar sus laboratorios [1]. Además, recientemente se incorporaron nuevas prácticas de laboratorio para las materias de la DCB, y la demanda de espacios de laboratorios y el incremento de equipo y materiales requeridos por parte del alumnado para la realización de sus actividades se convirtió en un problema a resolver.

Una de las prácticas adicionadas es: “Aplicación de las ecuaciones diferenciales en circuitos eléctricos”, para la materia de Ecuaciones Diferenciales, perteneciente al tronco común del plan de estudios de las ingenierías. Esta práctica fue impartida de forma presencial a dos grupos en los periodos semestrales 2019-1 y 2020-1, para un total de 79 estudiantes de Ciencias Básicas [2]; donde se busca que la alumna o el alumno pueda verificar experimentalmente el comportamiento de un circuito RLC en serie, poniendo en práctica sus conocimientos teóricos de la materia tras resolver las ecuaciones diferenciales asociadas.

Sin embargo durante la impartición de la práctica, se detectó la problemática del espacio actualmente disponible en los laboratorios, además de que fue difícil conseguir un lugar para su realización.

Por otra parte, algunos comentarios por parte de los docentes aplicadores referentes a los resultados obtenidos de los estudiantes a los que se les aplicó la práctica, fueron que presentaron dificultades para entender la redacción de la misma, ya que desconocen los nombres de los materiales y cómo utilizarlos. Se atribuye como una posible razón de esto al lapso de tiempo que se les da para realizar las actividades en su totalidad, considerando que el tiempo no es suficiente para familiarizarse y experimentar con los componentes electrónicos requeridos en la práctica, dado que en el semestre que se aplica, los alumnos, no han cursado materias de electrónica.

Es por esto que, como parte del proyecto PAPIME PE111218 “Diseño de prácticas de laboratorio para fortalecer el aprendizaje de conceptos matemáticos en Ciencias Básicas", este trabajo de tesis incursiona en el desarrollo de la práctica “Aplicación de las ecuaciones diferenciales en circuitos eléctricos”, sin la necesidad de asistir a un laboratorio, con la finalidad de solventar la situación de espacios; aprovechando el novedoso panorama que ofrecen las TIC al utilizar las tecnologías de realidad como herramienta para el apoyo docente, que permita la ejecución de algunas actividades experimentales prescindiendo de los materiales físicos necesarios y aplicado en dispositivos móviles, para permitir que un mayor número de estudiantes tengan acceso a las actividades y tiempo para familiarizarse con sus componentes

1.2 Objetivos

Este proyecto tiene como objetivo emplear la tecnología de realidad aumentada (RA) para desarrollar una aplicación móvil o *app* (acortamiento del inglés de *application*), que permita la realización de las actividades dentro de la práctica de laboratorio “Aplicación de las ecuaciones diferenciales en circuitos eléctricos” a través de un motor de interacción atractivo para el público usuario, siendo este el compuesto por alumnas y alumnos que cursen la materia de ecuaciones diferenciales de Ciencias Básicas, de tal manera que la *app* desarrollada sea accesible al mayor número posible de estudiantes dentro de la FI.

1.2.1 Objetivo general

Desarrollar una *app* como recurso tecnológico para facilitar la realización de las actividades referentes al armado y análisis del circuito RLC en serie, dentro del desarrollo de la práctica de laboratorio: “Aplicación de las ecuaciones diferenciales en circuitos eléctricos”, sin la necesidad de contar con los componentes electrónicos requeridos de forma física. La *app* desarrollada convivirá de forma paralela con la práctica presencial, ya que será la alternativa en dispositivos móviles para la realización de las actividades cuando así se requiera, permitiendo a los alumnos cumplir con los mismos objetivos en ambas modalidades.

1.2.2 Objetivos específicos

- Adecuar las actividades de la práctica “Aplicación de la ecuaciones diferenciales en circuitos eléctricos” para su realización dentro de la *app*.
- Crear un motor de interacción que permita la manipulación de objetos virtuales en el entorno de la *app*.
- Crear una interfaz gráfica que permita al usuario entender el funcionamiento del motor de interacción, además de brindarle información sobre las actividades de la práctica a desarrollar.
- Lograr que la *app* creada sea compatible con la mayoría de los dispositivos móviles que utiliza el alumnado de manera cotidiana.

1.3 Hipótesis

Este trabajo plantea como hipótesis que, la tecnología RA permite la creación de modelos virtuales que simulen en apariencia y funcionamiento a los componentes necesarios para realizar las actividades dentro de la práctica de laboratorio: “Aplicación de las ecuaciones diferenciales en circuitos eléctricos”, posibilitando el desarrollo de una aplicación móvil que permita armar y analizar un circuito RLC en serie sin la necesidad de asistir a un laboratorio de forma presencial y prescindiendo de los componentes físicos, cumpliendo así con los objetivos de este trabajo.

2 MARCO TEÓRICO

2.1 Antecedentes

Las Tecnologías de la Información y Comunicación (TIC), se han convertido en una herramienta de cambio cultural y social en la actualidad, a las que se les define como

> "el conjunto de herramientas, soportes y canales desarrollados y sustentados por las tecnologías (telecomunicaciones, informática, programas, computadores e internet) que permiten la adquisición, producción, almacenamiento, tratamiento, comunicación, registro y presentación de informaciones, en forma de voz, imágenes y datos, contenidos en señales de naturaleza acústica, óptica o electromagnética a fin de mejorar la calidad de vida de las personas" [3].

Debido a su versatilidad de desarrollo, las TIC suponen una amplia gama de áreas de aplicación, siendo una de ellas su potencial dentro de la educación, principalmente por la posibilidad de crear ambientes que proporcionen el acceso a información por medios digitales, sea de forma presencial, en línea o en un dispositivo móvil, lo que permite la abundancia de recursos así como distintas perspectivas dentro del ámbito docente [4].

En vista del avance tecnológico que se ha dado en los últimos años, estas tecnologías son cada vez más accesibles; diversas instituciones educativas han tenido el interés de generar recursos que permitan el apoyo a la práctica docente. Es así que por medio de diversos estudios se ha analizado el impacto del uso de herramientas TIC para este fin, como lo fue el realizado por UNIVERITIC [5], en donde se expone el análisis de indicadores para la evaluación docente con uso de las TIC dentro de universidades españolas. En sus resultados se revela que existe

una evolución en la comunidad universitaria por la formación de competencias utilizando las TIC; sin embargo, también señalan puntos de oportunidad, entre ellos la infraestructura para utilizar elementos multimedia, como lo es un proyector, conexión a internet o bien la actualización necesaria en la formación del docente para utilizar estas herramientas adecuadamente.

Para señalar el panorama de las TIC en México, se encuentra el trabajo de la ANUIES (Asociación Nacional de Universidades e Instituciones de Educación Superior) [6] que de manera similar a [5], expone indicadores de gestión y descripción de estas tecnologías, revelando su impacto en Instituciones de Educación Superior (IES) mexicanas a nivel nacional. En este trabajo se aborda entre otros temas, los servicios que mayormente disponen para el uso de las TIC; la seguridad de la información presentada e ingresada y la calidad de las herramientas desarrolladas según la norma ISO 9001:2008, para su aplicación en laboratorios o prácticas certificadas. Así mismo, señala la posibilidad de educación de forma no presencial, dentro de la cual muestra que las herramientas más demandadas por parte de la comunidad universitaria son los servicios a través de tecnologías móviles, debido a su portabilidad y el habitual acceso a estos dispositivos.

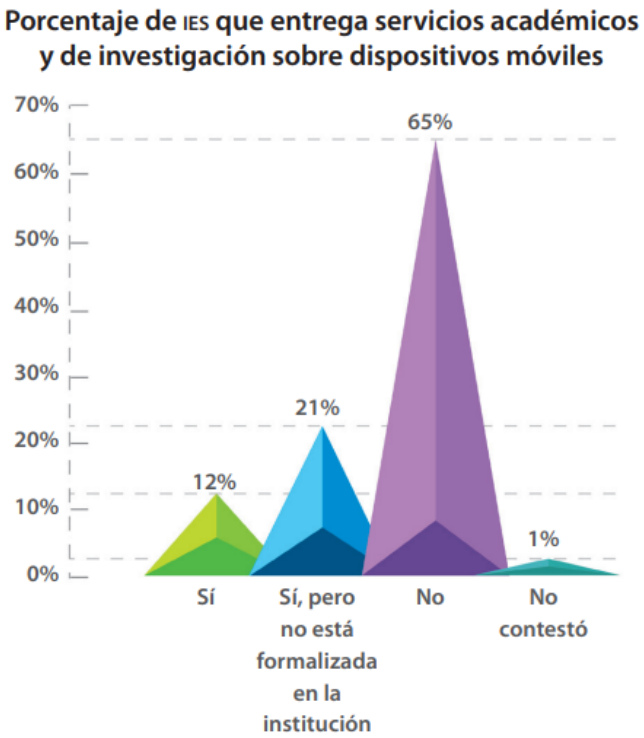


Figura 2.1 Porcentaje de IES que entrega servicios académicos y de investigación sobre dispositivos móviles [6].

De acuerdo a la información presentada en los resultados (figura 2.1), se observa que el 65% de las IES consideradas para el estudio, no cuentan con el servicio para dispositivos móviles, ya sean plataformas virtuales, avisos académicos etc. Sin embargo, es también señalado que el 21% muestra tener algún tipo de servicio de esta índole, sin formalizar. Con base en este panorama, se señala que las TIC representan una herramienta en desarrollo aplicable para apoyo a la docencia, esto mediante las distintas tecnologías pertenecientes a las TIC, entre las cuales se considera que los recursos en dispositivos móviles suponen una gran área de oportunidad.

Con el fin de evaluar la gama de tecnologías disponibles y en desarrollo para la educación, la empresa Gartner publica cada año lo que denominan *Hype Cycle,* en el que muestran la trayectoria pronosticada para el desarrollo de herramientas tecnológicas en diversos ámbitos. En torno a la educación, mostraron en 2018 el modelo sugerido en el que se observa el desarrollo de las TIC para brindar nuevas capacidades que impacten en el modelo educativo, enfocado en el uso de tecnologías móviles, virtuales y a distancia de cualquier institución [7].

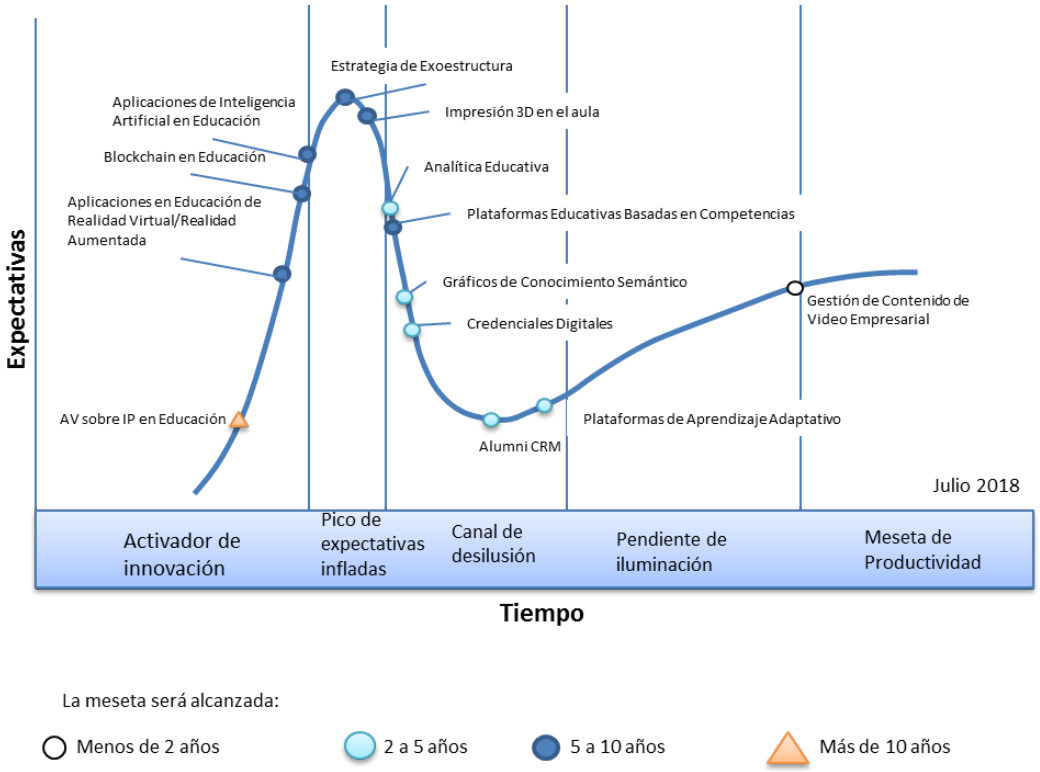


Figura 2.2 Hype Cycle para la educación, 2018 [7].

En el análisis del *Hype Cycle* (figura 2.2), se observa que en la etapa de activador de innovación se encuentran las tecnologías de realidad para la educación. Que se encuentren en esta etapa significa que se les considera con potencial de avance tecnológico significativo y que aunque su comercializabilidad aún no está probada, se mantienen en interés de los medios; a diferencia de las tecnologías que se encuentran en etapas posteriores, que suponen estar en áreas de expectativas infladas o bien en el canal de desilusión [8]. Por estas razones se considera a las tecnologías de realidad como un nuevo campo de progreso de las TIC aplicadas en el entorno educativo, en el que se pronostica un gran impacto y posibilidad de desarrollo que puede solventar el área de oportunidad demandada por el uso de dispositivos móviles, que ha expresado la comunidad universitaria de acuerdo a la investigación de la ANUIES.

2.2 Tecnologías de realidad

Como se ha hablado en el apartado anterior, las tecnologías de realidad, son un campo de desarrollo dentro de las TIC para el área docente y que satisfacen la demanda del uso de dispositivos móviles, sin embargo, estas tecnologías tienen una clasificación de acuerdo al tipo de interacción con la cual puedan encontrarse los usuarios, por lo que se pueden dividir en: Realidad Virtual (RV), Realidad Aumentada (RA) y Realidad Mixta (RM).

2.2.1 Realidad virtual

La idea de la tecnología de realidad virtual, surgió con el perfeccionamiento de la calidad gráfica ofrecida por la computadora, que permite la visualización de un mundo de tres dimensiones (3D), manipulable y con la posibilidad de ser animado [9], por lo que la RV se presenta con el fin de potenciar el mundo 3D. La primera idea de Ivan Sutherland en 1965 de que la virtualización debe acercarse a un modelo de la realidad física, establece de forma oficial el desarrollo de tecnologías para la generación de una respuesta digital lo más cercana a una respuesta real física, a través de diferentes sentidos fisiológicos como lo son el visual, el táctil y el auditivo [10].

Actualmente se puede definir a la RV como la que consiste en generar un entorno simulado y visible en 3D, en el cual la persona usuaria pueda interactuar con los objetos virtuales que no están relacionados físicamente con el entorno real, tal como se muestra en la figura 2.3.

Figura 2.3 Usuario en un sistema de inmersión RV (Expo DIMEI, Facultad de Ingeniería 2019).

Después de varias décadas de desarrollo posterior a la idea de Sutherland, debido a la variabilidad de los dispositivos para el acceso a la RV, esta puede tener varios grados de inmersión de acuerdo al nivel de interacción con el entorno virtual que permita el sistema. De acuerdo con el trabajo de Mazuryk y Gervautz, los niveles de inmersión en RV pueden clasificarse tal como se muestra en la tabla 2.1.

Tabla 2.1 Niveles de inmersión RV [9].

Desktop VR **(RV de escritorio)**	Es el nivel más simple de las aplicaciones virtuales, usualmente llamado ventana al mundo o *Window on World* (WoW), debido a que convencionalmente no existe otro tipo de dispositivo para la interacción más que un monitor gráfico en el que es posible visualizar el entorno virtual y manipularlo a través de algún tipo de periférico como puede ser un ratón en computadoras.
Fish Tank VR **(RV de pecera)**	En este nivel se mejora la visión del nivel *Desktop,* ya que es un sistema en el que se cuenta con algún dispositivo que permita el seguimiento de las señales de movimiento de cabeza con lentes de realidad, que proporcionan la sensación de libertad de visualización dentro del entorno virtual, sin embargo, usualmente no existe otro tipo de salida sensorial.
Immersive systems **(Sistemas inmersivos)**	Es el último nivel en el que se cuenta con todo tipo de sensores que permiten la mejora de la experiencia, a través del registro de movimientos corporales y con la transmisión de señales de audio y otros estímulos sensoriales.

2.2.2 Realidad aumentada

En busca del avance en la inmersión de un entorno virtual, el desarrollo de herramientas se encamina hacia la implementación de la tecnología de realidad en dispositivos móviles y que pueda interactuar en cierta medida con el entorno físico real, por lo que surge la realidad aumentada.

La realidad aumentada se puede definir como “aumentar la realimentación natural al usuario con señales simuladas” [11]. Por lo que esta realidad consiste en la creación de modelos virtuales, sean animaciones o imágenes independientes del ambiente virtual, es decir, que a diferencia de la RV, dentro de la RA los elementos 3D no conviven en un ambiente digital únicamente, ya que lo que se busca es que los componentes generados se dimensionen e interactúen con el entorno real físicamente, de tal manera que la persona usuaria no pierda la noción del mismo.

En el caso de la RA, no se busca la inmersión en un mundo virtual sino que tiene como objetivo que sea posible la visualización y manipulación de componentes virtuales de un mundo 3D en el ambiente físico real; es por ello que la mayoría de

los dispositivos para la interacción con RA son dispositivos móviles, como tabletas y teléfonos inteligentes o *smartphones* [12], como se muestra en la figura 2.4.

Figura 2.4 RA con uso de marcadores definidos en un smartphone.

A continuación, en la figura 2.5 se muestra un diagrama de funcionamiento del sistema de RA; en este se puede observar que no tiene niveles de inmersión como la RV. Para la RA su primera etapa es el mapeo o registro del entorno real, esto a través de los componentes electrónicos disponibles por el dispositivo utilizado. En el caso de los dispositivos móviles depende de su capacidad de *hardware*, ya que el registro del entorno real puede abarcar desde una visión proporcionada por una cámara única o bien ser tan robusto como para la utilización de distintos tipos de sensores para el registro de la mayor cantidad de información posible.

Figura 2.5 Diagrama general de un sistema de RA.

Una vez que se tiene la información del entorno real, el tipo de *software* utilizado en RA procesa esta información para desplegar en la pantalla del dispositivo los componentes virtuales que se requieran. Dependiendo de la capacidad de

procesamiento de información así como del conjunto de datos proporcionados por el entorno, los componentes virtuales pueden estar únicamente superpuestos visualmente, o incluso llegar a presentarse con la escala real del entorno mismo, interactuando así con los elementos reales físicamente.

Entre las herramientas que utiliza la tecnología RA para realizar el reconocimiento del entorno real y despliegue de los modelos virtuales, se encuentran los marcadores. Estos consisten en una imagen real manipulable con la impresión de un patrón formado por múltiples puntos característicos, los cuales pueden ser reconocidos por el *software* al realizar el procesamiento de la imagen y es posible superponer el modelo virtual. De esta forma, el usuario puede operar físicamente el marcador teniendo interacción en el modelo virtual. Sin embargo, en la constante evolución de la tecnología de realidad, se busca que la manipulación de los modelos virtuales resulte en una experiencia cada vez más intuitiva, por lo que las herramientas avanzadas de RA permiten la manipulación de los elementos virtuales con el seguimiento de movimientos de manos o incluso de los objetos reales tales como libros, muebles, paredes, etc., detectados al momento del reconocimiento del entorno [13].

2.2.3 Realidad mixta

La realidad mixta resulta de la evolución y combinación de las anteriores tecnologías (RA y RV) definiéndose como “el entorno predominantemente virtual donde los objetos o personas del mundo real se integran dinámicamente en mundos virtuales para producir nuevos entornos y visualizaciones, donde los objetos físicos y digitales coexisten e interactúan en tiempo real” [14]. Al surgir como la unión de las tecnologías de realidad, la RM posee un espectro de inmersión [14] que es visualizado en la figura 2.6. En este espectro se observa cómo es que la RM engloba características y elementos que conforman las tecnologías de realidad y a su vez el ambiente físico real.

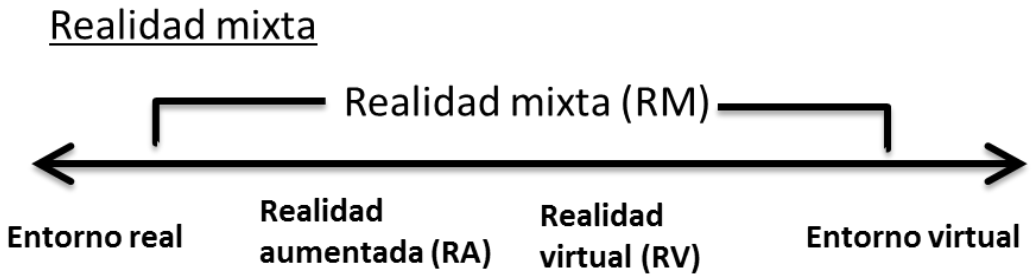


Figura 2.6 Espectro de realidad mixta (Mixed Reality Continuum) [14].

El ambiente creado por la RM acopla las ventajas de cada una de las realidades, por lo que se cuenta con diversos dispositivos, tal como los *Google Glasses* mostrados en la figura 2.7, que permiten tanto la visualización del mundo virtual, como su interacción con él. De manera similar a la RA, el primer paso de la RM es el mapeo del entorno, sin embargo, a diferencia de la RA, durante este paso la RM busca el mayor detalle posible del entorno en que se encuentre la persona usuaria, por lo que se integran diversos dispositivos como lo son sistemas de cámaras, escáneres y sensores de movimiento, de tal forma que puedan obtenerse los datos necesarios del entorno real con mayor precisión.

Figura 2.7 Uso de RM por medio de Google Glasses [15].

Seguido del mapeo, el despliegue del entorno virtual puede ser total como en RV o parcial como en RA, sea utilizando pantallas gráficas, dispositivos de visión periférica como lentes o bien proyectores de hologramas tal como se muestra en la figura 2.8. Por último, se tiene la interacción con el entorno virtual; este proceso, al igual que en la RA, busca realizarse a través de la naturalidad de los movimientos de la persona usuaria, procurando que la señal sensorial sea proporcionada por dispositivos auxiliares como guantes o trajes de cuerpo completo, cuya finalidad es que sea posible moverse libremente entre los elementos virtuales sin perder la percepción del espacio y entorno real.

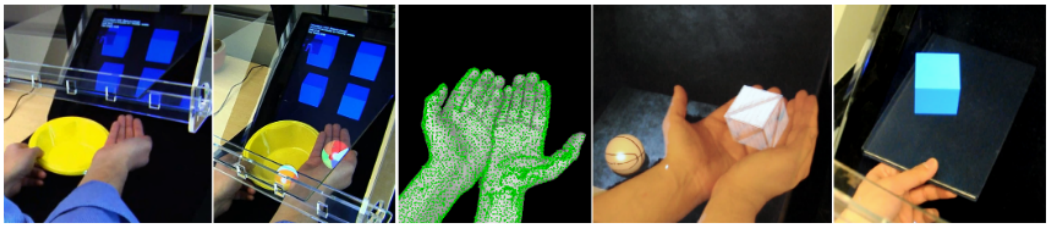


Figura 2.8 Interacción de RM por medio de hologramas del proyecto Holodesk [16].

2.3 Componentes de la realidad aumentada

En el apartado anterior se mencionaron los distintos tipos de tecnologías de realidad y la comparación entre ellas; en este apartado se abordará con mayor detalle la arquitectura que compone a la tecnología de RA y su funcionamiento.

2.3.1 Sistema de seguimiento

De acuerdo con la figura 2.5, el primer proceso para la tecnología RA es el mapeo del entorno o registro de marcadores, que es denominado *Tracking System* o sistema de seguimiento [17]. Este sistema está conformado por el conjunto de herramientas de *hardware* que permiten el reconocimiento espacial del entorno, tales como cámaras, GPS, giroscopios, acelerómetros, etc., que posibilitan la localización y determinación de la posición u orientación de los componentes físicos reales.

Se considera que este sistema otorga dos principales características para el desarrollo de la RA [18]: la primera es la determinación de la relación entre los modelos virtuales 3D a crear y la información de dirección y posición de la cámara o dispositivo que permita la correcta orientación para la visualización. La segunda es el proceso de *renderización,* el cual se define como la secuencia de pasos para generar imágenes que simulen ser texturas realistas en sus respectivos modelos 3D, lo que permite un correcto escalamiento de los componentes virtuales con el entorno real y precisión en su visualización.

Dependiendo de la robustez de este sistema, el desarrollo de la RA puede clasificarse de la siguiente manera: tecnología RA basada en marcadores y tecnología RA sin marcadores [19].

Tecnología RA basada en marcadores

Como se mencionó anteriormente el *s*istema de seguimiento registra la información del entorno real para que la tecnología RA pueda interactuar con este; en el caso de la RA basada en marcadores se reconoce un patrón / código cuando la información proporcionada es a través de una cámara contenida en algún dispositivo con sensores de orientación como lo son los *smartphones* y tabletas.

El marcador desempeña la función de activador para que sea posible superponer los modelos virtuales sobre la imagen captada del entorno real otorgada por la cámara; el patrón contenido en el marcador indica al sistema de seguimiento la ubicación, movimiento y perspectiva donde deben desplegarse los modelos virtuales, logrando así la interacción del entorno real y el modelo virtual mostrada en la figura 2.9.

Figura 2.9 Aplicación AR Solar System [20] con uso de marcadores QR.

Las implementaciones de realidad aumentada basadas en marcadores más comunes utilizan códigos de respuesta rápida QR (*Quick Response*) [19], ya que proporcionan un patrón plano único diseñado con puntos característicos para lograr un reconocimiento y procesamiento de la imagen. Sin embargo, actualmente existe *software* que permite que este marcador sea cualquier imagen seleccionada de una biblioteca ya sea por el desarrollador de la *app* o permitir al usuario seleccionar cualquier imagen plana que contenga los suficientes puntos característicos para que el procesamiento logre un correcto reconocimiento.

Tecnología RA sin marcadores

Se consideran tres tipos de sistemas RA sin marcadores: sistemas con mapas del entorno 3D previamente cargados, sistemas con extensiones de mapas conocidos desplegables para áreas desconocidas y sistemas con creación de mapas en tiempo real [21].

Los primeros dos casos de sistemas con mapas previamente conocidos, se estiman análogos a las soluciones basadas en marcadores, ya que en lugar de procesar puntos característicos de la imagen de los marcadores, se utilizan las características naturales existentes del entorno físico, tal como se muestra en la figura 2.10, por lo que se considera que cualquier superficie con suficiente textura puede convertirse en un punto característico para obtener la orientación y posición necesarias para el despliegue de los elementos virtuales. Esto crea el potencial para generar un banco de imágenes de superficie de RA predefinidas, almacenadas localmente o en la nube [21], para posteriormente recrear parcialmente o en su totalidad el entorno real captado en el conjunto de imágenes y, de esta forma, usarse para identificación y registro de orientación de cámara en un escenario de RA.

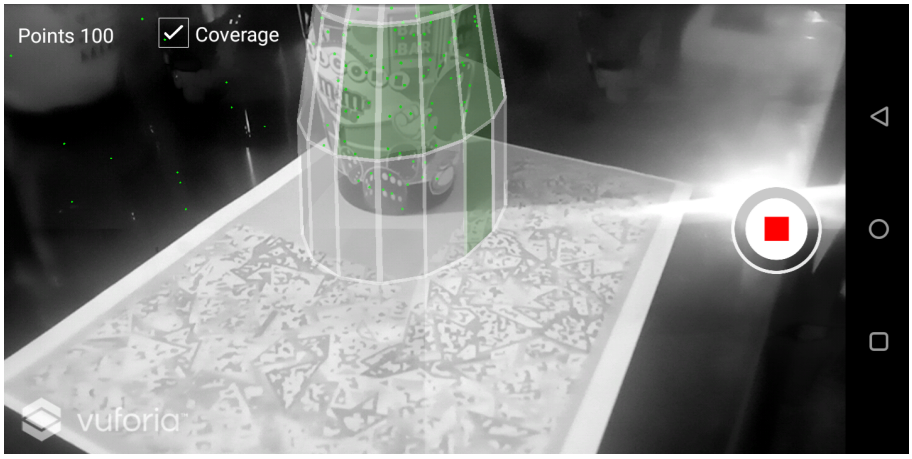


Figura 2.10 Escaneo de puntos característicos en texturas para un sistema de seguimiento sin marcadores con Vuforia [22].

En el caso de los sistemas con creación de mapas en tiempo real, la posición de la cámara de despliegue para los modelos virtuales, se estima mediante correspondencias de puntos característicos de un primer cuadro de imagen inicial y la comparación de estos mismos puntos con los siguientes cuadros del conjunto de imágenes obtenidas al mover la cámara.

Para lograr mayor robustez, este tipo de sistemas se apoya en adquirir información a partir de sensores de movimiento como se muestra en la figura 2.11, donde la información se extrae en función del movimiento captado en el cambio de posición de los puntos característicos de cuadro a cuadro del conjunto de imágenes obtenidas por la cámara. Las técnicas utilizadas, en este caso, se conocen como técnicas de localización y mapeo simultáneo (SLAM por sus siglas en inglés), donde la posición de la cámara y los mapas 3D se inicializan y actualizan simultáneamente [21], y tiene como desafío la inicialización del primer cuadro a partir de la secuencia de imágenes obtenidas por la cámara.

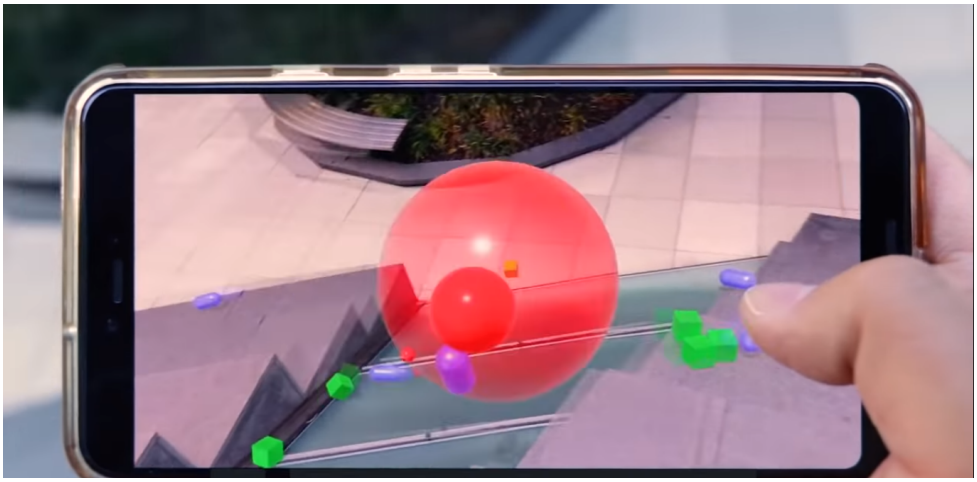


Figura 2.11 Tecnología RA sin marcadores con detección de movimiento, AR Core de Google [13].

2.3.2 Sistema de interacción

Debido a que la clasificación del sistema de seguimiento depende del uso de activadores definidos, el sistema de interacción dependerá de cuál tecnología RA sea empleada para el seguimiento.

En el caso del uso de marcadores, el usuario visualizará a través de la pantalla de despliegue, la imagen otorgada por la cámara donde será perceptible el marcador y superpuesto en él los modelos virtuales 3D, donde es posible apreciarlos como parte de la realidad; en otras palabras, cuando se interactúe con el marcador cambiándolo de posición, el objeto virtual se moverá junto con él en tiempo real. De esta forma es perceptible una interacción con los modelos virtuales desplegados al manipular físicamente el marcador registrado, tal como se muestra en la figura 2.12, por lo que si el marcador se pierde del cuadro en la cámara del dispositivo, los

modelos virtuales desaparecerán de la pantalla de despliegue, ya que la única referencia registrada por este sistema es la información otorgada por los puntos característicos procesados en la imagen del marcador en tiempo real.

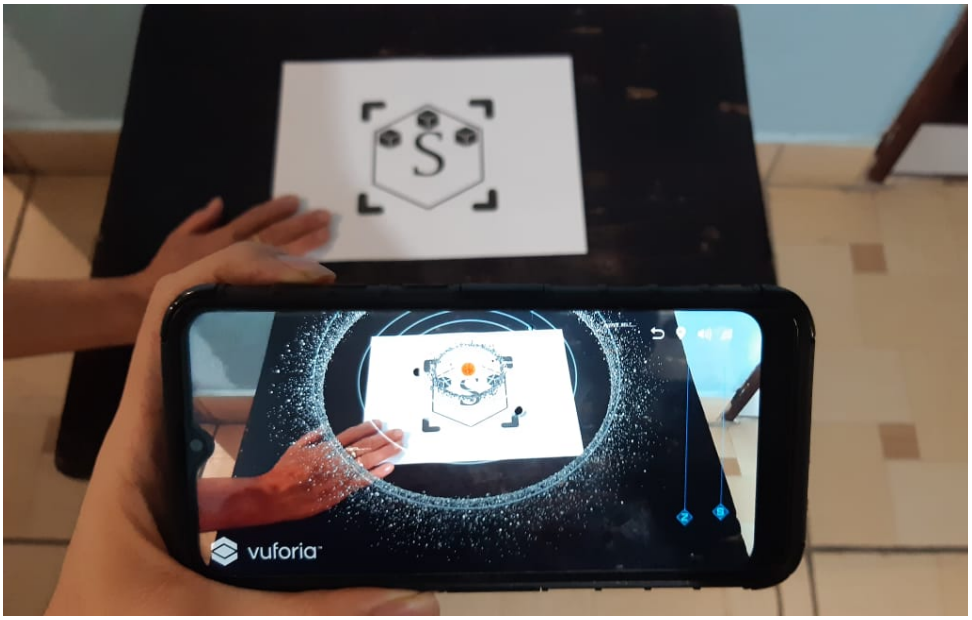


Figura 2.12 Interacción con el marcador para la aplicación AR Solar System [20].

Para la tecnología sin marcadores, la interacción permitida dependerá de la referencia definida por el sistema de seguimiento. En el caso de que el seguimiento sea a través del escaneo de algún elemento real 3D, la interacción se logrará al manipular este componente; de otra forma se dará al intervenir con el entorno completo, sea tocando las superficies planas registradas o todos los objetos detectados para el seguimiento. Conforme el sistema tenga mayor robustez y elementos de registro de datos como sensores de movimiento, es posible la interacción por medio del movimiento corporal de las extremidades del usuario, en el cuadro de la cámara del dispositivo que se esté utilizando, como es el caso mostrado en la figura 2.13.

Figura 2.13 Interacción por movimiento en la app Multi-hand Gesture Recognization Demo por OPEN AI LAB [23].

2.3.3 Despliegue

El dispositivo de despliegue es aquel que permite la visualización tanto del entorno real registrado como los objetos virtuales desplegados por la tecnología de RA. En el caso de que la visualización se realice por medio del despliegue de un monitor o pantalla de algún dispositivo móvil, se le considera un sistema VST (*video see-through* o vídeo transparente), ya que revela los objetos virtuales al capturar la imagen de escena real por medio de una cámara y superponiendo los modelos virtuales, proyectándose en la pantalla o monitor del dispositivo [24] (figura 2.14 a).

Otro tipo de despliegue se trata del sistema OST (*optical see-through* u óptica transparente), en el cual se fusiona el objeto virtual en una superficie, ya sea transparente, o bien, en una pantalla que utiliza como fondo la imagen captada en cámara del entorno real; en este caso las herramientas de despliegue pueden ser anteojos de material transparente como es el caso de los *Google Glasses* (figura 2.14 b) o visores de visión estereoscópica [17] (figura 2.14 c).

a) *b)* *c)*

Figura 2.14 Tecnologías de despliegue: a) pantalla de smartphone [25]; b) Google Glasses [15]; c) Samsung AR headset [26]; .

2.4 Estado de la técnica: realidad aumentada en la enseñanza universitaria

La RA aplicada con fines docentes ha sido objeto de investigación en numerosas ocasiones. Se han realizado trabajos y publicaciones referentes al impacto de diversas aplicaciones móviles, mostrando cómo esta tecnología responde correctamente a las necesidades y demandas educativas de la sociedad contemporánea [27].

La organización *New Media Consortium* en su publicación anual *Horizon Report* de 2016 dedicada a la educación superior, realizó un análisis de las tendencias sobre las tecnologías que apoyan la labor docente a nivel global, desde los desafíos que presentan hasta su desarrollo a corto, mediano y largo plazo. Dentro de este análisis se planteó un periodo de 2 a 3 años para la implementación de las tecnologías de RA como RV en el ámbito docente. Señala que el principal potencial de la RA es el apoyo visual y dinámico que pueden tener los estudiantes de forma práctica, ya que al permitir incorporar información digital como imágenes, videos y señales de audio dentro de espacios reales, refleja un impacto positivo para el aprendizaje en entornos cercanos a la realidad, donde pueden aplicar conocimientos anteriormente estudiados en un ambiente controlado para beneficiar su experiencia en la práctica profesional [28].

En su reporte del año 2020 [29], se analizó el impacto previsto en su reporte del 2016 sobre las aportaciones de estas tecnologías, englobando tanto RA como RV y RM en una misma categoría para hablar de ellas en conjunto acerca de los parámetros a considerar para su evaluación y comparación respecto a otros recursos educativos, aspectos como la receptividad del profesorado hacia su uso, los costos para la adquisición de equipo y su impacto en la enseñanza. Se describe que un factor de importancia para la aceptación de estas tecnologías por parte de la comunidad docente es que “su implementación sea compatible con las prácticas ya existentes de los profesores, y que su costo no sea significativamente mayor al de otras alternativas ya en uso” [29].

Se encontró que las tecnologías de realidad pueden utilizarse de manera efectiva para apoyar métodos de enseñanza basados en habilidades y competencias; y que pueden ampliar el rango de experiencia práctica de aprendizaje; siempre que “su uso esté integrado en diseños holísticos de instrucción y aprendizaje” [29].

Si bien se puede analizar el impacto de las tecnologías de realidad en la educación de manera conjunta, es importante señalar cuáles son las ventajas y desventajas (tabla 2.2) del uso de estas y su relevancia dentro del ámbito académico.

Tabla 2.2 Ventajas y desventajas en tecnologías de realidad.

Tecnología de realidad	Ventajas	Desventajas
Realidad virtual	• Es posible la generación y manipulación de un mundo virtual • Permite la visualización de elementos abstractos en un ambiente simulado por computadora • El ambiente virtual puede ser tan realista como se desee	• La persona usuaria puede no recibir estímulos de su entorno real • La mayoría de los dispositivos que soportan esta tecnología pueden no ser portables • La interacción se limita al número de periféricos con los que cuente el sistema • Cada periférico es una herramienta extra de *hardware*
Realidad aumentada	• Facilidad en el acceso a dispositivos móviles donde puede utilizarse • Los usuarios pueden compartir experiencias en tiempo real debido a la portabilidad de los dispositivos • La interacción con el entorno físico real y virtual simultáneamente, provee de una experiencia con mayor libertad de movimiento	• Se requiere de la manipulación de un dispositivo móvil en todo momento • Se limita a un único nivel de inmersión parcial • La interacción con los elementos virtuales puede requerir de un componente físico para su despliegue como los marcadores
Realidad mixta	• Permite la interacción con elementos virtuales sin dejar de recibir un estímulo sensorial del entorno real • Permite la creación y manipulación de un entorno virtual al igual que RV • Puede utilizarse en dispositivos portables al igual que RA	• Requiere de una gran cantidad de procesamiento de datos para funcionar de manera fluida • Incrementa el uso de *hardware* para lograr el acoplamiento de realidades • El procesamiento de mapeo al igual que la interacción requiere de dispositivos especializados.

Tras analizar la tabla 2.2, se observa que la tecnología de RM es el sistema más completo para la inmersión de visualización y con componentes para lograr una interacción a través de movimientos corporales intuitivos, sin embargo, involucra el uso de una gran cantidad de sensores, periféricos y *hardware* en general, lo que en el área docente puede significar una dificultad para su adquisición, sobre todo en el uso de dispositivos especializados. A pesar de esto, la RA en comparación con las otras dos tecnologías de realidad, tiene una ventaja significativa al resolver la demanda del uso de dispositivos móviles portables de fácil acceso para la comunidad estudiantil y que permitan la visualización e interacción con componentes virtuales, proporcionando un apoyo asequible a la comunidad docente.

El artículo de investigación *Advantages and challenges associated with augmented reality for education: A systematic review of the literature* (Ventajas y desafíos asociados con la realidad aumentada para la educación: una revisión sistemática de la literatura) [30] recopila los artículos que se encuentran en el índice de referencias de investigaciones científicas y sociales SSCI (Social Science Citation Index) publicados hasta 2015, que abordan temas del uso educativo de la RA, que en total son 68 publicaciones. En esta investigación se muestra la preferencia del 60% sobre el total de los artículos estudiados, por el uso de dispositivos móviles como herramienta para utilizar aplicaciones con tecnología de RA debido a su portabilidad sobre otros dispositivos para fines educativos (figura 2.15).

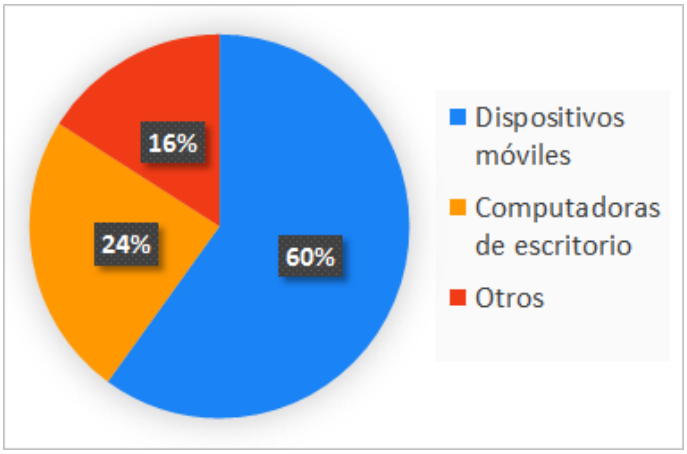


Figura 2.15 Distribución del uso de aplicaciones de RA con fines docentes según los dispositivos utilizados para su aplicación [30].

En la misma investigación [30], se menciona la tendencia de uso en diferentes niveles educativos de estas *apps* (figura 2.16), las cuales están presentes en todos los niveles educativos, desde educación básica hasta educación superior.

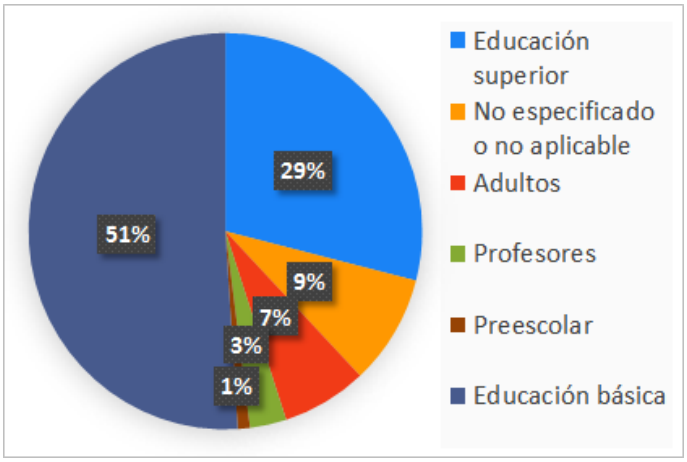


Figura 2.16 Distribución del uso de aplicaciones de RA con fines docentes según su uso en distintos niveles educativos [30].

Aproximadamente la mitad de los artículos considerados para la investigación (51%) fueron destinados a la educación de estudiantes de nivel básico; se plantea que una posible razón para que exista esta incidencia en educación básica es que estudiantes de este nivel están en una etapa de aprendizaje en la que deben ver, escuchar, o utilizar sus sentidos fisiológicos para conocer y aprender de su entorno, por lo que las características de visualización e interacción que ofrece la RA los vuelve sujetos adecuados para implementar y analizar el efecto de tecnologías de RA en la educación.

A pesar de la preferencia por el uso de la RA en la educación básica, debido a que el segundo nivel de incidencia (29%) mostrada en la figura 2.16, es la educación superior, despierta el interés de aprovechamiento que puede ofrecer en las distintas áreas de la educación universitaria. Por esta razón, Bacca-Acosta en su publicación *Augmented Reality Trends in Education: A Systematic Review of Research and Applications* (Tendencias de realidad aumentada en educación: una revisión sistemática de investigación y aplicaciones) [31], considera el análisis estadístico de 32 estudios publicados entre los años 2003 y 2013, con el fin de detallar el campo de aplicación, el nivel educativo objetivo, las ventajas, limitaciones, posibilidades y su efectividad en el ámbito docente de la RA. De acuerdo a los resultados que obtiene, mostrados en la figura 2.17, concluye que la RA ha sido aplicada con gran

incidencia en áreas afines a la educación superior, como lo son las ciencias, humanidades y la ingeniería.

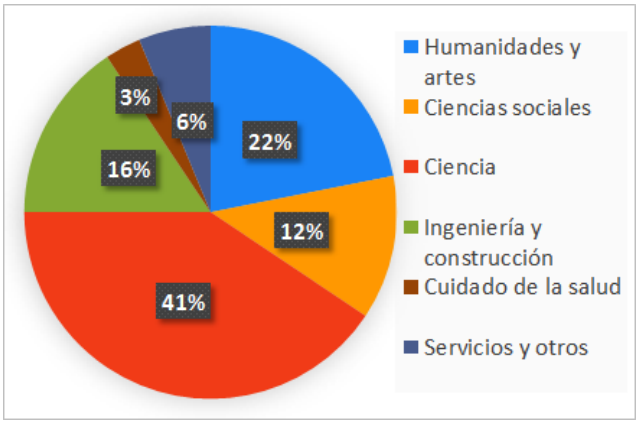


Figura 2.17 Distribución del uso de aplicaciones de RA con fines docentes en diversas áreas del conocimiento [31].

En respuesta al potencial vislumbrado que ofrece la RA para su uso en la educación superior, diversas instituciones universitarias han incursionado en el desarrollo de *apps* que permitan el apoyo al aprendizaje y docencia en distintas áreas y materias.

En la Universidad de Florida Central, Estados Unidos, se desarrolló la *app* de RA *CAM-ART* [32], como una herramienta para el reforzamiento del aprendizaje de temas de construcción estudiados en la carrera de ingeniería civil. Desarrollada con el *software* Junaio, cuenta con un sistema de seguimiento de marcadores que detecta las imágenes contenidas en un libro referente al tema de interés, para sobreponer en este la visualización de modelos 3D, vídeos y otros contenidos multimedia (figura 2.18).

Tras realizar pruebas, se comparó el aprendizaje mostrado por parte de un grupo de estudiantes que utilizaron la *app* como apoyo durante su educación, contrastado con el aprendizaje de estudiantes que no hicieron uso de esta herramienta y concluyeron sus estudios a través de un método tradicional. Los alumnos que emplearon la *app CAM-ART* como complemento de sus estudios, presentaron una mayor retención de información a la del grupo que tomó clase de manera tradicional, al responder con notable mejoría un examen aplicado un mes después de haber estudiado los temas con ayuda de la *app*. Aunado a esto, el grupo que utilizó la *app CAM-ART,* respondió una encuesta en escala Likert, calificando la *app* como una herramienta altamente efectiva y recomendada para su uso en otras clases por otros estudiantes [32].

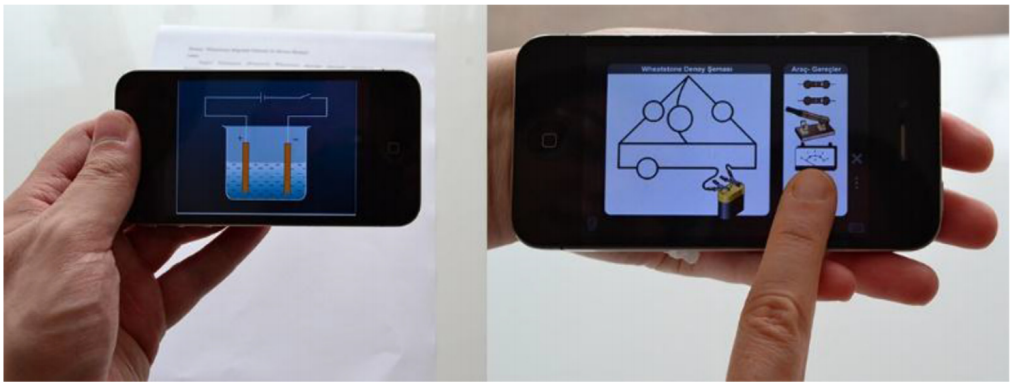
a) *b)*

Figura 2.18 Uso de la aplicación CAM-ART: a) vista en tercera persona; b) captura de pantalla del dispositivo móvil [32].

En la Universidad Kırıkkale, en Turquía [33], se desarrollaron 5 *apps* de RA, con el objetivo de mejorar el desempeño por parte del alumnado que realiza las distintas prácticas experimentales dentro del laboratorio de la materia Física General II. Cada *app* comprende respectivamente el contenido de las prácticas: electrólisis del agua, ley de Ohm, puente de Wheatstone, leyes de Kirchhoff y conexión de un motor trifásico, propias del laboratorio. La *app* utiliza marcadores para desplegar y permitir la visualización de videos, gráficos y enlaces a material complementario, con los cuales los estudiantes interactúan a través de la pantalla de su dispositivo (figura 2.19). El objetivo del trabajo fue la comparación del desempeño al realizar las prácticas por parte de los estudiantes que utilizaron estas herramientas, respecto a las habilidades mostradas por parte de un grupo de control que concluyó los experimentos por medio de métodos tradicionales.

Se evaluaron los conocimientos de los estudiantes de ambos grupos antes y después del experimento, a través de un cuestionario de evaluación de habilidades de laboratorio que consideraba aspectos como capacidad de realizar suposiciones e hipótesis experimentales, evaluación de procedimientos experimentales, y la comunicación y análisis de los resultados de un experimento. Se determinó que antes del experimento no había diferencias significativas entre las habilidades de ambos grupos, mientras que al terminar las 5 semanas, existía una diferencia importante a favor del grupo que utilizó las *apps* como complemento de la realización de sus prácticas de laboratorio. Además de esto, se realizaron entrevistas con los alumnos en las que comentaron que con el uso de las *apps* de RA “fueron capaces de terminar los experimentos en menor tiempo comparándolas con su trabajo previo en otros laboratorios, mencionando que los contenidos

visuales de la *app* les ayudó a realizar los experimentos, siendo capaces de terminarlos de manera más cómoda y fácil" [33].



a) *b)*

Figura 2.19 Ejemplos de apps de RA con experimentos de física hechos para verificar su uso en un laboratorio: a) electrólisis del agua; b) ley de Ohm [33].

En México en el ámbito nacional, la Universidad Autónoma Metropolitana [34] desarrolló una *app* que permite la identificación automática de circuitos integrados (CI) comúnmente utilizados para realizar operaciones lógicas básicas, cuyo objetivo es apoyar en el estudio de estos circuitos a alumnos en los cursos iniciales de ingeniería eléctrica y electrónica. Para realizar el reconocimiento, la *app* comienza con la captura de la posición del circuito armado en una tableta de prototipos y la identificación de los puntos invariantes de la imagen en cuadro obtenida a través de la cámara del dispositivo móvil. Posteriormente, identifica el número de matrícula del CI utilizando procesamiento digital de la imagen, para desplegar diferentes capas de información sobre la imagen, como información del circuito eléctrico, orden de los pines y diagrama lógico del CI (figura 2.20).

Para verificar su utilidad, se realizó un estudio cualitativo a través de una encuesta realizada a alumnos que utilizaron la *app* como complemento de una actividad de laboratorio. En este estudio, los estudiantes expresaron tener un alto nivel de satisfacción con el despliegue de información del CI utilizado, permitiéndoles realizar sus prácticas de circuitos lógicos de manera cómoda, en comparación con tener que consultar constantemente las hojas de especificaciones de los componentes. Además de esto, mencionaron que el sistema de RA propuesto atrae su atención y disposición hacia los temas del curso.

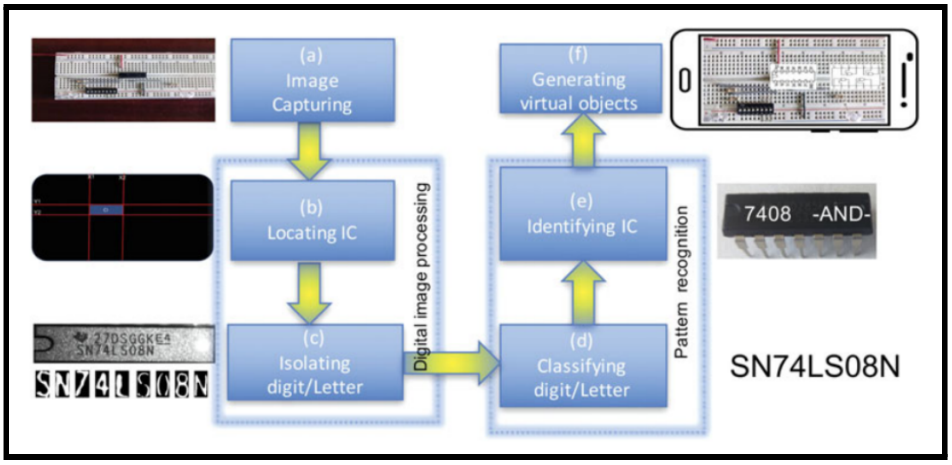


Figura 2.20 Diagrama de funcionamiento de la app para el reconocimiento de circuitos integrados [34].

En la Universidad Nacional Autónoma de México (UNAM) se han desarrollado proyectos con el uso de la tecnología RA para fines educativos, como es el caso de la *app Fetoteca* [35], que se desarrolló en la Facultad de Ingeniería con el objetivo de tener una colección de modelos virtuales de fetos en distintos tiempos de gestación, los cuales puedan ser visualizados y estudiados por el alumnado de la licenciatura de medicina; resolviendo la problemática que representa la conservación y constante manipulación de un material de este tipo. Fue hecha en el entorno de desarrollo Unity, apoyándose en el *software* ARCore para hacer uso de la RA. Utiliza un sistema de seguimiento sin marcadores por lo que es necesario detectar únicamente una superficie plana sobre la que hace el despliegue de los modelos virtuales de los fetos (figura 2.21). A través de una encuesta, se determinó la satisfacción del uso de la *app* en el cumplimiento de sus objetivos.

Figura 2.21 Modelo de feto en la app Fetoteca [35].

Otro aporte importante de la UNAM es la *app SQ sustancias: elementos y compuestos* [36], que busca apoyar la enseñanza de la materia de Química a alumnos de nivel licenciatura a través de la visualización de enlaces químicos entre diferentes elementos utilizando RA. Desarrollada en Unity con apoyo del *software* Vuforia, esta *app* realiza un seguimiento de distintos marcadores que representan cada uno, un elemento químico distinto. Al detectarse la imagen, se superpone un modelo 3D del elemento para su visualización y, al identificar contacto entre dos marcadores de elementos compatibles, se muestra el enlace químico que se produciría (figura 2.22).

Para comprobar su utilidad, se realizó una encuesta de uso según el estándar *System Usability Scale* (Escala de uso del sistema o SUS), obteniendo un 87.5 de 100 puntos máximos.

Figura 2.22 Modelos de elementos químicos en la app SQ sustancias: elementos y compuestos [36].

Según los estudios consultados y las *apps* descritas, la RA ha tenido una vasta implementación en el área docente, en el que resaltan sus aportaciones en la enseñanza de educación superior. En estas *apps* se pueden observar resultados favorables en aspectos como el mejoramiento del aprendizaje, la facilitación de tareas complejas, la manipulación y observación de materiales de difícil acceso, así como el aumento del interés y la disposición por parte de los estudiantes hacia los temas expuestos y, en general, un gran nivel de aceptación de esta tecnología para el apoyo del aprendizaje.

2.5 Práctica: aplicación de las ecuaciones diferenciales en circuitos eléctricos

Como se planteó en los objetivos de esta tesis, la *app* desarrollada busca facilitar la realización de la práctica de laboratorio “Aplicación de las ecuaciones diferenciales en circuitos eléctricos” fuera del laboratorio, a través del uso de la tecnología de RA.

Con fin de establecer los parámetros a seguir en el diseño de la *app* de acuerdo a las instrucciones de la práctica elegida (incluida en el anexo 1), se identifican los puntos principales que cubre de forma presencial, los cuales son:

Los objetivos a cumplir.

Esta práctica consta de 3 objetivos, que los alumnos cumplen a través de una serie de actividades que los guían en el entendimiento de los conceptos, los cuales son:

- Determinar el valor de la inductancia en un circuito RL
- Verificar experimentalmente el valor de la resistencia que se necesita para que un circuito RLC en serie sea críticamente amortiguado, y además corroborar el rango de valores que aquél puede tener para que el sistema tenga una respuesta subamortiguada o sobreamortiguada
- Comprobar la respuesta en estado permanente en el circuito RLC en serie con fuente de alimentación sinusoidal.

De acuerdo con los comentarios por parte de la profesora aplicadora de la práctica, Adriana Yoloxóchil Jiménez Rodríguez, el formato se ha presentado con el cambio del primer objetivo, ya que se consideró que una forma de agilizar la actividad es prescindiendo del cálculo de la inductancia, siendo preferible que este valor, al igual que su resistencia asociada, se les proporcione a los alumnos como constantes conocidas, para posteriores cálculos de interés en el circuito RLC en serie. Por esta razón, los objetivos se adaptan dentro del desarrollo de la *app*, permitiendo a los alumnos cumplirlos en su totalidad, considerando que el primer objetivo es un dato proporcionado por el profesor aplicador, con el fin de lograr una simulación aproximada al trabajo que se realiza de forma presencial al seguir con las instrucciones de la práctica.

Los materiales y componentes requeridos.

Se identifican los materiales necesarios para la realización del experimento, y en la tabla 2.3 se determina el tipo de representación que tendrá cada componente dentro del ambiente virtual de la *app*.

Tabla 2.3 Material requerido para la práctica presencial y su correspondiente representación virtual.

Material en práctica presencial	**Representación dentro de la *app***
Osciloscopio	Modelo 3D con graficador de funciones en tiempo real
Generador de funciones	Modelo 3D con simulación de funciones: cuadrada y sinusoidal; ambas variables con valores de amplitud y frecuencia
Resistores de 68, 180, 820, 1000 y 1800 Ω	Modelo 3D con valor constante de las denominaciones requeridas
Un condensador de 0.22 μF	Modelo 3D con valor constante de la denominación requerida
Inductor de 50 mH	Modelo 3D con valor constante de la denominación requerida
Una tableta de experimentación (*protoboard*)	Modelo 3D anexado al generador de funciones, con espacio para interacción y conexión de los componentes eléctricos

El desarrollo del experimento a realizar.

Las actividades de la práctica consisten en el armado y análisis de respuesta de los circuitos eléctricos que se describen brevemente a continuación:

1. Armar un circuito RLC serie, alimentado por una señal cuadrada, y verificar el tipo de respuesta en el osciloscopio (subamortiguado, sobreamortiguado, críticamente amortiguado). Repetir este proceso para tres valores de resistencia diferentes.
2. Para el circuito anterior, cambiar la alimentación a sinusoidal, observando amplitud y frecuencia de las señales de entrada y salida, así como el ángulo de desfase entre estas.

Debido a que los objetivos del presente trabajo y de la práctica a realizar están relacionados con el tema de análisis de un circuito RLC en serie, se considera

apropiado hacer un breve análisis del desarrollo de este circuito, las ecuaciones que lo caracterizan y su respuesta a una alimentación de corriente directa y de corriente alterna.

2.5.1 Generalidades de las ecuaciones diferenciales de un circuito RLC en serie

Un circuito RLC conectado en serie tiene la configuración mostrada en la figura 2.23. Se trata de un sistema físico que consta de un resistor, un condensador eléctrico y un inductor, conectados en serie a una fuente de alimentación. Dado su comportamiento, puede ser utilizado como un filtro en el procesamiento de señales analógicas.

Para analizar el comportamiento del circuito, es conveniente comprender el funcionamiento eléctrico de cada uno de los elementos que lo componen, por lo que a continuación se realiza el análisis de cada uno de estos componentes con base en el trabajo de Jaramillo Morales y Alvarado Castellanos [37] .

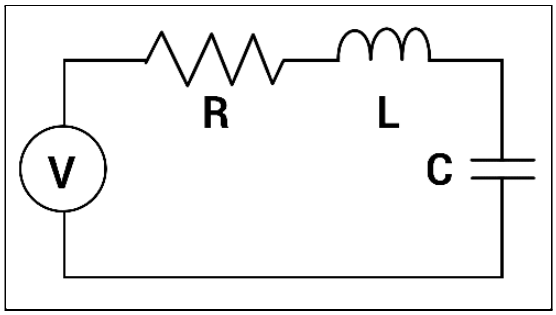


Figura 2.23 Circuito RLC en serie.

La capacitancia C se define como la cantidad de carga eléctrica que un dispositivo puede almacenar entre sus terminales por unidad de diferencia de potencial entre ellas. El condensador es entonces, un dispositivo electrónico pasivo capaz de almacenar energía sustentando un campo eléctrico. Cuando una corriente eléctrica I circula por las terminales del condensador, comienza a almacenar carga y generar una diferencia de potencial entre sus terminales, permitiendo un paso de corriente proporcional al cambio de esta diferencia de potencial respecto al tiempo. La relación entre el voltaje en el condensador V_C y la corriente I que circula entre sus terminales está dada por la ecuación 2.1.

$$I = C \frac{d}{dt} V_C \tag{2.1}$$

La inductancia L se define como la relación entre la corriente que circula por un dispositivo y el flujo electromagnético que se genera en este; en un fenómeno conocido como autoinducción, este flujo magnético genera a su vez una fuerza electromotriz (FEM) inducida en el dispositivo. El inductor se puede definir como un dispositivo electrónico pasivo capaz de almacenar energía sustentando un campo electromagnético. Cuando se aplica una diferencia de potencial entre las terminales de un inductor, comienza a circular la corriente que genera el flujo magnético y la FEM en el dispositivo, haciendo variar el voltaje entre sus terminales de manera proporcional a la variación de la corriente respecto al tiempo. La relación entre el voltaje del inductor V_L y la corriente I que circula entre sus terminales está dada por la ecuación 2.2.

$$V_L = L \frac{d}{dt} I \tag{2.2}$$

La resistencia R es la medida de la oposición al flujo de corriente a través de cualquier elemento conductivo, y se define como la relación entre el voltaje en las terminales de la resistencia V_R y la corriente I que circula a través de este, como se muestra en la ecuación 2.3.

$$V_R = R\, I \tag{2.3}$$

De las ecuaciones 2.1 y 2.2, se observa que los cambios en los valores de voltaje y corriente de un condensador y un inductor están relacionados con una variación diferencial respecto al tiempo, por lo que para describir un circuito RLC, se necesita plantear su ecuación diferencial (ED). En particular, se resolverá la ecuación diferencial que describe el comportamiento del voltaje del condensador, con el que se puede obtener la corriente del circuito a través de la ecuación 2.1 y con esta el voltaje de los demás componentes con las ecuaciones 2.2 y 2.3.

Aplicando la ley de tensiones de Kirchhoff al circuito de la figura 2.23, se obtiene la ecuación 2.4, siendo V_f el voltaje de alimentación del circuito.

$$V_f = V_R + V_L + V_C \tag{2.4}$$

Sustituyendo los valores de V_R y V_L de las ecuaciones 2.2 y 2.3 en la ecuación 2.4:

$$V_f = R\,I + L\,\frac{d}{dt}I + V_C \tag{2.5}$$

Sustituyendo el valor de la corriente I de acuerdo con la ecuación 2.1:

$$V_f = R\,(C\,\frac{d}{dt}\,V_C) + L\,\frac{d}{dt}\,(C\,\frac{d}{dt}\,V_C) + V_C \tag{2.6}$$

Simplificando y despejando la ecuación anterior se obtiene la ecuación 2.7, que se trata de la ecuación diferencial normalizada de segundo orden que describe el voltaje del condensador en el circuito. Se dice que es de segundo orden porque la variable de interés V_C, es derivada dos veces respecto al tiempo.

$$\frac{d^2}{dt^2}V_C + \frac{R}{L}\,\frac{d}{dt}V_C + \frac{1}{LC}V_C = \frac{1}{LC}V_f \tag{2.7}$$

Una vez planteada la ecuación diferencial que describe el comportamiento del circuito, esta se puede resolver a través de diversos métodos. Uno de ellos es el descrito por Zill y Wright [38], en el que se resuelve a través de un elemento conocido como operador anulador.

Antes de resolver la ED del circuito, se escribe una ecuación diferencial de segundo orden de forma general:

$$\frac{d^2}{dt^2}\,y + a\,\frac{d}{dt}\,y + b\,y = f(t) \tag{2.8}$$

La solución completa y de la ecuación 2.8 consta de una suma de dos partes: una solución homogénea y_H y una solución particular y_P. Se define como la solución particular o de estado permanente a la que depende únicamente del término independiente $f(t)$ de la ecuación, y refleja el comportamiento al que tiende la ecuación tras un tiempo infinito; y se define como solución homogénea o transitoria a la que depende de la ecuación con un término independiente nulo (en forma homogénea), la cual representa el comportamiento de la ecuación entre un estado inicial y la solución particular. Entonces, la solución general de la ecuación 6 se expresa en la ecuación 2.9.

$$y = y_H + y_P \tag{2.9}$$

Físicamente, estas soluciones se pueden asociar al comportamiento del circuito RLC en serie, siendo la respuesta de estado permanente su comportamiento

cuando el tiempo de excitación tiende a infinito, el cual depende de su señal de alimentación y sus componentes; y siendo la respuesta transitoria su comportamiento al inicio de su excitación y que depende únicamente de sus componentes y de sus condiciones iniciales.

Para obtener la respuesta transitoria y_H de la ecuación 2.7 correspondiente al circuito RLC, se analiza en su forma homogénea (considerando una excitación nula), y considerando el voltaje del condensador V_C como la variable y tal como se muestra en la ecuación 2.10.

$$\frac{d^2}{dt^2}\, y + \frac{R}{L}\, \frac{d}{dt}\, y + \frac{1}{LC}\, y = 0 \tag{2.10}$$

Las derivadas de la ecuación se pueden representar con la notación del operador D , tal que la ecuación anterior queda de la siguiente forma:

$$D^2\, y + \frac{R}{L}\, D\, y + \frac{1}{LC}\, y = 0 \; ; \; D^n y = \frac{d^n}{dt^n}\, y \tag{2.11}$$

Factorizando la variable de interés y, se obtiene la ecuación 2.12:

$$L_H(y) = \left[D^2 + a\, D + b\right] y \tag{2.12}$$

Donde a y b representan las constantes $\frac{R}{L}$ y $\frac{1}{LC}$ respectivamente, mientras que L_H es el operador diferencial de la ecuación diferencial homogénea. Un operador diferencial L tiene una ecuación auxiliar Pasociada conocida también como ecuación característica, la cual es un polinomio cuyos exponentes dependen del orden de las derivadas del operador anulador. Por lo que la ecuación auxiliar es de la forma:

$$L = D^n + D^{n-1} + \ldots + D^1 + n \Rightarrow P = s^n + s^{n-1} + \ldots + s^1 + n \tag{2.13}$$

Para obtener la solución homogénea y_H, se obtienen las raíces o valores característicos de la ecuación auxiliar P_H asociada al operador diferencial L_H, de tal forma que P_H auxiliar del circuito es representado en la ecuación 2.14:

$$P_H = s^2 + a\, s + b \tag{2.14}$$

Por lo que la solución homogénea y_H de la ED es una suma de términos que dependen de las raíces s_1 y s_2 de s en el polinomio $P_H$mostrado en la ecuación 2.14. Al establecer el valor de todas las funciones independientes en cero y suponer una solución con forma exponencial, la solución de y_H queda representada en las ecuaciones 2.15 - 2.17 de la siguiente forma:

- Con raíces reales iguales:

$$s_1 = s_2 \Rightarrow y_H = c_1 e^{s_1 t} + c_2 t\, e^{s_2 t} \tag{2.15}$$

- Con raíces reales diferentes:

$$s_1 \neq s_2 \Rightarrow y_H = c_1 e^{s_1 t} + c_2 e^{s_2 t} \tag{2.16}$$

- Con raíces complejas:

$$s_{1,2} = \alpha \pm j\beta \Rightarrow y_H = c_1 e^{\alpha t} cos(\beta t) + c_2 e^{\alpha t} sen(\beta t) \tag{2.17}$$

Siendo c_1 y $c_2$constantes de integración que se obtienen al evaluar la función en sus condiciones de frontera.

Determinada la solución homogénea y_H, para obtener la solución particular y_P es necesario seleccionar y emplear el correcto operador anulador L_P con fin de anular el término V_f que hace que la ED 2.7 no sea homogénea, en otras palabras, la selección del operador anulador dependerá de la forma de la función que represente la señal de alimentación $f(t)$ del circuito RLC tal que sea suficientemente derivable con el fin de que esta sea reducida a cero. Por lo que el operador L_P se puede definir como:

$$L_P(\, f(t)\,) = 0 \tag{2.18}$$

Para el caso de que la señal de alimentación V_f tenga la forma $f(t) = t^n$, su operador anulador es:

$$f(t) = t^n \Rightarrow L_P = \left[D^{n+1}\right] \tag{2.19}$$

Sin embargo cuando la señal de alimentación V_f es una señal periódica de la forma $f(t) = sen(\omega t)$, $f(t) = cos(\omega t)$, su operador anulador es:

$$f(t) = sen(\omega t)\,,\ f(t) = cos(\omega t) \Rightarrow L_P = \left[D^2 + \omega^2\right] \quad (2.20)$$

Identificado la forma del operador anulador L_P, de manera similar a la solución homogénea, la solución particular y_P es una suma de términos que se obtienen a través de las raíces de la ecuación auxiliar $P_P$obtenida a partir de la forma del operador anulado. Entonces, con base en la ecuación 2.9, la solución completa de la ED se puede escribir de la siguiente forma:

$$y = y_H + y_P = y_{h1} + \ldots + y_{hn} + y_{p1} + \ldots + y_{pn} \quad (2.21)$$

Siendo y_h los términos generados de la solución homogénea y y_p los términos de la solución particular.

La solución completa de una ED depende de la función de la señal de entrada, por lo que en los próximos dos apartados de este capítulo, se resuelve la ED del circuito RLC considerando excitaciones de corriente directa y de corriente alterna.

2.5.2 Respuesta del circuito RLC serie con alimentación de corriente directa

Para resolver la ED del circuito RLC en serie alimentado con una fuente de corriente directa (CD), se obtiene su respuesta particular considerando una entrada constante de magnitud V_m.

$$V_f = V_m \quad (2.22)$$

A partir de la ecuación 2.7, que describe el comportamiento del voltaje del condensador en el circuito, se obtiene el operador anulador L_P de su término no homogéneo, su polinomio asociado P_P y su raíz s_1 del mismo:

$$f(t) = \frac{1}{LC}V_m \Rightarrow L_P = [D] \Rightarrow P_P = s \Rightarrow s_1 = 0 \quad (2.23)$$

A partir de la raíz del polinomio P_P, se obtiene la solución particular de la ecuación 2.7 con una entrada constante, la cual describe el estado permanente del circuito.

$$\left(V_C\right)_P = c_1 \tag{2.24}$$

En la ecuación 2.24 se observa que si el circuito eléctrico se excita con una fuente de CD, la respuesta en estado permanente del voltaje de cada componente será, de igual manera, un valor continuo, por lo que con una alimentación de CD es más importante analizar el estado transitorio del circuito.

Para describir la respuesta transitoria del circuito, se obtiene la solución homogénea de la ecuación 2.7. Se considera la ecuación 2.7 homogeneizada y se obtienen su operador L_H y su ecuación asociada P_H:

$$L_H = \left[D^2 + (\tfrac{R}{L})D + \tfrac{1}{LC}\right] \Rightarrow P_H = s^2 + \tfrac{R}{L}s + \tfrac{1}{LC} = 0 \tag{2.25}$$

La ecuación auxiliar P_H se puede reescribir en función de los parámetros que definen el comportamiento de la respuesta transitoria del sistema:

$$\omega_0 = \frac{1}{\sqrt{LC}} \quad \xi = \frac{R\sqrt{LC}}{2L} \tag{2.26}$$

$$P_H = s^2 + 2\xi\omega_0 s + \omega_0^2 \tag{2.27}$$

Siendo ξ el factor de amortiguamiento y ω_0 la frecuencia natural de oscilación del sistema. Obteniendo las raíces de la ecuación auxiliar P_H:

$$s_{1,2} = -\xi\omega_0 \pm \omega_0\sqrt{\xi^2 - 1} \tag{2.28}$$

Se observa la importancia de estos parámetros en la solución de la ED, ya que dan lugar a 3 tipos de soluciones en función del factor de amortiguamiento [39] .

- Raíces reales negativas diferentes ($\xi > 1$): Respuesta sobreamortiguada

$$\left(V_C\right)_{H1} = c_2 e^{s_1 t} + c_3 e^{s_2 t} \tag{2.29}$$

- Raíces reales negativas iguales ($\xi = 1$): Respuesta críticamente amortiguada

$$s_{1,2} = -\xi\omega_0 \Rightarrow \left(V_C\right)_{H2} = (c_1 + c_2 t)\, e^{-s_1 t} \tag{2.30}$$

- Raíces complejas con parte real negativa ($0 < \xi < 1$): Respuesta subamortiguado

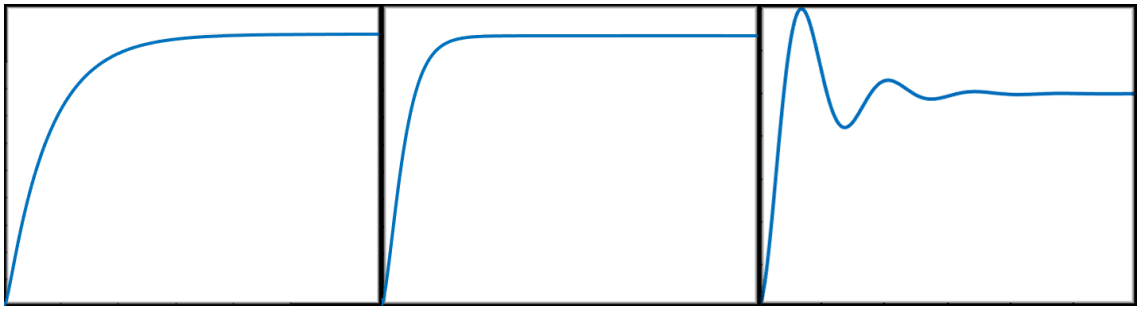
$$s_{1,2} = \alpha \pm j\beta \Rightarrow \left(V_C\right)_{H3} = c_2\, e^{\alpha t} cos(\beta x) + c_3 e^{\alpha t} sen(\beta x)\,;\ \alpha < 0 \tag{2.31}$$

Siendo c_2 y c_3 constantes de integración, definidas al evaluar la solución completa de la ecuación en sus condiciones de frontera. En estas posibles respuestas transitorias se puede observar que todas están multiplicadas por una exponencial negativa, por lo que en un tiempo infinito tienden a ser nulas.

Finalmente, con base en la ecuación 2.21, la respuesta completa de la ecuación que describe el comportamiento del voltaje del condensador en el circuito RLC con alimentación de CD tiene la forma:

$$V_C = (V_C)_H + (V_C)_P = (V_C)_H + c_1 \tag{2.32}$$

Sustituyendo cada una de las respuestas transitorias del voltaje del condensador de las ecuaciones 2.29 a 2.31 en la ecuación 2.32, se obtienen las gráficas mostradas en la figura 2.24.



a) *b)* *c)*

Figura 2.24 Gráficas de voltaje del condensador en un circuito RLC con alimentación de CD: a) sobreamortiguado; b) críticamente amortiguado; c) subamortiguado.

La respuesta subamortiguada presenta además los parámetros de interés señalados en la gráfica de la figura 2.25.

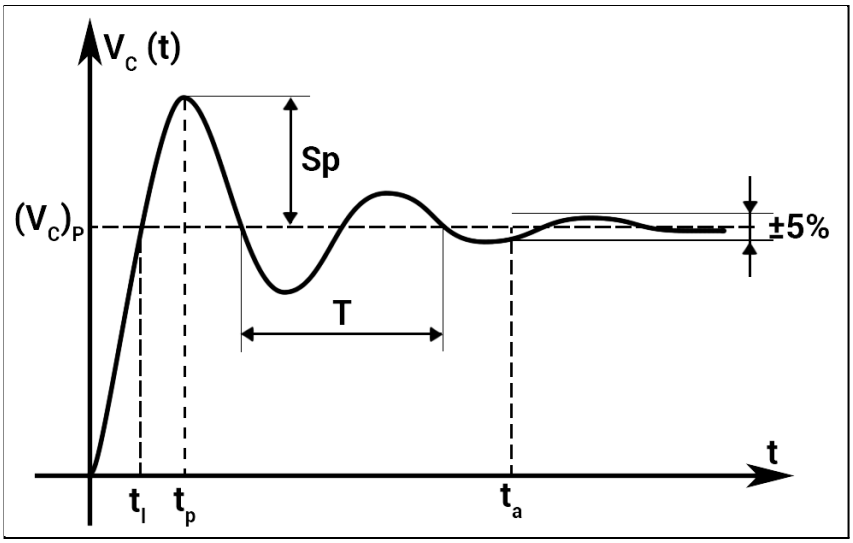


Figura 2.25 Parámetros de interés de la respuesta transitoria del voltaje de un condensador en un circuito RLC subamortiguado con alimentación de corriente directa.

Estos parámetros son:

1 Frecuencia de oscilación (f). Número de oscilaciones que tiene la respuesta del sistema por unidad de tiempo.

$$f = \frac{\omega_0\sqrt{1-\xi^2}}{2\pi} \tag{2.33}$$

2 Periodo de oscilación (T). Tiempo que transcurre en una oscilación completa de la respuesta del sistema.

$$T = \frac{1}{f} \tag{2.34}$$

3 Sobrepaso (Sp). Valor máximo de la respuesta, considerando una respuesta permanente unitaria ($V_F = 1$).

$$Sp = exp\left(\frac{-\xi\pi}{\sqrt{1-\xi^2}}\right) \tag{2.35}$$

4 Tiempo de sobrepaso (t_p). Tiempo necesario para que la respuesta alcance su valor máximo.

$$t_p = \frac{\pi}{\omega_0\sqrt{1-\xi^2}} \tag{2.36}$$

5 Tiempo de levantamiento (t_l). Tiempo necesario para que la respuesta alcance su valor permanente por primera vez.

$$t_l = \frac{\pi - cos^{-1}(\xi)}{\omega_0\sqrt{1-\xi^2}} \tag{2.37}$$

6 Tiempo de asentamiento (t_a). Tiempo que transcurre para que la respuesta oscile entre el 95% y el 105% de su respuesta de estado permanente.

$$t_a = \frac{3}{\xi\,\omega_0} \tag{2.38}$$

Para calcular los voltajes de cada componente, se sustituye el valor de la respuesta completa del voltaje del condensador de la ecuación 2.32 en la ecuación 2.1, para obtener la corriente del circuito, y a su vez la corriente se sustituye en las ecuaciones 2.2 y 2.3 para calcular el voltaje del inductor y el resistor, respectivamente.

De la solución general en la ecuación 2.32, se puede observar que la forma de la gráfica del voltaje del condensador en el tiempo se define principalmente por la solución homogénea de su ecuación diferencial, la cual describe el estado transitorio del circuito.

2.5.3 Respuesta de un circuito RLC con alimentación de señal periódica cuadrada

La entrada de voltaje de un circuito RLC puede ser una sucesión de valores continuos, que oscilan entre un voltaje nulo y un voltaje V_m, como se muestra en la figura 2.26. A una señal de este tipo se le conoce como señal cuadrada.

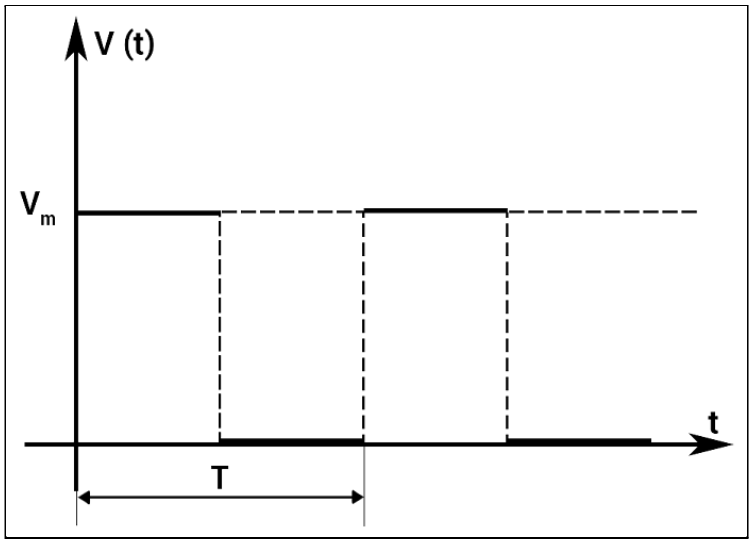


Figura 2.26 Forma de una señal cuadrada.

Esta señal se puede definir como una función de la siguiente forma:

$$V_f(t) = \begin{cases} V_m & (n)T \le t < \left(n+\frac{1}{2}\right)T \\ 0 & \left(n+\frac{1}{2}\right)T \le t < (n+1)T \end{cases} ; n \in \mathbb{Z} \tag{2.39}$$

En la ecuación 2.39, $V_f(t)$ no es una función contínua y por lo tanto no es derivable en todos sus puntos, por lo que no se puede utilizar un operador anulador al integrarla en la ED 2.7. Por esta razón se exploró una solución diferente; la resolución de la ED a través de un método numérico. Este método no requiere que la expresión sea una función derivable, por lo que se puede utilizar la expresión de la ecuación 2.39 como señal de excitación del circuito RLC en la ecuación 2.7.

Los métodos numéricos son métodos matemáticos de resolución de ecuaciones en variable discreta. Son útiles en situaciones en las que no es posible obtener una solución analítica de un problema, cuando el volumen de datos es muy grande, o cuando una solución con variables discretas es más apropiada por diversos factores, como es el caso de tener una función no derivable.

Para resolver una ecuación diferencial con un método numérico, se puede usar alguno de los métodos propuestos por Runge-Kutta. Se trata de una familia de métodos de aproximación punto a punto usados en discretización temporal para la resolución de ecuaciones diferenciales ordinarias, realizado a partir de un conjunto de valores iniciales y una distancia entre cada punto en el que se aproxima la

función, conocido como paso. Este método se aplica paso a paso para calcular valores posteriores y, por lo tanto, trazar la trayectoria de la solución. En particular, se describe el método Runge Kutta de cuarto orden, llamado así por aproximar en 4 pasos la pendiente entre un punto y otro de una función discreta [40]**.**

En el apéndice C se detalla el procedimiento para que, a partir de una ecuación diferencial de segundo orden como la ecuación 2.7 y un conjunto de valores iniciales, se obtenga la aproximación punto a punto de los valores siguientes de la variable V_C a partir de las ecuaciones C.14 a C.17, C.21 a C.24, C.39 y C.40.

La resolución de la ED del circuito con alimentación de CD a través del método descrito, está enfocado a la obtención y análisis de su respuesta transitoria, caso contrario a un circuito alimentado con una señal de corriente alterna, como se analiza en el siguiente apartado.

2.5.4 Respuesta de circuito RLC serie con alimentación de corriente alterna

En las ecuaciones 2.29 a 2.31, se observa que la respuesta homogénea de la ED que describe el estado transitorio del voltaje del condensador en el circuito RLC tiende a cero después de un tiempo determinado, por lo que al analizar la respuesta del circuito con alimentación de corriente alterna (CA) es más importante su respuesta de estado permanente, dado que la entrada es una señal periódica que oscila indefinidamente.

Para describir la respuesta de estado permanente del voltaje del condensador, se obtiene la solución particular de la ecuación 2.7 considerando una entrada cosinusoidal de la forma:

$$V_f = V_m cos(\omega t - \theta_f) \tag{2.40}$$

Siendo ω la frecuencia de la señal y θ_f el ángulo de desfase de la señal. Suponiendo un desfase de cero, la señal de voltaje es:

$$V_f = V_m cos(\omega t) \tag{2.41}$$

Se obtiene su operador anulador L_P y las raíces de su polinomio característico P_P

$$L_P = \left[D^2 + \omega^2\right] \Rightarrow P_P = s^2 + \omega^2 \Rightarrow s_{1,2} = \pm j\omega \quad (2.42)$$

Con estas raíces, se construye la solución particular de la ecuación diferencial

$$\left(V_C\right)_P = c_1\, cos(\omega t) + c_2\, sen(\omega t) \quad (2.43)$$

Siendo c_1 y c_2 constantes de integración. Esta ecuación se puede simplificar como

$$\left(V_C\right)_P = k\, cos(\omega t - \alpha) \quad (2.44)$$

Siendo k y α constantes obtenidas de la simplificación trigonométrica de la suma de seno y coseno. Sustituyendo el voltaje del condensador de la ecuación 2.44 en la ecuación 2.1 y simplificando, se obtiene la ecuación 2.45 que describe la corriente del circuito.

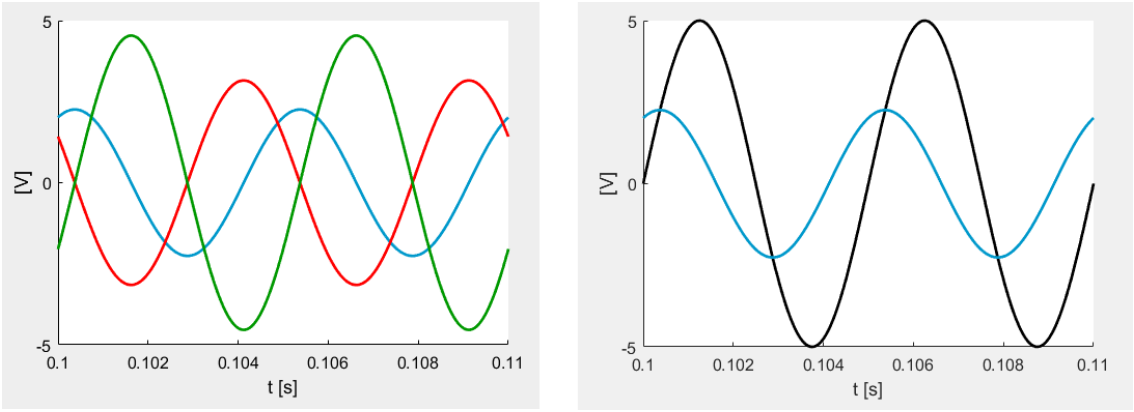
$$I = C\, k\, \omega\, cos(\omega t - \alpha - 90°) \quad (2.45)$$

Sustituyendo la corriente de la ecuación 2.45 en las ecuaciones 2.2 y 2.3, se obtiene el voltaje del inductor y la resistencia, respectivamente.

$$\left(V_L\right)_P = L\, C\, k\, \omega^2\, cos(\omega t - \alpha - 180°) \quad (2.46)$$

$$\left(V_R\right)_P = C\, k\, \omega^2\, cos(\omega t - \alpha - 90°) \quad (2.47)$$

De las ecuaciones 2.44, 2.46 y 2.47, se puede observar que el voltaje del condensador e inductor tienen un ángulo de desfase de 90° del voltaje del resistor, atrasándose y adelantándose respectivamente. Por otra parte, el voltaje de la resistencia (que está en fase con la corriente del circuito de la ecuación 2.71) tiene un ángulo de desfase con respecto a la señal de entrada del voltaje, que depende de los valores de resistencia, capacitancia e inductancia de los componentes del circuito y la frecuencia de la señal de entrada. Estos comportamientos se pueden observar en la figura 2.27.



a) *b)*

Figura 2.27 Respuesta de un circuito RLC con alimentación sinusoidal: a) gráficas de voltaje de entrada (negro) y voltaje de la resistencia (azul) b) gráficas de voltajes de resistencia (azul), condensador (verde) e inductor (rojo, aumentada 40 veces).

2.5.5 Análisis fasorial de un circuito RLC serie con alimentación de corriente alterna

Al analizar el comportamiento del circuito RLC alimentado con una fuente de corriente alterna (CA), resulta conveniente un análisis enfocado a este tipo de señal, conocido como análisis fasorial, de acuerdo al desarrollo planteado por Robbins y Miller [41].

Un fasor se define como un vector rotatorio cuya proyección en el eje vertical se usa para representar cantidades que varían de forma sinusoidal, como se muestra en la figura 2.28.

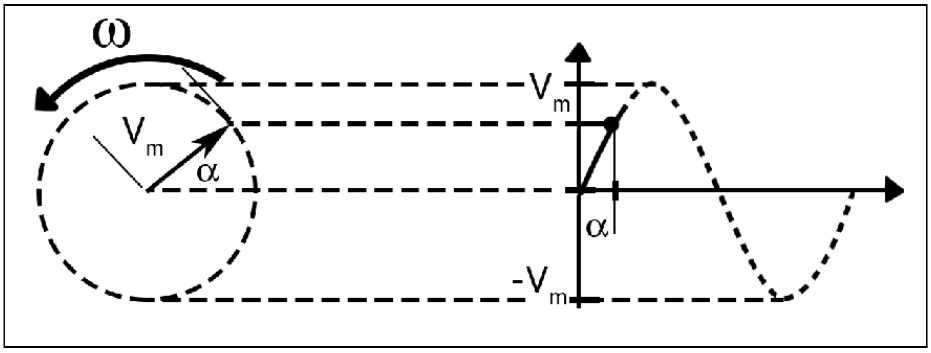


Figura 2.28 Representación gráfica de la relación entre un fasor y una señal sinusoidal [41].

Si el fasor rota con una velocidad angular ω, el ángulo α es $\alpha = \omega t$ y su proyección vertical es una ecuación senoidal de la forma de la ecuación 2.41. Esta representación se cumple si el fasor comienza a girar desde un ángulo $\alpha = 0$ cuando la gráfica está en un punto $t = 0$. Por otra parte, si el fasor comienza desde un ángulo diferente, representa una ecuación senoidal con un desfase similar a la ecuación 2.40, como se muestra en la figura 2.29.

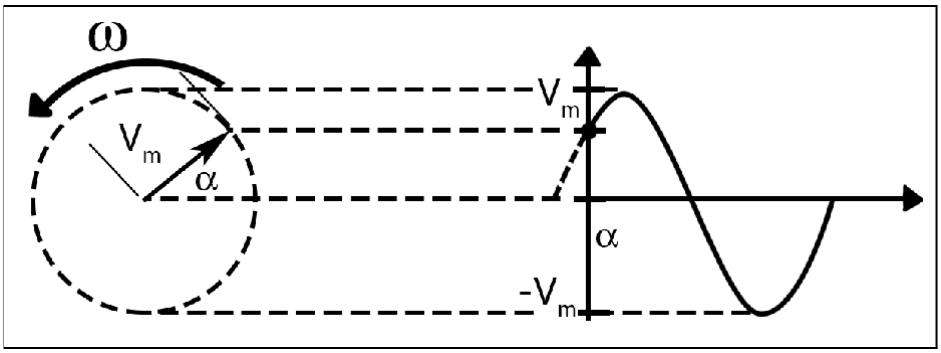


Figura 2.29 Fasor con un ángulo de desfase inicial y la sinusoidal que representa [41].

Dadas estas propiedades, el fasor se puede utilizar para representar las señales de voltaje y corriente sinusoidales de un circuito. Un fasor se escribe con la siguiente notación a partir de una señal sinusoidal.

$$y = y_m cos(\omega x) \Rightarrow \overline{Y} = Y_p \angle \theta_Y \tag{2.48}$$

Siendo Y_p la amplitud de la sinusoidal y θ_Y su ángulo de desfase. En forma fasorial, la ecuación 2.41 que describe la entrada de voltaje del circuito se escribe como:

$$\overline{V_f} = V_m \angle 0 \tag{2.49}$$

Para continuar con este análisis, se debe definir también el concepto de reactancia en cada uno de los componentes del circuito RLC.

Para describir la reactancia del inductor, se toma la ecuación 2.2 que describe la relación entre su voltaje y corriente, y se desarrolla de la siguiente forma:

$$V_L = L \frac{d}{dt} I_m sen(\omega t) = \omega L \, (I_m cos(\omega t)) = \omega L \, I_m (sen(\omega t + 90°)) \tag{2.50}$$

Por la identidad trigonométrica $cos(\omega t) = sen(\omega t + 90°)$, el voltaje del inductor se puede escribir como una ecuación trigonométrica de la forma

$$V_L = V_{lm}\, sen(\omega t + 90°) \tag{2.51}$$

Siendo V_{lm} el voltaje pico del inductor, donde:

$$V_{lm} = \omega L\, I_m \tag{2.52}$$

Despejando se obtiene:

$$\frac{V_{lm}}{I_m} = X_L = \omega L \tag{2.53}$$

La razón expresada en la ecuación 2.53 se define como reactancia inductiva, se representa con el símbolo X_L y tiene unidades de ohms. Esta relación representa la oposición que presenta la inductancia al paso de la corriente para el caso de CA sinusoidal. En la ecuación 2.51, se puede observar además que el voltaje de un inductor tiene un ángulo de desfase de 90° respecto a su corriente.

Para el condensador, se sigue un procedimiento similar, partiendo de la ecuación 2.1 y suponiendo un circuito capacitivo puro.

$$I_C = C\frac{d}{dt} V_C = C\frac{d}{dt}\left(V_{Cm} sen(\omega t)\right)$$

$$I_C = \omega C\, V_{Cm} cos(\omega t) = \omega C\, V_{Cm} sen(\omega t + 90°) \tag{2.54}$$

La corriente del condensador se puede escribir como:

$$I_C = I_m\, sen(\omega t + 90°) \tag{2.55}$$

Siendo I_m la corriente pico del condensador, donde:

$$I_m = \omega C\, V_{Cm} \tag{2.56}$$

Despejando:

$$\frac{V_{Cm}}{I_m} = X_C = \frac{1}{\omega C} \tag{2.57}$$

La razón expresada en la ecuación 2.57 se define como la reactancia capacitiva, se representa con el símbolo X_Cy tiene unidades de ohms. Esta relación representa la oposición que presenta la capacitancia al paso de la corriente para el caso de CA

sinusoidal. En la ecuación 2.55, se observa que la corriente del condensador tiene un ángulo de desfase de 90° respecto a su voltaje.

Finalmente, la reactancia del resistor X_R se describe en la ecuación 2.59 a partir de la ecuación 2.3 aplicando la ley de Ohm. En la ecuación 2.58 se observa que la corriente y el voltaje de un resistor no tienen un ángulo desfase entre sí.

$$V_R = R\,I_R = R\,I_m\,sen(\omega t) \tag{2.58}$$

$$\frac{V_R}{I_R} = X_R = R \tag{2.59}$$

Conociendo estas relaciones, se puede plantear el análisis fasorial del circuito. En el dominio fasorial, el circuito de la figura 2.23 se representa como se muestra en la figura 2.30.

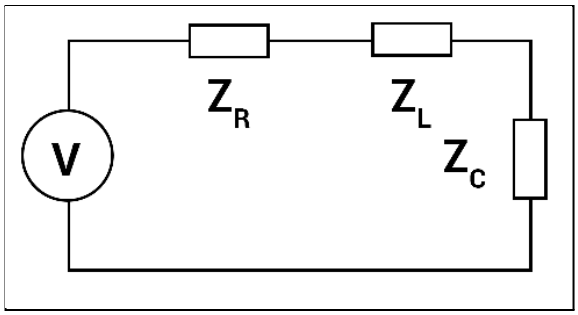


Figura 2.30 Representación de un circuito RLC en términos de las impedancias de cada componente.

La impedancia de cada componente, se define como la oposición que un elemento del circuito presenta al paso de la corriente alterna, definida de manera general por la ecuación 2.60.

$$Z = \frac{\overline{V}}{\overline{I}} \tag{2.60}$$

En el caso del resistor, se define su impedancia a partir de la ecuación 2.60 como:

$$Z_R = \frac{\overline{V_R}}{\overline{I_R}} = \frac{V_{Rm}\angle\theta_R}{I\angle\theta_R} = \frac{V_{Rm}}{I}\angle 0° = R\angle 0°$$

$$Z_R = R \tag{2.61}$$

Para el inductor:

$$Z_L = \frac{\overline{V_L}}{\overline{I_L}} = \frac{V_{Lm}\angle 0° + 90°}{I\angle 0°} = \omega L \angle 90° = j\omega L$$

$$Z_L = j\omega L \tag{2.62}$$

Para el condensador:

$$Z_C = \frac{\overline{V_C}}{\overline{I_C}} = \frac{V_{Cm}\angle 0°}{I\angle 0° + 90°} = \frac{1}{\omega C}\angle -90° = -j\frac{1}{\omega C}$$

$$Z_C = -j\frac{1}{\omega C} \tag{2.63}$$

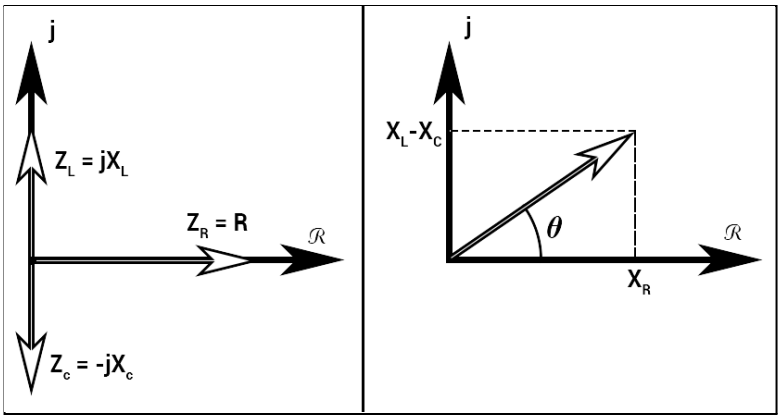
La impedancia total del circuito se define en la ecuación 2.64 como la suma de las impedancias de cada componente:

$$Z_T = Z_R + Z_L + Z_C \tag{2.64}$$

Sustituyendo las impedancias de cada componente de las ecuaciones 2.61 a 2.63 en la ecuación 2.64:

$$Z_T = R + j\omega L - j\frac{1}{\omega C} \tag{2.65}$$

El valor de las impedancias está dado por un número complejo, ya que contienen el término j. En la figura 2.31a se muestra el valor de cada impedancia en el plano complejo y en la figura 2.31b se muestra la resultante de la suma de estos valores. Se observa que la impedancia de la resistencia es un valor real puro, y la impedancia del condensador e inductor son valores imaginarios puros.



a) *b)*

Figura 2.31 Representación de impedancias del circuito RLC en el plano complejo: a) impedancias separadas; b) resultante de la suma de impedancias

Utilizando la relación trigonométrica en el triángulo del diagrama complejo, se obtienen el módulo y el ángulo del vector que representa la impedancia total del circuito.

$$\left|Z_T\right| = \sqrt{R^2 + \left(X_L - X_C\right)^2} \tag{2.66}$$

$$\theta_T = arc\,tan\left(\frac{X_L - X_C}{R}\right) \tag{2.67}$$

Debido a que los elementos del circuito RLC analizado están en serie, la corriente en cualquier punto es la misma y se define a través de la ley de Ohm como:

$$\bar{I} = \frac{\overline{V_f}}{Z_T} \tag{2.68}$$

Sustituyendo las ecuaciones 2.66 y 2.67 en la ecuación 2.68, se obtiene el fasor de la corriente del circuito:

$$\bar{I} = \frac{V_m \angle 0°}{\left|Z_T\right| \angle \theta_T} = I_m \angle -\theta_T \tag{2.69}$$

En donde:

$$I_m = \frac{V_m}{\left|Z_T\right|} \tag{2.70}$$

De manera similar a la ecuación 2.68, en la ecuación 2.71 se define de forma general el fasor de voltaje de cada componente a través de la ley de Ohm y su respectiva impedancia:

$$\overline{V} = Z\,\bar{I} \tag{2.71}$$

Finalmente, sustituyendo el valor de corriente de la ecuación 2.70 y el de la impedancia de cada componente de las ecuaciones 2.61 a 2.63 en la ecuación 2.71, se obtiene el fasor de voltaje de cada componente.

$$\overline{V_L} = Z_L\,\bar{I} = j(\omega L)\,\bar{I} \;\Rightarrow\; \overline{V_L} = \omega L\, I_m \angle -\theta_T + 90° \tag{2.72}$$

$$\overline{V_C} = Z_C\,\bar{I} = -j\left(\frac{1}{\omega C}\right)\bar{I} \;\Rightarrow\; \overline{V_C} = \frac{1}{\omega C}\, I_m \angle -\theta_T +- 90° \tag{2.73}$$

$$\overline{V_R} = Z_R\,\bar{I} = (R)\,\bar{I} \;\Rightarrow\; \overline{V_R} = R\, I_m \angle -\theta_T \tag{2.74}$$

Reescribiendo estas ecuaciones en función de la reactancia de cada componente de acuerdo a las ecuaciones 2.53, 2.57 y 2.59:

$$\overline{V_L} = X_L\, I_m \angle -\theta_T + 90° \tag{2.75}$$

$$\overline{V_C} = X_C\, I_m \angle -\theta_T - 90° \tag{2.76}$$

$$\overline{V_R} = X_R\, I_m \angle -\theta_T \tag{2.77}$$

Una vez realizado el análisis fasorial del circuito, las ecuaciones que describen la corriente del circuito y los voltajes de cada componente se pueden reescribir en función del tiempo, a partir de las ecuaciones 2.69 y 2.74 a 2.76, respectivamente, utilizando la nomenclatura utilizada para representar un fasor de la ecuación 2.48:

$$I = I_m \cos(\omega t - \theta_T) \tag{2.77}$$

$$V_L = X_L\, I_m \cos(\omega t - \theta_T + 90°) \tag{2.78}$$

$$V_C = X_C\, I_m \cos(\omega t - \theta_T - 90°) \tag{2.79}$$

$$V_R = X_R\, I_m \cos(\omega t - \theta_T) \tag{2.80}$$

Como se puede observar en las ecuaciones 2.77 a 2.80, las ecuaciones que describen el comportamiento en el tiempo de los componentes del circuito RLC obtenidas a través del análisis fasorial, describen el mismo comportamiento que las

obtenidas a través de la resolución de la ecuación diferencial, presentando el mismo ángulo de desfase entre los componentes.

2.6 Metodología de diseño

Las diversas metodologías de diseño están estructuradas para adaptarse a diferentes tipos de proyecto, buscando un enfoque particular en su desarrollo, o bien, dependiendo del producto final que se obtiene de éstos. Por lo que, se investigaron las metodologías de diseño de *software* existentes, con el fin de seleccionar una que se adecuara mejor a los objetivos planteados para este trabajo una vez analizada la práctica a desarrollar, sus componentes y sus fundamentos matemáticos. Es así que se eligió la metodología híbrida Aegis-MD [42], la cual toma los mejores elementos de metodologías ágiles de diseño de *software*: XP (*Extreme Programming,* Programación extrema), SCRUM y TDD (*Test Driven Development*, Desarrollo basado en pruebas), respecto al enfoque de desarrollo de aplicaciones móviles.

Esta metodología consiste de cinco fases: Exploración, Gestación, Construcción, Estabilización y Cierre. Dependiendo de la fase, se requieren actividades de entrada, o bien, se generan productos de salida a próximas fases de acuerdo a un ciclo iterativo, tal como se muestra en la figura 2.31.

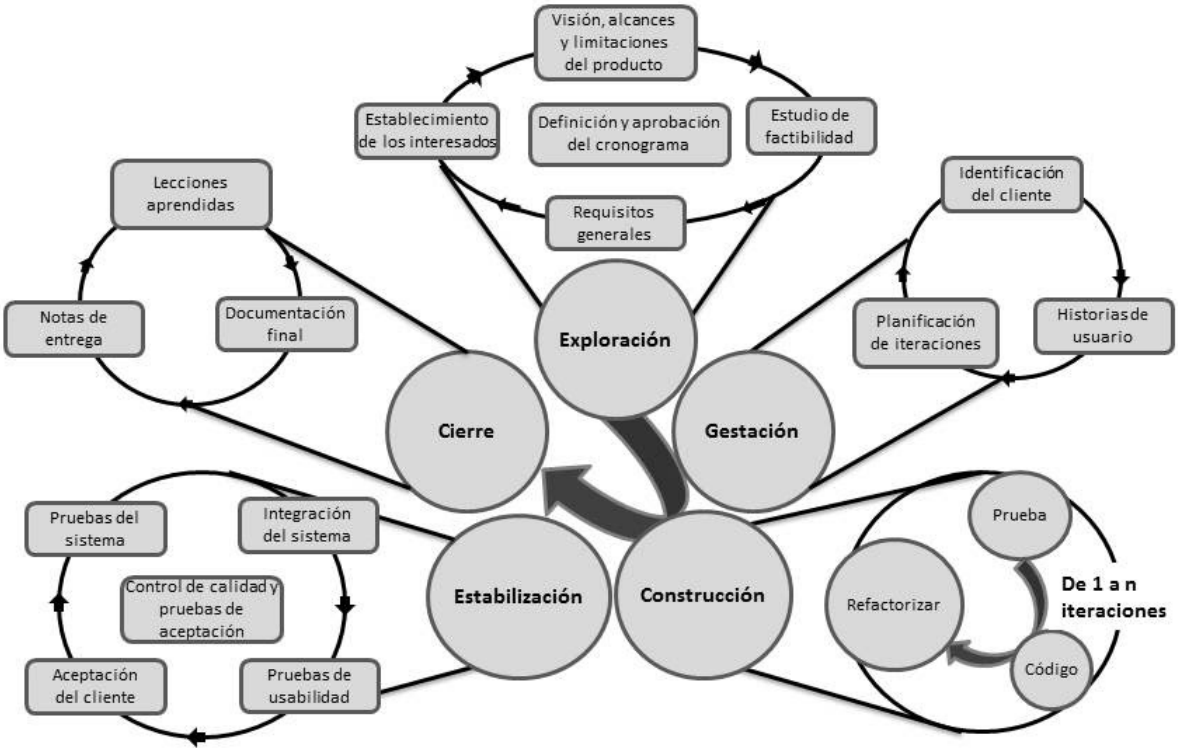


Figura 2.31 Guía metodológica AEGIS-MD [42].

La metodología comienza con el ciclo propuesto para la fase de Exploración, donde se busca realizar la planificación y establecimiento del proyecto, determinando los alcances y limitaciones del producto final.

De forma general durante esta primera fase se consideran las siguientes entradas que, posteriormente, se deberán particularizar de acuerdo a los objetivos del proyecto a desarrollar:

- Documento de requisitos generales
- Requerimientos
- Necesidades.

Continuando con la fase de Gestación, se realizan las actividades y estudios pertinentes para la identificación y caracterización de los usuarios finales del producto a desarrollar, donde las historias de usuario se identifican como aquellas que otorguen la información necesaria para los contextos en los que se utilizará la *app* a desarrollar y sus atributos.

Para esta fase se identifican las siguientes entradas:

- Identificación de usuarios

- Historias de usuario.

Durante la fase de Construcción, se deben realizar las actividades pertinentes de acuerdo al desarrollo de *software* para la *app*, cumpliendo con las necesidades y requerimientos planteados durante la fase de Exploración y conforme a las decisiones tomadas con base en la información evaluada de las historias de usuario durante la fase de Gestación.

Se considera la fase de Construcción como un ciclo iterativo incremental de tres pasos los cuales son:

1 Escribir la especificación del requisito (el objetivo de cada actividad planeada a programar en código)
2 Implementar el código para su ejecución
3 Refactorizar (reestructurar el código) para eliminar errores y hacer mejoras.

El número de iteraciones es ilimitado por lo que se puede repetir cada una de las actividades programadas hasta que todas estén completas, por lo que se realiza para cada una de ellas los tres pasos anteriores.

Para esta fase las entradas se consideran las siguientes:

- División e identificación de subsistemas correspondientes a cada necesidad identificada
- Se establecen las tareas a desarrollar por subsistema y la elección de sus herramientas para llevarla a cabo.

Posteriormente, la fase de Estabilización tiene como objetivo realizar una versión funcional integrada del sistema. De acuerdo a la fase de Construcción, las actividades se desarrollaron de manera independiente, por lo cual al finalizar esta fase, las actividades deberán funcionar correctamente por sí mismas. Durante la fase de Estabilización, todas las actividades implementadas en código deben unirse llevando a cabo pruebas de calidad y verificación de uso. Al igual que la fase de Construcción, este ciclo es iterativo hasta completar la integración total del sistema final.

Para esta fase las entradas son :

- Definición de pruebas y su forma de evaluación
- Recopilación de retroalimentación por parte del público usuario.

Para finalizar, en la fase de Cierre se ejecutará el sistema una vez revisado y con la documentación pertinente para su uso, así como la contemplación de posibles

puntos de mejora para una siguiente versión. La etapa culmina con la liberación final del sistema en su totalidad, siendo su formato un archivo ejecutable como *app* para dispositivos móviles.

Por último, se consideran las salidas de esta fase:

- Documentación sobre la *app* terminada
- Planteamiento de los puntos de mejora para futuras versiones y las conclusiones alcanzadas.

En los siguientes capítulos se describe a detalle el desarrollo de este proyecto, de acuerdo a las fases presentadas por esta metodología y la definición de las entradas y salidas, según corresponda al diseño particular del trabajo.

3 EXPLORACIÓN DE APLICACIÓN RACE: REALIDAD AUMENTADA EN CIRCUITOS ELÉCTRICOS

Dada la guía de la metodología Aegis-MD [42] elegida para el desarrollo de esta aplicación y una vez definidos los objetivos y el concepto general a cumplir, se procede a la primera iteración del ciclo a través de la fase de Exploración. En esta primera etapa se declaran los parámetros del proyecto, por lo que se consideran las siguientes entradas:

- Documento de requisitos generales: Práctica a realizar “Aplicación de las ecuaciones diferenciales en circuitos eléctricos”
- Necesidades y requerimientos
- Alcances

Cada una de estas entradas tienen como objetivo establecer los criterios a cumplir, con lo que se evalúa la obtención de un *software* considerado como producto, así como las limitantes a las que se estará atendiendo en el proceso de diseño.

3.1 Necesidades y requerimientos

Una vez analizada la problemática a tratar y definidos los objetivos a cumplir, así como los aspectos principales a realizar de acuerdo a la práctica presencial, se identifican las siguientes necesidades correspondientes al desarrollo de la *app* propuesta como solución.

Necesidades

- La *app* debe ser compatible con la mayoría de los dispositivos móviles que los alumnos dentro de la Facultad de Ingeniería utilizan de manera cotidiana
- Debe permitir la realización de las actividades de la práctica presencial de laboratorio, con tecnología de RA.

Identificadas estas necesidades, se señalan los requerimientos que permitan cumplir con las mismas.

Requerimientos

- La *app* debe estar desarrollada con una tecnología que permita ser ejecutada por el sistema operativo y *hardware* más utilizado por los estudiantes definidos como usuarios
- Para el cumplimiento de las actividades definidas por la práctica presencial de laboratorio, la *app* debe permitir tomar los datos necesarios del experimento realizado
- Los modelos virtuales de los componentes utilizados deben ser cercanos a la realidad, de tal manera que brinden al alumnado una experiencia visual similar a la de realizar una actividad con los componentes reales dentro de un laboratorio de forma presencial.

3.2 Alcances

La definición de alcances de un proyecto permite delimitar las características de su producto final, por lo que es importante que sean definidos de acuerdo a los objetivos establecidos de forma particular para este trabajo.

Se propone que el producto final establecido sea una *app* en la que los alumnos que cursen la materia de Ecuaciones Diferenciales, de la División de Ciencias Básicas en la Facultad de Ingeniería, puedan realizar la práctica “Aplicación de las ecuaciones diferenciales en circuitos eléctricos”. Tras la realización de esta práctica, el alumnado podrá obtener datos necesarios para el correcto entendimiento de los fenómenos que observa, tales como la gráfica de la respuesta de voltaje de los componentes de un circuito RLC, así como valores representativos de la misma. Será capaz de tomar lectura de estos datos de una manera similar a como lo haría en un osciloscopio real y podrá manipular los componentes necesarios para el armado del circuito.

Esta *app* será implementada como un elemento adicional a la práctica que se aplica presencialmente y será distribuida al alumnado únicamente a través de la figura docente interesada en adoptar su aplicación.

En este punto de diseño de acuerdo a los objetivos, necesidades y requerimientos planteados, se le denomina a la *app* a desarrollar con el nombre de RACE, atendiendo a las siglas: Realidad Aumentada en Circuitos Eléctricos. Este nombre se decidió de acuerdo a la asociación de la temática a la que pertenece la práctica a realizar y la tecnología de realidad utilizada para su implementación.

4 GESTACIÓN

De acuerdo a la metodología de diseño, en esta etapa se particularizan las siguientes entradas:

- Estudios descriptivos para selección de *hardware* objetivo
- Evaluación de herramientas de *software* para el desarrollo de la *app*.

Se considera para la identificación de usuarios, la definición de una muestra descriptiva de alumnos para quienes se está dirigiendo el desarrollo de este trabajo, de acuerdo a los alcances previamente establecidos, por lo cual se decide realizar un estudio descriptivo tipo encuesta (apéndice A), que arroje los datos necesarios para seleccionar el *hardware* más utilizado por los estudiantes considerados como futuros usuarios. Así mismo, se considera posteriormente la evaluación de herramientas existentes para construir el ambiente de RA dentro del aplicativo móvil, de tal manera que sea soportado por el *hardware* objetivo así como las preferencias de uso que expresen los alumnos. En los siguientes apartados se describen en detalle los resultados obtenidos para estas entradas.

4.1 Estudio descriptivo para selección de *hardware* objetivo

Las aplicaciones de realidad aumentada se encuentran limitadas por sus diferentes componentes, como lo es la capacidad en *hardware* del que se disponga, así como la compatibilidad y versatilidad que permita el *software* seleccionado para su desarrollo. De acuerdo con el objetivo de este trabajo, es necesario considerar y evaluar las limitaciones de los componentes de acuerdo con la opinión personal de una muestra representativa del universo de alumnos pertenecientes a la Facultad de Ingeniería de la UNAM, a quienes se les considera como el público usuario de la

app, ya que son los estudiantes que se encuentran cursando o cursarán la materia de Ecuaciones Diferenciales y podrán realizar la práctica: “Aplicación de las ecuaciones diferenciales en circuitos eléctricos”.

A partir de la definición de muestras para un estudio simple [43, Cap. 1], se consideró un estudio probabilístico por conglomerado, con el fin de generalizar las características del dispositivo más utilizado por parte del alumnado, así como su capacidad de *software*, definido como el grupo de conglomerado a los estudiantes propuestos por parte del profesorado de la DCB. Se trata de un grupo heterogéneo, el cual está integrado por alumnos tanto de Ciencias Básicas como de las otras divisiones.

Para determinar el tamaño de la muestra, se consideró primeramente como población a la totalidad de estudiantes inscritos en la Facultad de Ingeniería, 12,908 [44]; con ello se define el número de personas para la muestra utilizando la siguiente fórmula [45]:

$$n = \frac{z^2 Npq}{e^2(N-1)+z^2 pq} \tag{4.1}$$

donde:

z = 1.96, es el nivel de confianza para el 95%,

p = 0.5 y q = 0.5, dado que no se conoce la distribución del fenómeno,

e = 0.1, es el error esperado para el 10%,

N = 12,908, es el tamaño de la población

con lo que se obtuvo como resultado una muestra de n = 95 personas a entrevistar.

Para comprobar que los datos obtenidos tienen validez para el grupo objetivo, en el apéndice B se muestran gráficas comparativas en las que se observa que, entre las personas encuestadas, no hay una diferencia significativa en la distribución de las respuestas dadas por alumnos pertenecientes a la DCB en comparación con los que pertenecen a otras divisiones.

En la figura 4.1 se muestra la distribución de respuestas obtenidas sobre el sistema operativo predominante, tratándose de Android con un 76.8% de preferencia y

teniendo a su versión 8 como la más utilizada, reflejándose en la figura 4.2 un 27.37% de dominancia, que incluye todas sus actualizaciones.

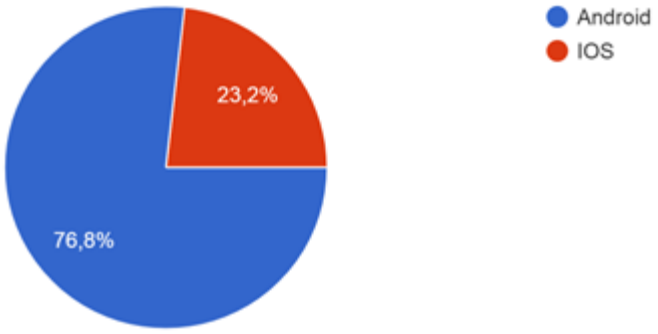


Figura 4.1 Gráfico de respuestas para sistema operativo utilizado.

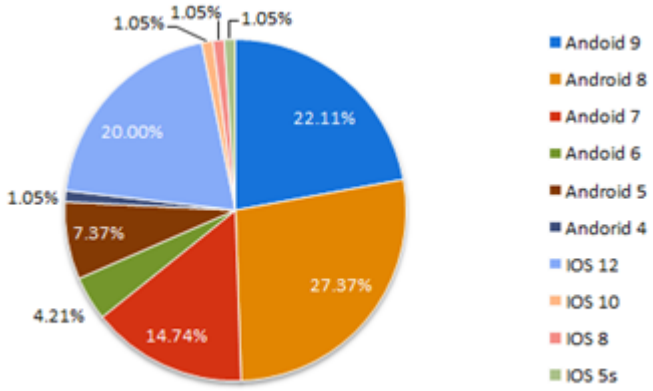


Figura 4.2 Gráfico de respuestas para versión de sistema operativo utilizado.

De acuerdo con estos resultados, se definió como sistema operativo de desarrollo para la aplicación el sistema Android, marcando como versión objetivo su actualización 8 y que es compatible con versiones superiores y previas hasta la 5, con la finalidad de englobar la distribución obtenida por la encuesta.

De manera similar, dentro del mismo estudio se señala una preferencia por parte del alumnado por una *app* de realidad mixta, justificando, entre otras razones, por tener una “Mayor interactividad”. Dicha distribución se muestra en la figura 4.3.

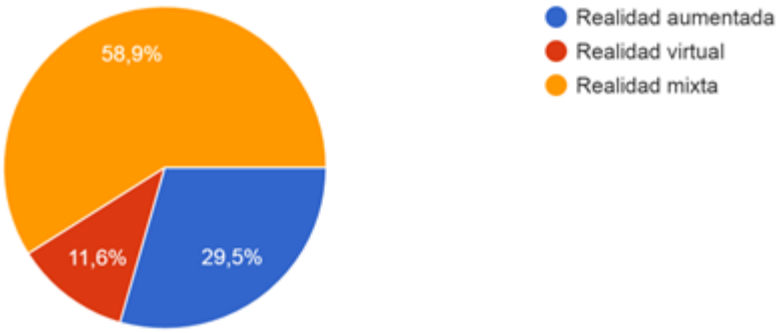


Figura 4.3 Gráfico de respuestas para preferencia en tecnología de realidad.

Sin embargo, se observa de acuerdo a los datos expresados en la figura 4.4, una mayor aceptación, con un 37.9%, por el uso de las herramientas sugeridas en RA.

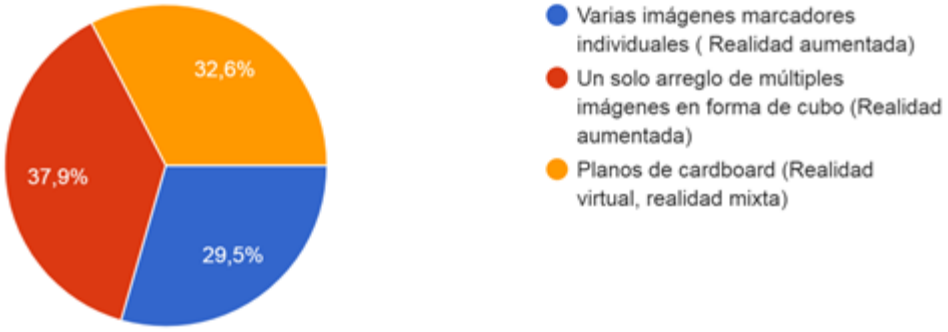


Figura 4.4 Gráfico de respuestas para preferencia por herramientas para utilizar en las distintas tecnologías de realidad.

Debido a que las justificaciones para la selección de realidad aumentada son, “su practicidad” y “familiaridad debido a juegos y otras aplicaciones”, se estableció la elección de esta tecnología de realidad como la más adecuada para cumplir el objetivo de este trabajo, ya que requiere de menor empleo de *hardware* con respecto a las tecnologías de realidad virtual y realidad mixta, además de que las alumnas y los alumnos expresaron contar con algún tipo de experiencia previa en el manejo de la realidad aumentada.

4.2 Evaluación de herramientas de *software* para realidad aumentada

Una vez seleccionado el sistema operativo objetivo, la siguiente consideración a revisar fue la selección del *software* para el desarrollo de la aplicación. Para ello, se evaluaron las soluciones que actualmente son nombradas dentro del ámbito para desarrolladores y anteriormente mencionadas dentro del estado de la técnica, que permiten una guía y facilidad para la construcción de aplicativos móviles en realidad aumentada. Para exponer las opciones y su selección, se expresan las ventajas y desventajas consideradas para cada herramienta en la tabla 4.1.

Tabla 4.1 Ventajas y desventajas de las distintas herramientas de realidad aumentada.

Herramienta para RA	Ventajas	Desventajas
ARCore	-Paquete completo para desarrollo de *apps* con RA -Mapeo de terreno para interacción con elementos escaneados en tiempo real -Fluidez en movimientos de elementos simulados -Reducción de interferencia debido a variantes de luz -No ocupa herramientas adicionales como marcadores	-Selección limitada de dispositivos que soportan el paquete -Compatible únicamente para algunos dispositivos con sistema Android 8 y superiores

vuforia™	-Soportado por cualquier dispositivo con cámara -Gama de posibles marcadores (detección de plano, objetos 3D, imágenes, marcador seleccionado por usuario) -Cuenta con paquetes predeterminados de acuerdo al tipo de *app* a desarrollar	-Presenta fallas por cambios de luminosidad -Requiere de marcadores seleccionados
OpenGL OpenCV	-Cuenta con mayor libertad de desarrollo al no utilizar paquetes predeterminados -Puede modificarse la visión artificial utilizada de acuerdo a la programación que se utilice	-Es necesario conocimiento en visión artificial y reconocimiento de imágenes para poder intervenir en el desarrollo de una *app* -Su compatibilidad con diversas plataformas está delimitada de acuerdo al requerimiento de procesador utilizado para la visión artificial.

Después de analizar la tabla 4.1, se puede apreciar que la herramienta con mayores ventajas es el paquete de desarrollo ARCore; sin embargo, presenta una desventaja significativa, debido a su limitada compatibilidad con los dispositivos. Esta desventaja supone una exclusión del objetivo de este trabajo, el cual es permitir la accesibilidad a la aplicación por parte del alumnado de la Facultad de Ingeniería, quienes reportan, a través de la encuesta realizada, que no disponen en su mayoría de dispositivos adecuados para utilizar esta herramienta. Por esta razón, se seleccionó el paquete de desarrollo Vuforia, ya que permite mayor facilidad para el desarrollo de la tecnología, con una guía adecuada y compatibilidad con una amplia gama de dispositivos.

4.3 Motor gráfico Unity para ambiente virtual

Una vez que se ha seleccionado el *software* de desarrollo para la tecnología de RA, es necesario definir el *software* que se desempeñe como el motor gráfico para el desarrollo del ambiente virtual dentro de la aplicación, y que a su vez sea compatible con la herramienta de RA previamente elegida. De acuerdo a un estudio realizado en 2019 por la compañía de desarrollo de software Innovecs *[46]*, la plataforma de desarrollo Unity posee una cuota de mercado del desarrollo de videojuegos y visualización 3D del 45%, con sus competidores más cercanos (Unreal Engine, GameMaker y CryEngine) que llegan apenas al 17% en conjunto (figura 4.5). Este elevado índice se debe al alto número de plataformas para las que se puede diseñar y exportar desde Unity, incluyendo plataformas móviles, entre otras.

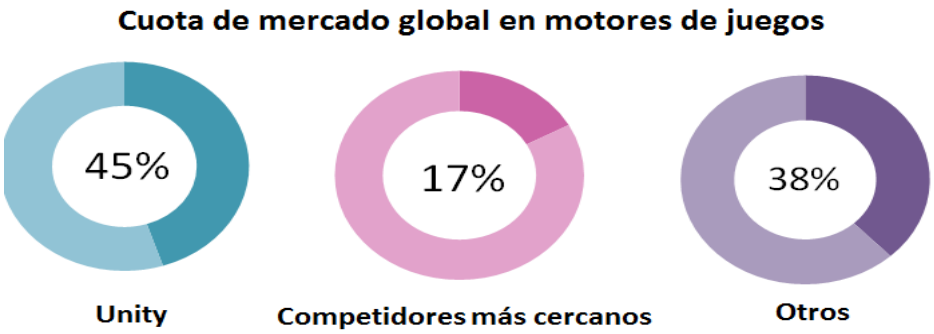


Figura 4.5 Cuota de mercado (Market Share) de desarrollo de videojuegos [46].

Por esta razón, Unity ha sido ampliamente utilizado para el desarrollo de *apps* con distintos fines, siendo incluso empleado en los trabajos previamente citados en el estado de la técnica dentro de este trabajo, por lo que se seleccionó como el motor gráfico a utilizar, considerando como sus mayores ventajas los siguientes puntos:

- Permite el diseño y *renderización* de objetos virtuales 3D, así como sus animaciones, incluyendo interacciones con simulaciones físicas y materiales
- Cuenta con paquetería para la compatibilidad con otras plataformas de desarrollo de tecnologías, incluyendo Vuforia para RA
- Su versatilidad de desarrollo permite la generación de proyectos ejecutables por diversas plataformas, como lo son dispositivos móviles Android y IOS
- Sus herramientas de desarrollo en programación permite la generación de códigos, tanto en Java como C#.

5 CONSTRUCCIÓN

Durante esta etapa, el desarrollo se llevó a cabo por medio de un ciclo iterativo que consta de tomar la especificación de un requerimiento, implementarlo en la *app* a través de un producto final, ya sea código o elemento virtual y finalmente llevar a cabo una prueba de funcionalidad para valorar esta implementación.

Se establecieron como entradas a esta fase las siguientes:

- Los requisitos del proyecto correctamente identificados (capítulo 3)
- Descripción del usuario a través de estudios de identificación (capítulo 4).

De acuerdo con la metodología de diseño, para poder cumplir adecuadamente con las entradas de esta fase, se consideró la división de subsistemas que conforman el trabajo de desarrollo, por lo que se establecieron los siguientes sistemas:

- Virtualización, generación de texturas y modelado de componentes virtuales
- Mecánica de interacción por medio de marcadores
- Despliegue de osciloscopio en tiempo real
- Interfaz de usuario.

En los siguientes apartados se describen las tareas que comprenden cada sistema y sus respectivas implementaciones.

5.1 Virtualización y modelado de componentes

Como se planteó en el capítulo 3, uno de los requerimientos de la *app* se describió de la siguiente manera:

Los modelos virtuales de los componentes utilizados deben ser cercanos a la realidad, de tal manera que brinden al usuario una experiencia visual similar a la de realizar una actividad con los componentes reales.

Para cumplir con este requerimiento, el modelado de los elementos virtuales se realizó priorizando el parecido de estos con los componentes reales. Como se mencionó en la tabla 2.3, los elementos a virtualizar son:

- Osciloscopio
- Módulo de protoboard y generador de funciones
- Inductor
- Resistor
- Condensador.

Los modelos tridimensionales utilizados por plataformas de desarrollo como Unity están constituidos por puntos, los cuales se unen a través de líneas formando los planos que representan las caras del modelo final. Algunos de los *software* de diseño 3D que tienen esta construcción de sus modelos son Blender, 3DS Max, OpenSCAD, entre otros, los cuales generan archivos con extensiones *.obj o *.fbx. Se optó por utilizar el *software* de licencia abierta Blender para el diseño de los modelos virtuales de la *app*, debido a que sus modelos tridimensionales cumplen con las características ya mencionadas, además de permitir hacer modificaciones y visualizarlas al instante en Unity.

Para el diseño de los componentes virtuales se tomaron como referencia imágenes de componentes electrónicos reales, con el fin de que los modelos virtualizados mantuvieran proporciones y detalles similares. A continuación, se describe de forma representativa el proceso seguido para realizar el modelo del resistor.

Se inició la virtualización colocando la imagen de referencia del componente real en el fondo del espacio de trabajo. La forma del modelo se estableció a partir de un cilindro que conforma el cuerpo de la figura a modelar (figura 5.1a), donde se definen las dimensiones y el número de caras del prisma.

Una vez colocado el cilindro, se hicieron subdivisiones a lo largo de su eje horizontal (figura 5.1b), de tal manera que los círculos transversales creados fueron escalados para crear las protuberancias del resistor y darle su forma final (figura 5.1c).

Para las terminales se utilizó un proceso similar; insertando, modificando y subdividiendo cilindros independientes de cada lado, para posteriormente unirlos con el cuerpo en un solo elemento y, finalmente, realizar una subdivisión de las

líneas del modelo, dando una apariencia final con menor cantidad de bordes sin aumentar excesivamente el número de puntos que constituyen el modelo (figura 5.1d).

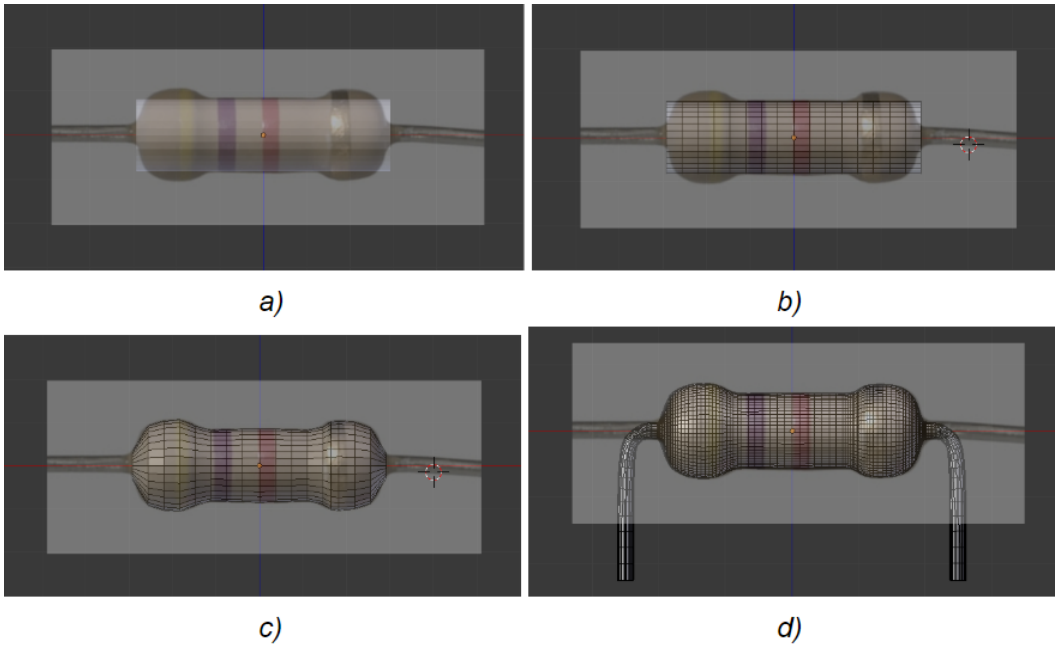


Figura 5.1 Proceso para la creación del modelo tridimensional de un resistor: a) creación y dimensionamiento del cilindro inicial; b) marcado de subdivisiones en la figura; c) escalamiento de las subdivisiones; d) insertado de las terminales en la forma final.

Se siguió un procedimiento similar al anteriormente descrito, para la creación de los modelos de condensador, inductor y osciloscopio, siendo sus formas finales las mostradas en la en la figura 5.2.

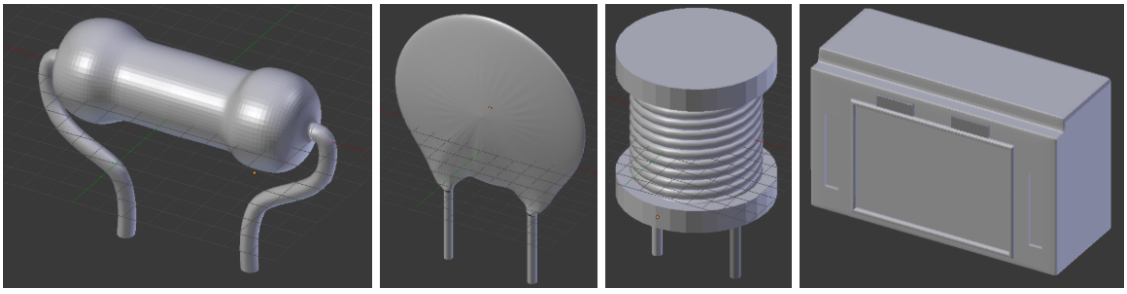
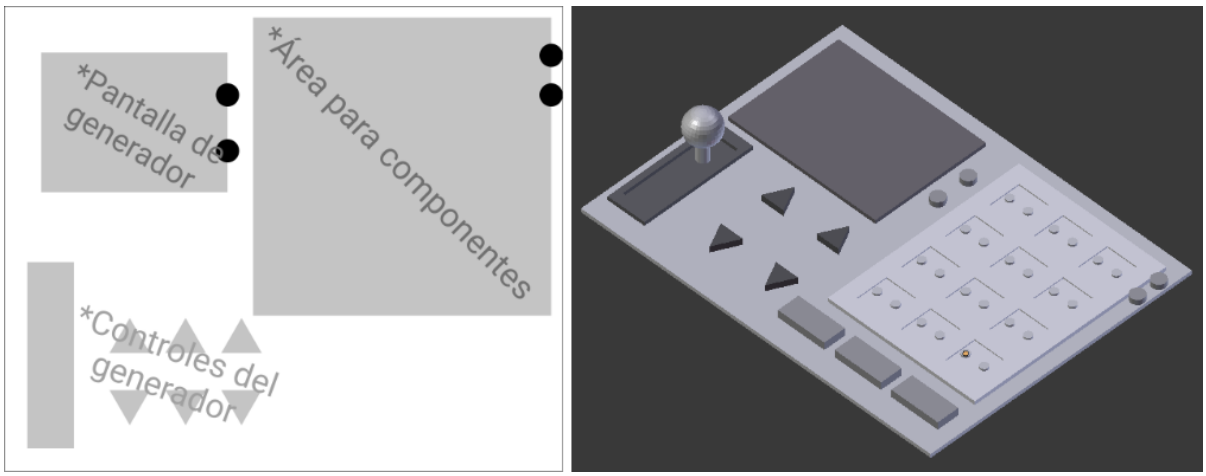


Figura 5.2 Modelos virtuales en 3D de los componentes: a) modelo de un resistor; b) modelo de un condensador; c) modelo de un inductor; d) modelo de un osciloscopio.

Para el caso del módulo de protoboard y generador, se comenzó por un boceto de la base en el que se aprecia de manera general la distribución de sus elementos (figura 5.3a). Se contempló en su estructura un arreglo de 5x5 espacios del lado derecho para colocar componentes y donde se realizará el alambrado del circuito, mientras del lado izquierdo una pantalla donde será visible la representación de la señal que el generador está emitiendo.

Se incluyeron controles en la parte inferior que permiten la modificación de las características de la señal, siendo los botones con forma de flecha los destinados a variar el valor de amplitud y frecuencia, mientras que la palanca habilitará la selección del tipo de onda. Posteriormente, se creó el modelo virtual del módulo propuesto en boceto (figura 5.3b), siguiendo un procedimiento similar al utilizado en los componentes anteriores.



a) *b)*

Figura 5.3 Módulo de protoboard y generador: a) boceto del módulo; b) modelo virtualizado del módulo.

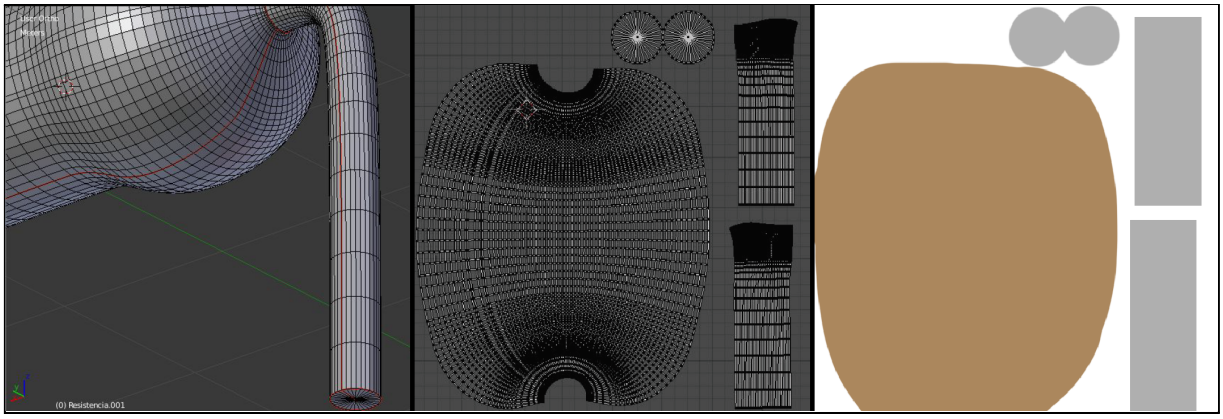
Para un mayor detalle del proceso de diseño de los componentes y de las iteraciones realizadas para su elaboración, consultar el apéndice D.

5.2 Generación de texturas virtuales

Con el fin de otorgar al modelo virtual una apariencia similar a la del componente real, se realizó el proceso de *renderizado*, en el cual utilizando imágenes que se superponen en la superficie de las caras del modelo, se genera una apariencia externa con detalles conocida como textura. Para colocar texturas de manera

correcta se realizó un proceso conocido como mapeo UV, el cual consiste en tomar un modelo 3D y dividirlo a través de líneas en conjuntos de polígonos, los cuales se colocan en una imagen plana (mapa de texturas) para posteriormente crear la imagen (textura) con base en la disposición en la que se hayan colocado los polígonos.

Para llevar a cabo este proceso en Blender, se seleccionaron las líneas de corte para dividir el modelo y se marcaron como *costuras* (término utilizado por Blender). En el caso del resistor, se seleccionó la circunferencia que marca la posición de sus terminales y el final de las mismas, así como una línea a lo largo del eje transversal del cuerpo y de sus terminales en las partes inferior y trasera respectivamente. Posteriormente, en un *software* de edición de imágenes se crearon las texturas del modelo, con base en este mapa. Las líneas de costura, el mapa de texturas y la textura final se muestran en la figura 5.4.



a) *b)* *c)*

Figura 5.4 Modelo y texturizado de resistor: a) modelo de un resistor con líneas de costura; b) mapa de texturas del resistor; c) textura utilizada para el resistor.

Siguiendo con este proceso se obtuvo el mapa de texturas de cada modelo y, con base en estas, se crearon sus respectivas texturas. En el apéndice E se puede observar la secuencia de creación de texturas para cada componente.

Finalmente, se importaron en Unity los modelos tridimensionales de los componentes y se les aplicaron sus texturas correspondientes. La versión final de los modelos se puede apreciar en la figura 5.5.

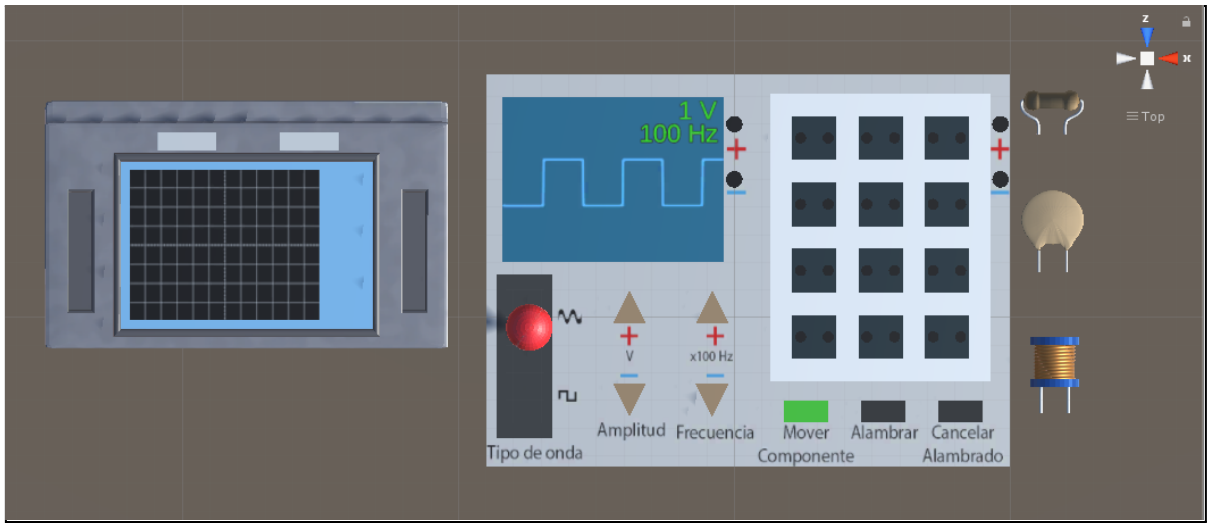


Figura 5.5 Modelos tridimensionales de los componentes con texturas desplegados en Unity.

5.3 Mecánica de interacción por medio de marcadores

En el capítulo 4.2 se compararon las diferentes herramientas de *software* que permiten el desarrollo de RA; en ese mismo apartado se decidió utilizar el paquete de desarrollo Vuforia, debido a su notable ventaja de ser compatible con una amplia gama de dispositivos móviles.

La paquetería de Vuforia permite tanto el uso de tecnología RA, basada en marcadores, como RA sin marcadores; sin embargo de acuerdo a su documentación, las herramientas de RA sin el uso de marcadores están disponibles para ciertos dispositivos que cumplen los requerimientos [47], lo que implica que de utilizar este sistema de seguimiento se perdería la principal ventaja de compatibilidad que ofrece Vuforia. Por esta razón se decidió utilizar como sistema de seguimiento la tecnología RA basada en marcadores.

Dentro del entorno de desarrollo Unity, el SDK de Vuforia permite utilizar como marcadores para RA distintas variantes, entre ellas están:

- *Image Target* (Imagen objetivo)
- *Multi Target* (Múltiples objetivos)
- *Cylinder Target* (Objetivo cilíndrico)
- *User Defined Target* (Objetivo definido por el usuario).

En el caso de *Image Target*, se trata de un único marcador plano definido como una imagen con los suficientes puntos característicos para que el sistema de seguimiento de Vuforia pueda reconocerla como marcador. Sin embargo, en el caso particular de esta opción, la imagen debe estar predefinida en la biblioteca al momento de programar la *app*, por lo que se considera que esta imagen se debe proporcionar a los usuarios para que puedan imprimirla y hacer uso de este tipo de marcador.

Por otra parte la herramienta de *Multi Target*, es un arreglo geométrico comúnmente en forma de cubo, cuyas caras son individualmente distintas *Image Target*. Este tipo de arreglo permite que el sistema de seguimiento sea más robusto al momento de manipular el marcador, ya que se reduce la oclusión ocasionada por variantes de luz o la perspectiva de la cámara, debido a que posee varias referencias en vez de una sola imagen como es el caso de *Image Target*. La variante de esta herramienta es *Cylinder Target,* donde el arreglo geométrico se trata de un cilindro, para ofrecer la posibilidad de utilizar esta geometría en lugar de la disposición en forma de cubo.

Adicionalmente, la herramienta *User Defined Target* permite que el usuario de la *app* defina en tiempo real la imagen plana que utilizará como marcador a través de la cámara de su dispositivo móvil. En este caso, la programación del reconocimiento se basa en realizar el despliegue de los elementos virtuales cuando el *software* de Vuforia detecta una imagen en la cámara con los suficientes puntos característicos.

Tal como se describió en los apartados 2.3.1 y 2.3.2, la herramienta de marcador permite el despliegue, manipulación de orientación y de posición de los elementos virtuales en RA; sin embargo, debido a los objetivos de este trabajo, la *app* a desarrollar cuenta con múltiples componentes con los cuales el usuario debe poder interactuar y manipular para realizar las actividades del armado del circuito. Por esta razón, el emplear un único marcador tanto para despliegue como para interacción, supondrá distintas dificultades para el usuario, ya que no podrá manipular aisladamente los componentes del circuito. Por otra parte, utilizar un marcador distinto para cada componente resultaría en un aumento de las herramientas físicas a controlar y supondría igualmente una interacción problemática para el usuario.

Para solventar esta dificultad, se decidió hacer uso de las distintas herramientas de marcadores que ofrece Vuforia, con tal de definir el empleo de dos marcadores; uno para el despliegue y posicionamiento inicial de los elementos virtuales, y el segundo

como marcador de interacción para que los usuarios puedan manipular los componentes virtuales de forma individual.

En la tabla 5.1 se describen las especificaciones de cada marcador de acuerdo a su función.

Tabla 5.1 Especificaciones de marcadores e interacción del usuario.

Función del marcador	**Especificaciones de interacción**
Marcador de despliegue	● Marcador único plano ● Realizará el despliegue de los elementos virtuales en su punto y posición de origen ● Delimitará la posición espacial de los elementos virtuales en el entorno real ● Modificará la posición espacial de todos los componentes en conjunto dentro del entorno real
Marcador de interacción	● Marcador geométrico ● Permitirá la interacción aislada de un componente virtual una vez que todos hayan sido desplegados ● El cambio de posición afectará únicamente a los componentes virtuales de resistor, inductor y condensador individualmente ● Actuará como indicador para manipular botones, palanca y alambrado dentro del modelo virtual de protoboard ● Permitirá la selección del compontente virtual de osciloscopio, para poder observar la señal resultante del circuito alambrado

Tal como se expresa en la tabla 5.1, el marcador de despliegue se considera un marcador plano, ya que deberá delimitar el punto de origen dentro del espacio real para efectuar el despliegue de los elementos virtuales, no obstante se considera utilizar la herramienta de marcador *User Defined Target,* debido a que esta permite que la selección de la imagen a utilizar como marcador sea en tiempo real y no una que deba ser programada con anterioridad. De esta manera se le otorga al usuario la libertad de seleccionar su área de trabajo.

Por otra parte, se considera para el marcador de interacción una disposición geométrica ya que al tratarse del elemento físico que permite la manipulación de los elementos virtuales, se busca el sistema de seguimiento óptimo y de menor riesgo que ofrece esta herramienta de marcadores, con fin de evitar los problemas que puedan ocasionar las variantes de luz o enfoque de cámara al estar en constante movimiento.

En los siguientes apartados se describe a detalle la programación y construcción de estos marcadores.

5.3.1 Programación de un marcador definido por el usuario

El procedimiento para establecer este marcador consistió en delimitar inicialmente el funcionamiento a programar de la herramienta seleccionada *User Defined Target*, de acuerdo a la señal de entrada que debe registrar, las condiciones para establecer como marcador la imagen captada por la cámara, definir la función de activación del marcador y, consecuentemente, obtener como salida el despliegue de los elementos virtuales. En la figura 5.6, se muestra el diagrama de bloques que describe las interacciones del procedimiento programado.

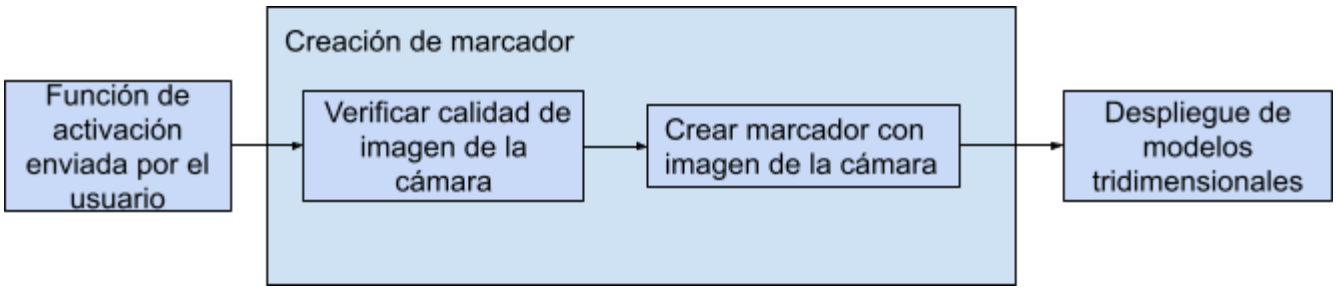
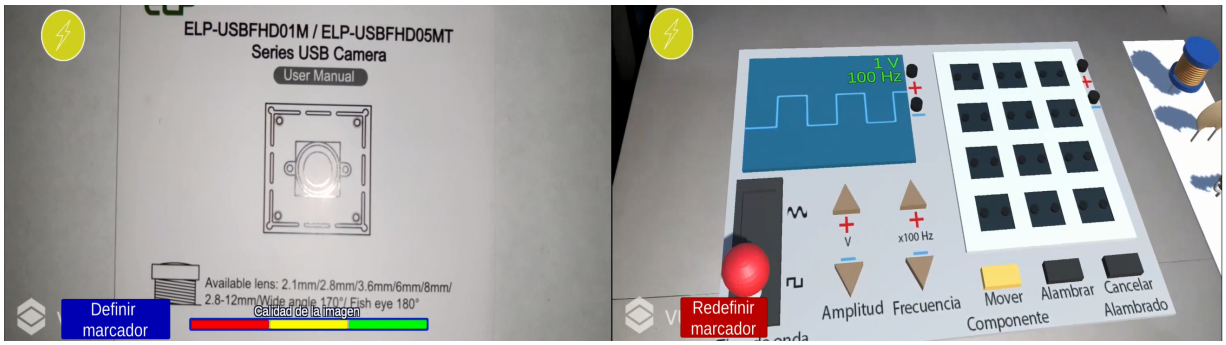


Figura 5.6 Diagrama de bloques del proceso para la creación del marcador plano de despliegue.

Dentro de la programación, Vuforia permite seleccionar de forma predeterminada la activación de la herramienta *User Defined Target,* habilitando automáticamente la cámara del dispositivo que se encuentre ejecutando la *app*, por lo tanto el dispositivo comenzará a buscar imágenes con puntos característicos con tal de identificar un marcador. Con el fin de que la imagen a seleccionar pueda ser identificada por el usuario de acuerdo a su calidad, se programó dentro de la interfaz visible un indicador de semáforo, que delimite la calidad de la imagen de acuerdo a la cantidad de puntos característicos registrados. El color verde del semáforo es el que está asociado a mayor cantidad de puntos característicos,

significando una alta calidad de imagen, mientras que el color rojo corresponde a una baja cantidad de puntos característicos siendo baja la calidad de la imagen.

De esta forma, al hacer uso del indicador de semáforo, el usuario al momento de utilizar la *app* puede seleccionar una imagen que se registre en el color verde, indicando que la imagen es óptima para utilizarse como marcador. Una vez que la imagen se encuentre dentro de la calidad óptima, la función de activación para definirla como marcador es ejecutada al momento que el usuario pulse el botón programado para este procedimiento. De esta manera, en el acto que la cámara registre una imagen indicada en color verde y el usuario presione el botón del activador, se desplegarán los elementos virtuales teniendo como resultado la ejecución de la tecnología de RA por medio del marcador seleccionado, tal como se muestra en la figura 5.7.



a) *b)*

Figura 5.7 Funcionamiento del marcador de despliegue: a) identificación de imagen a utilizar como marcador; b) despliegue de objetos 3D sobre el marcador definido.

La implementación en código del marcador plano de despliegue se puede consultar en el apéndice F. La plataforma Unity usa de manera predeterminada el lenguaje de programación C#, por lo que todo el código fuente de la *app* se realizó con este lenguaje.

5.3.2 Diseño y construcción de un marcador geométrico

En el proceso iterativo de diseño del marcador de interacción, se optó por una disposición cilíndrica debido a que la forma permite la semejanza de un dedal, lo que otorga al usuario la posibilidad de manipular el marcador de una forma ergonómica, colocándolo en el dedo índice de su mano diestra.

Para poder definir las dimensiones se determinó primeramente el promedio de la longitud del dedo índice de la población latinoamericana, tanto de mujeres y hombres entre los 19 a 24 años. Esta magnitud fue obtenida a partir de los valores expresados en el estudio de R. Avila Chaurand "*Dimensiones antropométricas de la población latinoamericana : México, Cuba, Colombia, Chile"* [48], del que se establece una longitud promedio de 7 cm tanto para hombres como para mujeres; sin embargo, para asegurar la movilidad del dedo y buscando la manipulación ergonómica, se concluyó que la longitud del marcador no cubra más que las dos primeras falanges del dedo, con tal de permitir un movimiento con menor limitación. Por lo tanto se establece como longitud final de 4.5 cm para la altura del cilindro usado como marcador.

Para mantener los parámetros de ergonomía para la manipulación del marcador cilíndrico, se determinó el diámetro de las caras superior e inferior a partir de las medidas estandarizadas de anillos [49], de tal forma que el diámetro máximo fuera de 2 cm para ambas caras, siendo suficiente el margen para mantener concéntricamente los diferentes tamaños de ajustes de acuerdo al estándar de anillos mencionado, con el fin de proporcionar un acoplamiento al dedo índice del usuario dentro del patrón de recorte a diseñar.

Una vez definidas las medidas del marcador, se diseñaron los patrones de imagen para la cara superior y lateral. Estas imágenes fueron creadas a partir de un proceso iterativo de prueba y error, con objeto de incluir una variedad de figuras y formas con la mayor cantidad de puntos característicos y de esta manera facilitar su detección por parte del sistema de seguimiento de Vuforia. La agilidad de reconocimiento de estas imágenes fue evaluada a través del portal de desarrollo en línea de Vuforia [50]. En la figura 5.8 se puede observar el resultado final de la evaluación de las imágenes, arrojando para ambas una calificación de 5 estrellas, la más alta que brinda esta herramienta; se observan también los puntos característicos reconocidos en cada imagen resaltados en color amarillo.

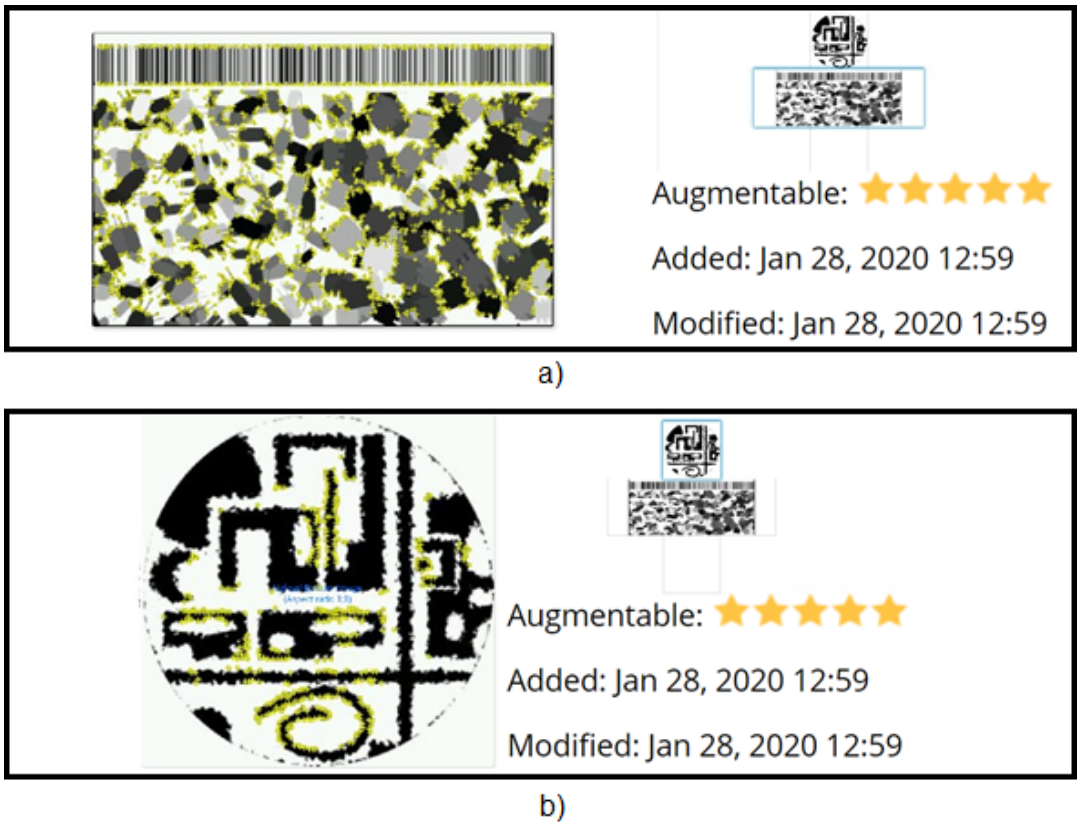
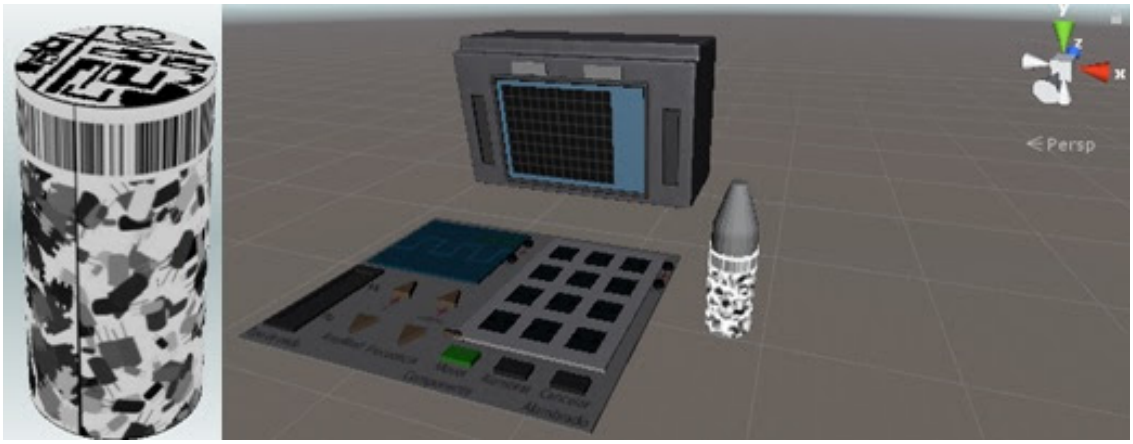


Figura 5.8 Evaluación de imágenes del marcador cilíndrico en el portal de desarrollo de Vuforia: a) cara lateral; b) cara superior.

Asimismo, dentro del portal de desarrollo de Vuforia, se establecieron las dimensiones del marcador geométrico con la forma cilíndrica y las imágenes diseñadas. Una vez que la figura fue definida en este portal, se logró que fuera importada dentro del ambiente de desarrollo de Unity manteniendo la relación de dimensiones, con tal que no hubiera interferencia en el escalamiento de los componentes virtuales desplegados en RA y el uso del marcador físico en el entorno real. En la figura 5.9, se observa el modelo virtual del marcador cilíndrico dentro del portal de Vuforia y su respectivo escalamiento dentro del entorno de Unity.



a) *b)*

Figura 5.9 Modelo virtual del marcador de interacción: a) modelo dentro del portal de desarrollo de Vuforia; b) escalamiento en el entorno de Unity.

Para permitir la interacción del marcador físico con los elementos virtuales, fue necesario asociarle un componente virtual en RA, de esta forma, al momento que la cámara del dispositivo móvil reconoce el marcador, despliega sobre el mismo la punta virtual que se muestra en la figura 5.10. Esta punta es un modelo creado en el entorno de Unity que al entrar en colisión con otros componentes virtuales puede tener interacción con ellos, y al estar asociada con el marcador cilíndrico, brinda la posibilidad de manipular los componentes necesarios. En el apartado 5.3.3, se explica con mayor detalle la mecánica de interacción entre estos elementos .

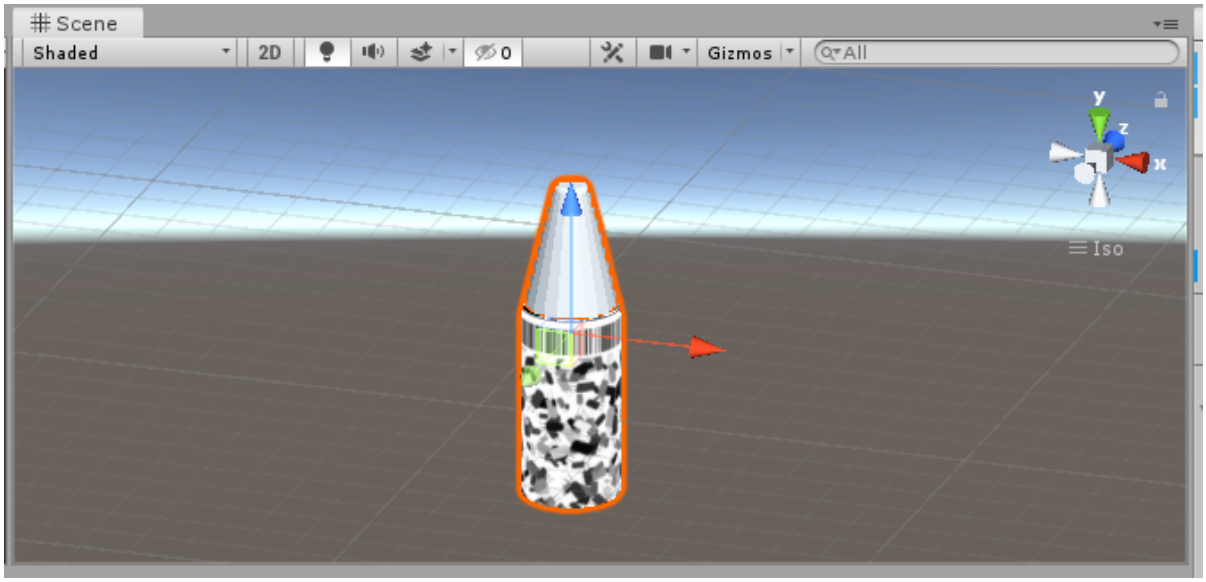


Figura 5.10 Modelo de marcador de interacción con punta virtual en el entorno de Unity.

Finalmente, a partir de la disposición plana de las caras del modelo del marcador, se creó el patrón de recorte tal como se muestra en la figura 5.11. Este patrón se diseñó con pestañas que permiten ensamblar de manera intuitiva el cilindro; en la cara inferior se colocaron las marcas de recorte con los diámetros concéntricos correspondientes a las medidas de anillos, con la finalidad de que el usuario elija recortar hasta la marca de su preferencia y pueda manipular cómodamente el marcador.

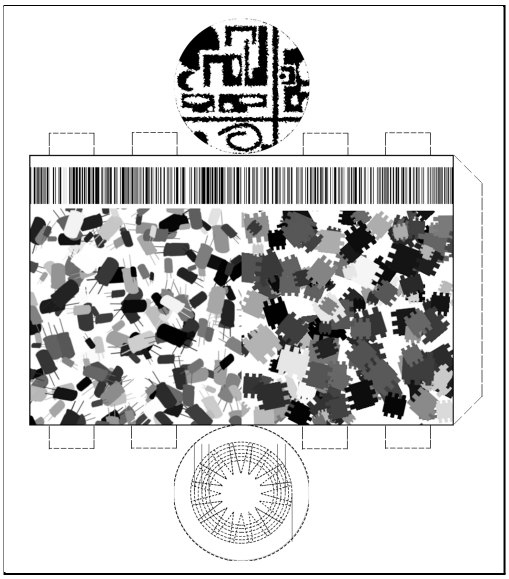


Figura 5.11 Patrón de recorte para marcador de interacción.

Las distintas iteraciones en la modificación de las imágenes dentro de las caras del cilindro se muestran en el apéndice G.

5.3.3 Despliegue e interacción con los elementos virtuales

Una vez que el usuario define el marcador plano de despliegue, los componentes virtuales se visualizan con la tecnología RA en una disposición que permite la manipulación de estos, tal como se muestra en la figura 5.12. De forma paralela al marcador de despliegue, se coloca el módulo de protoboard y generador, detrás de este se encuentra el osciloscopio, y del lado derecho se creó un espacio destinado para la selección de los componentes electrónicos antes de que sean colocados en el módulo de protoboard.

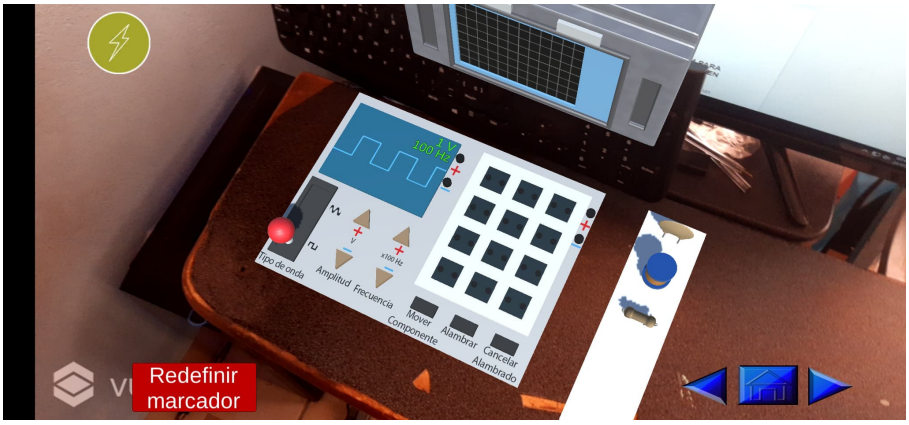


Figura 5.12 Disposición de los componentes virtuales en RA.

Una vez definida la disposición y espacio que ocupan los elementos virtuales al ser desplegados en el entorno real, se delimitan los elementos interactivos que el usuario podrá manipular haciendo uso del marcador cilíndrico de interacción y los elementos destinados para mostrar información. En el caso del módulo de protoboard con generador, la figura 5.13 muestra la ubicación de estos elementos de acuerdo a la siguiente lista de clasificación:

Elementos para mostrar información:

1 Pantalla del generador donde se grafica la señal con la que se alimentará el circuito
2 Amplitud de la señal de alimentación, en V
3 Frecuencia de la señal de alimentación, en Hz

Elementos interactivos:

4 Palanca del generador para cambiar la forma de la señal
5 Botones para aumentar y disminuir el voltaje pico de la señal de alimentación
6 Botones para aumentar y disminuir la frecuencia de la señal de alimentación
7 Botón para habilitar la manipulación de los componentes electrónicos
8 Botón para habilitar la selección de las terminales del generador y cables de alambrado
9 Botón para cancelar un proceso de alambrado
10 Terminales en las que se pueden colocar componentes para posteriormente alambrar el circuito
11 Terminales del generador de funciones
12 Terminales del osciloscopio
13 Espacio de selección de componentes electrónicos.

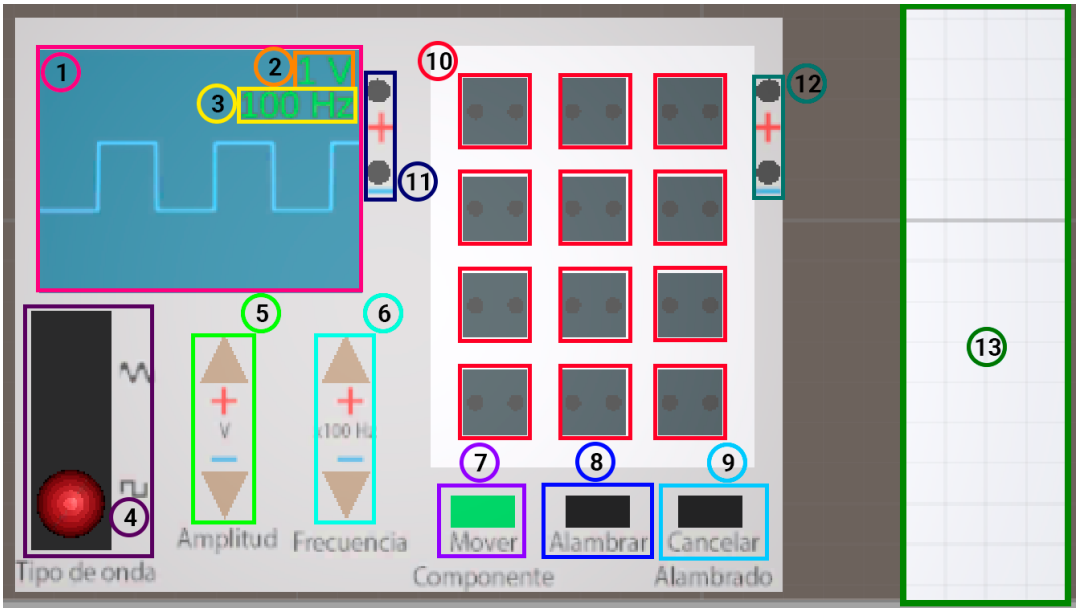


Figura 5.13 Elementos de interacción del módulo de protoboard y generador.

Los elementos de interacción se pueden ver afectados tanto por su selección con el marcador cilíndrico como por una colisión con otros elementos de esta misma categoría; por esta razón, para poder identificar y llevar el control de las interacciones de cada elemento, se les colocó un polígono transparente alrededor, dedicado a verificar el contacto de un elemento con otro (conocido como *hitbox*). Así mismo, estos elementos cuentan con variables booleanas a las que se les llamó banderas, que funcionan como indicadores para monitorear el estado de los componentes, ya sea que estén teniendo contacto con otro componente, o bien si son manipulados por el marcador cilíndrico. Estas banderas se enlistan en la tabla 5.2.

Tabla 5.2 Banderas de los elementos de interacción.

Componente virtual	Banderas asociadas
Marcador cilíndrico	MC1 Interacción con componentes electrónicos virtuales de resistor, inductor y condensador habilitada MC2 Interacción con terminales del módulo protoboard y cables habilitada MC3 Movimiento de componente electrónico en proceso MC4 Alambrado en proceso
Componente electrónico	CE1 Movimiento de un componente en proceso
Terminales electrónicas	TE1 Componente colocado (para terminales de protoboard) TE2 Cable de alambrado colocado TE3 Cable de alambrado hacia osciloscopio conectado TE4 Es una terminal de la protoboard TE5 Es una terminal del generador TE6 Es una terminal del osciloscopio
Palanca del generador	PG1 El generador produce una señal cuadrada PG2 El generador produce una señal sinusoidal.

En cada actualización de pantalla de la *app*, se verifica el estado de la *hitbox* de los componentes, para revisar el contacto que tienen con otros elementos y levantar la bandera correspondiente a esa colisión. Cuando se detecta una colisión, se verifica de qué componente se trata y su estado, si se trata de una interacción válida se inicia un proceso de carga de 1.5 segundos con un círculo que en este tiempo se llena de forma radial de color verde, mostrado gráficamente en la esquina superior derecha de la pantalla, visible para el usuario (figura 5.14). Cuando se termina el proceso de carga, se realiza la acción correspondiente. Este proceso se ejemplifica en el diagrama de bloques de la figura 5.15.

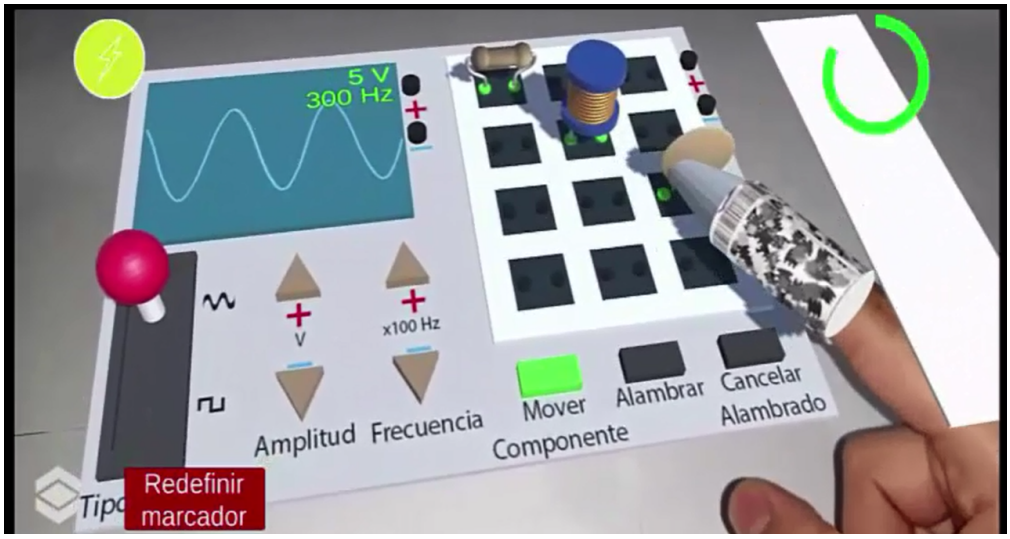


Figura 5.14 Círculo de carga mostrado en la esquina superior derecha durante la interacción de componentes.

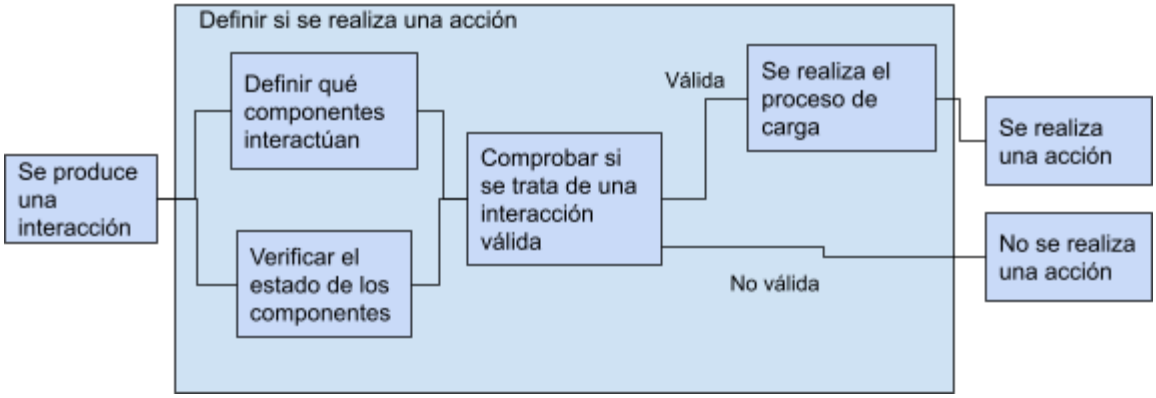


Figura 5.15 Diagrama de bloques del proceso de interacción del usuario con los componentes dentro de la app.

Se estableció que el uso del marcador cilíndrico sea el de un seleccionador, de esta forma cuando el usuario efectúe el contacto de este marcador con algún otro elemento, ocurrirá la colisión que levantará su respectiva bandera y modificará el estado del componente, realizando una acción correspondiente a las banderas habilitadas de acuerdo a la lista de la tabla 5.2.

Las posibles combinaciones de banderas con sus respectivas acciones están mostradas en la tabla 5.3, donde se observa que el marcador cilíndrico puede tener colisión con los elementos de la columna 1, procede a verificar el valor inicial de las banderas involucradas tanto del marcador como del componente y mencionadas en la columna 2; posteriormente, en caso de que la colisión de el marcador se mantenga en contacto con el componente durante el tiempo de carga, se valida la

interacción y consecuentemente se modifica el valor de las banderas en ambos componentes a un estado final definido en la columna 3, y finalmente se lleva a cabo la acción descrita en la columna 4.

Tabla 5.3 Interacciones posibles del marcador cilíndrico con los componentes de la app.

Componente con el que colisiona	Estado inicial de las banderas habilitada (✓) o deshabilitada (X)	Estado final de las banderas habilitada (✓) o deshabilitada (X)	Acción
Botón “Mover componentes”	Marcador: MC2(X), MC4(X)	Marcador: MC1(✓)	Habilita la interacción con componentes electrónicos (condensador, resistor o inductor)
Botón “Alambrar”	Marcador: MC1(X), MC3(X)	Marcador: MC2(✓)	Habilita la interacción con terminales y cables
Botón “Cancelar alambrado”	Marcador: MC2(✓), MC4(✓) Terminal: TE2(✓), TE4(✓)	Marcador: MC4(X) Terminal: TE2(X)	Cancela el último alambrado de componentes iniciado y regresa la terminal de protoboard a color verde
Botón “Cancelar alambrado”	Marcador: MC2(✓), MC4(✓) Terminal: TE2(✓), TE5(✓)	Marcador: MC4(X) Terminal: TE2(X)	Cancela el último alambrado de componentes iniciado y regresa la terminal de generador a color gris
Botón “Cancelar alambrado”	Marcador: MC2(✓), MC4(✓) Terminal: TE3(✓), TE6(✓)	Marcador: MC4(X) Terminal: TE3(X)	Cancela el último alambrado iniciado desde el osciloscopio y regresa la terminal a color gris
Palanca del generador	Palanca: PG1(✓)	Palanca: PG1(X), PG2(✓)	Cambia la señal cuadrada a una sinusoidal en el generador
Palanca del generador	Palanca: PG2(✓)	Palanca: PG1(✓), PG2(X)	Cambia la señal sinusoidal a una cuadrada en el generador
Componente electrónico	Marcador: MC1(✓), MC3(X)	Marcador: MC3(✓)	El componente comienza a moverse a la misma

	Componente: CE1(X)	Componente: CE1(✓)	posición del marcador cilíndrico hasta que se realice otra acción
Terminal	Marcador: MC2(✓), MC4(X) Terminal: TE1(✓), TE2(X), TE4(✓) o TE2(X), TE5(✓)	Marcador: MC4(✓) Terminal: TE2(✓)	La terminal se ilumina de color amarillo y se inicia el proceso de alambrado entre componentes
Terminal	Marcador: MC2(✓), MC4(✓) Terminal: TE1(✓), TE2(X), TE4(✓) o TE2(X), TE5(✓)	Marcador: MC4(X) Terminal: TE2(✓)	La terminal se ilumina de color amarillo y se termina el proceso de alambrado entre componentes, creando un cable que une esta terminal y la seleccionada anteriormente
Terminal	Marcador: MC2(✓), MC4(✓) Terminal: TE3(X), TE6(✓)	Marcador: MC4(X) Terminal: TE3(✓)	La terminal se ilumina de color azul y se inicia el proceso de alambrado al osciloscopio
Terminal	Marcador: MC2(✓), MC4(✓) Terminal: TE3(✓), TE4(✓)	Marcador: MC4(X) Terminal: TE3(✓)	La terminal se ilumina de color azul y se termina el proceso de alambrado al osciloscopio, creando un cable que une esta terminal y la selecionada anteriormente.

De acuerdo al procedimiento de validación de interacciones habilitando las banderas asociadas de cada componente, se especificó el caso en que el marcador cilíndrico hubiera realizado una colisión con un componente electrónico y este, a su vez, cambió su posición a la misma donde se encontraba el marcador cilíndrico; es decir que el marcador cilíndrico tuviera asociado un componente tal como se muestra en la figura 5.16. Para este caso en particular, el componente electrónico que está siendo manipulado puede llegar a interactuar con otros elementos durante su desplazamiento, por lo que dependiendo del componente con el que tenga contacto, se realicen determinadas acciones o se modifiquen. Estos posibles escenarios están descritos en la tabla 5.4.

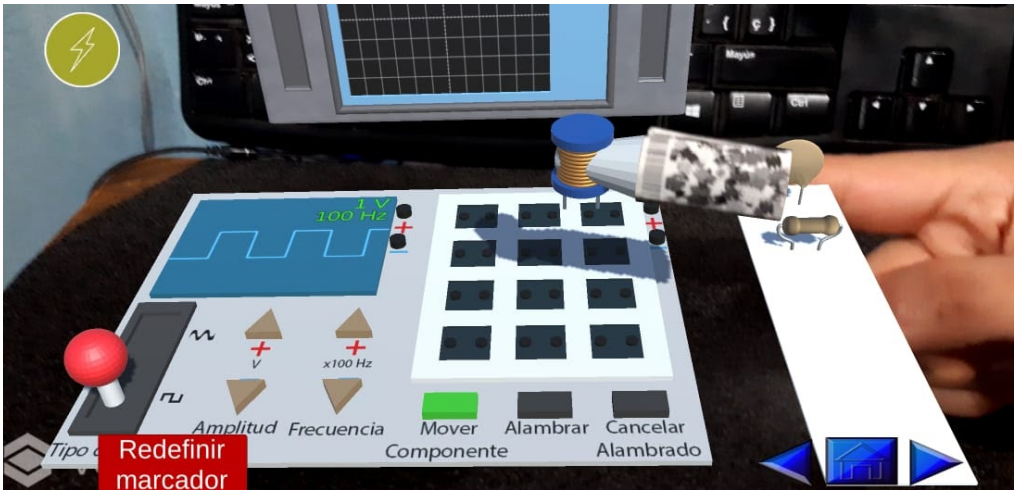


Figura 5.16 Marcador cilíndrico que manipula un componente electrónico para cambiar su posición.

Tabla 5.4 Interacciones posibles de un componente electrónico al ser desplazado con el marcador cilíndrico.

Componente con el que colisiona	Estado inicial de las banderas habilitada (✓) o deshabilitada (X)	Estado final de las banderas habilitada (✓) o deshabilitada (X)	Acción
Terminal del módulo de protoboard	Componente: CE1(✓) Terminal: TE1(X), TE4(✓)	Componente: CE1(X) Terminal: TE1(✓)	La terminal se ilumina de color verde y se coloca el componente sobre las terminales
Espacio para selección de componentes	Componente: CE1(✓)	Componente: CE1(X)	El componente se coloca en el espacio de selección de componentes electrónicos.

El desarrollo de la mecánica de interacción fue un proceso iterativo de pruebas con diferentes versiones del marcador de interacción, con objeto de definir la geometría con menor dificultad de manipulación y habilitación de banderas para realizar las acciones de acuerdo a cada interacción. La descripción y análisis de las interacciones previas está mostrado en el apéndice H, mientras que el código final de la mecánica aquí descrita se presenta en el apéndice I.

5.4 Despliegue de un osciloscopio en tiempo real

En el apartado 3.1 se planteó como un requerimiento de la *app*:

> *"Para el cumplimiento de las actividades definidas por la práctica presencial de laboratorio, la app debe permitir tomar los datos necesarios del experimento realizado"*

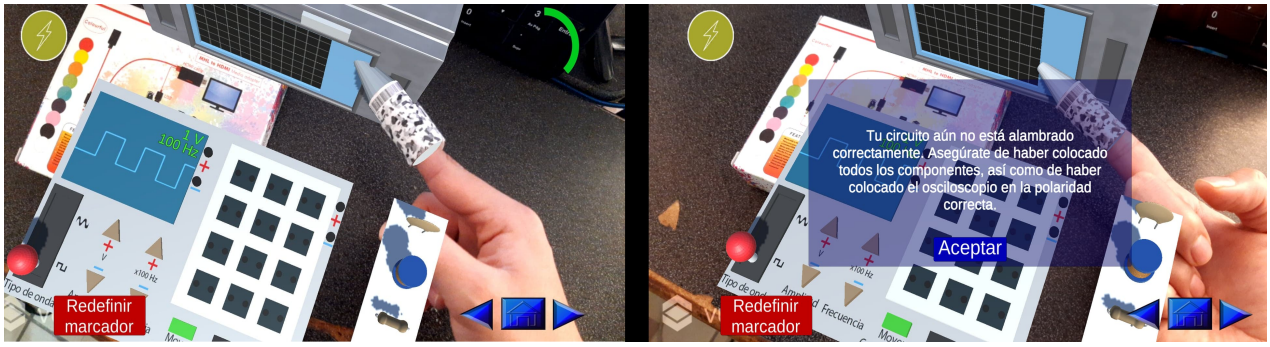
Para cumplir con este requerimiento, se implementó una interfaz de simulación virtual de un osciloscopio, que permitiera al usuario observar el comportamiento y tomar mediciones en tiempo real del circuito alambrado en RA.

Una vez que el usuario haya completado el armado del circuito con ayuda de los marcadores y la tecnología de RA, puede interactuar con el modelo tridimensional del osciloscopio al seleccionarlo con ayuda del marcador cilíndrico, tal como se muestra en la figura 5.17a. Al hacerlo, la *app* comprobará el estado del alambrado de su circuito, verificando las siguientes condiciones definidas a partir de los circuitos propuestos en la práctica:

- Que se haya alambrado un condensador, un inductor y un resistor en serie con la fuente de alimentación
- Que el osciloscopio esté alambrado en paralelo con un componente únicamente y la polaridad correcta, con la terminal positiva del componente más cercana a la terminal positiva del generador
- Que no haya conexiones en paralelo entre componentes (una disposición de alambrado distinta a la propuesta en la práctica).

Cada uno de los componentes electrónicos que son desplegados en RA, tienen asociada una etiqueta interna para su identificación, por lo que durante el alambrado del circuito y la habilitación de banderas que conlleva la mecánica de interacción, es posible reconocer cuáles son los componentes colocados y su posición dentro del modelo de protoboard. Con ayuda de estas etiquetas se programó la revisión de la estructura del circuito así como la identificación de las terminales conectadas tanto a la fuente del generador como hacia el modelo de osciloscopio, comprobando el cumplimiento de las condiciones establecidas para el circuito.

Si alguna de las condiciones no se cumple, la *app* muestra en pantalla al usuario el mensaje: "*Tu circuito aún no está alambrado correctamente. Asegúrate de haber colocado todos los componentes, así como de haber colocado el osciloscopio en la polaridad correcta.*", como se muestra en la figura 5.17b.



a) *b)*

Figura 5.17 Interacción con el osciloscopio dentro de la app: a) marcador cilíndrico del usuario en contacto con el osciloscopio; b) mensaje mostrado al usuario si no se cumplen las condiciones de alambrado correcto.

Si todas las condiciones se cumplen correctamente, la *app* termina con el despliegue de RA y cambia de pantalla para mostrar un osciloscopio en un ambiente completamente virtual; la razón del cambio de tecnología de realidad es para permitir que el usuario pueda manipular cómodamente los controles del osciloscopio y, de ser necesario, realice notas de las señales que observe.

En la imagen 5.18 se pueden apreciar los controles del osciloscopio, así como la información que ofrece al usuario de acuerdo a la siguiente lista:

1 Control para la escala de voltaje por división vertical en la pantalla del osciloscopio
2 Control para la escala de tiempo por división horizontal en la pantalla del osciloscopio
3 Botón para cambiar la señal afectada por el control de escala de voltaje por división
4 Botón para mostrar u ocultar la señal del generador de funciones
5 Botón para mostrar u ocultar la señal de comportamiento del componente que se alambró en paralelo al osciloscopio
6 Pantalla del osciloscopio
7 Gráfica de la señal de voltaje en el componente que se alambró en paralelo al osciloscopio (azul)
8 Gráfica de la señal del generador de funciones (verde)
9 Texto con la escala de tiempo por división horizontal correspondiente a la señal (amarillo)
10 Texto con la escala de voltaje por división vertical correspondiente a la señal del circuito alambrado (azul)

11 Texto con la escala de voltaje por división vertical correspondiente a la señal del generador (verde).

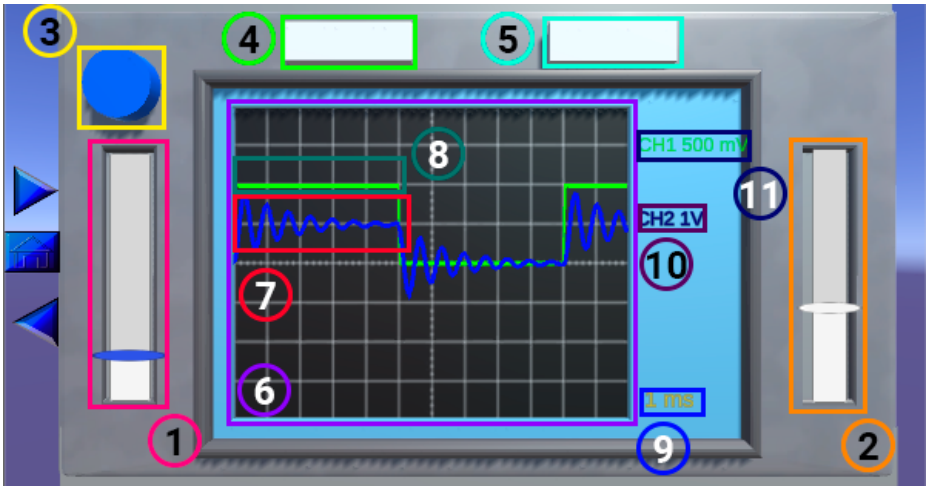


Figura 5.18 Osciloscopio virtual mostrado dentro de la app.

Al momento que el usuario ingresa a la pantalla del osciloscopio, se resuelve en tiempo real la ecuación diferencial asociada a dicho circuito, con los métodos matemáticos descritos en el capítulo 2.5. Utilizando el método fasorial como una función en código, cuando la señal del generador es sinusoidal, y con el empleo del método Runge-Kutta de cuarto orden cuando se trata de una señal de tren de pulsos. En ambas resoluciones se incluyen los valores establecidos para la resistencia interna del generador de funciones (50 Ω) y del inductor (150 Ω), de acuerdo a los comentarios por parte de los profesores aplicadores de la práctica presencial. De esta manera, permite que el usuario pueda visualizar la señal de la respuesta respectiva a los distintos valores de cualquiera de los componentes que integran el circuito RLC en serie.

La implementación en código del algoritmo para verificar el correcto alambrado del circuito, así como el graficado de la información en el osciloscopio y la modificación de sus parámetros de visualización, se pueden consultar en el apéndice J.

5.5 Interfaz de usuario de entorno virtual

La programación dentro del motor de Unity fue a través de lo que se denominan escenas, las cuales conforman cada una de las pantallas visibles dentro de la *app*, por lo que la pantalla de armado del circuito con RA y la que muestra el osciloscopio virtual son escenas independientes que permiten al usuario el desarrollo de las actividades propuestas por la práctica a desarrollar, sin embargo, se requirió de una

interfaz la cual le brindará al usuario información acerca del uso general de la *app* y lo guiará a través del proceso de realización de la práctica.

Con este propósito, se crearon un conjunto de pantallas en las que el usuario recibirá la información de uso, así como las instrucciones de las actividades a realizar. Cada una de las escenas cuenta con distintos botones de acción que dependen de la información presentada y las actividades a realizar. La descripción de estas pantallas se muestra en la tabla 5.5.

Tabla 5.5 Escenas que conforman las pantallas dentro de la app RACE.

Escena	Propósito e información presentada en pantalla	Botones y acciones
Inicio (figura 5.19)	Pantalla con el menú de bienvenida a la *app*	● Botón de instrucciones: cambia a la pantalla de instrucciones ● Botón de inicio: cambia a pantalla de actividad ● Botón de créditos: cambia a pantalla de créditos ● Botón de marcador: abre una liga de internet que permite descargar el patrón de recorte del marcador cilíndrico de interacción ● Botón i: cambia a pantalla de contacto ● Botón salir: termina los procesos y cierra la *app* en el dispositivo móvil ● Botones de navegación
Contacto (figura 5.20)	Pantalla para mostrar información de contacto en caso de que se requiera contactar a los desarrolladores	● Botones de navegación
Créditos (figura 5.21)	Pantalla para mostrar los agradecimientos por parte de los desarrolladores	● Botones de navegación

Instrucciones (figura 5.22)	Pantalla con información respecto al uso general de la *app* y las imágenes descriptivas de cada proceso	● Botones de navegación
Actividad (figura 5.23)	Pantalla con las indicaciones para desarrollar la respectiva actividad de la práctica: “Aplicación de las ecuaciones diferenciales en circuitos eléctricos”	● Botón de sistemas: cambia a pantalla de gráficas de sistemas ● Botones de navegación
Gráficas de sistemas (figura 5.24)	Pantalla que muestra las gráficas correspondientes a los tipos de sistemas que puede tener la señal de salida del circuito eléctrico armado	● Botón de manual de práctica: abre enlace de internet para descargar el formato de la práctica “Aplicación de las ecuaciones diferenciales en circuitos eléctricos” ● Botones de navegación
Selección de componentes (figura 5.25)	Pantalla que permite seleccionar los componentes disponibles para armar el circuito.	● Modelos virtuales de los componentes: al hacer click en estos modelos animados, se despliega una versión más grande en la parte central de la pantalla para su visualización en detalle ● Botones de valores: estos botones permiten seleccionar los valores nominales del componente desplegado en la parte central ● Botón de agregar: añade a la lista, el componente visualizado con el valor seleccionado, que posteriormente estará disponible para utilizar en el armado del circuito ● Botón X: permite eliminar de la lista el componente seleccionado ● Botones de navegación

Alambrado en RA (figura 5.26)	Pantalla que despliega la tecnología de RA para llevar a cabo el desarrollo de las actividades de armado de circuitos y selección de osciloscopio	• Botón de flash: este botón activa el flash del dispositivo móvil para mejorar la cantidad de luz del entorno • Botón de definir marcador: cuando el semáforo de calidad de imagen se encuentre en verde, al presionar este botón se desplegarán los componentes con RA, en la imagen captada por la cámara • Botón de redefinir marcador: una vez que los componentes se encuentren visibles en RA y se requiera de cambiar de imagen de despliegue, al presionar este botón se ocultaran los componentes virtuales y se permite la selección de una nueva imagen para el despliegue • Botones de navegación
Osciloscopio virtual (figura 5.27)	Pantalla que muestra el osciloscopio virtual con la gráfica en tiempo real del circuito armado en la pantalla de alambrado	• Botones de navegación.

Figura 5.19 Pantalla de inicio de la app.

Figura 5.20 Pantalla de contacto y reporte de errores.

Figura 5.21 Pantalla de créditos.

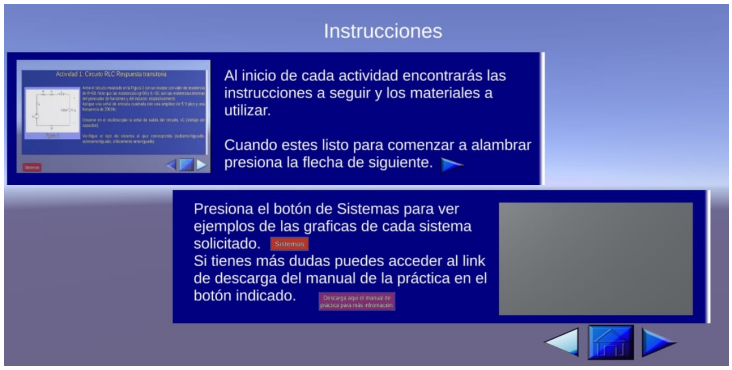


Figura 5.22 Pantalla de instrucciones de uso de la app.

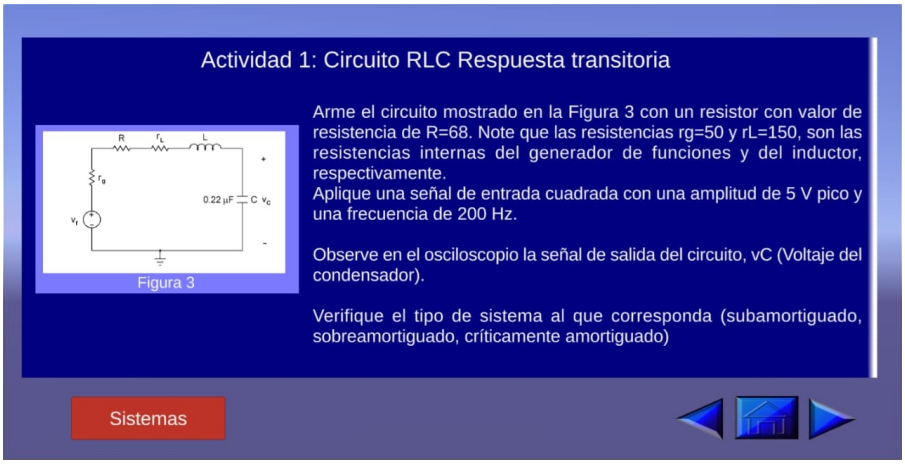


Figura 5.23 Pantalla de actividad a desarrollar.

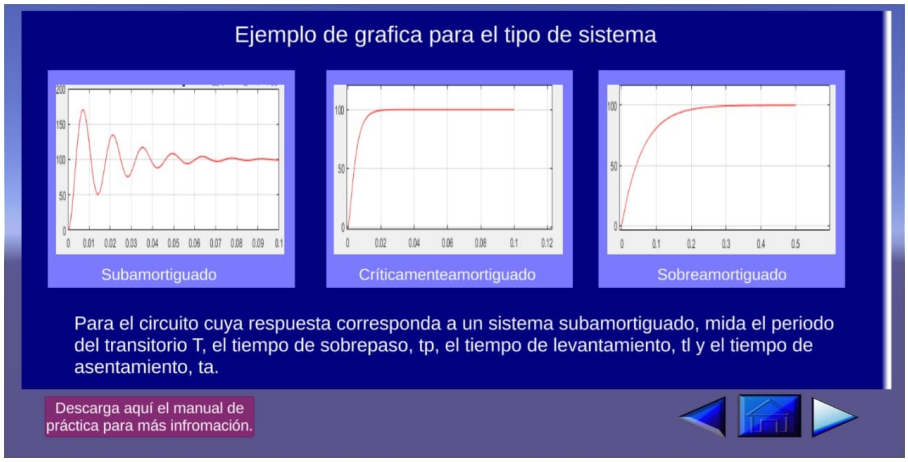


Figura 5.24 Pantalla de gráficas de sistemas.

Figura 5.25 Pantalla de selección de componentes.

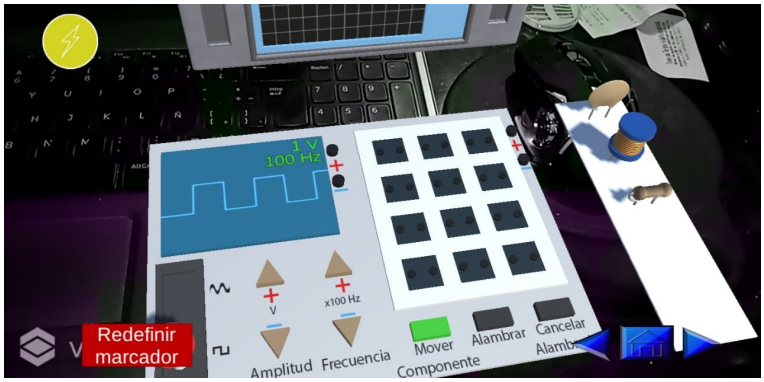


Figura 5.26 Pantalla de alambrado del circuito en RA.

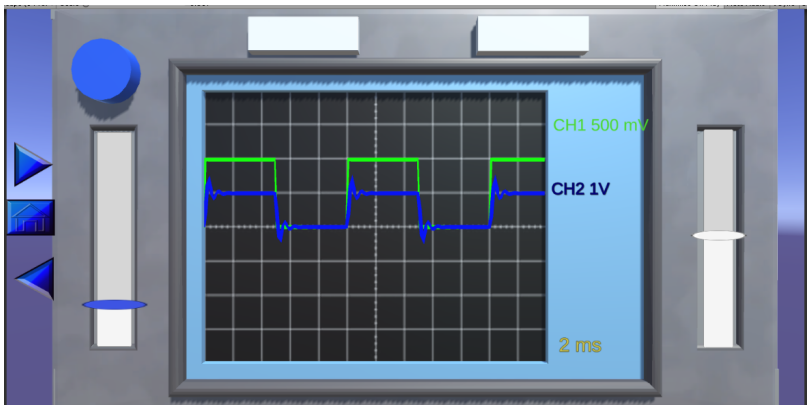


Figura 5.27 Pantalla de osciloscopio virtual.

Cada una de las pantallas cuenta con un panel conformado por los botones de navegación que se observan en la figura 5.28. Estos botones sirven para poder cambiar a la pantalla anteriormente cargada al presionar la flecha izquierda, avanzar a la pantalla siguiente con la flecha derecha y regresar al menú de inicio al presionar el botón central. Este panel tiene como fin que el usuario pueda desplazarse por las pantallas de la aplicación para acceder tanto a la información de interés como a las escenas de interacción y desarrollar las actividades que conforman la práctica a realizar.

Figura 5.28 Botones de navegación.

Para seleccionar la paleta de colores utilizada en el diseño de la interfaz de usuario, se decidió por las variaciones de tono en color azul (figura 5.29), debido a que el color azul es considerado en el diseño de UI (interfaz de usuario por las siglas en inglés de *User Interface*) como un color neutro que no influye en el diseño como

femenino o masculino, siendo que esto respalda la transmisión de una personalidad organizada y con datos accesibles [51, Cap. 6].

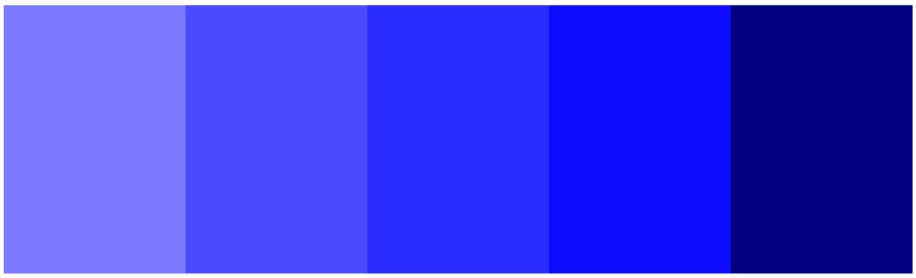


Figura 5.29 Paleta de color para interfaz de usuario.

Se agregaron el uso de tonos brillantes para los fondos del entorno virtual en contraste de lienzos en tonos más oscuros y sombras para la presentación de información y botones de acción utilizando las variaciones de color de acuerdo al uso visual de cada elemento [51, Cap. 6]. De acuerdo a Zeltīte Barševska en su artículo “Color in UI design”, los colores ya tienen sus propias asignaciones de acuerdo a la percepción de tonos cálidos y fríos, en las que la mayoría de los usuarios asocia el color azul como un tono frío para presentación de datos, educación y comodidad, y por esta razón que es asociado con tecnologías e innovación [52], por lo tanto considerado como adecuado para este trabajo.

Todas las escenas se diseñaron de manera gráfica dentro del entorno de Unity, siguiendo el orden descrito por el diagrama de flujo de la figura 5.30. Esta secuencia de escenas guía al usuario dentro de las pantallas que conforman la *app*, en el que la secuencia de inicio de las actividades es el proceso principal para llevar a cabo la práctica.

En el apéndice K se encuentran los bocetos y maquetas de las pantallas para la interfaz de usuario, los cuales sirvieron como base para lograr la disposición final descrita en este capítulo. En el apéndice L se encuentran los códigos correspondientes para cada una de las escenas utilizadas en el desarrollo de este proyecto.

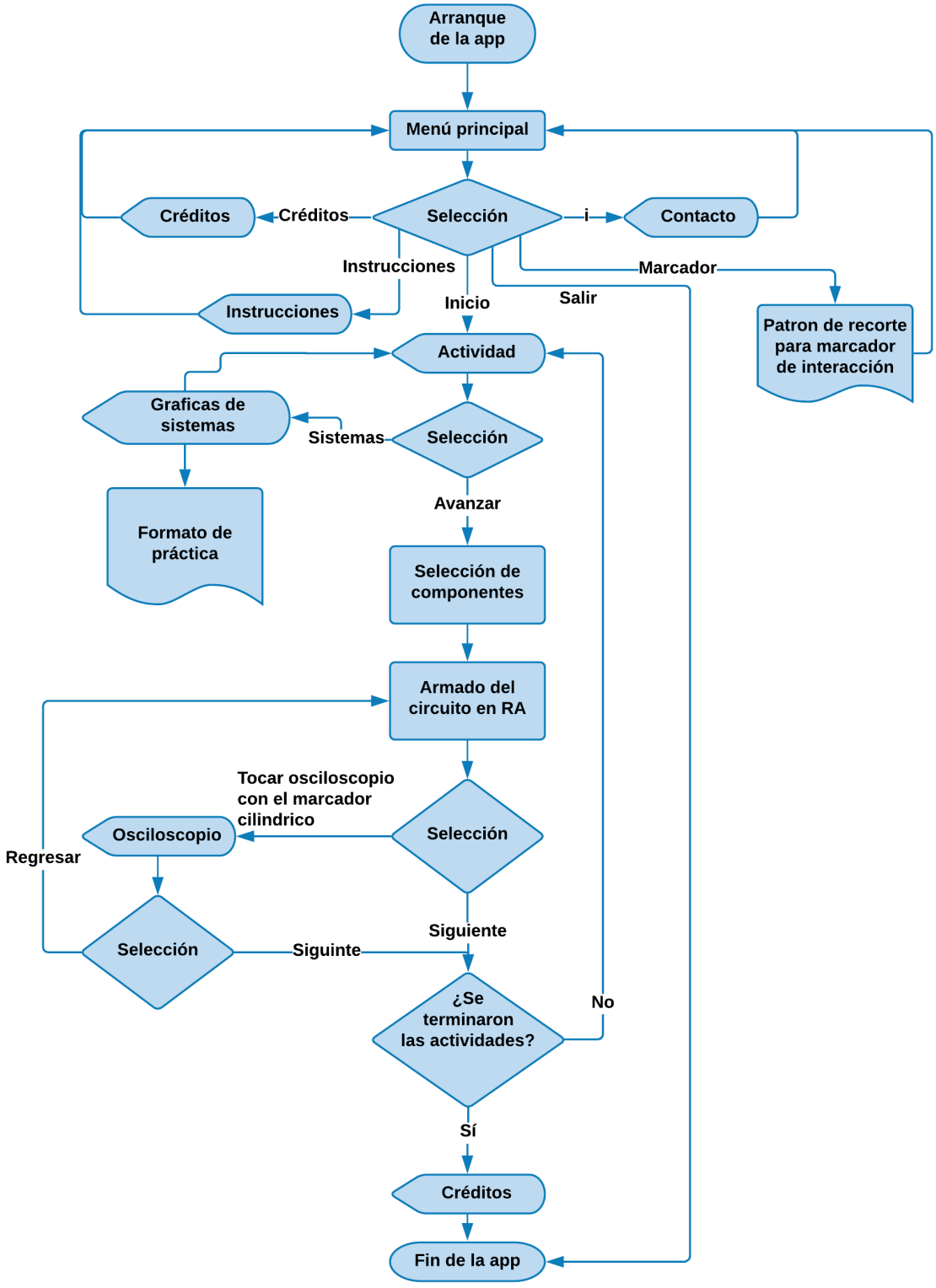


Figura 5.30 Diagrama de flujo de aparición de las escenas en la app.

6 ESTABILIZACIÓN

Finalizado el proceso iterativo de construcción de los subsistemas independientes que conforman la *app* RACE, se realizó la integración y unificación del sistema completo en un producto final, para este caso, un archivo ejecutable y listo para instalarse en los dispositivos móviles establecidos como objetivo durante la fase de gestación. Se consideraron las siguientes entradas a esta fase:

- El resultado de la última iteración del desarrollo de los subsistemas de la *app* durante la fase de construcción (capítulo 5)
- Descripción del hardware objetivo para la *app* (capítulo 4).

En este capítulo se describe la fase de estabilización del sistema unificado, a través de las herramientas de compilación dentro del motor de Unity, así como la prueba de verificación de uso al desarrollar las actividades programadas para el cumplimiento de los objetivos de la práctica.

6.1 Compilación de la app dentro de Unity

Para la creación del archivo de instalación de la *app*, se definieron los siguientes parámetros correspondientes a las características del proyecto:

- Nombre del producto con el que se mostrará la *app* una vez que sea instalada. En este caso, “RACE”
- Ícono o imagen con la que se representa la *app* (figura 6.1)
- Orientaciones permitidas en la rotación; dado que todas las escenas fueron diseñadas considerando una pantalla horizontal, se estableció que esta orientación en el dispositivo sea la única permisible para la visualización de la *app* en ejecución

- API o Interfaz de Programación de Aplicaciones, por sus siglas en inglés (*Application Programing Interface*). Este parámetro define la versión de Android para la cual estará optimizada la aplicación y la versión más antigua en la que podrá ser instalada. De acuerdo con la información recopilada durante la fase de gestación, se estableció como versión más antigua compatible Android 5 y como versión objetivo Android 8.

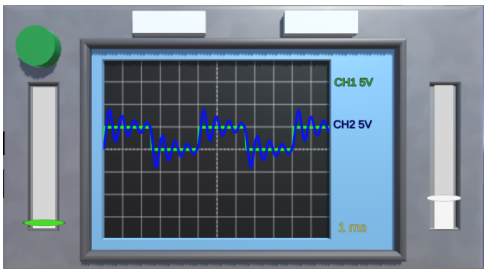


Figura 6.1 Imagen usada como icono de la app RACE.

Las aplicaciones hechas para dispositivos móviles Android están embebidas dentro de un archivo de código Java, por lo que para su creación se necesita del kit de desarrollo de Java (*Java Development Kit* o JDK) y una máquina virtual de Java (*Java Virtual Machine* o JVM). Unity integra estas herramientas de manera automática, siendo únicamente necesaria la actualización de la versión compatible para la API seleccionada, de acuerdo al sistema operativo que se estableció para la ejecución. La selección de herramientas externas se muestra en la figura 6.2.

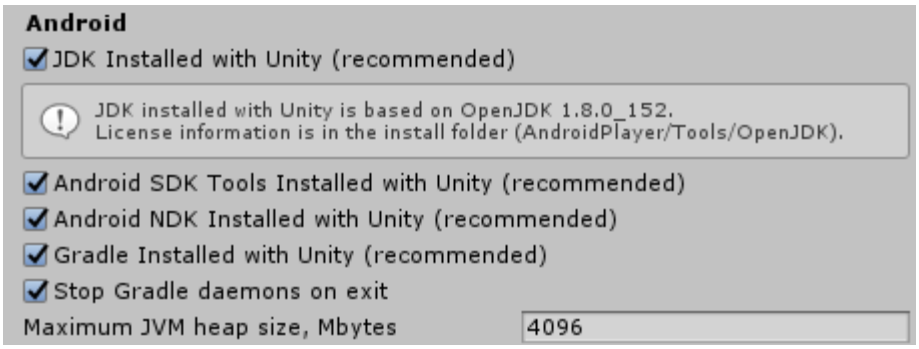


Figura 6.2 Preferencias de herramientas externas dentro de Unity.

Como se mencionó en el apartado 5.5, la *app* está conformada a partir de escenas, que son las diferentes pantallas que se muestran al usuario. Para la creación del archivo ejecutable, se seleccionaron y ordenaron las escenas correspondientes, de acuerdo a la secuencia de ejecución prevista que tendrá la *app*, es decir, que la pantalla inicial será la primera escena que se haya colocado dentro de esta configuración. En la figura 6.3 se observan las escenas seleccionadas para la

construcción y el orden de secuencia que tienen, siendo la escena “Inicio” la que muestra la pantalla inicial.

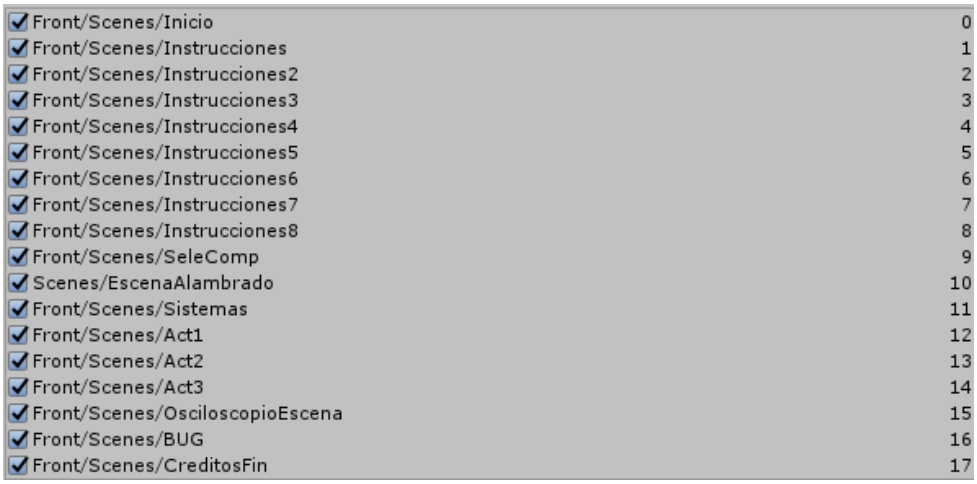


Figura 6.3 Escenas seleccionadas para la construcción de la app.

De forma predeterminada, Unity carga las escenas de forma sucesiva de acuerdo al orden en el que fueron ordenadas en las propiedades del compilador, por lo que al descartar una escena muestra la siguiente según el orden de la lista. De acuerdo al diagrama de flujo mostrado en la figura 5.30, la secuencia de escenas dentro de la *app* RACE no es lineal, y algunas se muestran más de una vez dentro del desarrollo de la práctica. Por esta razón, se implementó un archivo de código que se encarga de ordenar la presentación de las escenas dentro de la *app*, llevando el registro de escenas mostradas hasta el momento, para que cuando el usuario solicite un cambio de pantalla, busque la escena correspondiente, la cargue y la despliegue, todo esto a partir de su nombre, el cual es único de cada escena. Este código se puede encontrar en el apéndice L, y un fragmento del mismo se muestra en la figura 6.4.

```
        0 referencias
136     public void SelecComp()
137     {
138         SceneManager.LoadScene("SeleComp");
139     }
140
        0 referencias
141     public void Alambrado()
142     {
143         SceneManager.LoadScene("EscenaAlambrado");
144     }
```

Figura 6.4 Fragmento del código encargado de administrar la aparición de escenas dentro de la app.

Con estos parámetros definidos, se integraron los subsistemas que conforman la solución completa correspondiente a cada escena construida y, posteriormente, se creó el archivo .apk de instalación de la *app,* con ayuda del compilador de Unity.

6.2 Prueba de ejecución

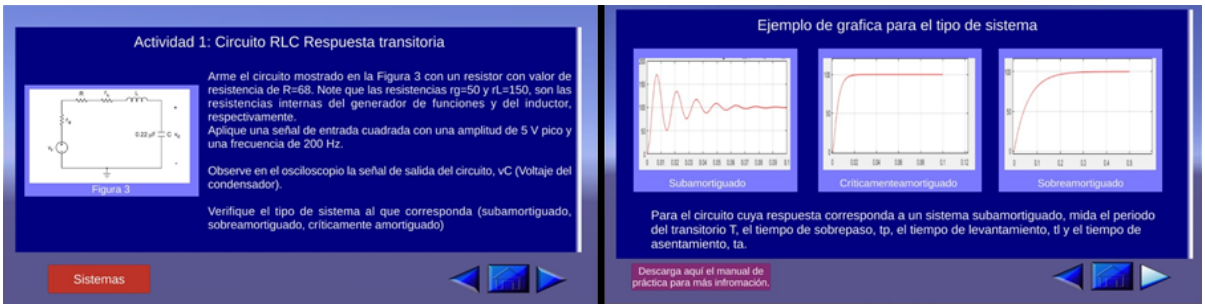
Una vez que el desarrollo completo de la práctica: "Aplicación de las ecuaciones diferenciales en circuitos eléctricos" ha sido integrado en un archivo ejecutable, este fue instalado en un dispositivo de prueba con el fin de comprobar la secuencia de ejecución de uso prevista.

Para la realización de la prueba, se utilizó un dispositivo móvil con sistema operativo Android 5, por ser el sistema mínimo compatible con la *app*.

La secuencia de ejecución prevista para el usuario tiene la finalidad de brindarle una experiencia similar a la de realizar la práctica de forma presencial en un laboratorio, dándole la posibilidad de experimentar y cometer algunos errores de manera controlada. Esto permite que la *app* no guíe de manera excesiva al usuario, ya que esto reduciría su experiencia al seguir una simulación guiada.

De acuerdo al diagrama de flujo del apartado 5.5 (figura 5.30), al abrir la *app*, el usuario comienza en la pantalla de “Inicio”, en la que puede acceder a las instrucciones de la práctica, los créditos de los desarrolladores, datos de contacto o iniciar la realización de la práctica.

Luego de presionar el botón de inicio, se presenta la pantalla correspondiente a la actividad 1, donde se describe el circuito que debe armar, los componentes a utilizar y sus respectivos valores. Dentro de esta pantalla, puede presionar el botón “Sistemas”, como se muestra en la figura 6.5a, cambiando a la pantalla de gráficas de sistemas (figura 6.5b), donde se muestra la identificación gráfica del comportamiento de la señal esperada del circuito a armar, además de un botón con el enlace para ver o descargar el formato original de la práctica, en caso de necesitarlo.



a) *b)*

Figura 6.5 Pantallas para actividad: a) pantalla de actividad 1; b) pantalla de gráficas de sistemas.

Desde la pantalla de actividad, presionando el botón para continuar, el usuario avanza a la pantalla de “Selección de componentes”. En esta pantalla se muestran los componentes de la lista mostrada en la tabla 6.1, los cuales incluyen además de los componentes indicados por la práctica, valores nominales adicionales para permitir que el usuario experimente con las variaciones de la señal.

Para seleccionar los componentes, el usuario presiona el modelo tridimensional del componente que desea agregar, haciendo que dicho elemento aparezca en el centro de la pantalla y de mayor tamaño. Debajo de este modelo aparece el valor nominal del componente con flechas a la izquierda y derecha, para disminuir o aumentar su valor respectivamente. Al presionar el botón con el texto “Agregar”, el componente se muestra en la lista que aparece del lado derecho de la pantalla, habilitando el botón con una cruz junto al nombre del componente, que al presionarlo permite quitar el componente de la selección.

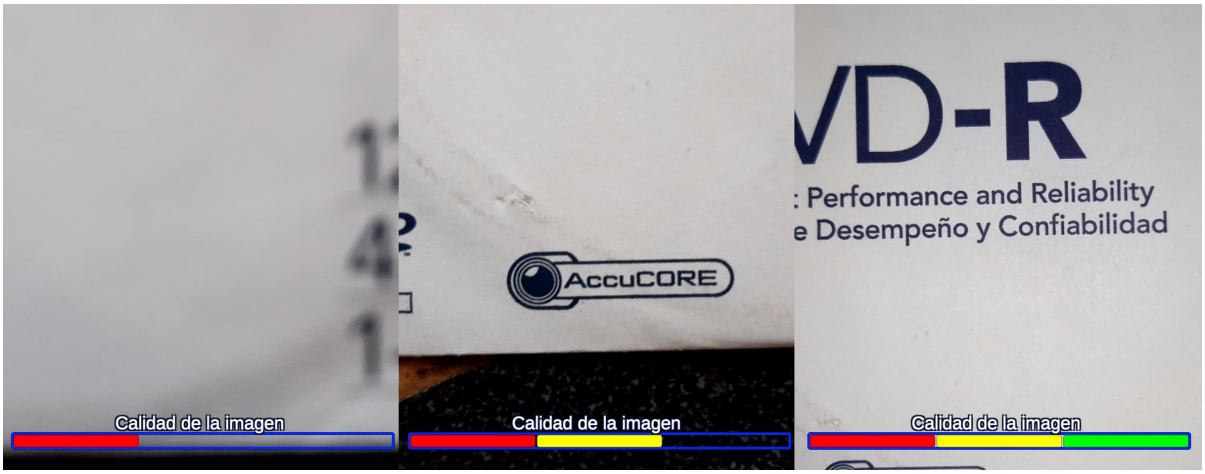
Tabla 6.1 Valores nominales de los componentes electrónicos.

Componente electrónico	**Valores nominales**
Condensador	● 0.12 µF ● 0.22 µF ● 0.33 µF ● 0.47 µF ● 0.56 µF
Resistor	● 68 Ω ● 180 Ω ● 820 Ω ● 1000 Ω ● 1800 Ω
Inductor	● 30 mH ● 40 mH ● 50 mH ● 60 mH ● 70 mH

El usuario debe seleccionar al menos un componente para continuar a la pantalla de alambrado del circuito. Si trata de avanzar sin haber seleccionado algún componente, aparecerá una ventana con el texto "*Debes seleccionar al menos 1 componente para poder realizar la actividad siguiente*", como se muestra en la figura 6.6. Se tomó esta decisión ya que una escena sin componentes no permitiría al usuario realizar algún tipo de actividad. Cabe mencionar que no se obliga al usuario a seleccionar todos los componentes que sean necesarios para la actividad en curso, o bien, seleccionar los valores correctos de los mismos.

Figura 6.6 Mensaje al usuario si no selecciona componentes.

Después de la selección de componentes, se presenta al usuario la pantalla de "Alambrado del circuito en RA". Como se describió en el apartado 5.3, se solicita inicialmente la selección de un marcador plano para poder desplegar los elementos virtuales, lo cual se realiza enfocando una imagen por completo con la cámara del dispositivo. La imagen enfocada es indicada al usuario a través del semáforo de colores en la parte inferior de la pantalla (figura 6.7).



a) *b)* *c)*

Figura 6.7 Semáforo que indica la calidad de la imagen enfocada por la cámara: a) imagen sin detalles (rojo); b) imagen con pocos detalles (amarillo); c) imagen con detalles suficientes (verde).

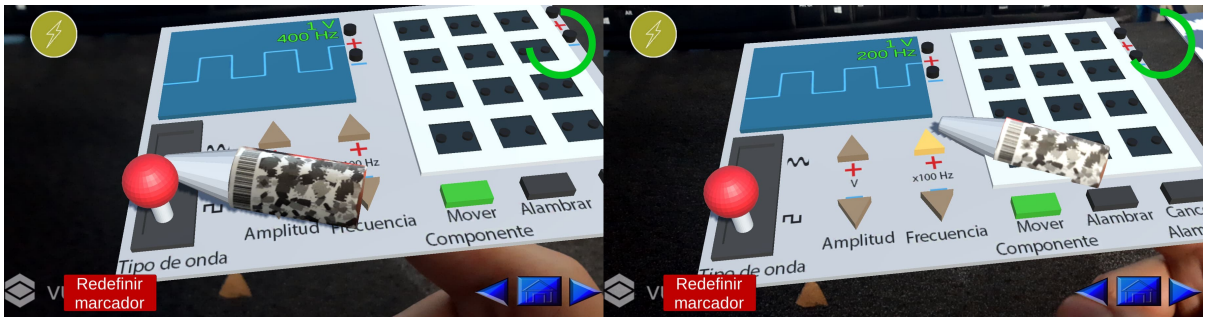
Para ajustar el enfoque de la imagen apropiadamente, el usuario puede presionar cualquier parte de la pantalla, además de hacer uso del botón amarillo de la interfaz para encender y apagar la linterna de su dispositivo, y de esta froma ayudar a la iluminación de la imagen.

Cuando la imagen es adecuada para usarse como marcador, el usuario presiona el botón con el texto "Desplegar", para seleccionar la imagen y los elementos

tridimensionales de la práctica se desplieguen sobre la misma. En cualquier momento, es posible cambiar la imagen seleccionada como marcador plano, presionando el botón rojo con el texto “Redefinir marcador” y, así, repetir el proceso para seleccionar una nueva imagen.

Con los componentes virtuales desplegados, el usuario debe interactuar con ellos haciendo uso del marcador cilíndrico descrito en el apartado 5.3.2.

Dentro de las funciones a realizar, se selecciona el tipo de señal de alimentación del circuito manipulando la palanca del módulo de generador (figura 6.8a) y se modifican los parámetros al aumentar o disminuir el voltaje pico y frecuencia, por medio de la interacción con los botones correspondientes (figura 6.8b), de acuerdo a los valores solicitados por cada actividad.



a) *b)*

Figura 6.8 Interacción con los controles del generador de señales: a) manipulación de la palanca para selección de la señal de alimentación; b) manipulación de controles para selección de amplitud y frecuencia de la señal.

Mientras el botón con el texto “Mover componente” se encuentre iluminado de color verde, el usuario puede seleccionar con el marcador cilíndrico, los componentes a colocar desde el panel de componentes desplegado del lado derecho, y cambiar su posición hasta ubicarlo sobre las terminales del módulo de protoboard. Una vez que el componente se haya colocado, las terminales se iluminarán de color verde tal como muestra la figura 6.9.

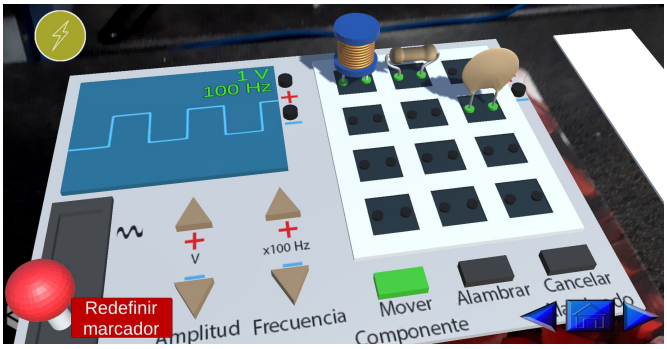


Figura 6.9 Componentes colocados en el módulo de protoboard con generador.

Para comenzar el alambrado del circuito, el usuario debe interactuar con el botón con el texto “Alambrar”, haciendo que se ilumine de color amarillo. De esta forma, habilitará la interacción con las terminales para realizar el alambrado. Al seleccionar una terminal se iluminará de color amarillo e inmediatamente después se creará una conexión con la siguiente terminal a seleccionar. Las conexiones entre componentes son de color naranja y las correspondientes tanto al generador de funciones como las del osciloscopio son de color rojo y negro para sus polaridades positiva y negativa, respectivamente.

Las terminales del osciloscopio se iluminan de color azul al interactuar con ellas, y sólo pueden ser conectadas a una terminal de componente que ya haya sido conectada con otro componente dentro del circuito. Las conexiones descritas se muestran en la figura 6.10.

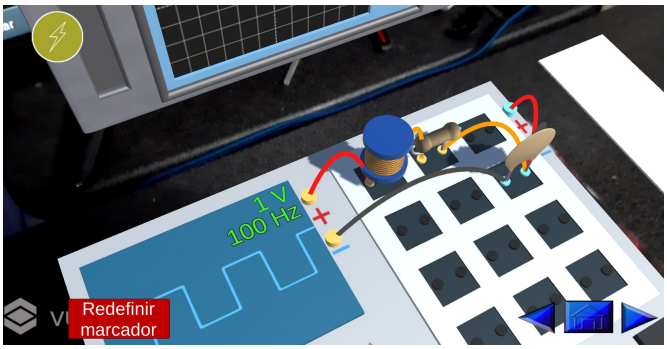


Figura 6.10 Componentes conectados en el módulo de protoboard y generador.

Durante el modo de alambrado, el usuario puede cancelar la selección de alguna terminal a conectar ya iluminada en amarillo, para esto debe interactuar con el botón de “Cancelar alambrado”. Por otro lado, cuando existe un cable de conxión entre dos componentes, el usuario debe interactuar con este cable hasta que el circulo de carga de color rojo se llene por completo y el cable sea eliminado, tal como se observa en la figura 6.11.

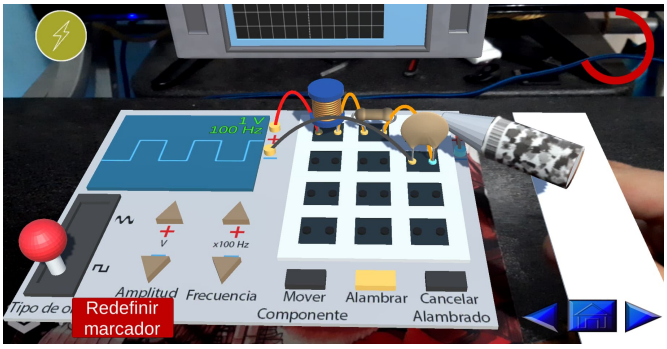


Figura 6.11 Eliminación de una conexión entre componentes.

Una vez que sea concluido el alambrado del circuito y corresponda a la disposición solicitada por la práctica, el usuario puede acceder a la ventana del osciloscopio manteniendo el contacto del marcador cilíndrico con este modelo virtual (figura 6.12).

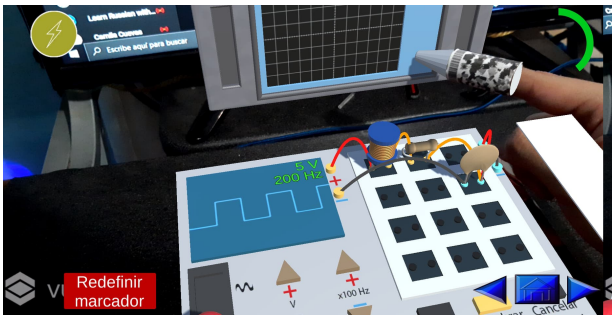


Figura 6.12 Interacción del usuario con el osciloscopio.

En la escena de “Osciloscopio virtual”, se observan las señales graficadas en tiempo real, y el usuario puede manipularlas para obtener la información necesaria con el fin de cumplir con los objetivos de la práctica o bien avanzar a la siguiente actividad a desarrollar.

Este proceso se repite hasta que el usuario complete en su totalidad las tres actividades de armado de circuitos. Al terminar se le muestra el mensaje de despedida mostrado en la figura 5.21, concluyendo así la realización de la práctica “Aplicación de las ecuaciones diferenciales en circuitos eléctricos”.

6.3 Consumo de recursos durante ejecución

Para asegurar el correcto funcionamiento de la *app* en los dispositivos objetivo, se realizó la prueba de ejecución descrita en el apartado 6.2, utilizando la herramienta *Profiler* de Unity. Esta herramienta es un monitor que permite observar en tiempo

real el uso de recursos del dispositivo móvil donde se encuentre en ejecución la *app*, así como la tasa de refresco de la pantalla, medido en cuadros por segundo o FPS por sus siglas en inglés (*Frames Per Second*).

Se adquirieron datos en cuatro escenas representativas. Las gráficas observadas en la parte superior de las imágenes del monitor de recursos indican los FPS a los que se ejecuta la *app*, las gráficas intermedias muestran el consumo de memoria RAM y el texto en la parte inferior es un detalle del tiempo que tardan en ejecutarse los diversos procesos de la *app*, ordenados jerárquicamente y de mayor a menor tiempo de ejecución.

- Pantalla “Instrucciones”. Esta escena sirvió como referencia para adquirir datos del uso general de la *app* en las pantallas de presentación de información. En la figura 6.13 se observa que la *app* actualizó la pantalla cada 33.12 ms, correspondientes a aproximadamente 30 FPS, con un uso total de 12.5 MB de memoria RAM. En la parte inferior se puede observar que 4.15 ms fueron utilizados en la instrucción “*Camera.Render*”, la cual es encargada del despliegue del aspecto gráfico de la interfaz de usuario de la *app*, y la mayoría del tiempo restante (20.76 ms) en la instrucción “*WaitForTargetFPS*”, la cual tiene únicamente la función de esperar para que la actualización de la pantalla coincida con con una tasa de refresco que Unity considera óptima, que en este caso fueron 30 FPS.

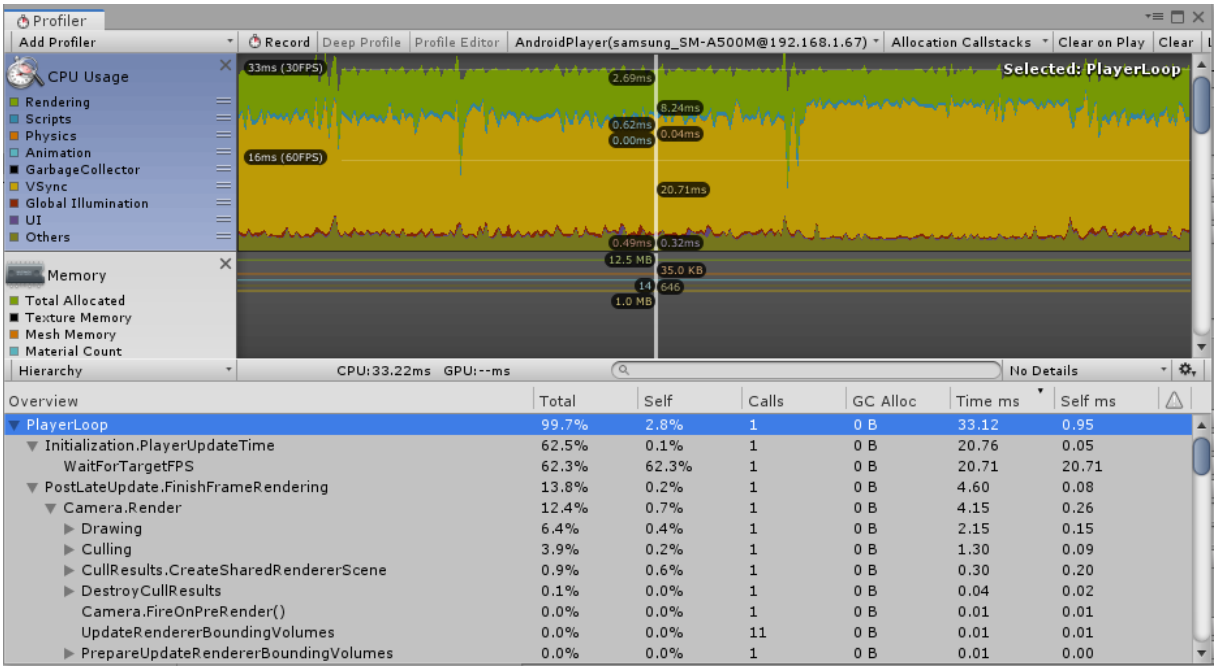


Figura 6.13 Monitor de recursos durante la escena “Instrucciones”.

- Pantalla “Selección de componentes”. Los datos de esta escena permitieron conocer el consumo de recursos del dispositivo cuando realiza el despliegue y animación de modelos tridimensionales sin hacer uso de la cámara. En la figura 6.14 se observa que al dispositivo le tomó 36.20 ms procesar un cuadro de la imagen, por lo que esta pantalla se actualizó a aproximadamente 28 FPS. La mayoría de este tiempo (23.3 ms) corresponde al proceso “*Camera.Render*”, el cual es llamado cuando se genera y actualiza un elemento gráfico; en este caso, la animación de giro de los componentes que puede seleccionar el usuario. En esta pantalla, la *app* consumió 20.6 MB de memoria RAM.

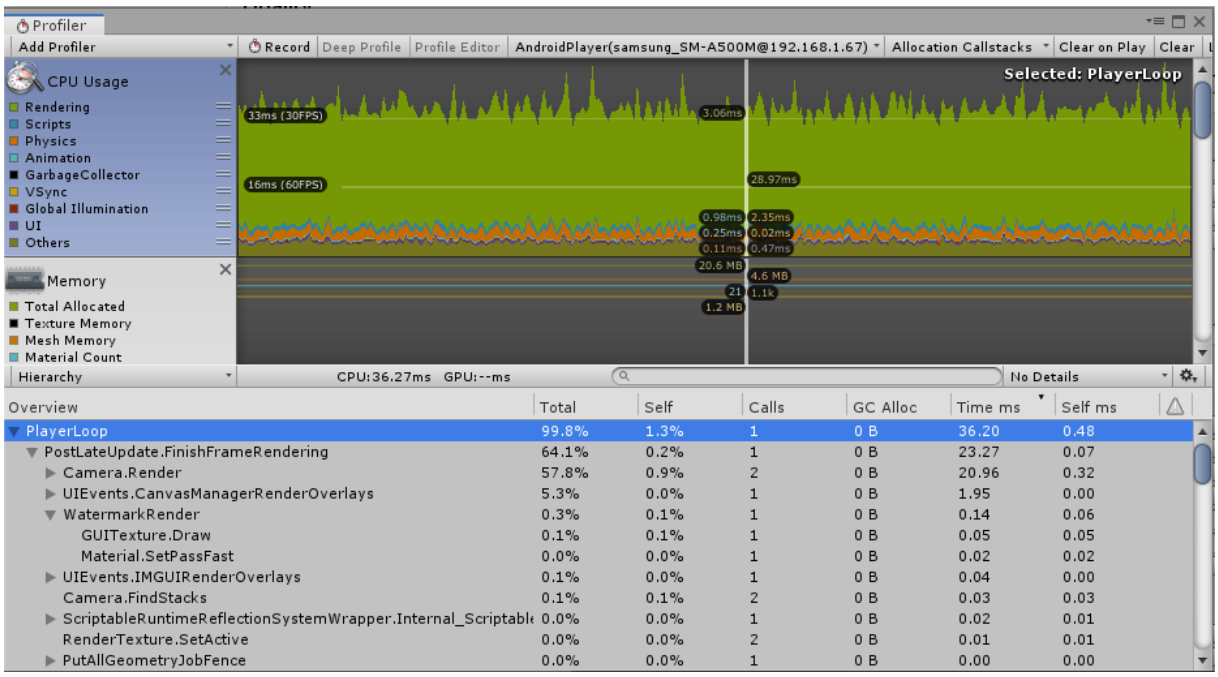


Figura 6.14 Monitor de recursos durante la escena “Selección de componentes”.

- Pantalla “Alambrado del circuito en RA”. En esta escena se aplica la tecnología de RA, por lo que se busca conocer el desempeño y consumo de recursos del dispositivo al ejecutarla. En las figuras 6.15 y 6.16 se observa el consumo de recursos de esta pantalla durante la actualización de dos cuadros de la pantalla, los cuales le tomaron al dispositivo 51.19 ms y 27.92 ms para actualizar, lo que equivale a 19.6 y 35.8 FPS, respectivamente. Las gráficas superiores de ambas imágenes muestran que esta variación se debe a un proceso que el monitor de recursos clasifica como “*Other*”, que corresponde al *Tracking System* realizado por Vuforia para la ubicación y el seguimiento de los marcadores, el cual consume más recursos en las actualizaciones de pantalla en las que realiza el reconocimiento de los

marcadores (figura 6.15), disminuyendo su consumo en las actualizaciones en las que sólo se despliegan elementos tridimensionales (figura 6.16). Para determinar una tasa de refresco en esta pantalla, se promedió el tiempo de actualización de ambos casos, resultando que la *app* se ejecutó a aproximadamente 25 FPS en esta escena, con un consumo de memoria RAM de 27.1 MB.

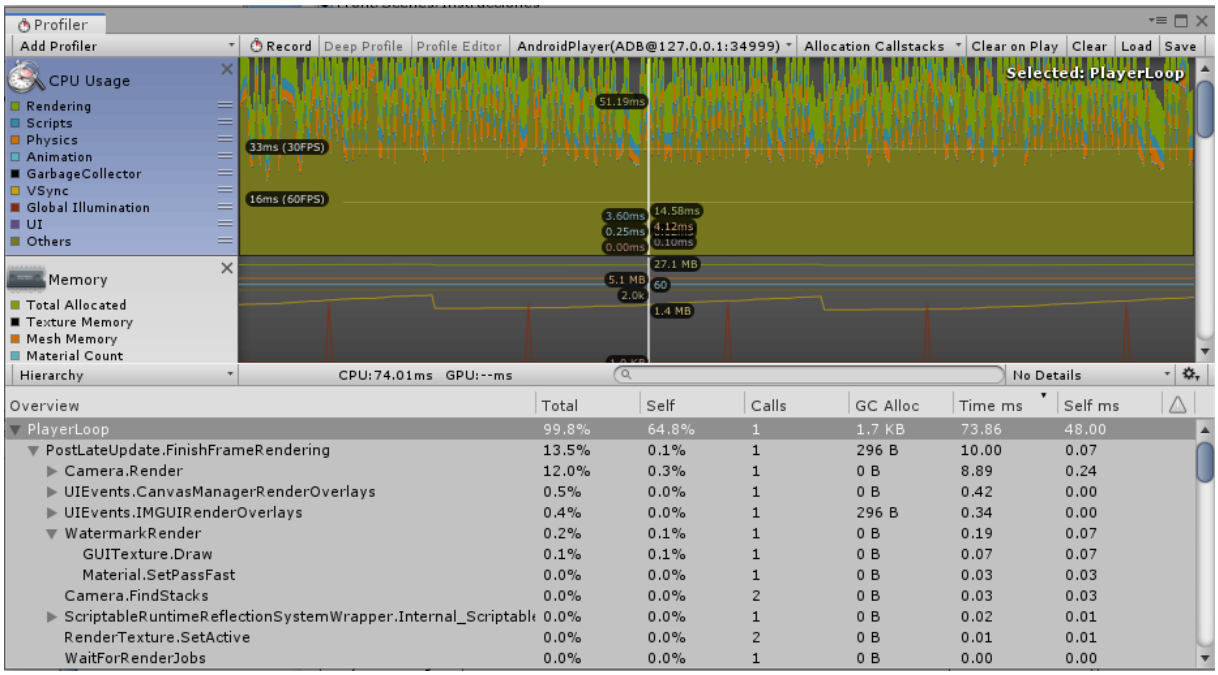


Figura 6.15 Monitor de recursos durante la escena "Alambrado del circuito en RA" durante el mayor consumo de recursos.

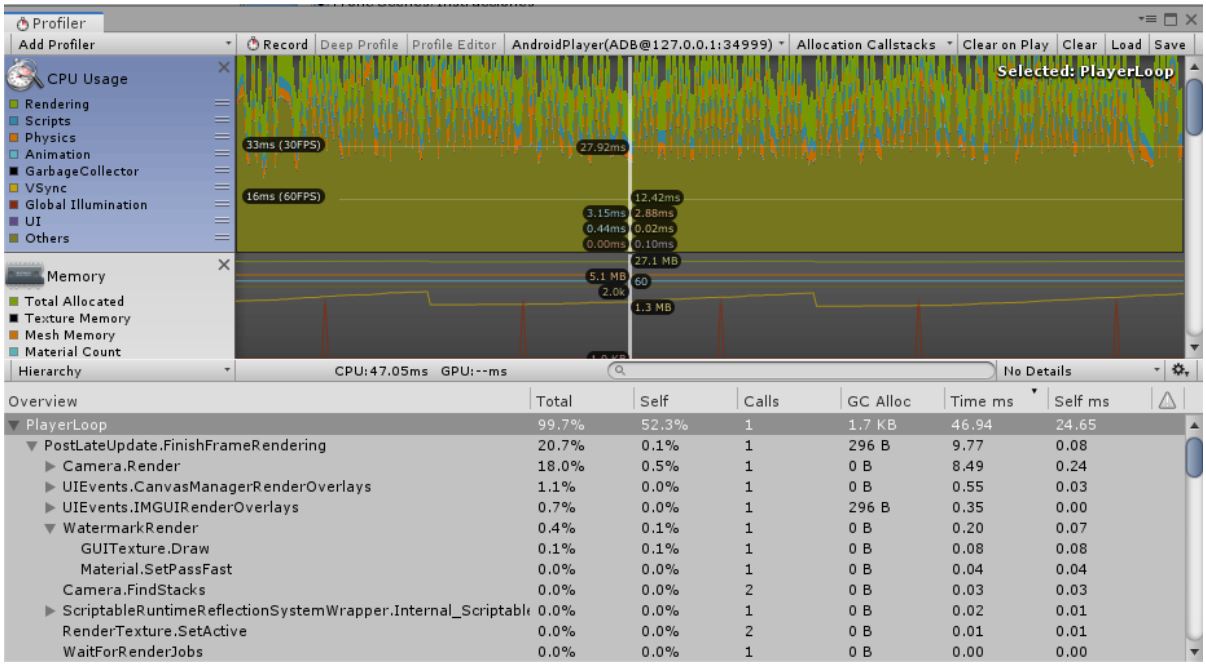


Figura 6.16 Monitor de recursos durante la escena "Alambrado del circuito en RA" durante el menor consumo de recursos.

- Pantalla "Osciloscopio virtual". En esta escena se resuelve la ecuación diferencial correspondiente al circuito alambrado por el usuario; el consumo de datos durante su ejecución permitió verificar que la ejecución de los métodos implementados no exijan al dispositivo de forma excesiva. En la figura 6.17 se observa que la *app* necesitó 41.77 ms para actualizar la pantalla, por lo que se ejecutó a una tasa de refresco de aproximadamente 24 FPS, con un consumo de memoria RAM de 35 MB. La mayoría de su consumo se debió a la tarea "*Device.Present*", encargada también del aspecto gráfico de la interfaz de usuario.

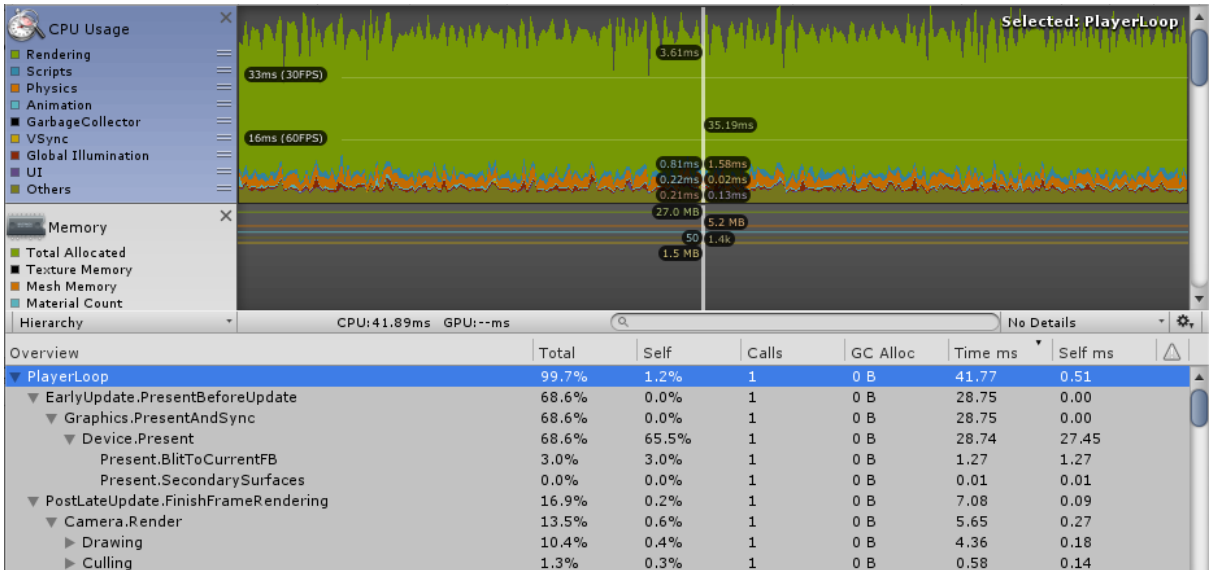


Figura 6.17 Monitor de recursos durante la escena "Osciloscopio virtual".

Los datos presentados se monitorearon con el fin de prevenir el consumo desmedido de recursos al ejecutar la *app*, ya que utilizar gran cantidad de memoria RAM afecta directamente el rendimiento del dispositivo tras provocar que aplicaciones en segundo plano necesarias para el correcto funcionamiento, no se ejecuten correctamente, por lo que tras analizar los datos mostrados en el monitor de recursos de acuerdo a las escenas más representativas, se pudo asegurar que el consumo de memoria de la *app* no provocará inestabilidad en el dispositivo ejecutante.

De acuerdo al artículo "*Animation principles for UX and UI designers*" [53], el ojo humano requiere que una aplicación digital de cualquier tipo tenga una tasa de refresco mínima de 24 FPS para percibir la sucesión de imágenes en una pantalla como un movimiento fluido. En la prueba de ejecución realizada, se observó que la *app RACE* trabaja con una tasa de refresco promedio de todas sus escenas de 27.4 FPS, por lo que, en dispositivos con el mínimo de sistema operativo compatible, tiene un rendimiento suficiente para dar una buena experiencia al usuario. De esta forma se garantiza el funcionamiento adecuado en la plataforma mínima compatible, asegurando también que en dispositivos con versiones superiores del sistema operativo, la *app* tendrá un mejor rendimiento.

Finalmente, con la obtención del archivo de instalación de la *app*, así como la realización de la prueba de ejecución y verificación de consumo de recursos, se concluye un producto listo para la implementación de pruebas con usuarios objetivo.

7 ANÁLISIS DE RESULTADOS

7.1 Prueba de facilidad de uso

Para verificar que la *app* RACE permite a los usuarios realizar la práctica "Aplicación de las ecuaciones diferenciales en circuitos eléctricos" de manera satisfactoria, se implementó una prueba de facilidad de uso. De acuerdo a la norma ISO 9241-11 [54], la facilidad de uso de un sistema se puede medir a partir de:

> "*El grado en el que un sistema, producto o servicio puede ser utilizado por usuarios específicos para lograr objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso específico*".

Este concepto permite saber si un usuario es capaz de manipular un sistema apropiadamente desde el primer uso, además de cumplir objetivos planteados de manera precisa y en su totalidad, lo que finalmente permite al usuario cumplir sus necesidades y expectativas. De acuerdo a la investigación de Jakob Nielsen [55], realizar una prueba de facilidad de uso con 5 usuarios es suficiente para detectar los problemas que pueda tener un sistema de forma general. Las pruebas posteriores a esta cifra aportan menor grado de información al respecto, ya que los problemas encontrados en las primeras pruebas serán nuevamente encontrados por usuarios posteriores, por lo que es improbable obtener datos relevantes nuevos.

La prueba de facilidad de uso de la *app* RACE fue aplicada a alumnos de la DCB, quienes fueron considerados como los usuarios objetivo en los alcances del proyecto en el capítulo 3.2, así como a los profesores de la misma división, debido a que son quienes utilizarán la *app* como una herramienta de apoyo y pueden validar si se cumplen los objetivos educativos de la misma.

La prueba consiste en que el usuario ejecute la *app* en su totalidad, cumpliendo con los puntos listados a continuación:

- Instalación de la *app*
- Navegación entre los distintos menús, incluyendo el acceso a información e instrucciones
- Manipulación del marcador cilíndrico con el dedo índice
- Selección del marcador plano
- Realización de la práctica (descrita en el capítulo 6.2).

Se estableció que las pruebas con ambos grupos fueran realizadas a través de sesiones de video llamadas en línea. La prueba realizada con el grupo conformado por profesores constó de dos partes. Primero, una demostración de la ejecución de la *app*, en la que se les mostraron y describieron todos los pasos necesarios para su correcto uso. La demostración se realizó bajo las siguientes condiciones:

- Se les mostró la pantalla de un dispositivo móvil en el que se ejecutaba en tiempo real la *app* RACE
- Se les dio una explicación detallada de las acciones mientras estas se llevaban a cabo
- Durante la sesión, los profesores externaron sus dudas respecto al uso general de la *app*
- Al finalizar la sesión, los profesores realizaron comentarios acerca del diseño y funcionamiento de la *app*.

Posteriormente, los profesores hicieron uso de la *app* por su cuenta, siguiendo las instrucciones dentro de la misma, con el fin de comprobar si eran capaces de llevar a cabo las actividades de la práctica sin inconvenientes.

Para la prueba aplicada al grupo de alumnos, se buscó simular una situación de uso real de la *app*, por lo que el archivo de instalación y el marcador recortable fueron proporcionados a través de su profesora. La prueba se realizó bajo las siguientes condiciones:

- Los alumnos ejecutaron la *app* por su cuenta
- Durante la sesión, los alumnos sólo recibieron ayuda en caso de tener dudas que les impidiera continuar con las actividades.

Con el fin de valorar los aspectos relacionados a la facilidad de uso de la *app* RACE al finalizar las sesiones con los grupos establecidos, se aplicaron encuestas basadas en la Escala de Uso para Sistemas (SUS, por sus siglas en inglés *System*

Usability Scale) [56]. Esta escala está conformada por 10 enunciados a los que el usuario muestra estar de acuerdo o desacuerdo con la afirmación expresada.

Los enunciados propuestos en la SUS están estructurados para cubrir aspectos del uso de un sistema en general, tales como la necesidad de asistencia o entrenamiento para su uso, así como la complejidad del mismo. Las oraciones impares (1, 3, 5, 7 y 9) son afirmaciones positivas del sistema a evaluar, mientras que las pares (2 ,4, 6, 8 y 10) son afirmaciones negativas. La característica de incluir afirmaciones tanto positivas como negativas del sistema, así como presentarlas de forma intercalada, tiene como objetivo verificar que no existan contradicciones en las respuestas del usuario, además de evitar que se presente un sesgo de aquiescencia, en el que se tiende a contestar una pregunta de manera afirmativa sin importar su contenido. Por esta razón, al hacer cualquier adaptación de esta encuesta, se debe tener cuidado en no cambiar la intención y el orden de las preguntas.

Para construir la encuesta que se aplicó a los usuarios, se modificaron las preguntas propuestas en la SUS para que hicieran referencia a la *app*. Además de esto, se incluyeron preguntas adicionales para obtener información referente a la condición académica del alumno o profesor, su opinión respecto al cumplimiento de los objetivos y su preferencia sobre la aplicación de la práctica a través de la *app*. Las encuestas creadas para alumnos y profesores se pueden consultar en el apéndice M.

7.2 Encuesta de facilidad de uso

Las pruebas se llevaron a cabo de acuerdo a las condiciones descritas en el apartado 7.1. Debido a situaciones externas a este trabajo, durante el desarrollo de las pruebas, se contó únicamente con el apoyo de 3 profesores y 4 alumnos para las mismas.

El puntaje que los usuarios dan a cada pregunta basada en la SUS está representado de la siguiente manera:

- “Fuertemente en desacuerdo” con el número 1
- “En desacuerdo” con el número 2
- “Ni de acuerdo, ni en desacuerdo” con el número 3
- “De acuerdo” con el número 4
- “Fuertemente de acuerdo” con el número 5.

El puntaje correspondiente a cada respuesta dada por alumnos y profesores se muestra en las tablas 7.1 y 7.2, respectivamente.

Tabla 7.1 Puntaje a respuestas de alumnos a la encuesta basada en la SUS.

Alumno	**Pregunta**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	5	2	4	4	4	2	3	2	4	2
2	4	2	4	1	5	1	5	2	5	2
3	5	2	4	1	4	1	4	2	4	2
4	4	4	3	2	5	1	4	3	3	1

Tabla 7.2 Puntaje a respuestas de profesores a la encuesta basada en la SUS.

Profesor	**Pregunta**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	4	2	4	1	2	1	5	1	5	2
2	5	2	4	1	5	1	2	3	5	3
3	5	1	4	1	5	1	3	1	5	1

Las preguntas adicionales en la encuesta para profesores estuvieron enfocadas en saber si, como herramienta docente, la *app* RACE permite cumplir de manera apropiada con los objetivos de la práctica. Por otra parte, las preguntas hechas a los alumnos se enfocaron en saber si consideran que la tecnología de RA les permite reforzar los conceptos prácticos aprendidos. Se incluyeron además preguntas abiertas, con el fin de conocer la razón de la preferencia del formato de aplicación de la práctica, así como cualquier comentario u observación adicional.

7.3 Análisis de los resultados de las encuestas

Dado que las oraciones de una encuesta basada en la SUS son afirmaciones tanto positivas como negativas y tienen un orden intercalado, el puntaje final que da un usuario respecto a la facilidad de uso de un sistema a través de dicha encuesta se debe calcular de forma específica. Para las preguntas 1, 3, 5, 7 y 9 (afirmaciones positivas), el puntaje se obtiene tomando el valor de la respuesta y restando 1. Para las preguntas 2, 4, 6, 8 y 10 (afirmaciones negativas), el puntaje es 5 menos el valor de la respuesta. Finalmente, se suman todos los puntajes y se multiplican por 2.5 para obtener un puntaje final, el cual estará en un rango entre 0 y 100. Este cálculo se determina de acuerdo a la ecuación 7.1:

$$P = 2.5[(a_1 - 1) + (5 - a_2) + (a_3 - 1) + (5 - a_4) + (a_5 - 1) + (5 - a_6) + (a_7 - 1) + (5 - a_8) + (a_9 - 1) + (5 - a_{10})] \quad (7.1)$$

Siendo P el puntaje que el usuario da a la facilidad de uso y a_n la respuesta del usuario a la afirmación n de la encuesta.

De forma demostrativa, para calcular el puntaje con el que el profesor número 1 en la tabla 7.2 califica la facilidad de uso de la *app* RACE, se realiza el proceso descrito anteriormente, sustituyendo las respuestas de la primera fila de la tabla 7.2 en la ecuación 7.1 obteniendo el resultado expresado en la ecuación 7.2:

$$P = 2.5[(4 - 1) + (5 - 2) + (4 - 1) + (5 - 1) + (2 - 1) + (5 - 1) + (5 - 1) + (5 - 1) + (5 - 1) + (5 - 2)] = 82.5 \quad (7.2)$$

Realizando los respectivos cálculos, en las tablas 7.3 y 7.4 se muestran los resultados obtenidos de las encuestas aplicadas a alumnos y profesores, respectivamente, además del puntaje promedio que los usuarios dan a la facilidad de uso de la *app*.

Tabla 7.3 Resultado de las encuestas aplicadas a alumnos.

Alumno	1	2	3	4	Promedio
Puntos	70	87.5	82.5	70	77.5

Tabla 7.4 Resultado de las encuestas aplicadas a profesores.

Profesor	1	2	3	Promedio
Puntos	82.5	77.5	92.5	84.16

Los alumnos otorgaron a la *app* RACE un puntaje promedio de 77.5, mientras que los profesores de 84.16. Un sistema con un puntaje final a partir de los 68 puntos puede considerarse sobre el promedio aceptable respecto a su facilidad de uso [57], por lo que la *app* RACE se valora como un sistema que puede ser ejecutado y manipulado para un usuario promedio sin tener inconvenientes en su uso.

Adicional a los puntajes obtenidos, en la figura 7.1 se muestran los resultados de las preguntas realizadas a los alumnos, donde se observa que el 100% de los alumnos encuestados opinaron que la *app* RACE les permitió cumplir los objetivos de la práctica de manera satisfactoria (figura 7.1b). Este mismo porcentaje de alumnos indican que habían cursado la materia Ecuaciones Diferenciales con anterioridad (figura 7.1a), por lo que se puede decir que poseen conocimientos previos de la materia y dieron su opinión respecto al cumplimiento de los objetivos conociendo las actividades y los resultados esperados de las mismas.

De igual forma, en la figura 7.1c se muestra que el 100% de los alumnos consideraron que la tecnología de RA les ayudó a comprender y desarrollar las actividades de la práctica, reafirmando su opinión con comentarios como "[...] *Al principio cuesta un poco de trabajo adaptarse a la interfaz, pero una vez que practicas funciona bastante bien y da el efecto de tener las cosas físicamente*". La aceptación del uso de la tecnología de RA se vio reflejada en la opinión de la totalidad de los alumnos quienes, como se muestra en la figura 7.1d, prefirieron realizar la práctica en un formato mixto, ya que consideraron la *app* como una herramienta útil para "[...] *complementar, corroborar y apoyarnos de manera didáctica*", sin embargo, también consideraron importante la experiencia de realizar la práctica de manera tradicional.

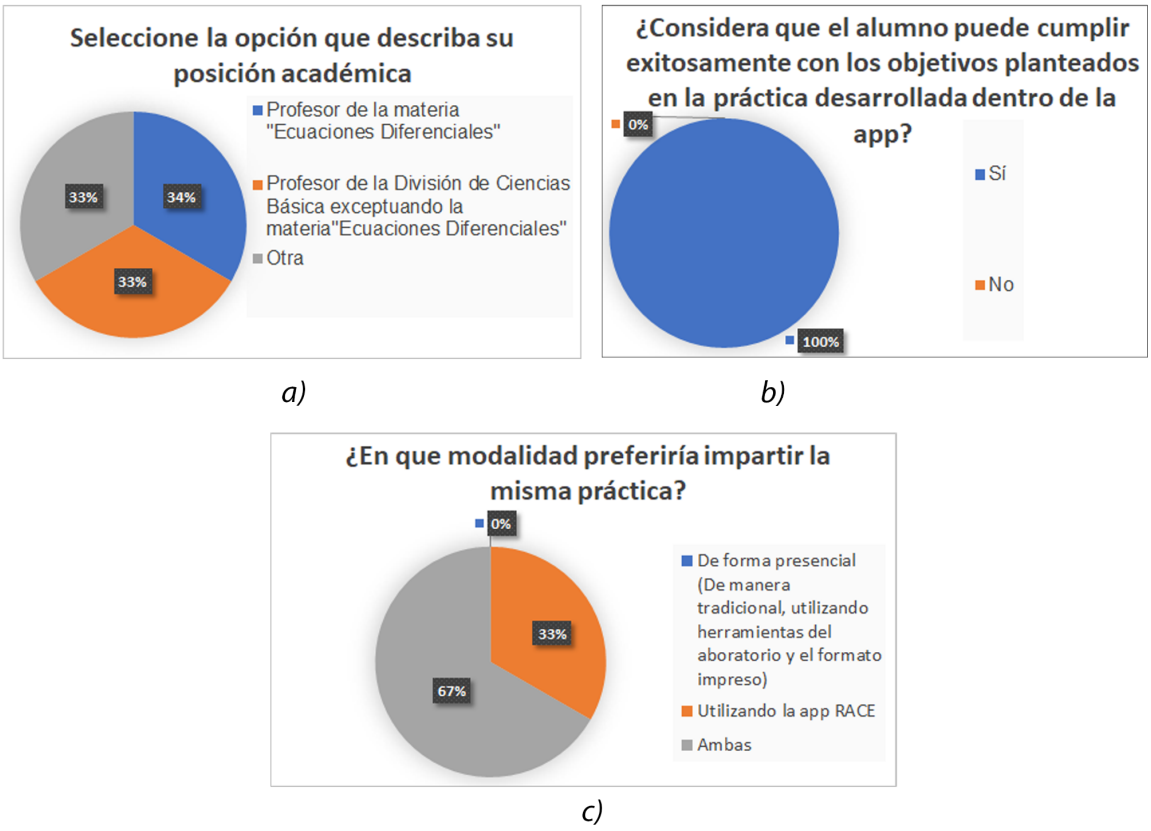
Figura 7.1 Resultado de las preguntas adicionales a alumnos: a) situación académica; b) objetivos planteados en la práctica desarrollada ; c) tecnología de realidad aumentada; d) modalidad de preferencia para realizar la misma práctica.

Dentro de los comentarios adicionales, la mitad de los alumnos encuestados mencionó que, al inicio de la realización de la práctica, tuvieron problemas relacionados con el enfoque y la iluminación para definir el marcador plano de despliegue, pero pudieron resolverlos a los pocos minutos de uso. Con base en todos los comentarios expresados por los alumnos, podemos concluir que el uso de la herramienta de RA les resultó útil e interesante, y a pesar de presentar algunas complicaciones durante su uso, no fueron lo suficientemente significativas para cambiar su opinión respecto a la utilidad y aceptación de la *app* RACE.

Respecto a las preguntas adicionales hechas a los profesores, en la figura 7.2a se muestra que el 33% son profesores de la materia Ecuaciones Diferenciales, 33% son profesores de la DCB pero no de la materia, y 33% tiene otra situación académica. El 100% de los profesores encuestados consideraron que la *app* puede permitir a los alumnos cumplir con los objetivos de la práctica, como se muestra en la figura 7.2b. Sin embargo, mientras que el porcentaje correspondiente a profesores de la DCB y de otra situación académica opinaron su preferencia por

que la práctica se aplique en un formato mixto, el 33% correspondiente a profesores de la DCB optaron por la posibilidad de aplicar la práctica utilizando únicamente la *app* RACE "*Por una cuestión de insuficiencia de laboratorio, y que puede realizarse fuera de la hora de clase*"; este mismo 33% fue quien dio la menor calificación en la prueba de usabilidad a la *app*, y comentó que tuvo problemas con la iluminación y enfoque de la cámara para la definición del marcador plano.

Estos resultados nos indican que todos los profesores encuestados expresaron un alto grado de aceptación para el uso de la *app* RACE, considerando que desde un punto de vista docente es útil para llevar a cabo los objetivos de aprendizaje de la materia; en particular, los profesores encuestados de la materia Ecuaciones Diferenciales, a pesar de haber tenido problemas con su uso, la vieron como una alternativa viable para la aplicación de la práctica, basándose en su experiencia de la materia y de los conocimientos que se espera que adquieran los alumnos al realizar esta práctica.



a) *b)*

c)

Figura 7.2 Resultado de las preguntas adicionales a profesores: a) posición académica; b) cumplimiento de los objetivos planteados en la práctica desarrollada; c) modalidad de preferencia para impartir la misma práctica.

Tanto alumnos como profesores, en los comentarios adicionales, hicieron la sugerencia de incluir una sección de preguntas frecuentes, en la que se describan posibles problemas de uso como la correcta forma de enfoque de cámara e iluminación de la zona de trabajo, así como la forma de solucionarlo.

De manera general, todos los usuarios encuestados coincidieron en que la *app* RACE les pareció didáctica e interesante, resultando útil para el cumplimiento de los objetivos de la práctica “Aplicación de las ecuaciones diferenciales en circuitos eléctricos”. Si bien es cierto que el desarrollo de la *app* tuvo algunos puntos en los que el diseño podría mejorar, esta prueba de facilidad de uso ha permitido identificarlos para poder mejorarlos en un trabajo a futuro.

8 CONCLUSIONES Y TRABAJO A FUTURO

Como resultado del proceso de diseño, se logró programar la *app* RACE, que implementa el desarrollo de la práctica “Aplicación de las ecuaciones diferenciales en circuitos eléctricos” de la materia Ecuaciones Diferenciales de la Facultad de Ingeniería, a través del uso de la tecnología de RA. De acuerdo a los requerimientos establecidos, la *app* es compatible con los dispositivos más utilizados por los alumnos de la FI, e incorpora todas las actividades necesarias para llevar a cabo los objetivos de la práctica.

Con este trabajo se comprueba la hipótesis planteada, en el que se expone el apoyo que ofrece la tecnología de RA como recurso docente en el reforzamiento de conceptos matemáticos que presentan las prácticas de laboratorio, dotando a éstas de un mayor atractivo, al permitir realizar las actividades sin disponer de los componentes físicos.

La *app* construida, muestra y guía al usuario a través de las actividades necesarias para cumplir con los objetivos de la práctica, permitiendo la selección y visualización de los componentes requeridos así como su manipulación con el uso del motor de interacción desarrollado con marcadores para la construcción virtual de los circuitos solicitados, así como la posibilidad de observar su simulación de funcionamiento en tiempo real, promoviendo el aprendizaje por medio de la libertad en la modificación de la construcción de los mismos.

Con base en los resultados de las pruebas de facilidad de uso realizadas a alumnos y profesores, se concluye que la *app* logra cumplir con los objetivos académicos que busca el desarrollo de esta práctica de laboratorio, además de ser aceptada con una facilidad de uso superior al promedio, teniendo un puntaje de 77.5, otorgado por los alumnos quienes son los usuarios objetivo, lo que demuestra que

como sistema de aplicación móvil, es una herramienta muy buena en su ejecución y no representa dificultad para su empleo.

Adicional a la facilidad de uso, tanto alumnos como profesores expresaron su aprobación hacia la tecnología de RA, ya que ofrece una gran utilidad como recurso docente en el reforzamiento de los conceptos que presentan las prácticas de laboratorio, permitiendo realizarlas sin disponer de los componentes físicos necesarios al realizarlas de forma tradicional y sin la necesidad de acudir a un laboratorio, solventando la falta de espacios actuales dentro de la FI.

En general, los usuarios no tuvieron complicaciones importantes durante el uso de la *app*, ya que todos fueron capaces de utilizarla con facilidad tras pocos minutos de empleo. Sin embargo, a partir de los resultados de las pruebas se consideran mejoras a futuro, como la implementación de una sección de preguntas frecuentes, en la que se muestren algunas complicaciones comunes que los usuarios pudieran encontrarse al comenzar a utilizar la *app*. Se propone además como una posible mejora a largo plazo la modificación del mecanismo de interacción, para que funcione con sistemas más sofisticados de rastreo y descartar el uso de marcadores, tal como la detección de planos o *handtracking*, sistemas que no se implementaron en esta versión de la *app* debido a las limitaciones en la tecnología de los dispositivos móviles más usados por los usuarios objetivo.

Dados los resultados positivos del uso de la *app* RACE en la realización de una práctica a través del uso de RA, se considera como un trabajo a largo plazo la incorporación de una mayor variedad de prácticas de laboratorio para diversas materias de la DCB, haciendo uso de esta misma tecnología.

REFERENCIAS

[1] “Plan de Desarrollo de la División de Ciencias Básicas”, UNAM, Facultad de Ingeniería, 2015 [En línea]. Disponible en: http://dcb.ingenieria.unam.mx/wp-content/uploads/Documentos/PDD_DCB.pdf. [Consultado: 10-nov-2019].

[2] E. Salazar, “Reporte de la realización de prácticas de Laboratorio para evidenciar conceptos de Ecuaciones Diferenciales Coordinación de Ciencias Aplicadas”, UNAM, Facultad de Ingeniería, 2020.

[3] W. Ávila, “Hacia una reflexión histórica de las TIC”, *Hallazgos*, vol. 10, núm. 19, pp. 213–233, 2013.

[4] E. Smeets, “Does ICT contribute to powerful learning environments in primary education?”, *Computers & Education*, vol. 44, núm. 3, pp. 343–355, 2005.

[5] J. Gómez, *Universitic 2016: análisis de las TIC en las Universidades Españolas*. Crue Universidades Españolas, 2016.

[6] J. L. Ponce, “Estado actual de las tecnologías de la información y las comunicaciones en las instituciones de educación superior en México: estudio ejecutivo 2016”, *México, DF., ANUIES*, 2016.

[7] “*Hype Cycle for Education”, 2018*, Gartner. [En línea]. Disponible en: http://www.gartner.com/en/documents/3882872/hype-cycle-for-education-2018. [Consultado: 18-nov-2019].

[8] “*Hype Cycle Research Methodology”*, Gartner. [En línea]. Disponible en: https://www.gartner.com/en/research/methodologies/gartner-hype-cycle. [Consultado: 18-nov-2019].

[9] T. Mazuryk y M. Gervautz, “Virtual Reality History, Applications, Technology and Future History”, *Virtual Real.*, pp. 58–60, 1997.

[10] I. E. Sutherland, “The ultimate display”, *Multimedia: From Wagner to virtual reality*, vol. 1, pp. 506–509, 1965.

[11] D. Nincarean et ál., “Mobile Augmented Reality: the potential for education”, *Procedia-social and behavioral sciences*, vol. 103, núm. 0, pp. 657–664, 2013.

[12] J. Cabero Almenara y B. Fernández Robles, “Las tecnologías digitales emergentes entran en la Universidad: RA y RV”, *RIED. Revista Iberoamericana de Educación a Distancia*, vol. 21, núm. 2, pp. 119–138, 2018.

[13] "*AR home"*, Google AR & VR. [En línea]. Disponible en: https://arvr.google.com/ar/. [Consultado: 20-nov- 2019].
[14] "Reality Technologies", *What is Mixed Reality (MR)? The Ultimate Guide to Mixed Reality (MR)*, Reality Technologies. [En línea]. Disponible en: https://www.realitytechnologies.com/mixed-reality/. [Consultado: 3-ene-2020].
[15] "*Glass – Glass"*, Glass. [En línea]. Disponible en: https://www.google.com/glass/start/. [Consultado: 15-ene-2020].
[16] "*HoloDesk: Direct 3D Interactions with a Situated See-Through Display - Microsoft Research"*, Microsoft Research. [En línea]. Disponible en: https://www.microsoft.com/en-us/research/project/holodesk-direct-3d-interactions-with-a-situated-see-through-display/. [Consultado: 15-feb-2020].
[17] P. Wilson, "Business & Technology Surveillance An Introduction to Industrial Augmented and Virtual Reality", Analytics, Resiliency and Reliability (ARR) Work Group, 2019 [En línea]. Disponible en: https://www.cooperative.com/programs-services/bts/Documents/TechSurveillance/Surveillance-AR-VR-Technology-Overview-June-2019.pdf. [Consultado: 08-mar-2020].
[18] Y. Chen et ál., "An overview of augmented reality technology", *J. Phys. Conf. Ser.*, vol. 1237, núm. 2, p. 022082, jun. 2019.
[19] M. Figueiredo et ál., "Augmented Reality tools and techniques for developing interactive materials for mobile-learning", *Journal Recent Advances Educational Technologies and Methodologies*, vol. 395, núm. 7, pp. 63–72, 2014.
[20] A. Arzumanyan, "AR Solar System" [En línea]. Disponible en: https://play.google.com/store/apps/details?id=com.ar.solar&hl=es_MX. [Consultado: 21-jun-2020].
[21] K. Č. Pucihar y P. Coulton, "Exploring the Evolution of Mobile Augmented Reality for Future Entertainment Systems", *Comput. Entertain.*, vol. 11, núm. 2, pp. 1–16, ene. 2015.
[22] "*Vuforia Object Scanner"*. [En línea]. Disponible en: https://library.vuforia.com/articles/Training/Vuforia-Object-Scanner-Users-Guide. [Consultado: 21-feb-2020].
[23] "*CVGesture"*. Github[En línea]. Disponible enhttps://github.com/OAID/CVGesture. [Consultado: 21-feb- 2020].
[24] P. Cipresso et ál., "The Past, Present, and Future of Virtual and Augmented Reality Research: A Network and Cluster Analysis of the Literature", *Front. Psychol.*, vol. 9, p. 2086, nov. 2018.
[25] "Vuforia Core Samples". [En línea]. Disponible en: https://assetstore.unity.com/packages/templates/packs/vuforia-core-samples-99026 [Consultado: 21-feb- 2020].
[26] J. Teji, (2019, noviembre.11), "*Is There A New Samsung AR Headset In The Making?", TEJ*, TEJ. [En línea]. Disponible en: https://tej.ie/new-samsung-ar-headset-in-the-making/. [Consultado: 4-mar-2020].
[27] G. A. Gallego Trujillo, "Modelo para el análisis de aplicaciones visuales educativas en Realidad Aumentada desde la perspectiva de la semiótica

visual". [En línea]. Disponible en: http://openaccess.uoc.edu/webapps/o2/handle/10609/98606.
[28] N. I. Horizon, "Edición Superior de Educación", *Austin, Texas: The*, 2016.
[29] M. Brown et ál., "2020 Educause Horizon Report Teaching and Learning Edition", EDUCAUSE, 2020[En línea]. Disponible en: https://www.learntechlib.org/p/215670/.
[30] M. Akçayır y G. Akçayır, "Advantages and challenges associated with augmented reality for education: A systematic review of the literature", *Educational Research Review*, vol. 20, pp. 1–11, feb. 2017.
[31] J. Bacca-Acosta et ál., "Augmented Reality Trends in Education: A Systematic Review of Research and Applications", vol. 17, núm. 4, pp. 133–149, oct. 2014.
[32] A. Shirazi y A. H. Behzadan, "Design and assessment of a mobile augmented reality-based information delivery tool for construction and civil engineering curriculum", *J. Prof. Issues Eng. Educ. Pract.*, vol. 141, núm. 3, p. 8, 2015.
[33] M. Akçayır et ál., "Augmented reality in science laboratories: The effects of augmented reality on university students' laboratory skills and attitudes toward science laboratories", *Computers in Human Behavior*, vol. 57. pp. 334–342, 2016 [En línea]. Disponible en: http://dx.doi.org/10.1016/j.chb.2015.12.054.
[34] C. Avilés-Cruz y J. Villegas-Cortez, "A smartphone-based augmented reality system for university students for learning digital electronics", *Comput. Appl. Eng. Educ.*, vol. 27, núm. 3, pp. 615–630, may 2019.
[35] R. Terpán y S. Rodríguez, "Sistema de realidad aumentada para fetoteca", Ingeniero Mecatrónico, UNAM, 2017.
[36] O. Ruíz, "Modelo computacional para calcular enlaces químicos utilizando realidad aumentada", Licenciado en Ciencias de la Computación, UNAM, 2018.
[37] G. A. J. Morales y A. A. A. Castellanos, "*Electricidad y magnetismo"*. Trillas, 1997.
[38] D. G. Zill, "*Ecuaciones diferenciales con valores en la frontera"*. Cengage Learning Editores S.A. de C.V., 2013.
[39] N. M. Morris, "Circuit Analysis", *Electrical Circuit Analysis and Design*. pp. 28–60, 1993 [En línea]. Disponible en: http://dx.doi.org/10.1007/978-1-349-22560-6_2.
[40] P. L. DeVries y R. P. Wolf, "A First Course in Computational Physics", *Computers in Physics*, vol. 8, núm. 2, pp. 178–179, mar. 1994.
[41] A. Robbins y W. C. Miller, "*Análisis de Circuitos: Teoría Y Práctica"*. Cengage Learning Latin America, 2008.
[42] Y. D. A. Balaguera, "Guía metodológica ágil, para el desarrollo de aplicaciones móviles 'AEGIS-MD'", *Revista De Investigaciones UNAD*, vol. 14, núm. 1, pp. 97–113, 2015.
[43] F. J. García et ál., "*Apuntes para la investigación en salud"*. Universidad Nacional Autónoma de México, 2014.
[44] "Tercer Informe de actividades 2017", UNAM, 2017-2018, feb. 2018 [En línea]. Disponible en: https://www.planeacion.unam.mx/informes/PDF/FI-2017-2018.pdf.

[45] M. R. Spiegel, "*Estadística"*. McGraw-Hill, 2009.
[46] "*More Game Development Opportunities with the Unity 2019 Update"*. [En línea]. Disponible en: https://www.innovecsgames.com/blog/unity-2019-update/. [Consultado: 13-nov-2019].
[47] "*Recommended Devices"*. [En línea]. Disponible en: https://library.vuforia.com/platform-support/vuforia-engine-recommended-devices.html. [Consultado: 20-nov-2019].
[48] R. Á. Chaurand et ál., "*Dimensiones antropométricas de población latinoamericana: México, Cuba, Colombia, Chile"*. 2001.
[49] "*Cómo saber tu talla de anillo"*. [En línea]. Disponible en: https://paperpile.com/shared/2UrUGc. [Consultado: 27-ene-2021].
[50] "*Vuforia Developer Portal"*. [En línea]. Disponible en: https://developer.vuforia.com/vui/develop/licenses. [Consultado:15-nov-2020].
[51] T. Schlatter y D. Levinson, "*Visual Usability: Principles and Practices for Designing Digital Applications"*. Newnes, 2013.
[52] Z. Barševska y O. Rakele, "*Color in UI Design*", Proceedings of the 61st International Scientific Conference of Daugavpils University, p. 79, Universidad de Daugavpils, 2019.
[53] V. Joyce, (2018, diciembre.13), "*Animation principles for UX and UI designers - UX Planet"*, UX Planet. [En línea]. Disponible en: https://uxplanet.org/animation-that-matters-adding-value-to-your-interface-65496fe4c182. [Consultado: 13-abr-2021].
[54] Technical Committee ISO/TC 159. Subcommittee SC y Ergonomics of human-system interaction, *ISO 9241-11:2018 "Ergonomics of Human-system Interaction: Egonomie de L'interaction Homme-système. Usability : definitions and concepts. Utilisabilité : définitions et concepts. Part 11. Partie 11"*. ISO, 2018.
[55] "*Why you only need to test with 5 users"*. [En línea]. Disponible en: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/. [Consultado: el 09-may-2021].
[56] J. Brooke, "SUS -- a quick and dirty usability scale", en *Usability Evaluation in Industry*, unknown, 1996, pp. 189–194.
[57] Assistant Secretary for Public Affairs, "System Usability Scale (SUS)", sep. 2013 [En línea]. Disponible en: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html [Consultado: 16-may-2021].
[58] C. Steven y C. Raymond, "Métodos numéricos para ingenieros", *Editorial McGraw-Hill, España*, 2003.

APÉNDICES

A Formato de encuesta realizada a alumnos de la Facultad de Ingeniería

Encuesta sobre dispositivos móviles utilizados en la Facultad de Ingeniería

Esta encuesta fue preparada con fines informativos sobre los dispositivos utilizados por el alumnado dentro de la Facultad de Ingeniería con fines docentes. La información obtenida por esta encuesta no será utilizada con otros fines más que los planteados. Dado que las condiciones del producto están fuera del control de la Facultad de Ingeniería, la UNAM no asume responsabilidad en conexión con el uso de esta información. Nada de lo aquí expresado será divulgado a externos.

*Obligatorio

1 ¿Actualmente cursas una o más materias de la División de Ciencias Básicas?*

Marca solo un óvalo.

- Sí
- No

2 ¿Qué dispositivo móvil utilizas más para tareas escolares? *

Marca solo un óvalo.

- Teléfono celular
- Tableta

3 Cuál es el modelo de tu dispositivo? (Ej: Samsung Galaxy A5) *

4 ¿Cual es su sistema operativo? *

Marca solo un óvalo.

- Android
- IOS
- Otro:

5 Por favor escribe la versión del sistema operativo de tu dispositivo (ej. Android 8.0) *

6 Si las prácticas de laboratorio de alguna materia pudieran realizarse con alguna de las siguientes tecnologías de realidad ¿Cuál preferirías utilizar? *

Marca solo un óvalo.

Realidad aumentada

Realidad virtual

Realidad mixta

7 Respecto a la pregunta anterior ¿por qué? *

_____

_____

_____

8 Las tecnologías de realidad necesitan de una o varias imágenes de referencia para ser utilizadas. ¿Cuál de los siguientes arreglos consideras más cómodo para que se te proporcione y puedas usar? *

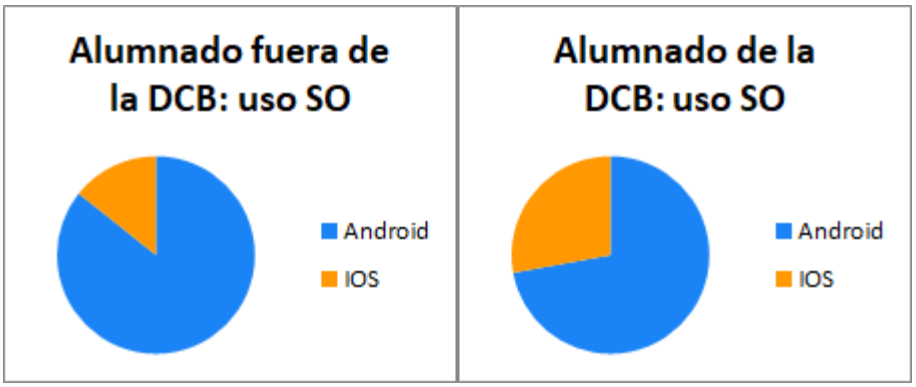
Marca solo un óvalo.

Varias imágenes marcadores individuales (Realidad aumentada)

Un solo arreglo de múltiples imágenes en forma de cubo (Realidad aumentada)

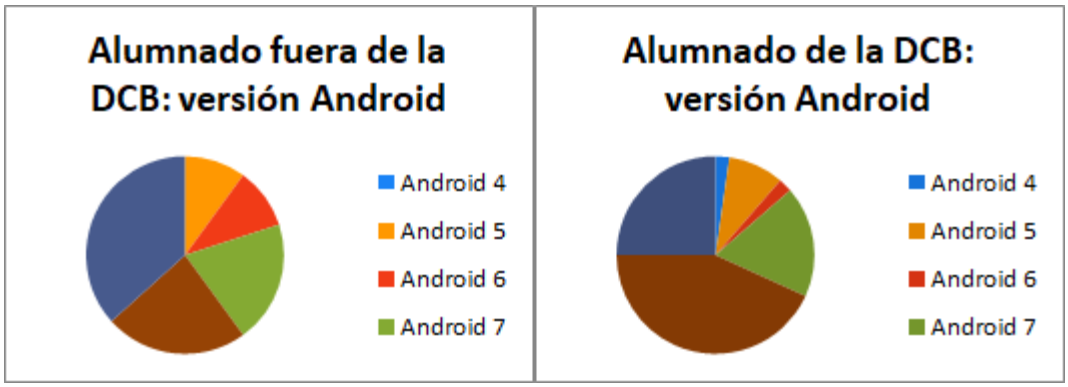
Planos de cardboard (Realidad virtual, realidad mixta)

B Gráficas comparativas



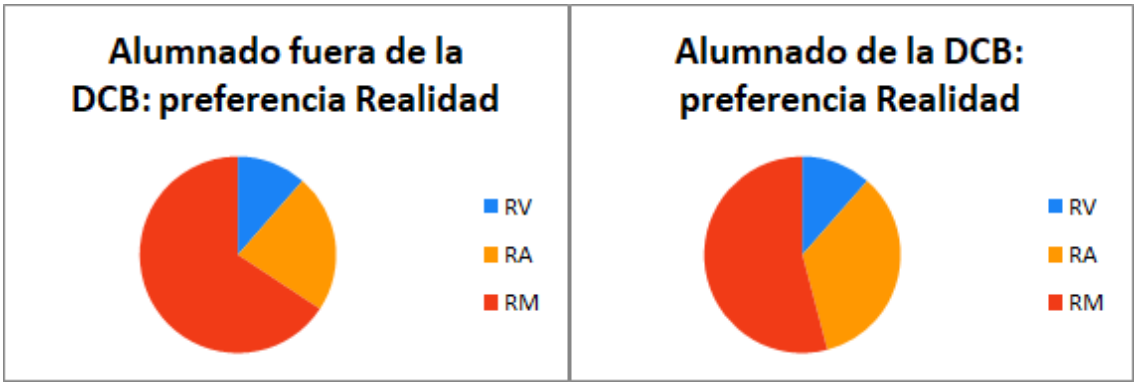
a) *b)*

Figura B.1 Gráfica de distribución comparativa sobre la preferencia del uso en sistemas operativos: a) respuestas por parte del alumnado no perteneciente a la DCB; b) respuestas por parte del alumnado perteneciente a la DCB.



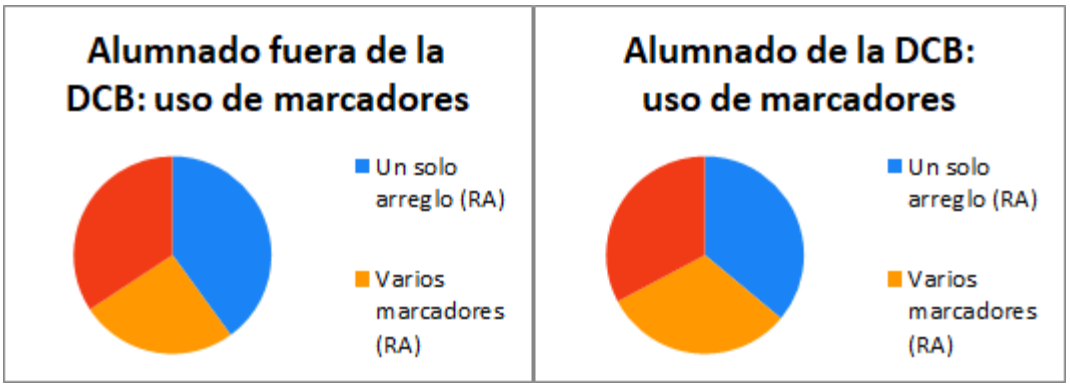
a) *b)*

Figura B.2 Gráfica de distribución comparativa sobre el uso de versiones en el sistema operativo Android: a) respuestas por parte del alumnado no perteneciente a la DCB; b) respuestas por parte del alumnado perteneciente a la DCB.



a) *b)*

Figura B.3 Gráfica de distribución comparativa sobre la preferencia del uso en tecnología de realidad: a) respuestas por parte del alumnado no perteneciente a la DCB; b) respuestas por parte del alumnado perteneciente a la DCB.



a) *b)*

Figura B.4 Gráfica de distribución comparativa sobre la preferencia del uso de marcadores en las tecnologías de realidad: a) respuestas por parte del alumnado no perteneciente a la DCB; b) respuestas por parte del alumnado perteneciente a la DCB.

C Método numérico Runge-Kutta de cuarto orden

Dada la ecuación diferencial ordinaria C.1 y conociendo su valor y_0 evaluada en un punto conocido t_0:

$$y'(t) = f(t, y(t)) ; y_0 = y(t_0) \quad \text{(C.1)}$$

Se realiza la aproximación punto a punto de la función y evaluada en un punto siguiente t_{n+1}, extrapolando y_{n+1} linealmente a partir del valor conocido y_n. Esta aproximación se obtiene sumando al valor conocido y_n, el producto de una pendiente y el valor de la distancia entre los puntos t_n y t_{n+1}, y tiene la forma general de la ecuación C.2.

$$y_{n+1} = y_n + \frac{1}{6} h (k_1 + 2k_2 + 2k_3 + k_4) \quad \text{(C.2)}$$

Donde h es el valor de la distancia entre los puntos t_n y t_{n+1}, y_{n+1} es el valor aproximado de la función evaluada en el punto siguiente t_{n+1}, y los coeficientes k son aproximaciones de la primera derivada (interpretable como la pendiente) de la función y dentro del intervalo entre t_n y t_{n+1}.

Como se muestra en la figura C.1, para evaluar la función $f(t, y(t))$ en los distintos puntos del intervalo entre t_n y t_{n+1} y obtener el valor de k_i, donde $i = 1, 2, 3, 4$; se aproxima el incremento de y_n con el producto del incremento de la variable t y la pendiente anteriormente aproximada k_{i-1}, de tal forma que los coeficientes k_i están definidos como se muestra en las ecuaciones C.3 a C.6. Con

estos coeficientes se realiza el promedio ponderado de la ecuación C.2 para obtener el valor interpolado de y_{n+1}.

$$k_1 = f\left(t_n, y_n\right) \tag{C.3}$$

$$k_2 = f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}h\,k_1\right) \tag{C.4}$$

$$k_3 = f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}h\,k_2\right) \tag{C.5}$$

$$k_4 = f\left(t_n + h, y_n + h\,k_3\right) \tag{C.6}$$

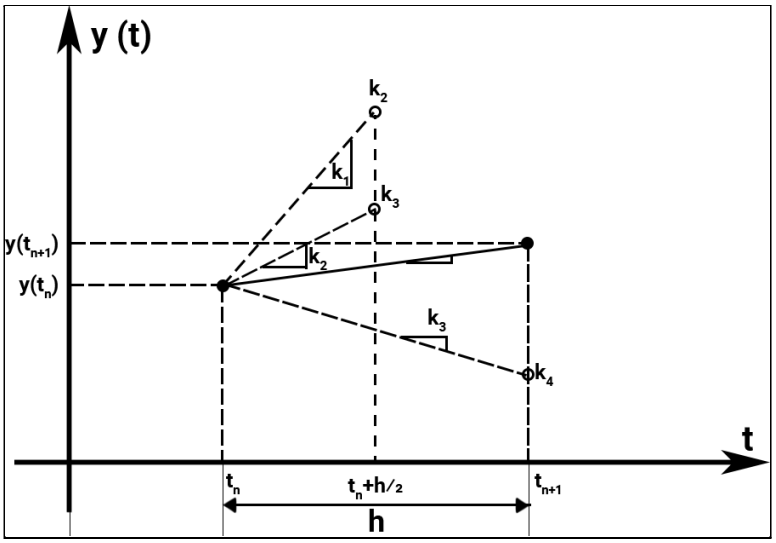


Figura C.1 Pendientes k_i aproximadas para la interpolación punto a punto de y de [58].

Dada la ecuación diferencial de segundo orden C.7.

$$y''(t) = f(t, y(t), y'(t)) \tag{C.7}$$

El método se aplica convirtiendo la ecuación en un sistema de dos ecuaciones diferenciales de primer orden. Realizando el cambio de variable propuesto en la ecuación C.8, se obtiene el sistema de ecuaciones diferenciales mostrado en las ecuaciones C.9 y C.10.

$$u = y(t) \;;\; w = y'(t) \tag{C.8}$$

$$u'(t) = w(t) \tag{C.9}$$

$$w'(t) = f(t, u(t), w(t)) \tag{C.10}$$

Con condiciones iniciales

$$y_0 = y(t_0); y'_0 = y'(t_0) \tag{C.11}$$

Por lo que

$$u_0 = y_0; w_0 = y'_0 \tag{C.12}$$

Al miembro derecho de las ecuaciones C.9 y C.10, es decir las derivadas de primer orden, se les denominará

$$f_1 = w \tag{C.13}$$

$$f_2 = f(t, u, w) \tag{C.14}$$

Cabe hacer notar que

$$f_1 = f_1(w) \tag{C.15}$$

$$f_2 = f_2(t, u, w) \tag{C.16}$$

De manera similar a la forma de la ecuación C.2, se aplica el método a las ecuaciones C.9 y C.10 para obtener los valores u_{n+1} y w_{n+1}.

$$u_{n+1} = u_n + \frac{1}{6} h (p_1 + 2p_2 + 2p_3 + p_4) \tag{C.17}$$

$$w_{n+1} = w_n + \frac{1}{6} h (q_1 + 2q_2 + 2q_3 + q_4) \tag{C.18}$$

Siguiendo con la forma general de la ecuación C.2, los valores de p_i son las aproximaciones de la primera derivada de la función u_n, dadas por la función w_n, y los valores de q_n son las aproximaciones de la primera derivada de w_n, dada por la función $f(t, u, w)$ evaluada en los puntos t_n, u_n y w_n. Los incrementos de u_n y w_n para ambas aproximaciones se obtienen con los valores de los coeficientes p_i y q_i determinados a partir de las ecuaciones C.3 a C.6, multiplicados por el incremento de la variable t_n, de tal manera que para el cálculo de los valores de p:

$$p_1 = f_1(w_n) \tag{C.19}$$

$$p_2 = f_1(w_n + \frac{1}{2} h q_1) \tag{C.20}$$

$$p_3 = f_1(w_n + \frac{1}{2} h\, q_2) \quad \text{(C.21)}$$

$$p_4 = f_1(w_n + h\, q_3) \quad \text{(C.22)}$$

Por otra parte, el cálculo de los valores de q se determina a partir de las ecuaciones C.23 a C.26:

$$q_1 = f_2\left(t_n, u_n, w_n\right) \quad \text{(C.23)}$$

$$q_2 = f_2\left(t_n + \frac{1}{2} h, u_n + \frac{1}{2} h\, p_1, w_n + \frac{1}{2} h\, q_1\right) \quad \text{(C.24)}$$

$$q_3 = f_2\left(t_n + \frac{1}{2} h, u_n + \frac{1}{2} h\, p_2, w_n + \frac{1}{2} h\, q_2\right) \quad \text{(C.25)}$$

$$q_4 = f_2\left(t_n + h, u_n + h\, p_3, w_n + h\, q_3\right) \quad \text{(C.26)}$$

Para resolver la ED que describe el comportamiento del voltaje del condensador en el circuito RLC, se retoma la ecuación 2.7 explicada en el capítulo 2.5.1 y se despeja la segunda derivada del voltaje del condensador V_C para llegar a la forma general de la ecuación C.7, tal que:

$$\frac{d^2}{dt^2} V_C = f(t, V_C, \frac{d}{dt} V_C) = \frac{1}{LC} V_f(t) - \frac{1}{LC} V_C - \frac{R}{L} \frac{d}{dt} V_C \quad \text{(C.27)}$$

Donde V_f es el voltaje de alimentación, L es la inductancia, C la capacitancia y R la resistencia de los elementos del circuito.

Se realiza el cambio de variable de acuerdo a la forma de la ecuación C.8, es decir, los cambios propuestos en C.28, obteniendo el sistema conformado por las ecuaciones C.29 y C.30.

$$u(t) = V_C(t) \;;\; w(t) = \frac{d}{dt} V_C(t) \quad \text{(C.28)}$$

$$u'(t) = f_1(w(t)) = w(t) \quad \text{(C.29)}$$

$$w'(t) = f_2(t, u(t), w(t)) = \frac{1}{LC} V_f(t) - \frac{1}{LC} u(t) - \frac{R}{L} w(t) \quad \text{(C.30)}$$

Siguiendo la forma de las ecuaciones C.13 y C.14, el método se aplica en el sistema de ecuaciones anterior, tal que las aproximaciones punto a punto del voltaje del condensador y su derivada se calculan a partir de las ecuaciones C.31 y C.32 respectivamente:

$$(V_C)_{n+1} = (V_C)_n + \frac{1}{6} h (p_1 + 2p_2 + 2p_3 + p_4) \quad \text{(C.31)}$$

$$\left(\frac{d}{dt} V_C\right)_{n+1} = \left(\frac{d}{dt} V_C\right)_n + \frac{1}{6} h (q_1 + 2q_2 + 2q_3 + q_4) \quad \text{(C.32)}$$

D Proceso iterativo de modelado de componentes tridimensionales

Para los modelos tridimensionales del condensador, resistor e inductor, se tomaron como referencia fotografías de los componentes, esto con el fin de mantener las proporciones de los modelos similares a las de los componentes reales.

D.1 Condensador

Se colocó un cilindro delgado con las dimensiones del condensador (figura D.1a) y se movieron los vértices de la parte inferior para darle la forma del cuerpo del condensador (figura D.1b). Se realizó una división radial de las caras laterales del cilindro, haciendo una división en la parte central y en las orillas (figura D.1c), para posteriormente adelgazar las orillas del cilindro y hacer sobresalir la parte central (figura D.1d).

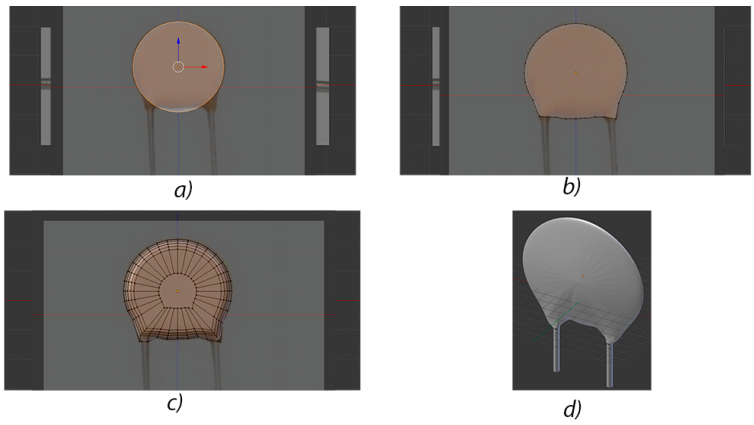


Figura D.1 Proceso de construcción del modelo tridimensional de un condensador: a) cilindro inicial; b) detalles de forma; c) redondeado de las orillas y suavizado de bordes; d) modelo final con terminales.

D.2 Resistor

Se colocó un cilindro con las proporciones del resistor (figura D.2a), se dividió transversalmente en 16 segmentos (figura D.2b) y se moldearon las dimensiones radiales de sus vértices para que el cilindro tuviera la silueta del resistor (figura D.2c). Se realizó un proceso de subdivisión de aristas automático para suavizar los bordes del modelo (figura D.2d). Se realizó un proceso similar con dos cilindros para añadir las terminales del modelo (figura D.2e).

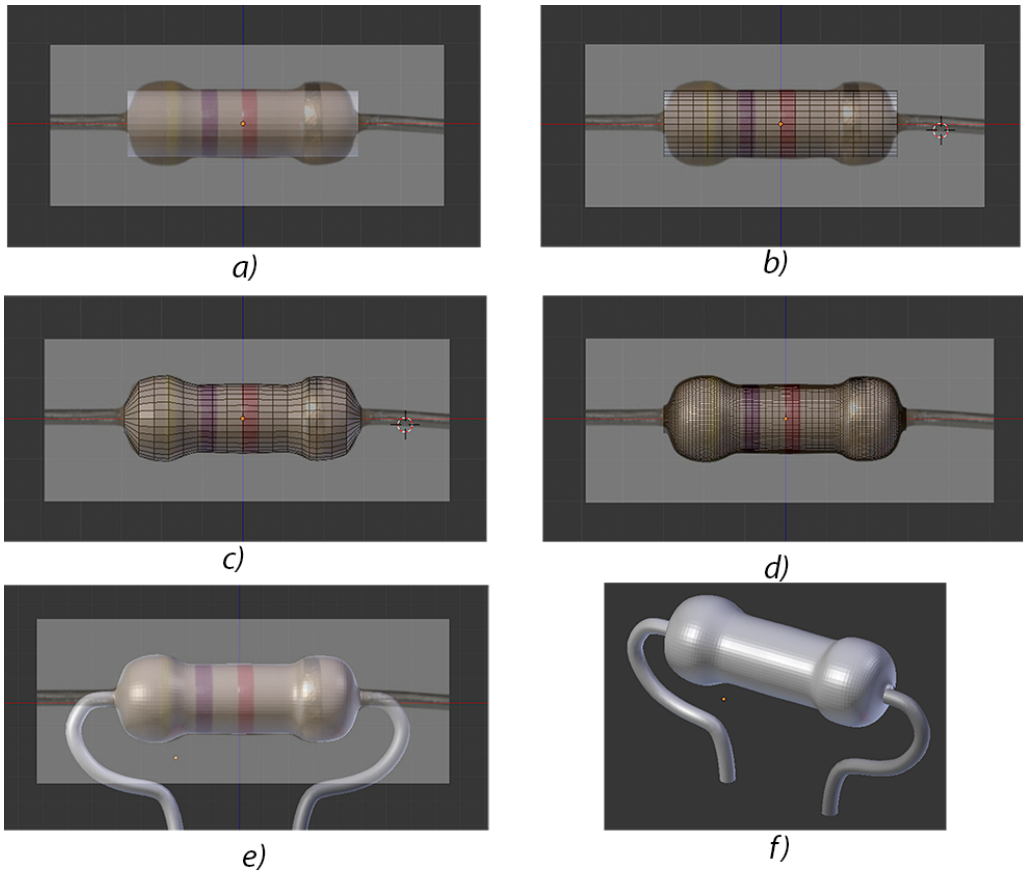


Figura D.2 Proceso de construcción del modelo tridimensional de un resistor: a) cilindro base; b) división del cilindro; c) definición de detalles; d) suavizado de bordes; e) se agregan terminales; f) modelo final.

D.3 Inductor

Este modelo está constituido por 5 partes separadas. La primera corresponde al devanado del inductor, el cual se construye a partir de un disco horizontal en la parte inferior del devanado en la imagen de referencia (figura D.3a); a este modelo se le hace una transformación denominada en Blender como "*Screw*" para generar

una espiral (figura D.3b). Posteriormente se agregan cuatro cilindros; dos cilindros gruesos, uno en la parte superior e inferior del devanado, y dos cilindros delgados en la parte inferior que corresponden a los soportes del devanado y las terminales del inductor, respectivamente (figura D.3c), con lo que se obtiene el modelo final (figura D.3d).

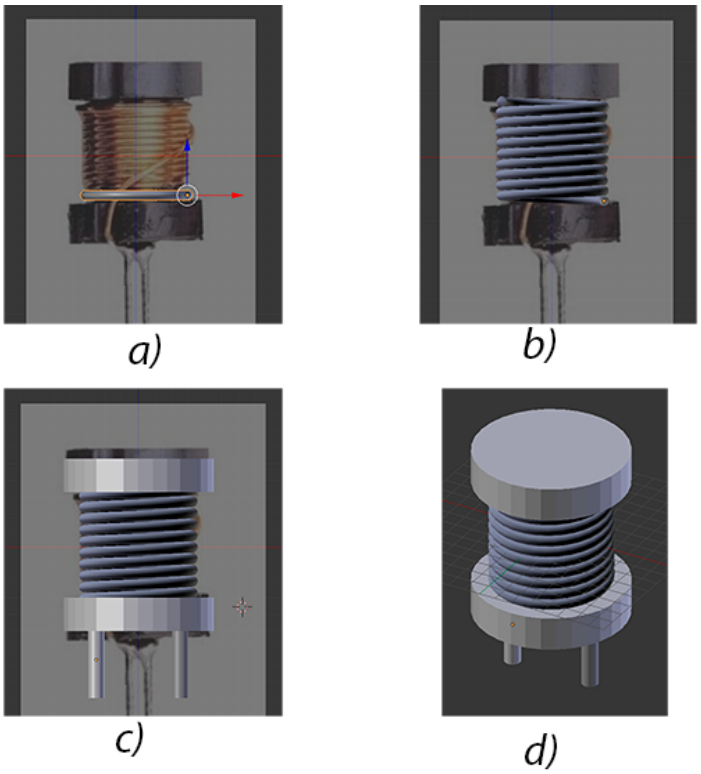


Figura D.3 proceso de construcción del modelo tridimensional de un inductor: a) rueda inicial; b) extrusión de la rueda para el bobinado del inductor; c) tapas y terminales del inductor; d) modelo final.

D.4 Osciloscopio

D.4.1 Primera iteración

Se construyó un prisma rectangular con los bordes suavizados (figura D.4a), en una cara del prisma se hicieron ranuras para representar la pantalla y las perillas de un osciloscopio (figura D.4b). Este concepto fue modificado debido a que se determinó que el uso de perillas en una pantalla táctil podría ser complicado para el usuario.

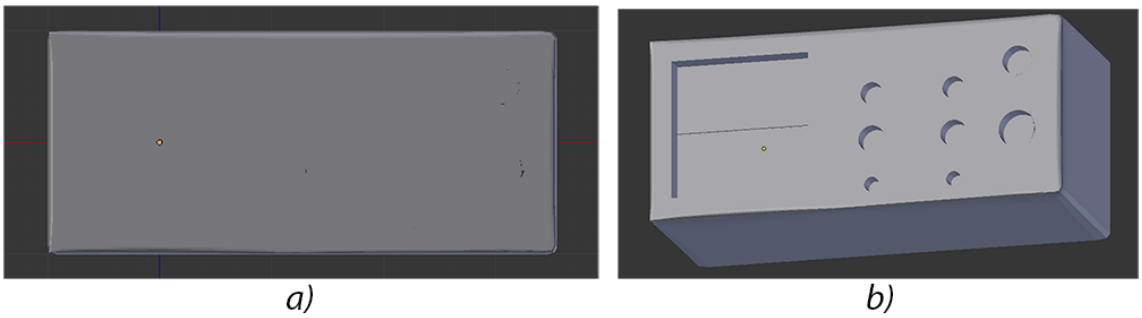


Figura D.4 Primera iteración del modelo tridimensional del osciloscopio: a) prisma rectangular con bordes redondeados; b) modelo con ranuras de perillas y pantallas.

D.4.2 Segunda iteración

A un prisma rectangular (figura D.5a) se le suavizaron los bordes y se le hizo una ranura en la parte frontal (figura D.5b). Se le añadieron salientes delgadas para marcar los límites de la pantalla del osciloscopio y de controles de tipo deslizador en cada lado (figura D.5c). Finalmente se agregaron prismas rectangulares en la parte superior que simulan botones para modificar la activación de las señales mostradas en la pantalla del osciloscopio (figura D.5d).

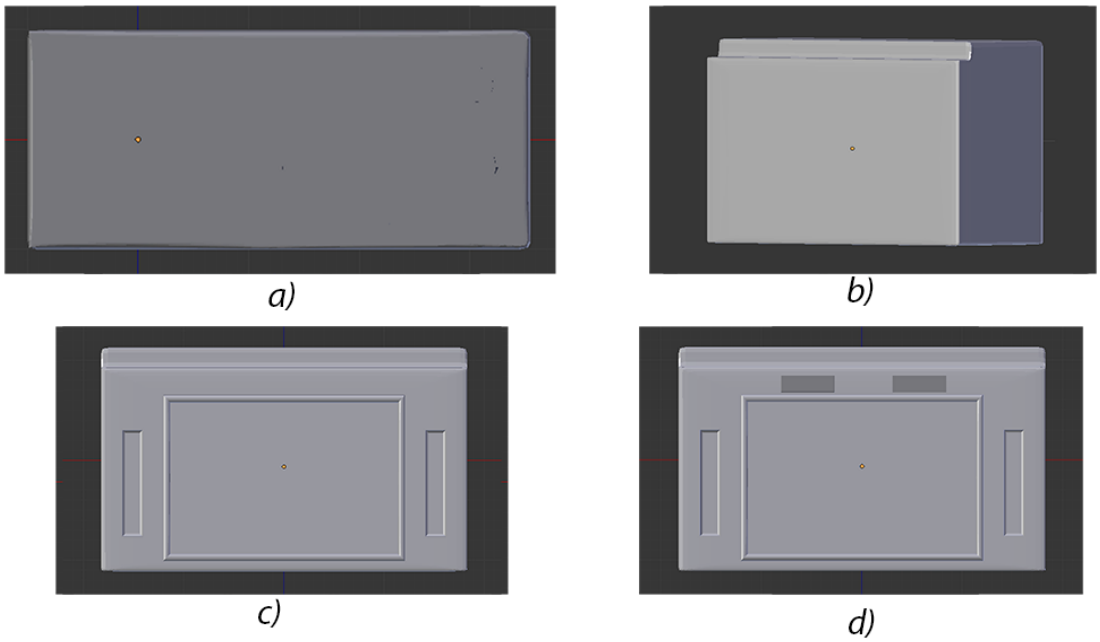


Figura D.5 Segunda iteración del modelo tridimensional del osciloscopio: a) prisma rectangular; b) bordes redondeados y cambio en la forma; c) espacio para pantalla y controles; e) modelo final con botones.

D.5 Módulo de protoboard y generador

D.5.1 Primera iteración

Para este modelo, primero se creó un boceto para indicar las dimensiones y ubicación de los componentes del módulo (figura D.6a), el cual posteriormente se utilizó como la imagen referencia para crear el modelo tridimensional. Se creó un prisma rectangular con las medidas del boceto (figura D.6b), sobre el cual se agregaron modelos tridimensionales adicionales para cada elemento, como la pantalla del generador, los botones y palanca para modificar la señal del generador, botones para las interacciones del usuario con los componentes y las terminales del generador y osciloscopio (figura D.6c). Del lado derecho se añadió un arreglo de 5x5 espacios para colocar componentes, en los cuales el usuario realizaría el proceso de alambrado (figura D.6d).

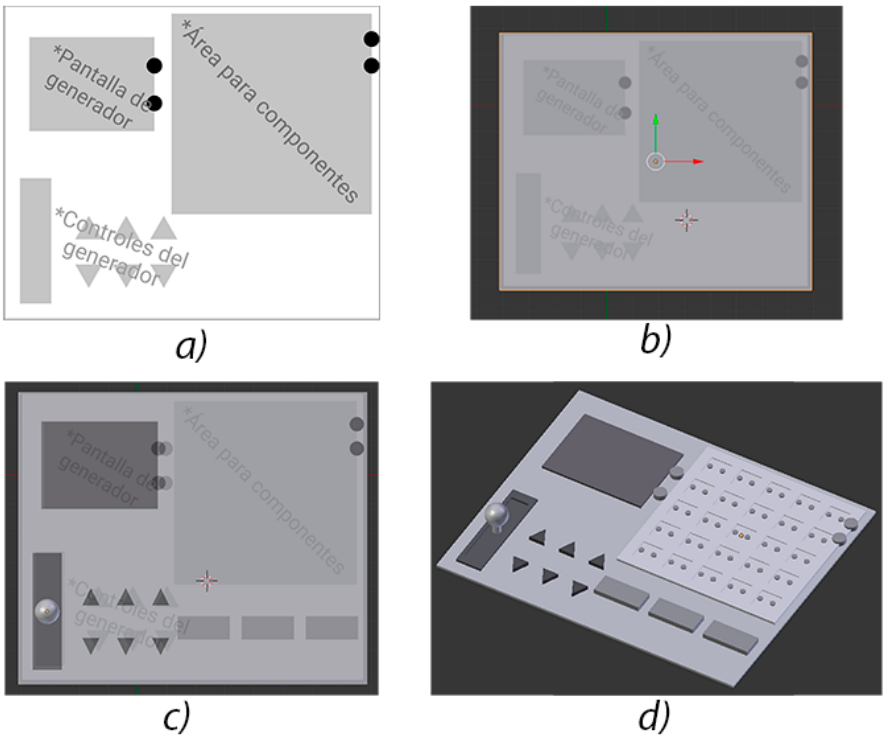


Figura D.6 Primera iteración del modelo tridimensional del módulo de protoboard y generador de señales: a) boceto; b) prisma rectangular base; c) se agregan terminales, pantalla, botones y perilla como modelos independientes; d) modelo final.

D.5.2 Segunda iteración

Se quitaron botones innecesarios del panel, y se disminuyó el arreglo de 5x5 espacios para componentes a 3x4, debido a que sólo son necesarias 3 terminales para realizar la práctica; además se buscó disminuir las dimensiones del modelo y permitir que se visualice por completo en las pantallas de los dispositivos móviles ejecutantes.

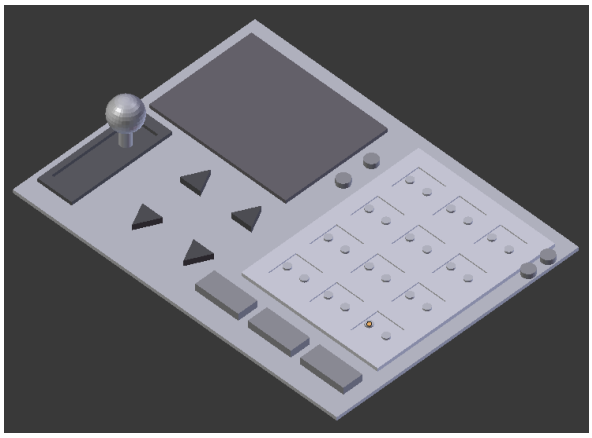


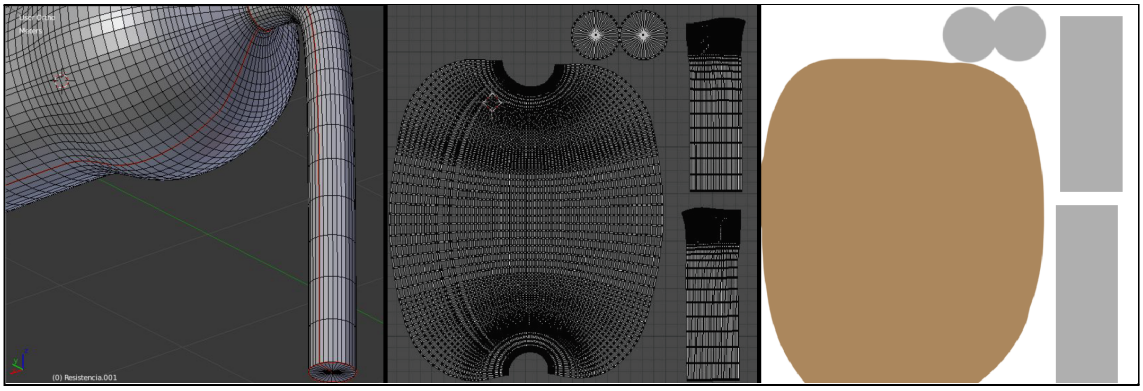
Figura D.7 Segunda iteración del modelo tridimensional del módulo de protoboard y generador de señales. Se modificó la proporción del modelo y se redistribuyeron los componentes.

E Proceso de diseño de texturas para los modelos tridimensionales

Para crear las texturas de los modelos tridimensionales, su superficie se debe dividir a partir de líneas de costura y se distribuye en una imagen plana denominada mapa de texturas, la cual sirve como guía para crear las imágenes que darán el detalle a los modelos.

E.1 Resistor

Se marcaron líneas de costura en los vértices ubicados a lo largo del eje longitudinal del resistor y sus terminales (figura E.1a), y se generó el mapa de texturas por Blender de manera automática a partir de las líneas (figura E.1b). Finalmente se creó la imagen de la textura del resistor (figura E.1c).

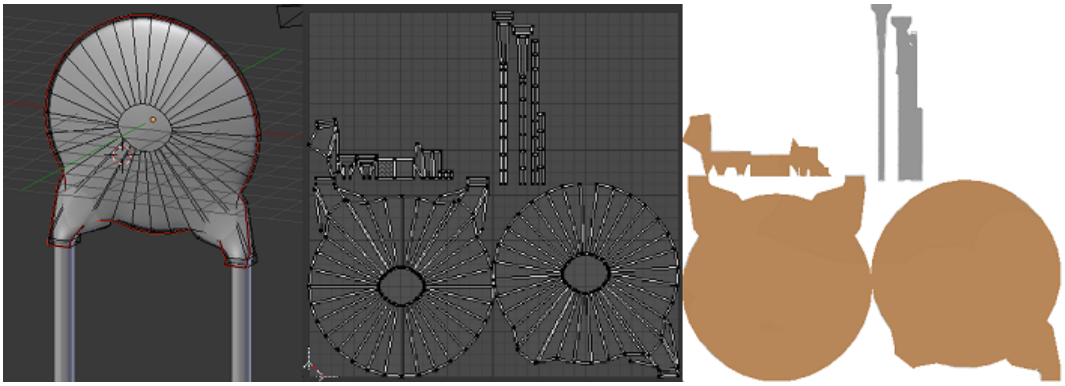


a) *b)* *c)*

Figura E.1 a) modelo del resistor con líneas de costura; b) mapa de texturas del resistor; c) textura utilizada para el resistor.

E.2 Condensador

Se marcaron líneas de costura en la orilla de la parte circular del condensador, así como en el eje longitudinal de sus terminales (figura E.2a). Se creó el mapa de texturas de forma automática por Blender (figura E.2b) y se dibujó la imagen para la textura del condensador (figura E.2c).

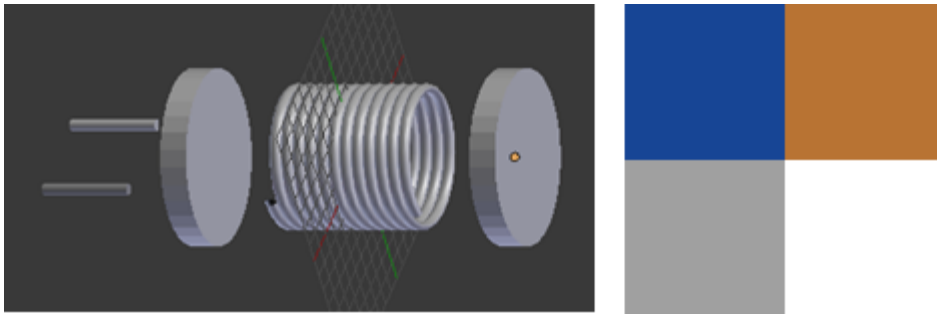


a) *b)* *c)*

Figura E.2 a) modelo del condensador con líneas de costura; b) mapa de texturas del condensador; c) textura utilizada para el condensador.

E.3 Inductor

El modelo del inductor consta de 4 piezas completamente separadas (figura E.3a), por lo que bastó usar un color sólido en cada una sin necesidad de definir costuras (figura E.3b).

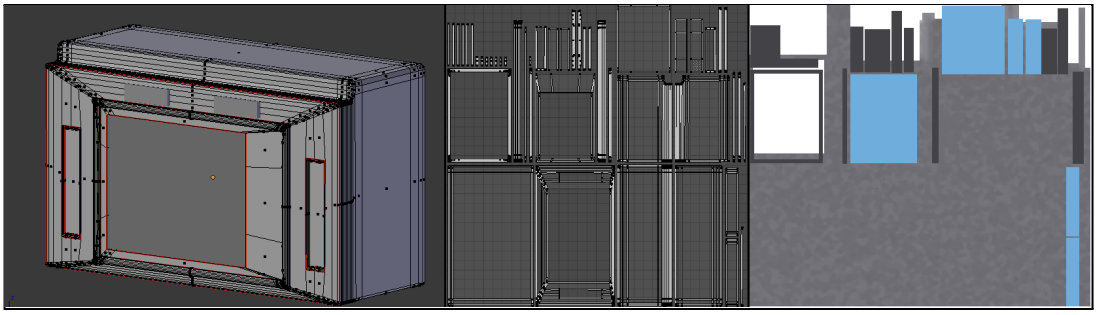


a) *b)*

Figura E.3 a) modelo del inductor separado en partes; b) colores sólidos utilizados en el modelo.

E.4 Osciloscopio

Las líneas de costura del osciloscopio se colocaron con el objetivo de definir claramente los espacios correspondientes a la pantalla, los controles y la cara frontal (figura E.4a). El mapa de texturas se creó automáticamente (figura E.4b) y a partir de este se dibujó la textura. El osciloscopio es de color gris, las partes en color gris oscuro son para los bordes que delimitan las partes del osciloscopio y las partes azules son las pertenecientes a la pantalla.

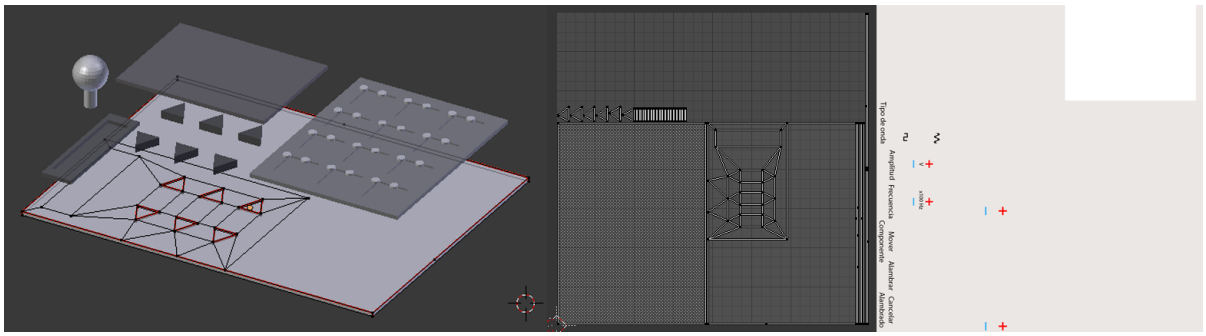


a) *b)* *c)*

Figura E.4 a) modelo del osciloscopio con líneas de costura; b) mapa de texturas del osciloscopio; c) textura utilizada para el osciloscopio.

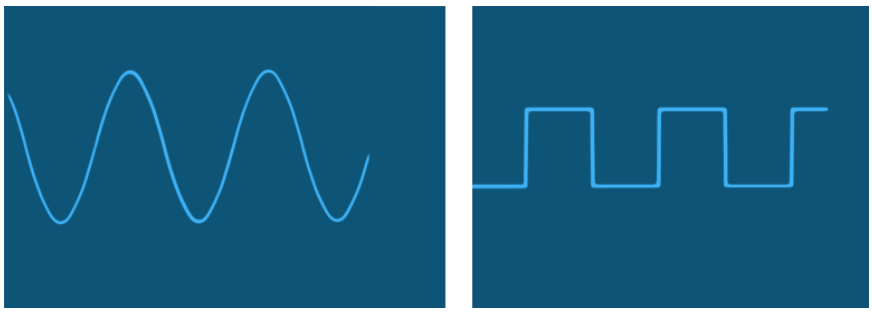
E.5 Módulo de protoboard y generador

El módulo está construido por una base rectangular y modelos tridimensionales adicionales que representan sus botones y terminales. Se crearon texturas para la base y para la pantalla del generador de funciones. La textura de la base (figura E.5) tiene los nombres de los botones, la acción que realizan los botones y la palanca del módulo. Para la pantalla se crearon cuatro texturas diferentes; la primera y la segunda son para indicar al usuario si el generador está produciendo una señal sinusoidal (figura E.6a) o cuadrada (figura E.6b); la tercera y la cuarta son texturas transparentes que se superponen a las primeras dos para indicar al usuario el voltaje (figura E.7) y la frecuencia (figura E.8) de la señal.



a) b) c)

Figura E.5 a) modelo del módulo de protoboard con generador separado en partes y con líneas de costura; b) mapa de texturas de la base del módulo de protoboard con generador; c) textura utilizada para la base del módulo.



a) b)

Figura E.6 Texturas utilizadas para indicar la señal del generador: a) señal sinusoidal; b) señal cuadrada.

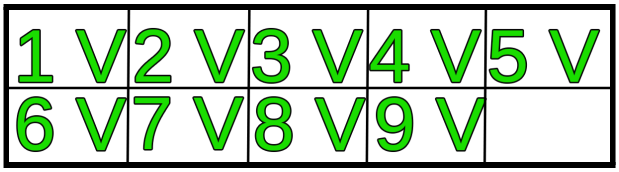


Figura E.7 Texturas utilizadas para desplegar el voltaje de la señal simulada por el generador.

Figura E.8 Texturas utilizadas para desplegar la frecuencia de la señal simulada por el generador.

F Implementación en código de marcador definido por el usuario

En todo el código, los comentarios descriptivos se muestran en color verde.

```
using System.Collections;
using UnityEngine;
using Vuforia;
public class UDTManager : MonoBehaviour, IUserDefinedTargetEventHandler {
    //Componente que construye el “target”
    UserDefinedTargetBuildingBehaviour udt_targetBuildingBehaviour;
    //Componente para la búsqueda y reconocimiento de imágenes
    ObjectTracker objectTracker;
    //Arreglo donde se encuentran los “targets” instanciados
    DataSet dataSet;
    ImageTargetBuilder.FrameQuality udt_FrameQuality;
    //”Prefab” del “target” con componentes base
    public ImageTargetBehaviour targetBehaviour;
    //Objeto en el que se instancia el panel completo con componentes
    ImageTargetBehaviour currentImageTarget;
    //Bandera de que ya se instanció una vez el marcador
    bool instanciado = false;
    int targetCounter;
    //Indicadores de calidad de imagen
    UnityEngine.UI.Image Rojo;
    UnityEngine.UI.Image Amarillo;
    UnityEngine.UI.Image Verde;
    UnityEngine.UI.Image Marco;
    //Lista de componentes que se desplegarán (pública para que se
    le pasen datos)
    public string[] Componentes;
    //Lista con los valores de los componentes anteriores (en el mismo orden)
    public float[] Valores_comp = new float[5];
    //Componentes que se instancian con base en este arreglo
    GameObject[] Componentes_Inst = new GameObject[5];
    //”Prefabs” de los componentes que se eligen para alambrar el circuito
    public GameObject Resistencia_Prefab;
    public GameObject Capacitor_Prefab;
    public GameObject Inductor_Prefab;
    //Botones de despliegue y cambio de marcador
```

```
    GameObject botonDesplegar;
    GameObject botonDestruir;
    GameObject ventanaEmergente;
    GameObject ventanaEmergenteAlambrado;
    //Tabla en la que se colocan los componentes al desplegar un
    marcador nuevo
    public GameObject espacio_componentes;
    //"GameObject" de tablero de componentes inicial
    GameObject InitialBoard;

    void Start() {
//Obtiene el constructor de "targets" del objeto actual
udt_targetBuildingBehaviour = GetComponent<UserDefinedTarget>();
//Obtiene "GameObject" e Imágenes de los indicadores para la
calidad de imagen
Rojo=GameObject.Find("Indicador_Bajo").GetComponent<UnityEngine.UI.Image>();
Amar=GameObject.Find("Indicador_Medio").GetComponent<UnityEngine.UI.Image>();
Verde=GameObject.Find("Indicador_Alto").GetComponent<UnityEngine.UI.Image>();
Marco=GameObject.Find("Indicador_Marco").GetComponent<UnityEngine.UI.Image>()
//Define el "GameObject" que se instanciará con el marcador, dependiendo del
estado de la transición de escenas
    }
    //Se llama al inicializar "UserDefinedTargetBehaviour"
    public void OnInitialized()    {
    //Instancia el objeto que reconoce imágenes nuevas o ya instanciadas
    objectTracker = TrackerManager.Instance.GetTracker<ObjectTracker>();
    }

    //Se llama al cambiar la calidad de imagen de la cámara
    public void OnFrameQualityChanged
    (ImageTargetBuilder.FrameQuality frameQuality)    {
        //Lee la calidad de la imagen actual
        udt_FrameQuality = frameQuality;
        //Cambia en color de los indicadores según la calidad de imagen
        //Calidad alta
        if (udt_FrameQuality == ImageTargetBuilder.FrameQuality
        .FRAME_QUALITY_HIGH){
            Verde.enabled = true;
            Amarillo.enabled = true;
        }
        //Calidad media
        elseif(udt_FrameQuality==ImageTargetBuilder.FrameQuality
        FRAME_QUALITY_MEDIUM{
            Verde.enabled = false;
            Amarillo.enabled = true;
        }
        //Calidad baja
        elseif(udt_FrameQuality== ImageTargetBuilder.FrameQuality
        .FRAME_QUALITY_LOW){
            Verde.enabled = false;
            Amarillo.enabled = false;        }    }
```

```
//Se llama al tratar de inicializar un marcador nuevo con el
botón de la GUI
public void BuildTarget()   {
    //Verifica que haya una calidad de imagen apropiada
    if (udt_FrameQuality == ImageTargetBuilder
    .FrameQuality.FRAME_QUALITY_HIGH) {
    //Llama la función del objeto que construye los marcadores
    (Nombre del     marcador, tamaño horizontal del marcador)
    //Esta función crea una imagen, lo que llamará a la
    función “OnNewTrackableSource()” para ahí instanciar los objetos
    con las relaciones apropiadas
    udt_targetBuildingBehaviour.BuildNewTarget("targetActual",
    targetBehaviour.GetSize().x);
    //Apaga los marcadores de indicador de imagen
    Verde.enabled = false;
    Amarillo.enabled = false;
    Rojo.enabled = false;
    Marco.gameObject.transform.Find("textoCalidad")
    .GetComponent<UnityEngine.UI.Text>().enabled = false;
    Marco.enabled = false;
    //Deshabilita el botón de desplegar y habilita
    el botón de cambiar marcador
    botonDesplegar.SetActive(false);
    botonDestruir.SetActive(true);
}

    //Al haber instanciado un nuevo marcador; recibe la imagen
    con la que se instanció
public void OnNewTrackableSource(TrackableSource trackableSource)  {
    targetCounter++;
    //Desactiva el “Dataset” creado al inicializar vuforia para poder
    incluir un nuevo objeto
    objectTracker.DeactivateDataSet(dataSet);
    //Instancia el “GameObject” creado al iniciar la escena para
    usarse con el marcador
    currentImageTarget = Instantiate(targetBehaviour)
    as ImageTargetBehaviour;
    //Coloca el tablero con los componentes como parte del marcador
    ComponentesAlambrados.Board.SetActive(true);
    Instantiate(ComponentesAlambrados.Board, currentImageTarget
    .gameObject.transform);
    ComponentesAlambrados.Board.SetActive(false);
    //Crea un nuevo objeto dentro del “Dataset” (imagen, “GameObject”
    que se relacionará con el “target”)
    dataSet.CreateTrackable(trackableSource,
    currentImageTarget.gameObject);
    //Activa el dataset
    objectTracker.ActivateDataSet(dataSet);
    //Bandera de marcador instanciado
    instanciado = true;
```

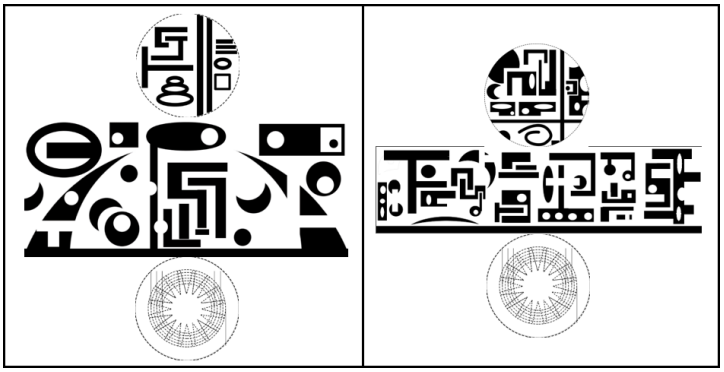
```
    //Hace que las coordenadas globales estén referidas
    al componente instanciado
    VuforiaARController.Instance.SetWorldCenter
    (currentImageTarget.GetComponent<TrackableBehaviour>());
}

public void PopUpWarning()
{       ventanaEmergente.SetActive(true);    }
public void PopUpWarningClose()
{       ventanaEmergente.SetActive(false);    }
public void PopUpWireClose()
{       ventanaEmergenteAlambrado.SetActive(false);    }

//Para cambiar la imagen a la que se asignó el “target”
public void DestroyTarget()    {
    if (targetCounter > 0)
    {
        //Desactiva el “Dataset”
        objectTracker.DeactivateDataSet(dataSet);
        //Destruye el objeto instanciado anteriormente
        dataSet.DestroyAllTrackables(true);
        //Reinicia el “Dataset”
        objectTracker.ActivateDataSet(dataSet);
        Debug.Log("Target destruido");
        targetCounter--;
        //Detiene la búsqueda de nuevos marcadores
        udt_targetBuildingBehaviour.StartScanning();
        Rojo.enabled = true;
        Marco.enabled = true;
        Marco.gameObject.transform.Find("textoCalidad")
        .GetComponent<UnityEngine.UI.Text>().enabled = true;
        //Deshabilita el botón de destruir y la ventana emergente,
        y habilita el botón de desplegar marcador
        botonDestruir.SetActive(true);
        botonDesplegar.SetActive(true);
        ventanaEmergente.SetActive(true);
    }   }
public void SaveBoard()    {
    if (GameObject.FindWithTag("FullBoard") != null)        {
        Destroy(ComponentesAlambrados.Board);
        ComponentesAlambrados.Board=Instantiate
        (GameObject.FindWithTag("FullBoard"));
        ComponentesAlambrados.Board.SetActive(false);
        DontDestroyOnLoad(ComponentesAlambrados.Board);
    } }
```

G Proceso iterativo de diseño del marcador geométrico

G.1 Primera iteración

Durante la primera iteración de la mecánica de interacción descrita en el apéndice H, se implementó el uso de dos marcadores cilíndricos, hechos para los dedos índice y pulgar del usuario. Esta primera iteración estaba encaminada a manipular los objetos virtuales realizando un movimiento de sujeción entre ambos dedos y los modelos tridimensionales.

Con base en las medidas de anillos estandarizadas [49] y los valores de longitud promedio para manos [48], se establecieron como medidas de 4.5cm de largo y 2cm de diámetro para el marcador del dedo índice, y 3cm de largo y 2.5cm de diámetro para el marcador del dedo pulgar. Los patrones en las imágenes para las caras de cada cilindro, se crearon tratando de que tuvieran un gran número de curvas y esquinas.



a) *b)*

Figura G.1 Primera iteración de marcadores cilíndricos con figuras geométricas simples: a) para el dedo índice; b) para el dedo pulgar.

Esta primera iteración de marcadores fue evaluada en el portal de Vuforia para conocer su viabilidad para usar como marcador. En las figuras G.2 y G.3 se observan los puntos característicos de cada imagen en color amarillo, además de su calificación en una escala de 1 a 5 estrellas. Como se puede observar, resultaron ser poco viables para su detección como marcador.

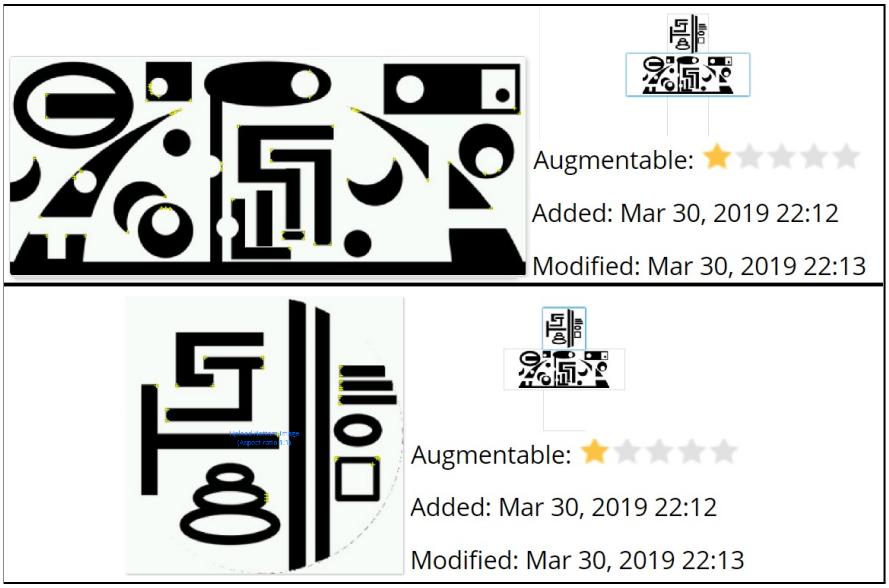


Figura G.2 Valoración en el portal de Vuforia del marcador cilíndrico del dedo índice de la primera iteración.

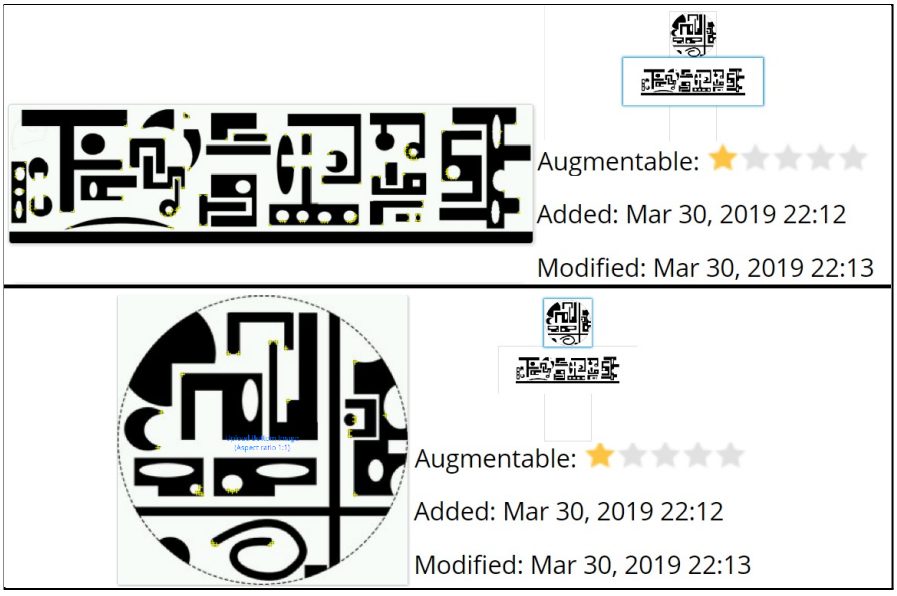
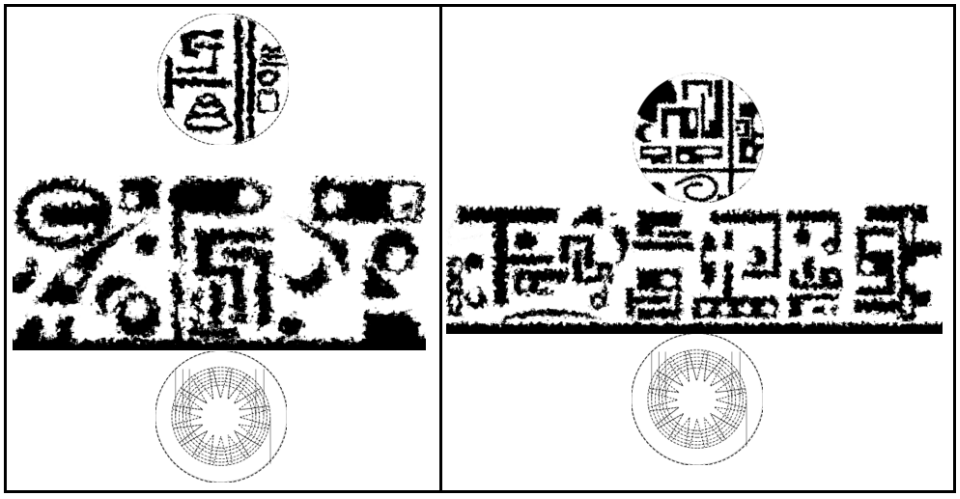


Figura G.3 Valoración en el portal de Vuforia del marcador cilíndrico del dedo pulgar de la primera iteración.

G.2 Segunda iteración

Los patrones de las imágenes de la primera iteración fueron distorsionadas para aumentar en gran medida su cantidad de puntos característicos, resultando en las imágenes de la figura G.4. Como se observa en las figuras G.5 y G.6, al evaluar esta modificación, las imágenes tuvieron una mayor cantidad de puntos característicos, obteniendo una calificación de 5 estrellas para su viabilidad como marcadores.



a) *b)*

Figura G.4 Segunda iteración de marcadores cilíndricos con figuras geométricas simples y distorsión: a) para el dedo índice; b) para el dedo pulgar.

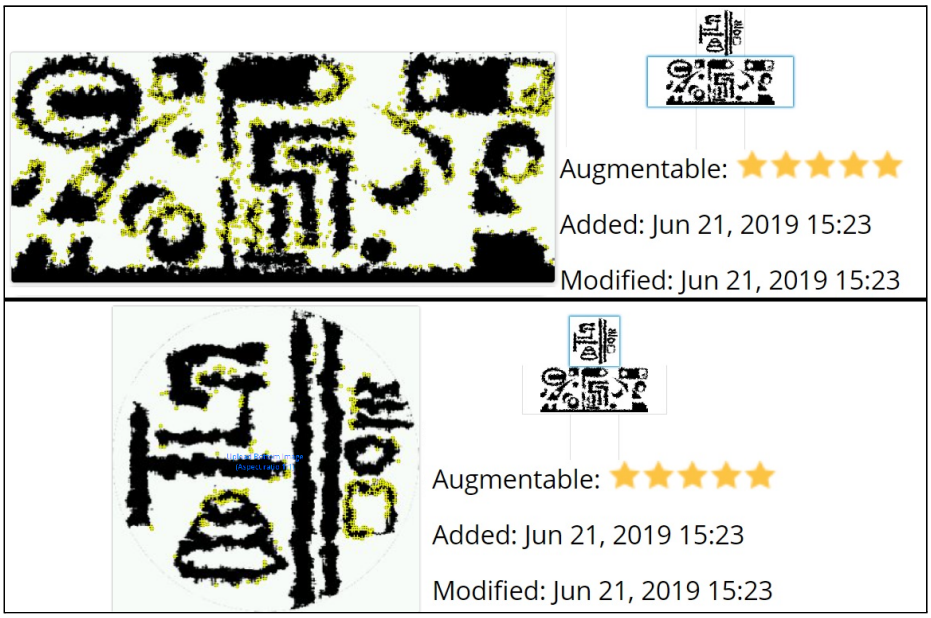


Figura G.5 Valoración en el portal de Vuforia del marcador cilíndrico del dedo índice de la segunda iteración.

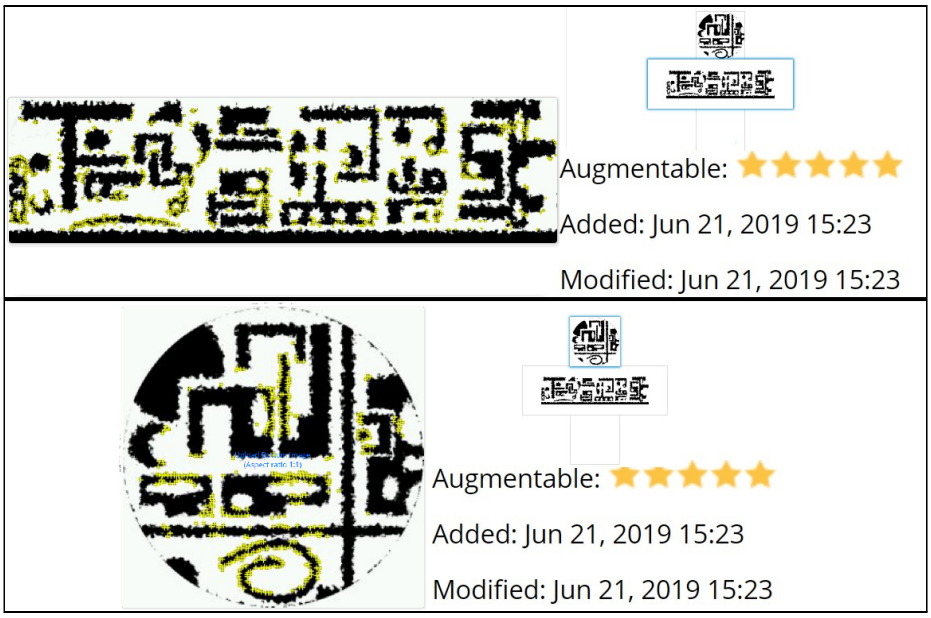


Figura G.6 Valoración en el portal de Vuforia del marcador cilíndrico del dedo pulgar de la segunda iteración.

G.3 Tercera iteración

Durante la segunda iteración de la mecánica de interacción descrita en el apéndice H, esta se modificó para que el usuario interactuara con los componentes utilizando únicamente un marcador. Por esta razón, se conservó el marcador cilíndrico del dedo índice.

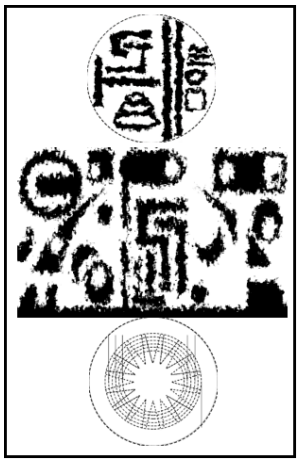


Figura G.7 Tercera iteración de marcadores cilíndricos. Se conserva un único marcador para el dedo índice.

G.4 Cuarta iteración

A pesar de que el marcador de la tercera iteración resultó ser apropiado de acuerdo al portal de Vuforia, al realizar pruebas se notó que tenía problemas para ser detectado durante el escalamiento dentro del entorno virtual de Unity. Este problema se debió a que sus puntos característicos no se distribuían uniformemente en la superficie del marcador, por lo que se creó la imagen mostrada en la figura G.8, donde los patrones son figuras geométricas con mayor densidad de formas para solventar este problema. En la figura G.9 se observa la distribución uniforme de sus puntos característicos, siendo óptima para su correcto escalamiento.

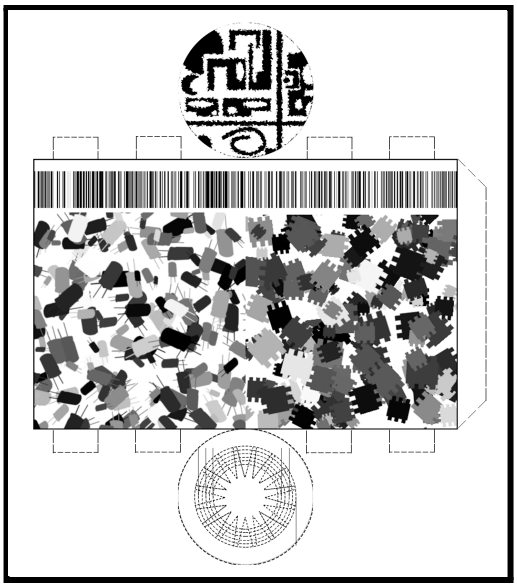


Figura G.8 Cuarta iteración de marcadores cilíndricos. Cambio de patrón de la imagen, se incluyen pestañas y líneas de recorte.

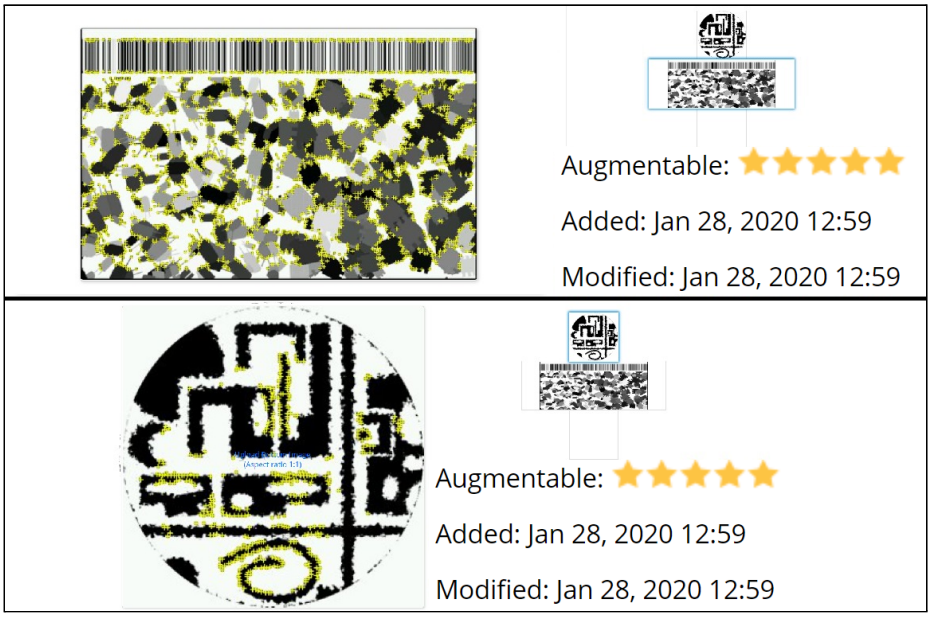


Figura G.9 Valoración en el portal de Vuforia del marcador cilíndrico del dedo índice de la cuarta iteración.

H Proceso iterativo de la mecánica de interacción

H.1 Primera iteración

Se implementó el uso de dos marcadores cilíndricos, hechos para los dedos índice y pulgar del usuario. Cada marcador tiene el modelo tridimensional de un cilindro en la punta para realizar las interacciones con los componentes virtuales. La interfaz cuenta con los siguientes elementos:

- Botón de despliegue del marcador con el texto “Desplegar”
- Botón para eliminar el marcador plano y volverlo a definir. Es de color rojo y con la leyenda “Destruir”
- Botón para activar y desactivar el flash del dispositivo con el texto “Flash”
- Botón con el texto “Manual” para que el dispositivo trate de enfocar la imagen que está observando con la cámara al realizar un toque en la pantalla
- Botón con el texto “Auto” para que el dispositivo trate de enfocar de manera automática
- Indicador con el texto “Collider” que se ilumina de color morado para indicar que los dos marcadores cilíndricos están en contacto
- Indicador cuadrado sin texto que se ilumina de color cyan cuando alguno de los marcadores cilíndricos está en contacto con un elemento tridimensional.

Se implementó un sistema de sujeción apoyado en los dos marcadores cilíndricos, donde con el movimiento de agarre entre ambos dedos y el modelo virtual, por lo que cuando ambos marcadores hacen contacto entre sí, se simula la acción de “tomar” y “soltar” el modelo; es decir, mientras ambos marcadores están en contacto con la acción de “tomar”, cambia la posición del modelo virtual, contrario a la acción de “soltar” cuando ambos marcadores son separados.

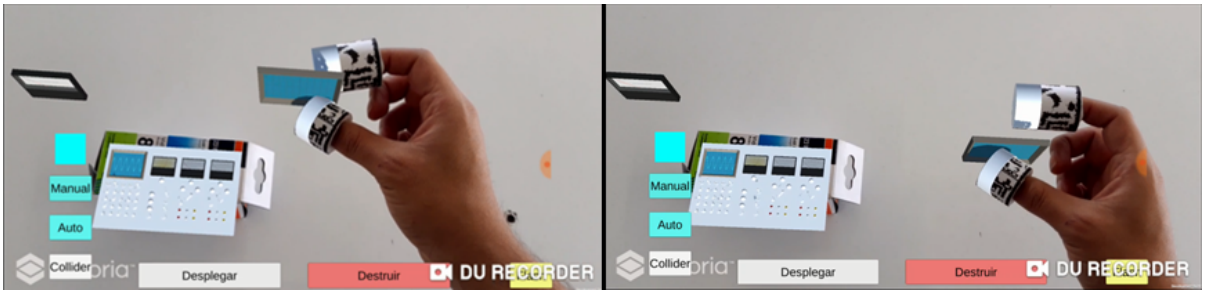


Figura H.1 Primera iteración de la manipulación de componentes virtuales con marcadores: a) los marcadores están en contacto entre sí y con un modelo virtual, el componente cambia su posición; b) cuando los marcadores se separan, el componente deja de moverse.

De esta iteración se observaron los siguientes aspectos:

- El sistema de sujeción podía resultar complejo para el usuario, dado que dependía que tres elementos estuvieran en contacto de manera simultánea y este contacto se debía mantener.
- El significado de los indicadores de colores para informar al usuario del contacto con objetos y entre los marcadores no resultaba claro
- El uso de los botones para el enfoque de la cámara era poco intuitivo.

H.2 Segunda iteración

Para solventar los problemas encontrados durante la primera iteración de la mecánica de interacción, se aplicaron las siguientes mejoras:

- Para disminuir la cantidad de elementos necesarios para la realización de la práctica, se omitió el uso del marcador cilíndrico del dedo pulgar
- Se pasó de la mecánica de sujeción a que el marcador cilíndrico interactuara con los componentes al tocarlos
- Se quitaron los indicadores de color que informaban al usuario del contacto entre los marcadores y un componente
- Se modificó el botón de la linterna del dispositivo a un botón circular que se enciende o apaga si la linterna lo hace
- Se quitaron los botones para el enfoque de la cámara, para ser suficiente con presionar la pantalla para que la *app* trate de enfocar la imagen
- Los botones para definir y modificar el marcador plano de despliegue se combinaron en uno solo, el cual es de color azul con el texto “*Definir*

marcador” si el marcador no está definido, y si lo esta, es de color rojo con el texto “*Redefinir marcador*”

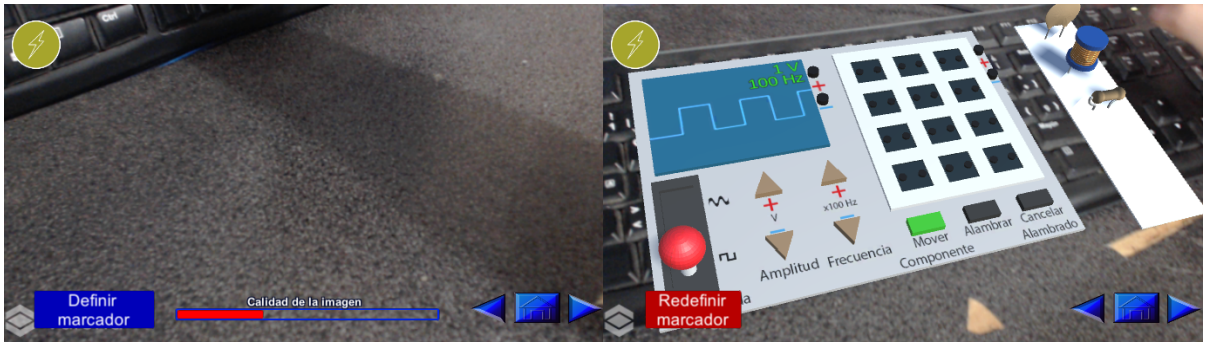


Figura H.2 Segunda iteración de la mecánica de interacción: a) interfaz sin el marcador definido; b) interfaz con el marcador definido.

Se observaron los siguientes aspectos de esta mecánica de interacción:

- Dado que el marcador interactuaba con los componentes al contacto, resultaba común que el usuario interactuara con un componente diferente por accidente
- Se identificó que la imagen del marcador cilíndrico, a pesar de tener los suficientes puntos característicos para considerarse adecuado, se dificultaba su detección por la poca densidad de puntos característicos dentro del escalamiento para el entorno virtual, por lo que se modifica como se describe en el apéndice G.4.

H.3 Tercera iteración

Se realizaron los siguientes cambios respecto a la segunda iteración, lo que permitió una mecánica con mayor facilidad de uso:

- Se cambió el modelo tridimensional en la punta del marcador cilíndrico a un cono, para facilitar el contacto con los componentes virtuales
- Se modificó la mecánica de interacción con los componentes para tener un tiempo de retraso, el cual sería mostrado gráficamente al usuario con un círculo de progresión llenado de forma radial mientras se mantiene el contacto entre el marcador cilíndrico y los componentes.

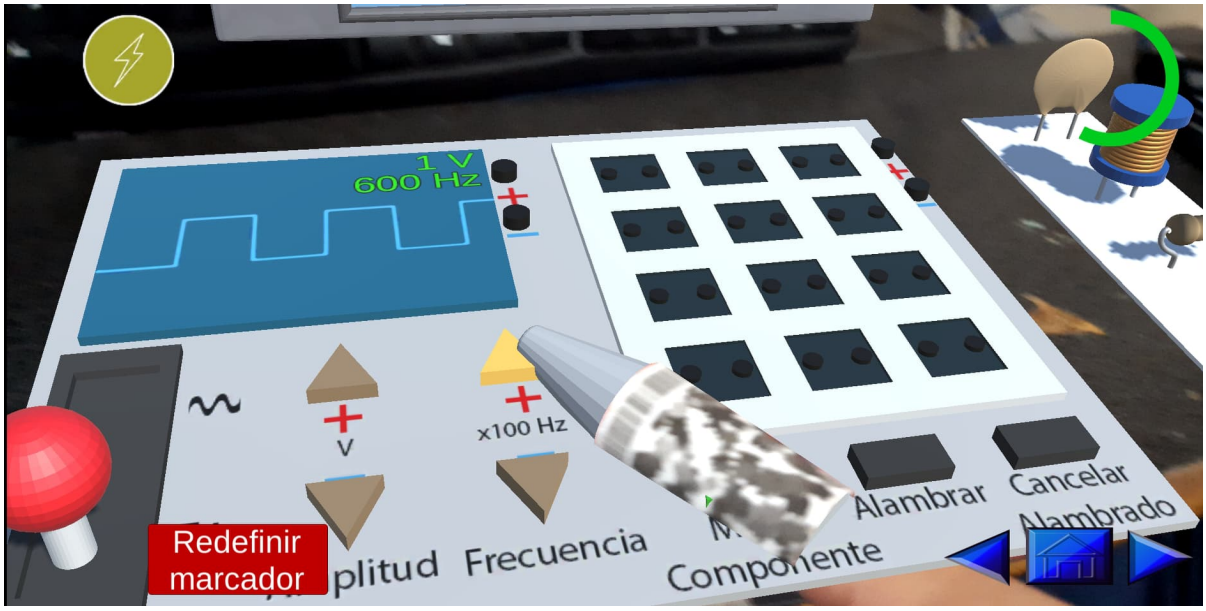


Figura H.3 Tercera iteración de la mecánica de interacción con el marcador y círculo de carga.

I Código final de la mecánica de interacción

I.1 Comportamiento del marcador cilíndrico

```
using UnityEngine;
public class ColliderManager_Principal : MonoBehaviour{
    private GameObject obj;
    public bool connected = false;
    public bool _enabled = false;
    //Componente que se está manipulando con los marcadores
    GameObject moviendo;
    public GameObject Toch_Indicador;
    UnityEngine.UI.Image Indicador;
    public GameObject Toch_Boton;
    UnityEngine.UI.Image _Boton;
    public ColliderManager_Secundario Toch_Sec;
    void Start() {
        _Boton = Toch_Boton.GetComponent<UnityEngine.UI.Image>();
        Indicador = Toch_Indicador.GetComponent<UnityEngine.UI.Image>();
        obj = GameObject.FindGameObjectWithTag("Comp"); }
    void OnTriggerEnter(Collider other) {
        if (other.tag == "Comp" && connected == false) {
            connected = true;
            moviendo = other.gameObject; } }
    void OnTriggerExit(Collider other) {
        if (other.tag == "Comp" && other.gameObject == moviendo)
        { connected = false; } }
    void Update() {
        if (connected == true && Toch_Sec._enabled == true) {
            Vector3 pos = moviendo.transform.position;
            pos = Vector3.Lerp(moviendo.transform.position,
transform.position, Time.deltaTime * 30);
            pos.x = pos.x;
            moviendo.transform.position = pos; } } }
```

I.2 Comportamiento de los elementos del circuito

```
using UnityEngine;
public class ComponentManager : MonoBehaviour{
    public bool comp_cap = false;
    public bool comp_res = false;
    public bool comp_ind = false;
    bool acc_carga = false;
    public float Value;
    public ColliderManager gripper;    //"Gripper"
    public UnityEngine.UI.Image Círculo;    //Círculo de carga
    int t_carga = 1;    //Tiempo que tarda en colocarse el componente
    public PinManager pin_asc;    //Pin al que entra el componente
    void Start() {
    //Obtiene "Gripper" y círculo de carga
    gripper = GameObject.FindGameObjectWithTag("Toch")
    .GetComponent<ColliderManager>();
    Círculo=GameObject.FindGameObjectWithTag
    ("Circulo_carga").GetComponent<UnityEngine.UI.Image>();
        //Verifica el tipo de componente al que está adjunto el "script"
        switch (gameObject.name.Substring(0, 3)) {
            case "cap": comp_cap = true; break;
            case "res": comp_res = true; break;
            case "ind": comp_ind = true; break; } }
    //Instanciamiento con valores
    void Update() {
        //Avanza el proceso de carga si es necesario
        if (acc_carga == true) {
            if (Círculo.fillAmount < 1)
            { Círculo.fillAmount += Time.deltaTime / t_carga; }
            else {
                acc_carga = false;
                gripper.acc_curso = true;
                Círculo.fillAmount = 0; } }
        //Mueve un componente si es el que lleva el "gripper"
        else if(gripper.acc_curso == true && gripper.obj_mov == gameObject){
            Vector3 pos = transform.position;
            pos = gripper.transform.position;
            pos.x = pos.x;
            transform.position = pos; } }
    private void OnTriggerEnter(Collider coll_entr) {
        //Verifica si se quiere mover al objeto con un "gripper"
        if(coll_entr.tag == "Toch" && !gripper.acc_curso && gripper.armando){
            //Verifica si el componente está puesto en un pin
            if (pin_asc != null)  {
                //Verifica si ninguno de los dos pines tienen cables
                if(!pin_asc.colocado_cable && !pin_asc.colocado_cable_osc &&
                !pin_asc.Pin_ot.GetComponent<PinManager>().colocado_cable &&
```

```
                !pin_asc.Pin_ot.GetComponent<PinManager>()
                .colocado_cable_osc) {
                    gripper.obj_mov = gameObject;
                    acc_carga = true; } }
            //Si no está puesto en un pin, lo trata de mover
            else {
                gripper.obj_mov = gameObject;
                acc_carga = true; } }
        //Verifica si entró a un pin
        else if (coll_entr.tag == "Pin") {
            pin_asc = coll_entr.gameObject.GetComponent<PinManager>(); } }
    private void OnTriggerExit(Collider coll_ext) {
        //Verifica si hay que cancelar la carga
        if (acc_carga == true && gripper.armando && coll_ext.tag == "Toch") {
            acc_carga = false;
            Círculo.fillAmount = 0;
            gripper.obj_mov = null; }
        //Verifica si entró a un pin
        else if (coll_ext.tag == "Pin") {
            pin_asc = null; } } }
```

I.3 Comportamiento de las terminales de conexión

```
using UnityEngine;
public class Terminal_Manager : MonoBehaviour{
    public ColliderManager gripper;    //"Gripper"
    public UnityEngine.UI.Image Círculo;    //Círculo de carga
    float t_alam = 1;   //Tiempo de carga para alambrar, en s
    float t_dest = 2;   //Tiempo de carga para eliminar un cable puesto, en s
    bool alam_inic = false;    //Bandera de inicio de rutina para alambrar
    public Material materialEncendido;    //Materiales de pines
    public Material materialApagado;
    public Material materialAlambrado;
    MeshRenderer meshRenderer;    //"Renderizador" del pin
    void Start() { gripper =
GameObject.FindGameObjectWithTag("Toch").GetComponent<ColliderManager>();
        Círculo =
GameObject.FindGameObjectWithTag("Circulo_carga").GetComponent<UnityEngine.UI
.Image>();
        meshRenderer = GetComponent<MeshRenderer>(); }

    void OnTriggerEnter(Collider coll_entr) {
         //Verifica si se está tratando de realizar una rutina de alambrado,
tipo de componente, estado de la variable que indica si se está alambrando,
si el pin ya tiene un componente y el "gripper" no trae cable de
este mismo pin
        if (coll_entr.tag == "Toch" && !gripper.armando &&
meshRenderer.material == materialApagado && gripper.cable != this) {
            alam_inic = true; } }
```

```
    void OnTriggerExit(Collider other) {
        //Verifica que se esté cancelando la acción de alambrar
        if (other.tag == "Toch" && !gripper.armando
        && meshRenderer.material == materialApagado) {
            alam_inic = false;            //Baja la bandera de alambrado
            Círculo.fillAmount = 0; } }   //Reinicia el círculo de carga
    void Update() {    //Si está arriba la bandera de alambrado
        if (alam_inic) {   //Avanza el círculo
            if (Círculo.fillAmount < 1)
            { Círculo.fillAmount += Time.deltaTime / t_alam; }
            //Si se llenó el círculo
            //Si el "gripper" no lleva un cable
            else if (gripper.cable == null) {
                Círculo.fillAmount = 0;    //Limpia el círculo
                //Indica al "gripper" que lleva un cable de este pin
                encenderPin_cable();  //Cambia el color del pin a verde
                alam_inic = false; }  //Termina la rutina de alambrado
          else if (gripper.cable != null) { //Si el "gripper"
          ya lleva un cable
                Círculo.fillAmount = 0; //Limpia el círculo
                //Manda la información de la conexión entre
                los dos pines a un gestor de conexiones
                encenderPin_cable();  //Cambia el color del pin a verde
                gripper.cable = null;  //Limpia la información del "gripper"
                alam_inic = false; } } }  //Termina la rutina de alambrado
    void encenderPin_cable() { meshRenderer.material = materialalambrado; } }
```

I.4 Comportamiento de las conexiones en el circuito

```
using UnityEngine;
public class WireManager : MonoBehaviour {
    //Bandera de destrucción en curso
    bool destrucción = false;
    float t_carga = 2;
    UnityEngine.UI.Image Círculo;
    AllWireManager allWireManager;
    //Pines en los que se construyó esta línea
    public PinManager PinIni;
    public PinManager PinFin;
    //Bandera de que se trata de un cable de osciloscopio
    public bool cable_osc = false;
    public ColliderManager gripper;
    void Start() {
    allWireManager = GameObject.FindGameObjectWithTag
    ("WireManag").GetComponent<AllWireManager>();
    gripper = GameObject.FindGameObjectWithTag
    ("Toch").GetComponent<ColliderManager>();
```

```
        Círculo = GameObject.FindGameObjectWithTag("Circulo")
        .GetComponent<UnityEngine.UI.Image>(); }
    void OnTriggerEnter(Collider coll_entr) {
        //Verifica que entre el “gripper” y esté en modo alambrado
        if (coll_entr.tag == "Toch" && !gripper.armando)
        { destrucción = true; } }
    void OnTriggerExit(Collider coll_sal)    {
        //Verifica que salga el “gripper” y esté en modo alambrado
        if (coll_sal.tag == "Toch" && !gripper.armando) {
            destrucción = false; Círculo.fillAmount = 0; } }
    void Update() {
        if (destrucción) {
            //Avanza el proceso de carga si es necesario
            if (Círculo.fillAmount < 1) { Círculo.fillAmount
            += Time.deltaTime/t_carga; }
            else { //Si ya se cargó el círculo
                //Reinicia los pines a su estado antes de colocar cables
                if (cable_osc) {
                    PinIni.colocado_cable_osc = false;
                    PinFin.colocado_cable_osc = false;
                    PinIni.apagarPin();
                    PinFin.encenderPin_cable(); }
                else {
if (PinIni.colocado_cable_osc) { //Verifica si hay cables de osciloscopio
PinIni.wire_osc.PinIni.apagarPin();  //Apaga el pin del osciloscopio
PinIni.colocado_cable_osc = false;  //Baja la bandera de pin del osciloscopio
en ambos pines
PinIni.wire_osc.PinIni.colocado_cable_osc = false;
allWireManager.DestroyLine(PinIni.wire_osc.gameObject); } //Destruye el cable
           if (PinFin.colocado_cable_osc) {  //Apaga el pin del osciloscopio
                       PinFin.wire_osc.PinIni.apagarPin();
                     //Baja la bandera del pin de osciloscopio en ambos casos
                       PinFin.colocado_cable_osc = false;
                       PinFin.wire_osc.PinIni.colocado_cable_osc = false;
allWireManager.DestroyLine(PinFin.wire_osc.gameObject); } //Destruye el cable
                    PinIni.colocado_cable = false;
                    PinFin.colocado_cable = false;
                    if (PinIni.pin_comp) PinIni.encenderPin();
                    else PinIni.apagarPin();
                    if (PinFin.pin_comp) PinFin.encenderPin();
                    else PinFin.apagarPin(); }
                Círculo.fillAmount = 0 ;} } }
    // Devuelve el otro pin (inicio o final) contrario del que se da
    public PinManager PairPin(PinManager Pin)    {
        PinManager res = null;
        if (Pin == PinIni) res = PinFin;
        else if (Pin == PinFin) res = PinIni;
        return res;    }
    public bool ContainsPin(PinManager Pin)  {
        bool res = false;
        if (Pin == PinIni || Pin == PinFin)  res = true;  return res; }
```

I.5 Comportamiento para la palanca de la señal de entrada

```
using UnityEngine;
public class SignalLeverManager : MonoBehaviour {
    ColliderManager gripper;    //"Gripper"
    UnityEngine.UI.Image Círculo;    //Círculo de carga
    bool acc_carga = false;    //Bandera de carga
    float t_carga = 1f;    //Tiempo de carga
    Animator animator;
 MeshRenderer pantalla; //Material de la pantalla donde se despliega la señal
    //Imagen de las señales
    Texture señalCuadrada;
    Texture señalSenoidal;
    void Start() {
        gripper = GameObject.FindGameObjectWithTag("Toch")
        .GetComponent<ColliderManager>();
        Círculo = GameObject.FindGameObjectWithTag("Circulo_carga")
        .GetComponent<UnityEngine.UI.Image>();
        animator = gameObject.GetComponent<Animator>();
        pantalla = transform.parent.parent.Find
        ("Panel prueba").Find("Panel Protofuente").Find("proto_comp -
        copia").Find("Pantalla").GetComponent<MeshRenderer>();

        if (VisualizaciónComp.ReiniciarTablero) {
            ComponentesAlambrados.SeñalEntrada = 1;
            pantalla.material.mainTexture = señalCuadrada; }
        if (ComponentesAlambrados.SeñalEntrada == 2) {
            pantalla.material.mainTexture = señalSenoidal;
            animator.SetBool("ToSineAnim", true);
            animator.SetBool("ToSquareAnim", false); } }

    void Update() {
        //Avanza el proceso de carga si es necesario
        if (acc_carga == true)        {
        if(Círculo.fillAmount<1){Círculo.fillAmount+=Time.deltaTime/t_carga;}
            else {
                acc_carga = false;
                Círculo.fillAmount = 0;
                //Verifica estado actual y deseado de la señal
                //La señal sinusoidal pasa a cuadrada
                if (ComponentesAlambrados.SeñalEntrada == 1)  {
                    //Inicia la animación de movimiento
                    animator.SetBool("ToSineAnim", true);
                    animator.SetBool("ToSquareAnim", false);
                    //Cambia el tipo de señal
                    ComponentesAlambrados.SeñalEntrada = 2;
                    //Cambia la textura de la pantalla
                    pantalla.material.mainTexture = señalSenoidal; }
```

```
                //Está señal senoidal y pasa a cuadrada
                else if (ComponentesAlambrados.SeñalEntrada == 2) {
                    //Inicia la animación de movimiento
                    animator.SetBool("ToSineAnim", false);
                    animator.SetBool("ToSquareAnim", true);
                    //Cambia el tipo de señal
                    ComponentesAlambrados.SeñalEntrada = 1;
                    //Cambia la textura de la pantalla
                    pantalla.material.mainTexture = señalCuadrada; } } } }

    private void OnTriggerEnter(Collider coll_ent) {
        //Verifica si se quiere cambiar la señal
        if (coll_ent.tag == "Toch") { acc_carga = true; } }
    private void OnTriggerExit(Collider coll_ext)  {
        //Verifica si se quiere cancelar el cambio
        if (coll_ext.tag == "Toch") {
            acc_carga = false;
            Círculo.fillAmount = 0; } } }
```

I.6 Comportamiento para las propiedades de la señal de entrada

```
using UnityEngine;
public class GeneratorButtonManager : MonoBehaviour {
    public ColliderManager gripper;    //”Gripper”
    public UnityEngine.UI.Image Círculo;    //Círculo de carga
    MeshRenderer voltajeDisplay;    //Despliegue de voltaje y frecuencia
    MeshRenderer FreqDisplay;
    //Imágenes de valores de voltaje y frecuencia
    Texture[] voltajes = new Texture[9];
    Texture[] frecuencias = new Texture[9];
    bool acc_carga = false;    //Bandera de acción en curso
    float t_carga = 1.4f;    //Tiempo de carga
    //Materiales del botón
    Material materialApagado;
    Material materialPresionado;
    MeshRenderer buttonRenderer;    //“Renderizador” del botón
    void Start() {
        gripper = GameObject.FindGameObjectWithTag("Toch")
        .GetComponent<ColliderManager>();
        Círculo = GameObject.FindGameObjectWithTag("Circulo_carga")
        .GetComponent<UnityEngine.UI.Image>();
        voltajeDisplay = transform.parent.parent.parent.Find("VDisplay")
        .GetComponent<MeshRenderer>();
        FreqDisplay = transform.parent.parent.parent.Find("HDisplay")
        .GetComponent<MeshRenderer>();
        buttonRenderer = gameObject.GetComponent<MeshRenderer>();
        if (VisualizaciónComp.ReiniciarTablero)  {
            ComponentesAlambrados.VoltajeEntrada = 1;
```

```
            ComponentesAlambrados.Frecuencia = 100;
ComponentesAlambrados.Periodo = 1 / (ComponentesAlambrados.Frecuencia); }
        voltajeDisplay.material.mainTexture =
voltajes[(int)ComponentesAlambrados.VoltajeEntrada - 1];
        FreqDisplay.material.mainTexture =
frecuencias[(int)(ComponentesAlambrados.Frecuencia / 100) - 1];  }

    void Update()  {
        if (acc_carga == true)  {
if (Círculo.fillAmount<1){ Círculo.fillAmount += Time.deltaTime / t_carga; }
            else {
                Círculo.fillAmount = 0;
//Verifica el tipo de botón que se presionó y que el valor actual pueda
cambiar
//...cambia los valores en el statusManager y los desplegados en la pantalla
if (name == "Boton_Amp_1" && ComponentesAlambrados.VoltajeEntrada < 9) {
                    //Aumenta el voltaje
                    ComponentesAlambrados.VoltajeEntrada++;
                    //Cambia la textura del despliegue
                    voltajeDisplay.material.mainTexture =
voltajes[(int)ComponentesAlambrados.VoltajeEntrada - 1];  }
if (name == "Boton_Amp_2" && ComponentesAlambrados.VoltajeEntrada > 1)  {
                    //Disminuye el voltaje
                    ComponentesAlambrados.VoltajeEntrada--;
                    //Cambia la textura del despliegue
                    voltajeDisplay.material.mainTexture =
voltajes[(int)ComponentesAlambrados.VoltajeEntrada - 1];  }
                if (name == "Boton_Frec_1" &&
ComponentesAlambrados.Frecuencia < 900)  {
                    //Aumenta la frecuencia
                    ComponentesAlambrados.Frecuencia += 100;
                    ComponentesAlambrados.Periodo = 1 /
(ComponentesAlambrados.Frecuencia);
                    //Cambia la textura del despliegue
                    FreqDisplay.material.mainTexture =
frecuencias[(int)(ComponentesAlambrados.Frecuencia / 100) - 1]; }
                if (name == "Boton_Frec_2" &&
ComponentesAlambrados.Frecuencia > 100) {
                    //Disminuye la frecuencia
                    ComponentesAlambrados.Frecuencia-=100;
                    ComponentesAlambrados.Periodo = 1 /
(ComponentesAlambrados.Frecuencia);
                    //Cambia la textura del despliegue
                    FreqDisplay.material.mainTexture =
frecuencias[(int)(ComponentesAlambrados.Frecuencia / 100) - 1]; } } } }

    private void OnTriggerEnter(Collider coll_ent) {
        //Verifica si se quiere cambiar la señal
        if (coll_ent.tag == "Toch") {
            acc_carga = true;
            buttonRenderer.material = materialPresionado; } }
```

```
    private void OnTriggerExit(Collider coll_ext) {
        //Verifica si se quiere cancelar el cambio
        if (coll_ext.tag == "Toch") {
            acc_carga = false;
            buttonRenderer.material = materialApagado;
            Círculo.fillAmount = 0; } } }
```

I.7 Comportamiento del botón “Mover componentes”

```
using UnityEngine;

public class Button_Motion : MonoBehaviour{
    //Materiales del botón
    Material materialApagado;
    Material materialButtonMotion;
    MeshRenderer buttonMotionRenderer; //“Renderizador” del botón
MeshRenderer buttonWiringRenderer; /“Renderizador” del otro botón
    //“Gripper”
    public ColliderManager gripper;
    //Circulo de carga
    public UnityEngine.UI.Image Círculo;
    //Tiempo de carga (s)
    float t_carga = 1f;
    //Variable de acción de presionar en proceso
    bool acc_carga = false;
    void Start() {
        gripper =
GameObject.FindGameObjectWithTag("Toch").GetComponent<ColliderManager>();
        Círculo =
GameObject.FindGameObjectWithTag("Circulo_carga").GetComponent<UnityEngine.UI
.Image>();
        buttonMotionRenderer = GetComponent<MeshRenderer>(); }

    void OnTriggerEnter(Collider coll_entr)    {
     //Verifica si se usa el “gripper”, si se está
     alambrando y si no trae cables
if (coll_entr.tag == "Toch" && !gripper.armando && gripper.cable == null) {
            acc_carga = true;
            buttonMotionRenderer.material = materialButtonMotion; }
    void OnTriggerExit(Collider other) {
        //Verifica si hay que cancelar la carga
        if (acc_carga == true) {
            acc_carga = false;
            Círculo.fillAmount = 0;
            buttonMotionRenderer.material = materialApagado; } }
    void Update() {
        //Avanza el proceso de carga si es necesario
        if (acc_carga == true) {
            if (Círculo.fillAmount < 1)
```

```
            { Círculo.fillAmount += Time.deltaTime / t_carga; }
            else  {
                acc_carga = false;
                Círculo.fillAmount = 0;
                //Cambia el método de uso del "gripper"
                gripper.armando = true;
                buttonWiringRenderer.material = materialApagado;  }  }  }  }
```

I.8 Comportamiento del botón "Alambrar"

```
using UnityEngine;

public class Button_Wiring : MonoBehaviour {
    //Materiales del botón
    Material materialApagado;
    Material materialButtonWiring;
    MeshRenderer buttonWiringRenderer;    //"Renderizador" del botón
MeshRenderer buttonMotionRenderer; /"Renderizador" del otro botón
    public ColliderManager gripper;    //"Gripper"
    public UnityEngine.UI.Image Círculo;    //Circulo de carga
    float t_carga = 1f;    //Tiempo de carga, en s
    bool acc_carga = false;    //Variable de acción de presionar en proceso

    void Start() {
        gripper = GameObject.FindGameObjectWithTag("Toch")
        .GetComponent<ColliderManager>();
        Círculo = GameObject.FindGameObjectWithTag("Circulo_carga")
        .GetComponent<UnityEngine.UI.Image>();
        buttonWiringRenderer = GetComponent<MeshRenderer>(); }
    void OnTriggerEnter(Collider coll_entr) {
        //Verifica si se quiere cambiar a alambrado:
        //Si se hace con el "gripper", si está armando
        y si no trae componentes
        if(coll_entr.tag == "Toch" && gripper.armando && !gripper.acc_curso){
            acc_carga = true;
            buttonWiringRenderer.material = materialButtonWiring; } }
    void OnTriggerExit(Collider other)    {
        //Verifica si hay que cancelar la carga
        if (acc_carga == true) {
            acc_carga = false;
            Círculo.fillAmount = 0;
            buttonWiringRenderer.material = materialApagado; } }
    void Update() {
        //Avanza el proceso de carga si es necesario
        if (acc_carga == true) {
            if (Círculo.fillAmount < 1)
            { Círculo.fillAmount += Time.deltaTime / t_carga; }
            else  {
```

```
                acc_carga = false;
                Círculo.fillAmount = 0;
                //Cambia el método de uso del “gripper”
                gripper.armando = false;
                buttonMotionRenderer.material = materialApagado; } } } }
```

I.9 Comportamiento del botón “Cancelar alambrado”

```
using UnityEngine;

public class Button_Cancel : MonoBehaviour{
    public ColliderManager gripper;    //“Gripper”
    public UnityEngine.UI.Image Círculo;    //Círculo de cancelación
    float t_carga = 1f;    //Tiempo de cancelación, en s
    bool acc_carga = false;    //Variable de acción de presionar en proceso
    //Materiales del botón
    Material materialApagado;
    Material materialButtonCancel;
    MeshRenderer buttonCancelRenderer;    //Renderizador del botón

    void Start() {
        gripper = GameObject.FindGameObjectWithTag("Toch")
        .GetComponent<ColliderManager>();
        Círculo = GameObject.FindGameObjectWithTag("Circulo_dest")
        .GetComponent<UnityEngine.UI.Image>();
        buttonCancelRenderer = GetComponent<MeshRenderer>(); }

    void OnTriggerEnter(Collider coll_entr) {
        //Verifica si se quiere cancelar un alambrado. Si el objeto es el
        “gripper”, se está alambrando y trae un cable
if (coll_entr.tag == "Toch" && !gripper.armando && gripper.cable != null) {
            acc_carga = true;
            buttonCancelRenderer.material = materialButtonCancel; } }
    void OnTriggerExit(Collider other) {
        if (acc_carga == true) { //Verifica si hay que cancelar la carga
            acc_carga = false;
            Círculo.fillAmount = 0;
            buttonCancelRenderer.material = materialApagado; } }
    void Update() {
        //Avanza el proceso de carga si es necesario
        if (acc_carga == true) {
            if (Círculo.fillAmount < 1)
            { Círculo.fillAmount += Time.deltaTime / t_carga; }
            else {
                Círculo.fillAmount = 0; //Limpia el círculo
                //Regresa el pin que trae el cable a su estado anterior
                if (gripper.cable.pin_comp)  {
                    gripper.cable.alam_inic = false;
```

```
    gripper.cable.encenderPin();  }
else  {
    gripper.cable.alam_inic = false;
    gripper.cable.alam_inic_osc = false;
    gripper.cable.apagarPin();  }
//Indica al “gripper” que ya no lleva cable
gripper.cable = null;
gripper.alam_inic = false;
acc_carga = false;
buttonCancelRenderer.material = materialApagado; } } }
```

J Código para la visualización de señales en el osciloscopio

J.1 Comprobación del correcto alambrado del circuito

```
using UnityEngine;

public class StatusManager : MonoBehaviour{
    //Objeto que verifica el alambrado del circuito (en otro “GameObject”)
    public AllWireManager allWireManager;
    public ColliderManager gripper;    //“Gripper”
    public UnityEngine.UI.Image Círculo;    //Círculo de carga
    float[] generador;    //Valores no públicos
    bool acc_carga = false;    //Bandera de colisionador
    float t_carga = 1f;    //Tiempo de carga
    //Ventana emergente de aviso de alambrado incorrecto
    GameObject ventanaEmergente;
    void Start() {
        gripper = GameObject.FindGameObjectWithTag("Toch")
        .GetComponent<ColliderManager>();
        Círculo = GameObject.FindGameObjectWithTag("Circulo_carga")
        .GetComponent<UnityEngine.UI.Image>();
        allWireManager = transform.parent.Find("WireManager_obj").
        GetComponent<AllWireManager>();
        ventanaEmergente = GameObject.Find("Canvas")
        .transform.Find("Aviso Alambrado Erroneo").gameObject; }

    void Update() {
        if (acc_carga == true) {
            if (Círculo.fillAmount < 1)
            { Círculo.fillAmount += Time.deltaTime / t_carga; }
            else {
                acc_carga = false;
                Círculo.fillAmount = 0;
                if (allWireManager.ComprobarCircuito()) {
                    Debug.Log("Circuito alambrado correctamente");
//Llama los valores de los componentes y los guarda en variables estáticas
```

```
                    float[] valores = allWireManager.Valores();
                    ComponentesAlambrados.ResistorValor = valores[0];
                    ComponentesAlambrados.InductorValor = valores[1];
                    ComponentesAlambrados.CapacitorValor = valores[2];
                    ComponentesAlambrados.ComponenteSalida = (int)valores[3];
                    //Guarda el estado actual del tablero
                    Destroy(ComponentesAlambrados.Board);
                    ComponentesAlambrados.Board =
Instantiate(GameObject.FindWithTag("FullBoard"));
                    ComponentesAlambrados.Board.SetActive(false);
                    DontDestroyOnLoad(ComponentesAlambrados.Board);
                    //Cambio de escena
UnityEngine.SceneManagement.SceneManager.LoadScene("OsciloscopioEscena"); }
                else { ventanaEmergente.SetActive(true); } } } }

    private void OnTriggerEnter(Collider coll_ent) {
        //Verifica si se quiere comprobar el circuito
        if (coll_ent.tag == "Toch") { acc_carga = true; } }
    private void OnTriggerExit(Collider coll_ext) {
    if (coll_ext.tag == "Toch") {
            acc_carga = false;
            Círculo.fillAmount = 0; } }
    private void OnDestroy()    {
        VisualizaciónComp.ReiniciarTablero = false; } }
```

J.2 Visualización y modificación de la señal del generador

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class LineEntrada : MonoBehaviour {
    #region Object Variables
    private GameObject lineGeneratorPrefab;
    private GameObject linePointPrefab;
    float lengthOfLineRenderer = 10f-1.35f;
    public int cambio = 0;
    float amplitud = 0.17f;
    float frecuencia = 2.7746f;
    GameObject newLineGen;
    LineRenderer lRend;
    Vector3 pos;
    GameObject[] allPoints;
    Vector3[] allPointPositions;
    #endregion
    float i = 0;
    public Boton0 entrada;
```

```
    public Boton1 salida;
    float W = ComponentesAlambrados.Frecuencia*2f * Mathf.PI;// frecuencia
    float T = ComponentesAlambrados.Periodo; //Periodo
    int cua = 0;
    float SeñalCuadra=0;
    //2 sinusoidal,1 cuadrada
    int SeñalEntrada = ComponentesAlambrados.SeñalEntrada;
    public TextMeshProUGUI CH1;
    public TextMeshProUGUI MS;
    float V = ComponentesAlambrados.VoltajeEntrada; // voltaje pico
    public void Start() {
        i = 0;
        //Verifica cada tipo de señal
        if (SeñalEntrada==1) {
        //Despliega puntos para dibujar la señal cuadrada
        hasta una longitud definida
            while (i < lengthOfLineRenderer) {
                for (cua = 0; cua < 25; cua++) {
                    SeñalCuadra = SeñalCuadra + Heaviside(i * (frecuencia /
1000), cua * T) - Heaviside(i * (frecuencia / 1000), (cua * T) + T / 2); }
                pos = new Vector3(i, amplitud * V * SeñalCuadra, 0);
                CreatePointMarker(pos);
                i = i + 0.05F;
                SeñalCuadra = 0; } }

        if (SeñalEntrada==2) {
//Despliega puntos para dibujar la señal sinusoidal
hasta una longitud definida
            while (i < lengthOfLineRenderer) {
pos = new Vector3(i,amplitud*V * Mathf.Sin(i * (frecuencia / 1000) * W), 0);
                CreatePointMarker(pos);
                i = i + 0.05F; } }
        GenerateNewLine(); }
//Función tren de pulsos a partir de un periodo “u” y un punto “t”
    public int Heaviside(float x, float u) {
        int a = 0;
        if (x > u) a = 1;
        return a; }
//Función para cambiar la amplitud de la señal activa. Lee el valor del
“slider” y asigna una amplitud a partir de intervalos
    public void CambioAmplitud(float amp) {
        if (amp >= 0.1 && amp <= 0.17) {
            amplitud = 0.17f;
            CH1.text = "CH1 5V"; }
        if (amp > 0.17 && amp <= 0.425) {
            amplitud = 0.425f;
            CH1.text = "CH1 2V"; }
        if (amp > 0.425 && amp <= 0.85) {
            amplitud = 0.85f;
            CH1.text = "CH1 1V"; }
        if (amp > 0.85 && amp <= 1.7) {
```

```
            amplitud = 1.7f;
            CH1.text = "CH1 500 mV"; }
        if (amp > 1.7 && amp <= 4.2) {
            amplitud = 4.25f;
            CH1.text = "CH1 200 mV"; }
ClearAllPoints(); //Destruye la línea que se había creado anteriormente
        Destroy(newLineGen);
        Destroy(lRend);
        i = 0;
        //Crea la línea con el nuevo valor de amplitud
        if (SeñalEntrada == 1) {
            while (i < lengthOfLineRenderer) {
                for (cua = 0; cua < 25; cua++) {
                    SeñalCuadra = SeñalCuadra + Heaviside(i * (frecuencia /
1000), cua * T) - Heaviside(i * (frecuencia / 1000), (cua * T) + T / 2); }
                pos = new Vector3(i, amplitud * V * SeñalCuadra, 0);
                CreatePointMarker(pos);
                i = i + 0.05F;
                SeñalCuadra = 0; } }
        if (SeñalEntrada == 2) {
            while (i < lengthOfLineRenderer) {
                pos = new Vector3(i, amplitud * V * Mathf.Sin(i * (frecuencia
/ 1000) * W), 0);
                CreatePointMarker(pos);
                i = i + 0.05F; } }
        ClearAllPoints();
        Destroy(newLineGen);
        Destroy(lRend);
        GenerateNewLine(); }
//Función para cambiar el periodo de visualización de la señal activa. Lee el
valor del "slider" y asigna una frecuencia a partir de intervalos
    public void CambioFrecuencia(float frec) {
        if (frec >= 0.2 && frec <= 0.2775) {
            frecuencia = 0.2775f;
            MS.text = "0.2 ms"; }
        if (frec > 0.2775 && frec <= 0.6937) {
            frecuencia = 0.6937f;
            MS.text = "0.5 ms"; }
        if (frec > 0.6937 && frec <= 1.3873) {
            frecuencia = 1.3873f;
            MS.text = "1 ms"; }
        if (frec > 1.3873 && frec <= 2.8) {
            frecuencia = 2.7746f;
            MS.text = "2 ms"; }
//Destruye la línea que se había creado anteriormente
ClearAllPoints();
        Destroy(newLineGen);
        Destroy(lRend);
        i = 0;
        //Crea la línea con el nuevo valor de frecuencia
        if (SeñalEntrada == 1) {
```

```
            while (i < lengthOfLineRenderer) {
                for (cua = 0; cua < 25; cua++) {
                    SeñalCuadra = SeñalCuadra + Heaviside(i * (frecuencia /
1000), cua * T) - Heaviside(i * (frecuencia / 1000), (cua * T) + T / 2); }
                pos = new Vector3(i, amplitud * V * SeñalCuadra, 0);
                CreatePointMarker(pos);
                i = i + 0.05F;
                SeñalCuadra = 0; } }
        if (SeñalEntrada == 2) {
            while (i < lengthOfLineRenderer) {
                pos = new Vector3(i, amplitud * V * Mathf.Sin(i * (frecuencia
/ 1000) * W), 0);
                CreatePointMarker(pos);
                i = i + 0.05F; } }
        ClearAllPoints();
        Destroy(newLineGen);
        Destroy(lRend);
        GenerateNewLine(); }
   //Función para cambiar la señal activa a la que se le modifica la amplitud
    public void CambioSeñal() {
        ClearAllPoints();
        Destroy(newLineGen);
        Destroy(lRend);
        ClearAllPoints();
        Destroy(newLineGen);
        Destroy(lRend);
        GenerateNewLine(); }

    public void CreatePointMarker(Vector3 pointPosition) {
        // Crea la imagen de un punto en una posición
        Instantiate(linePointPrefab, pointPosition, Quaternion.identity);
        linePointPrefab.SetActive(true); }

    public void ClearAllPoints() {
        //Obtiene un arreglo con los “GameObject” con el “tag” 'PointMarker'
        allPoints = GameObject.FindGameObjectsWithTag("PointMarker");
        //Destruye los objetos dentro del arreglo
        foreach (GameObject p in allPoints) { Destroy(p); } }
    public void GenerateNewLine() {
        //Obtiene un arreglo con los “GameObject” con el “tag” 'PointMarker'
        allPoints = GameObject.FindGameObjectsWithTag("PointMarker");
        // “Instancia” un arreglo de Vector3 de la longitud del arreglo
        allPointPositions = new Vector3[allPoints.Length];
        //Si hay dos o más puntos “instanciados”
        if (allPoints.Length >= 2) {
        //Ciclo para asignar los objetos Vector3 a la posición
        de los objetos en el arreglo “allPoints”
            for (int i = 0; i < allPoints.Length; i++) {
                allPointPositions[i] = allPoints[i].transform.position; }
     //Función para crear la línea
     SpawnLineGenerator(allPointPositions); } }
```

```
    public void SpawnLineGenerator(Vector3[] linePoints) {
            //Crea un objeto “LineHolder”
            newLineGen = Instantiate(lineGeneratorPrefab);
            //Obtiene el componente “LineRenderer” del objeto
            lRend = newLineGen.GetComponent<LineRenderer>();
            //Define las posiciones de “LineRenderer” a la cantidad de puntos
            lRend.positionCount = linePoints.Length;
            //Define las posiciones de “LineRenderer” usando
            el arreglo “LinePoints”
            lRend.SetPositions(linePoints); } }
```

J.3 Visualización y modificación de la señal de salida del circuito

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class Line2 : MonoBehaviour {
    #region Object Variables
    private GameObject lineGeneratorPrefab;
    private GameObject linePointPrefab;
    float lengthOfLineRenderer = 10f-1.35f;
    public int cambio = 0;
    float amplitud = 0.17f;
    float frecuencia = 2.7746f;
    GameObject newLineGen2;
    LineRenderer lRend2;
    Vector3 pos2;
    GameObject[] allPoints2;
    Vector3[] allPointPositions2;
    #endregion
    float i = 0;
    public Boton0 entrada;
    public Boton1 salida;
    int  EntradaActual = ComponentesAlambrados.SeñalEntrada;
    float res= ComponentesAlambrados.ResistorValor;
    float L = ComponentesAlambrados.InductorValor;
    float C = ComponentesAlambrados.CapacitorValor;
    float W = ComponentesAlambrados.Frecuencia* 2f * Mathf.PI;
    // 1= capacitor, 2=Inductor, 3=Resistencia, Otro= error
    int VolSalida = ComponentesAlambrados.ComponenteSalida;
    float Xl; //Reactancia inductiva
    float Xc; //Reactancia capacitiva
    float Z; //Reactancia Total
    float T=ComponentesAlambrados.Periodo;//Periodo
```

```
    float Theta; //Ángulo de desfase
    float I; //Intensidad de corriente
    float F = 0;
    float SeñalCuadra1 = 0;
    float h = 0.00001f;
    VoltagesRLC testApp;
    float V = ComponentesAlambrados.VoltajeEntrada; // voltaje pico
    public TextMeshProUGUI CH2;
    //Los valores de resistencias internas también están declarados en la
función “VoltagesRLC”
    float Rg = 50f;
    float Rl = 150f;

    public void Start() { // Cálculo de  reactancias
        i = 0;
        Xc = 1 / (W * C);
        Xl = W * L;
Z = Mathf.Sqrt(Mathf.Pow(res + Rl + Rg, 2) + (Mathf.Pow(Xl - Xc, 2)));
        I = V / Z;
        Theta = Mathf.Atan((Xl - Xc) / res);
//Función que calcula los valores de salida para una señal sinusoidal a
partir del método RK4, y crea un arreglo con estos valores
        testApp = new VoltagesRLC(res, L, C, V, h, T, 0, 0);
if (EntradaActual==2){//Si el circuito está alimentado por
una señal sinusoidal
            while (i < lengthOfLineRenderer) {
//Calcula los valores de la señal de salida a partir del método de fasores
para puntos definidos
                if (VolSalida == 1) {
F = (-Xc) * I * (Mathf.Cos(-Theta + (W * i * (frecuencia / 1000)))); }
                if (VolSalida == 2) {
F = (Xl) * I * (Mathf.Cos(-Theta + (W * i * (frecuencia / 1000)))); }
                if (VolSalida == 3) {
F = (res) * I * (Mathf.Sin(-Theta + (W * i * (frecuencia / 1000)))); }
                pos2 = new Vector3(i, amplitud * F, 0);
//Despliega puntos para dibujar la señal cuadrada hasta una longitud definida
                CreatePointMarker(pos2);
                i = i + 0.05F; } }
if (EntradaActual==1) {//Si el circuito está alimentado por una señal
periódica cuadrada
         if (VolSalida == 1) { //Si se alambró un condensador al osciloscopio
                while (i < lengthOfLineRenderer) {
//Realiza un submuestreo del arreglo de la solución ya calculada
para puntos definidos
                    SeñalCuadra1 = testApp.solveVc(i * (frecuencia/1000));
                    pos2 = new Vector3(i, amplitud * SeñalCuadra1, 0);
//Despliega puntos para dibujar la señal cuadrada hasta una longitud definida
                    CreatePointMarker(pos2);
                    i = i + 0.05F; } }
            if (VolSalida == 2) {//Si se alambró un inductor al osciloscopio
                while (i < lengthOfLineRenderer) {
```

```
//Realiza un submuestreo del arreglo de la solución ya calculada
para puntos definidos
                    SeñalCuadra1 = testApp.solveVl(i * (frecuencia / 1000));
                    pos2 = new Vector3(i, amplitud * SeñalCuadra1, 0);
//Despliega puntos para dibujar la señal cuadrada hasta una longitud definida
                    CreatePointMarker(pos2);
                    i = i + 0.05F; } }
        if (VolSalida == 3) { //Si se alambró una resistencia al osciloscopio
                while (i < lengthOfLineRenderer) {
//Realiza un submuestreo del arreglo de la solución ya calculada
para puntos definidos
                    SeñalCuadra1 = testApp.solveVr(i * (frecuencia / 1000));
                    pos2 = new Vector3(i, amplitud * SeñalCuadra1, 0);
//Despliega puntos para dibujar la señal cuadrada hasta una longitud definida
                    CreatePointMarker(pos2);
                    i = i + 0.05F; } } }
        GenerateNewLine(); }
//Función para cambiar la amplitud de la señal activa. Lee el valor del
“slider” y asigna una amplitud a partir de intervalos
    public void CambioAmplitud(float amp) {
        if (amp >= 0.1 && amp <= 0.17) {
            amplitud = 0.17f;
            CH2.text = "CH2 5V"; }
        if (amp > 0.17 && amp <= 0.425) {
            amplitud = 0.425f;
            CH2.text = "CH2 2V"; }
        if (amp > 0.425 && amp <= 0.85) {
            amplitud = 0.85f;
            CH2.text = "CH2 1V"; }
        if (amp > 0.85 && amp <= 1.7) {
            amplitud = 1.7f;
            CH2.text = "CH2 500mV"; }
        if (amp > 1.7 && amp <= 4.2) {
            amplitud = 4.25f;
            CH2.text = "CH2 200mV"; }
      ClearAllPoints(); //Destruye la línea que se había creado anteriormente
        Destroy(newLineGen2);
        Destroy(lRend2);
        i = 0;
        //Crea la línea con el nuevo valor de amplitud
        if (EntradaActual == 2) {
            while (i < lengthOfLineRenderer) {
//Calcula los valores de la señal de salida a partir del método de fasores
para puntos definidos
                if (VolSalida == 1) {
F = (-Xc) * I * (Mathf.Cos(-Theta + (W * i * (frecuencia / 1000)))); }
                if (VolSalida == 2) {
F = (Xl) * I * (Mathf.Cos(-Theta + (W * i * (frecuencia / 1000)))); }
                if (VolSalida == 3) {
F = (res) * I * (Mathf.Sin(-Theta + (W * i * (frecuencia / 1000)))); }
                pos2 = new Vector3(i, amplitud * F, 0);
```

```
                CreatePointMarker(pos2);
                i = i + 0.05F; } }
        if (EntradaActual == 1) {
            if (VolSalida == 1) {
                while (i < lengthOfLineRenderer) {
//Realiza un submuestreo del arreglo de la solución ya calculada para puntos
definidos, para generar la solución del voltaje del componente seleccionado
                    SeñalCuadra1 = testApp.solveVc(i * (frecuencia / 1000));
                    pos2 = new Vector3(i, amplitud * SeñalCuadra1, 0);
                    CreatePointMarker(pos2);
                    i = i + 0.05F; } }
            if (VolSalida == 2) {
                while (i < lengthOfLineRenderer) {
                    SeñalCuadra1 = testApp.solveVl(i * (frecuencia / 1000));
                    pos2 = new Vector3(i, amplitud * SeñalCuadra1, 0);
                    CreatePointMarker(pos2);
                    i = i + 0.05F; } }
            if (VolSalida == 3) {
                while (i < lengthOfLineRenderer) {
                    SeñalCuadra1 = testApp.solveVr(i * (frecuencia / 1000));
                    pos2 = new Vector3(i, amplitud * SeñalCuadra1, 0);
                    CreatePointMarker(pos2);
                    i = i + 0.05F; } } }
        ClearAllPoints();
        Destroy(newLineGen2);
        Destroy(lRend2);
        GenerateNewLine(); }
//Función para cambiar el periodo de visualización de la señal activa. Lee el
valor del “slider” y asigna una frecuencia a partir de intervalos
    public void CambioFrecuencia(float frec) {
        if (frec>=0.2&&frec<=0.2775) {
            frecuencia = 0.2775f; }
        if (frec > 0.2775 && frec <= 0.6937) {
            frecuencia = 0.6937f; }
        if (frec > 0.6937 && frec <= 1.3873) {
            frecuencia = 1.3873f; }
        if (frec > 1.3873 && frec <= 2.78) {
            frecuencia = 2.7746f; }
     ClearAllPoints();  //Destruye la línea que se había creado anteriormente
        Destroy(newLineGen2);
        Destroy(lRend2);
        i = 0;
        //Crea la línea con el nuevo valor de frecuencia
        if (EntradaActual == 2) {
            while (i < lengthOfLineRenderer) {
                if (VolSalida == 1) {
F = (-Xc) * I * (Mathf.Cos(-Theta + (W * i * (frecuencia / 1000)))); }
                if (VolSalida == 2) {
F = (Xl) * I * (Mathf.Cos(-Theta + (W * i * (frecuencia / 1000)))); }
                if (VolSalida == 3) {
F = (res) * I * (Mathf.Sin(-Theta + (W * i * (frecuencia / 1000)))); }
```

```
                pos2 = new Vector3(i, amplitud * F, 0);
                CreatePointMarker(pos2);
                i = i + 0.05F; } }
        if (EntradaActual == 1) {
            if (VolSalida == 1) {
                while (i < lengthOfLineRenderer) {
                    SeñalCuadra1 = testApp.solveVc(i * (frecuencia / 1000));
                    pos2 = new Vector3(i, amplitud * SeñalCuadra1, 0);
                    CreatePointMarker(pos2);
                    i = i + 0.05F; } }
            if (VolSalida == 2) {
                while (i < lengthOfLineRenderer) {
                    SeñalCuadra1 = testApp.solveVl(i * (frecuencia / 1000));
                    pos2 = new Vector3(i, amplitud * SeñalCuadra1, 0);
                    CreatePointMarker(pos2);
                    i = i + 0.05F; } }
            if (VolSalida == 3) {
                while (i < lengthOfLineRenderer) {
                    SeñalCuadra1 = testApp.solveVr(i * (frecuencia / 1000));
                    pos2 = new Vector3(i, amplitud * SeñalCuadra1, 0);
                    CreatePointMarker(pos2);
                    i = i + 0.05F; } } }
        ClearAllPoints();
        Destroy(newLineGen2);
        Destroy(lRend2);
        GenerateNewLine(); }
    //Función para cambiar la señal activa a la que se le modifica la amplitud
    public void CambioSeñal() {
        ClearAllPoints();
        Destroy(newLineGen2);
        Destroy(lRend2);
        ClearAllPoints();
        GenerateNewLine(); }

    public void CreatePointMarker(Vector3 pointPosition) {
        // Crea la imagen de un punto en una posición
        Instantiate(linePointPrefab, pointPosition, Quaternion.identity);
        linePointPrefab.SetActive(true); }

    public void ClearAllPoints() {
        //Obtiene un arreglo con los “GameObject” con el tag 'PointMarker'
        allPoints2 = GameObject.FindGameObjectsWithTag("PointMarker2");
        //Destruye los objetos dentro del arreglo
        foreach (GameObject p in allPoints2) { Destroy(p); } }

    public void GenerateNewLine() {
        //Obtiene un arreglo con los “GameObject” con el tag 'PointMarker'
        allPoints2 = GameObject.FindGameObjectsWithTag("PointMarker2");
        //“Instancia” un arreglo de Vector3 de la longitud del arreglo
        allPointPositions2 = new Vector3[allPoints2.Length];
        //Si hay dos o más puntos “instanciados”
```

```
        if (allPoints2.Length >= 2) {
        //Ciclo para asignar los objetos Vector3 a la posición de los objetos
en el arreglo “allPoints”
            for (int i = 0; i < allPoints2.Length; i++) {
                allPointPositions2[i] = allPoints2[i].transform.position; }
    SpawnLineGenerator(allPointPositions2); } } //Función para crear la línea

    public void SpawnLineGenerator(Vector3[] linePoints) {
        //Crea un objeto “LineHolder”
        newLineGen2 = Instantiate(lineGeneratorPrefab);
        //Obtiene el componente “LineRenderer” del objeto
        lRend2 = newLineGen2.GetComponent<LineRenderer>();
        //Define las posiciones de “LineRenderer” a la cantidad de puntos
        lRend2.positionCount = linePoints.Length;
        //Define las posiciones de “LineRenderer”
        usando el arreglo “LinePoints”
        lRend2.SetPositions(linePoints); }
    //Función tren de pulsos a partir de un periodo u y un punto t
    public float Heaviside(float x, float u) {
        float a = 0;
        if (x > -u)
            a = 5;
        return a; } }
//Clase abstracta con el método RK4 que resuelve una ED de segundo orden
public abstract class RungeKuttaSolver {
    //Variables de estado
    double x;
    double v;
    double t;
    //Paso para el método numérico
    public double h;
    public RungeKuttaSolver(float h, float V0, float dV0)    {
        this.h = h;
        this.x = V0;
        this.v = dV0;    }
    /// <summary>
    /// Método que resuelve la ED para el siguiente punto; guarda los datos
para sustituirlos en el siguiente punto la próxima vez que se llame al
método. Regresa float {V, dV,t}
    /// </summary>
    /// <returns>V, dV, t</returns>
    public double[] resolver()    {
        //variables auxiliares
        double k1, k2, k3, k4;
        double l1, l2, l3, l4;
        //estado en el instante t
        double x = this.x;
        double v = this.v;
        double t = this.t;
//Cálculo de las aproximaciones de la segunda derivada
después del cambio de variable
```

```
            k1 = v;
            l1 = f(x, v, t);
            k2 = (v + h * l1 / 2);
            l2 = f(x + h * k1 / 2, v + h * l1 / 2, t + h / 2);
            k3 = (v + h * l2 / 2);
            l3 = f(x + h * k2 / 2, v + h * l2 / 2, t + h / 2);
            k4 = (v + h * l3);
            l4 = h * f(x + h * k3, v + h * l3, t + h);
            x += h * (k1 + 2 * k2 + 2 * k3 + k4) / 6;
            v += h * (l1 + 2 * l2 + 2 * l3 + l4) / 6;
            t += h;
            //Actualiza estado en el instante t+h
            this.t = t;
            this.x = x;
            this.v = v;
            //Devuelve los valores calculados
            return new double[] { x, v, t }; }
        //Función f (miembro derecho de la segunda derivada despejada).
        Recibe los valores de los componentes
        abstract public double f(double x, double y, double t);
        //Señal cuadrada con tiempo de transición de 1% en subida y bajada
        public double PeriodicHeaviside(double t, double T)    {
            double a = 0;
            double tref = t % T;
            //Si es 1
            if (tref >= T * 0.005 && tref <= T / 2 - T * 0.005) a = 1;
            //Si es 0
            else if (tref >= T / 2 + T * 0.005 && tref <= T - T * 0.005) a = 0;
            //Primer transitorio: subida inicial
            else if (tref < T * 0.005) a = tref / (T * 0.005);
            //Segundo transitorio: bajada
            else if (tref > T / 2 - T * 0.005 && tref < T / 2 + T * 0.005)
                a = 1 - ((tref - (T / 2) + (T * 0.005)) / (T * 0.01));
            //Tercer transitivo: subida final (antes del siguiente periodo)
            else if (tref > T / 2 - T * 0.005 && tref < T / 2 + T * 0.005)
                a = (tref - T + T * 0.005) / (T * 0.005);
            return a; } }
//Clase que hereda de “RungeKuttaSolver” para la ED de segundo orden del
circuito RLC en particular
public class VoltagesRLC : RungeKuttaSolver{
float Rg = 50f; float Rl = 150f; double R; double L; double C; double V;
double T; double[] resultsVc; double[] resultsVr; double[] resultsVrT;
double[] resultsVl; double[] resultsTime;


    /// Inicialización de la ecuación diferencial a resolver. Se define paso,
periodo y condiciones iniciales
    public VoltagesRLC(float R, float L, float C, float V, float h, float T,
float V0, float dV0) : base(h, V0, dV0) {
        this.R = R; this.L = L; this.C = C; this.V = V; this.T = T;
```

```
        //Arreglos con los valores obtenidos de la resolución numérica
        int length = 5000; resultsVc = new double[length]; resultsVr = new
double[length]; resultsVrT = new double[length]; resultsVl = new
double[length]; resultsTime = new double[length]; double[] temp_res;
        resultsVc[0] = V0;
        resultsTime[0] = 0;
        for (int actual_pos_indx = 1; actual_pos_indx < length;
actual_pos_indx++) {
            temp_res = resolver();
            resultsVc[actual_pos_indx] = temp_res[0];
            resultsVrT[actual_pos_indx] = C * (R + Rg + Rl) * temp_res[1];
            resultsVr[actual_pos_indx] = C * R * temp_res[1];
            resultsVl[actual_pos_indx] = V *
PeriodicHeaviside(actual_pos_indx * h, T) - resultsVc[actual_pos_indx] -
resultsVrT[actual_pos_indx];
            resultsTime[actual_pos_indx] = temp_res[2]; } }
//Función para interpolar valores del voltaje del condensador
    public float solveVc(float t) {
        //Obtener índice con base en t y h
        double interm = t / h;
        double dec_part = interm % 1;
        int int_indx = (int)(interm - dec_part);
        //Interpola el resultado
        double res = resultsVc[int_indx + 1] - ((resultsVc[int_indx + 1] -
resultsVc[int_indx]) * (1 - dec_part));
        return (float)res; }
//Función para interpolar valores del voltaje del resistor
    public float solveVr(float t) {
        //Obtener índice con base en t y h
        double interm = t / h;
        double dec_part = interm % 1;
        int int_indx = (int)(interm - dec_part);
        //Interpola el resultado
        double res = resultsVr[int_indx + 1] - ((resultsVr[int_indx + 1] -
resultsVr[int_indx]) * (1 - dec_part));
        return (float)res; }
//Función para interpolar valores del voltaje del inductor
    public float solveVl(float t) {
        //Obtener índice con base en t y h
        double interm = t / h;
        double dec_part = interm % 1;
        int int_indx = (int)(interm - dec_part);
        //Interpola el resultado
        double res = resultsVl[int_indx + 1] - ((resultsVl[int_indx + 1] -
resultsVl[int_indx]) * (1 - dec_part));
        return (float)res; }
```

```
    /// Ecuación diferencial de segundo orden despejada. Para obtener valores
auxiliares de k
    public override double f(double x, double v, double t) {
//Función f (miembro derecho de la segunda derivada despejada) para el
voltaje del condensador en un circuito RLC en serie
        return (1 / (L * C)) * ((V * PeriodicHeaviside(t, T)) - x - (R + Rg +
Rl) * C * v); }
```

K Bocetos de la interfaz de usuario

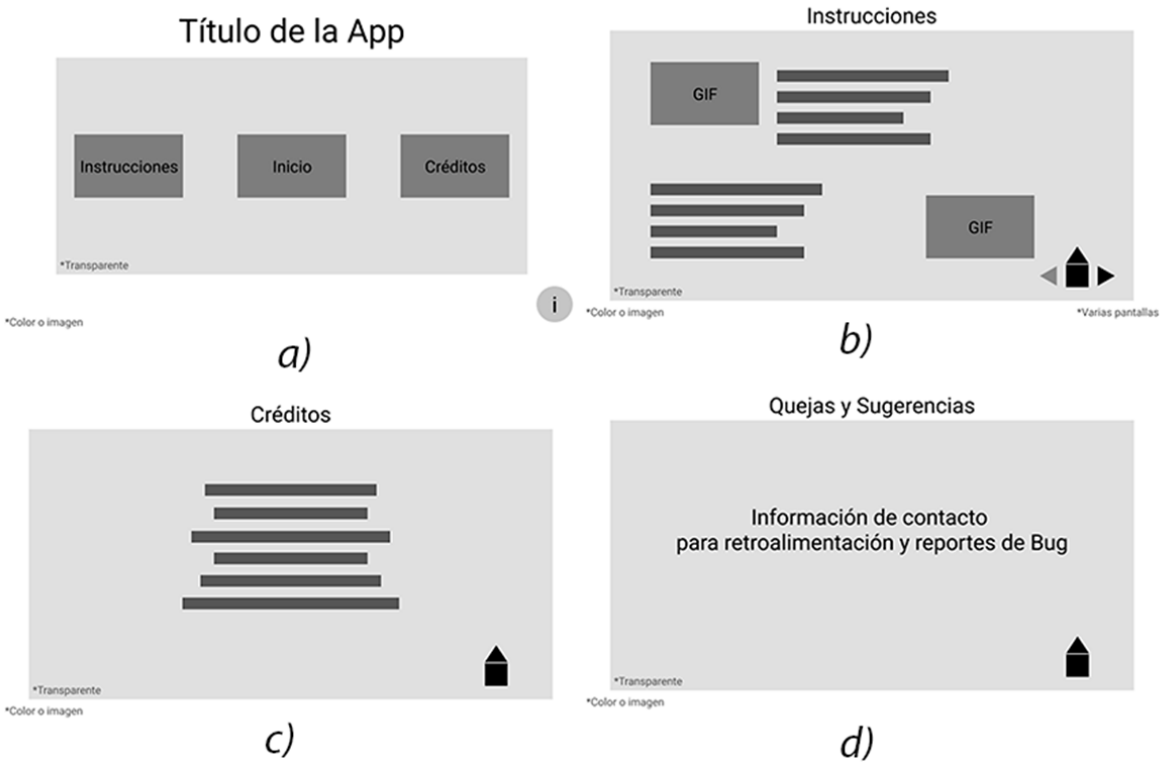


Figura K.1 Propuesta de la disposición de elementos dentro de las escenas de la app: a) inicio; b) instrucciones; c) créditos; d) contacto.

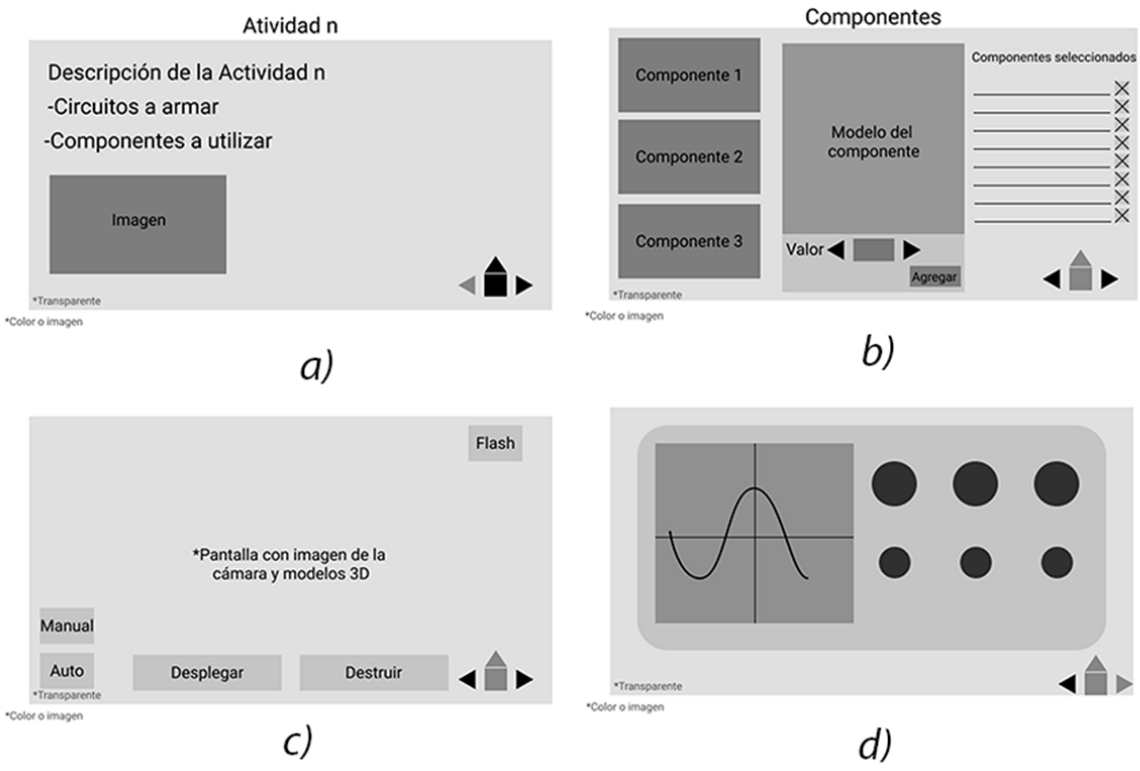


Figura K.2 Propuesta de la disposición de elementos dentro de las escenas de la app: a) actividad; b) selección de componentes; c) armado del circuito; d) osciloscopio.

Figura K.2 Bocetos con color de las escenas de la app: a) inicio; b) instrucciones; c) créditos; d) contacto; e) actividad; f) selección de componentes; g) armado del circuito; h) osciloscopio.

L Código de las escenas para la interfaz de usuario

L.1 Cambio de escenas

```
using System.Collections;
using UnityEngine;
using UnityEngine.SceneManagement;

public class CambioEcenas : MonoBehaviour{
    //Variable que registra la escena cargada actualmente
    public static int EscenasCargadas = 0;
    ///Funciones para realizar cambios de escena. Cada función se llama
    ///dependiendo de la escena actual y de las acciones del usuario

    //Carga la escena “Instrucciones” desde la pantalla de inicio
    public void InstruccionesPantalla()
        {SceneManager.LoadScene("Instrucciones");}
    //Carga una escena de actividad desde la pantalla de inicio o al terminar
    //el alambrado de otra actividad al presionar el botón con la flecha
    //hacia la derecha
    public void ActividadSiguiente() {
        //Incrementa el valor de la escena actual
        EscenasCargadas = EscenasCargadas + 1;
        //Con base en el nuevo valor, selecciona la actividad correspondiente
        if (EscenasCargadas == 1) { SceneManager.LoadScene("Act1"); }
        else if (EscenasCargadas == 2) { SceneManager.LoadScene("Act2"); }
        else if (EscenasCargadas == 3) { SceneManager.LoadScene("Act3"); }
        else if (EscenasCargadas >= 4) {
            //Si se terminaron las actividades, llama a la escena “Créditos”
            SceneManager.LoadScene("CreditosFin");
            EscenasCargadas = 0; } }
    //Carga una escena de actividad anterior desde la escena de alambrado de
    //otra actividad al presionar el botón con la flecha hacia la izquierda
    public void ActividadAnterior() {
        if (EscenasCargadas == 1) { SceneManager.LoadScene("Act1"); }
        else if (EscenasCargadas == 2) { SceneManager.LoadScene("Act2"); }
```

```
        else if (EscenasCargadas == 3) { SceneManager.LoadScene("Act3"); }
        else if (EscenasCargadas >= 4) {
            SceneManager.LoadScene("CreditosFin");
            EscenasCargadas = 0; } }
    //Carga a la escena “Inicio” desde cualquier escena al presionar el botón
    //con el símbolo de una casa. También reinicia el contador de escenas
    //de escenas cargadas
    public void Home()
    { SceneManager.LoadScene("Inicio"); EscenasCargadas = 0; }
    //Carga la escena “Sistemas” desde cualquier escena de actividades
    public void Sistemas()
    { SceneManager.LoadScene("Sistemas", LoadSceneMode.Additive); }
    //Redirige al usuario al archivo con el formato de la práctica original
    public void OpenPDF() {
Application.OpenURL("https://drive.google.com/file/d/1A1AqKVX-4Ma8xn7CG93BZJ5
XxTE-xoaF/view?usp=sharing"); }
    //Redirige al usuario al archivo del marcador cilíndrico
    public void MarcadorC() {
Application.OpenURL("https://drive.google.com/open?id=1b_P-zPIQUqtKDbnT1Am7wS
-deWk_mBKN"); }
    //Cierra la escena “Sistemas” para regresar a la anterior
    public void Anterior() { SceneManager.UnloadScene("Sistemas"); }
    //Funciones para cargar de forma ordenada las escenas relacionadas
    //con las instrucciones de uso de la app
    public void Inst2() { SceneManager.LoadScene("Instrucciones2"); }
    public void Inst3() { SceneManager.LoadScene("Instrucciones3"); }
    public void Inst4() { SceneManager.LoadScene("Instrucciones4"); }
    public void Inst5() { SceneManager.LoadScene("Instrucciones5"); }
    public void Inst6() { SceneManager.LoadScene("Instrucciones6"); }
    public void Inst7() { SceneManager.LoadScene("Instrucciones7"); }
    public void Inst8() { SceneManager.LoadScene("Instrucciones8"); }
    //Cierra la “app”
    public void Salir() { Application.Quit(); }
    //Carga la escena “Selección de componentes” desde una actividad
    public void SelecComp() { SceneManager.LoadScene("SeleComp"); }
    //Carga la escena para realizar el alambrado de un circuito desde
    //la escena “Selección de componentes”
    public void Alambrado() { SceneManager.LoadScene("EscenaAlambrado"); }
    //Carga la escena “Contacto” desde la pantalla de inicio
    public void INFO() { SceneManager.LoadScene("BUG"); }
    public void Creditos() { SceneManager.LoadScene("CreditosFin"); }
    //Carga la escena “Créditos” desde la pantalla de inicio
    public void Osciloscopio(){SceneManager.LoadScene("OsciloscopioEscena");}
```

L.2 Escena “Selección de componentes”

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using UnityEngine.SceneManagement;

public class VisualizaciónComp : MonoBehaviour{
    //Objeto que controla las interacciones con la pantalla táctil
    RaycastHit hitInfo;
    //Objetos para desplegar los modelos tridimensionales de los componentes
    private GameObject Cap;
    private GameObject Res;
    private GameObject Indu;
    //Objetos para el despliegue del texto con el valor de los
    //componentes seleccionados
    public TextMeshProUGUI Valor;
    public TextMeshProUGUI Compo1;
    public TextMeshProUGUI Compo2;
    public TextMeshProUGUI Compo3;
    public TextMeshProUGUI Compo4;
    public TextMeshProUGUI Compo5;
    //Botones para cancelar la selección de algún componente
    public Button Close1;
    public Button Close2;
    public Button Close3;
    public Button Close4;
    public Button Close5;
    //Variable para identificar componentes:
    //1 capacitor , 2 Resistencia, 3 Inductor
    int comp = 0;
    //Número de componentes seleccionados
    int indice = 1;
    //Variable que indica si se debe desplegar un modelo tridimensional
    int uso = 0;
    //Arreglos para almacenar el valor de los componentes seleccionados
    int[] C = new int [5];
    int[] R = new int[5];
    int[] I = new int[5];
    string Capacitor;
    string Resistencia;
    string Inductor;
    public static bool ReiniciarTablero;

    void Start()    {
        //Oculta los modelos tridimensionales al inicio de la escena
        Cap.gameObject.SetActive(false);
```

```
        Res.gameObject.SetActive(false);
        Indu.gameObject.SetActive(false);
        //Coloca los valores por defecto de los componentes en 0
        CapacitorValor1 = 0;
        CapacitorValor2 = 0;
        CapacitorValor3 = 0;
        CapacitorValor4 = 0;
        CapacitorValor5 = 0;
        InductorValor1 = 0;
        InductorValor2 = 0;
        InductorValor3 = 0;
        InductorValor4 = 0;
        InductorValor5 = 0;
        ResistorValor1 = 0;
        ResistorValor2 = 0;
        ResistorValor3 = 0;
        ResistorValor4 = 0;
        ResistorValor5 = 0;
        int[] C = new int[5];
        int[] R = new int[5];
        int[] I = new int[5]; }

    void Update() {
        //Verifica si se presiona la pantalla sobre algún componente para
        //desplegarlo y asignarle un valor inicial
        if (Input.GetMouseButtonDown(0)) {
            Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
            RaycastHit hitInfo;
            if (Physics.Raycast(ray, out hitInfo)) {
                if (hitInfo.collider.gameObject.tag == "Cap") {
                    Cap.gameObject.SetActive(true);
                    Res.gameObject.SetActive(false);
                    Indu.gameObject.SetActive(false);
                    comp = 1;
                    indice = 1;
                    uso = 1;
                    Valor.text = "0.12 uF"; }
                if (hitInfo.collider.gameObject.tag == "Res") {
                    Cap.gameObject.SetActive(false);
                    Res.gameObject.SetActive(true);
                    Indu.gameObject.SetActive(false);
                    comp = 2;
                    indice = 1;
                    uso = 1;
                    Valor.text = "68 ohm"; }
                if (hitInfo.collider.gameObject.tag == "Indu") {
                    Cap.gameObject.SetActive(false);
                    Res.gameObject.SetActive(false);
                    Indu.gameObject.SetActive(true);
                    comp = 3;
                    uso = 1;
```

```
                indice = 1; Valor.text = "30 mH"; } } } }

    //Función para desplegar en texto el valor de un componente
    //al modificarlo
    public void CambioValor() {
        if (comp == 1) {
            uso = 1;
            switch (indice) {
                case 1:
                    Valor.text = "0.12 uF"; break;
                case 2: Valor.text = "0.22 uF"; break;
                case 3: Valor.text = "0.33 uF"; break;
                case 4: Valor.text = "0.47 uF"; break;
                case 5: Valor.text = "0.56 uF"; break; } }
        if (comp == 2) {
            uso = 1;
            switch (indice) {
                case 1: Valor.text = "68 ohm"; break;
                case 2: Valor.text = "180 ohm"; break;
                case 3: Valor.text = "820 ohm"; break;
                case 4: Valor.text = "1000 ohm"; break;
                case 5: Valor.text = "1800 ohm"; break; } }
        if (comp == 3) {
            uso = 1;
            switch (indice) {
                case 1: Valor.text = "30 mH"; break;
                case 2: Valor.text = "40 mH"; break;
                case 3: Valor.text = "50 mH"; break;
                case 4: Valor.text = "60 mH"; break;
                case 5: Valor.text = "70 mH"; break;} } }

    //Función para incrementar el valor del componente que se
    //está seleccionando
    public void ValorSup() {
        if (uso == 1) {
            indice = indice + 1;
            if (indice >= 6) {
                indice = 1; } } }

    //Función para decrementar el valor del componente que se
    //está seleccionando
    public void ValorAnt() {
        if (uso == 1) {
            indice = indice - 1;
            if (indice <= 0) { indice = 1; } } }

    //Función para agregar un componente a la lista de seleccionados
    public void Agregar() {
        if (uso == 1) {
            //Verifica el estado de cada espacio de asignación de componentes
            //si está disponible, coloca la información del nuevo componente
```

```
        //de forma gráfica y en el arreglo correspondiente
        if (Compo1.text == "") {
            Compo1.text = Valor.text;
            Close1.interactable = true;
            if (comp == 1) {C[0] = 1; }
            else if (comp == 2) {R[0] = 1; }
            else if (comp == 3) { I[0] = 1; } }
        else if (Compo2.text == "") {
            Compo2.text = Valor.text;
            Close2.interactable = true;
            if (comp == 1) { C[1]= 2; }
            else if (comp == 2) { R[1] = 2; }
            else if (comp == 3) { I[1] = 2; }            }
        else if (Compo3.text == "") {
            Compo3.text = Valor.text;
            Close3.interactable = true;
            if (comp == 1) { C[2] = 3; }
            else if (comp == 2) { R[2] = 3; }
            else if (comp == 3) { I[2] = 3; } }

        else if (Compo4.text == "") {
            Compo4.text = Valor.text;
            Close4.interactable = true;
            if (comp == 1) { C[3] = 4; }
            else if (comp == 2) { R[3] = 4; }
            else if (comp == 3) { I[3] = 4; } }
        else if (Compo5.text == "") {
            Compo5.text = Valor.text;
            Close5.interactable = true;
            if (comp == 1) { C[4] = 5; }
            else if (comp == 2){ R[4] = 5; }
            else if (comp == 3){ I[4] = 5; } } }
    ReiniciarTablero = true; }

//Funciones para eliminar de la lista un componente previamente
//seleccionado al presionar el botón con una cruz roja
public void Eliminar1() {
    Compo1.text = "";
    Close1.interactable = false;
    if (C[0]== 1) { C[0] = 0; }
    else if (R[0] == 1) { R[0] = 0; }
    else if (I[0] == 1) { I[0] = 0; } }
public void Eliminar2() {
    Compo2.text = "";
    Close2.interactable = false;
    if (C[1] == 2) { C[1] = 0; }
    else if (R[1] == 2) { R[1] = 0; }
    else if (I[1] == 2) { I[1] = 0; } }
public void Eliminar3() {
    Compo3.text = "";
    Close3.interactable = false;
```

```
        if (C[2] == 3) { C[2] = 0; }
        else if (R[2] == 3) { R[2] = 0; }
        else if (I[2] == 3) { I[2] = 0; } }

    public void Eliminar4() {
        Compo4.text = "";
        Close4.interactable = false;
        if (C[3] == 4) { C[3] = 0; }
        else if (R[3] == 4) { R[3] = 0; }
        else if (I[3] == 4) { I[3] = 0; } }
    public void Eliminar5() {
        Compo5.text = "";
        Close5.interactable = false;
        if (C[4] == 5) {C[4] = 0;}
        else if (R[4] == 5) { R[4] = 0; }
        else if (I[4] == 5) { I[4] = 0; } }
    IEnumerator ExecuteAfterTime(float time) {
        Sincomp.gameObject.SetActive(true);
        yield return new WaitForSeconds(time);
        Sincomp.gameObject.SetActive(false); }

    //Función que guarda los valores de los componentes seleccionados para
    //pasar a la escena de alambrado del circuito
    public void SigComp() {
        //Verifica si se ha elegido por lo menos un componente de cada tipo
        if (C[0]==0 && R[0] == 0 && I[0] == 0) {
            StartCoroutine(ExecuteAfterTime(10f)); }
        else if (C.Length != 0 || R.Length != 0 || I.Length != 0) {
            //Extrae los valores de los capacitores seleccionados
            for (int j = 0; j <C.Length; j++) {
                if (C[j]==1) {
                    Capacitor = Compo1.text;
                    //Extrae únicamente el valor numérico de la cadena
                    //de texto que se muestra durante la selección
                    Capacitor = Capacitor.Substring(0, Capacitor.Length - 3);
                    float.TryParse(Capacitor, out CapacitorValor1);
                    CapacitorValor1 = CapacitorValor1 * 0.000001f; }
                if (C[j] == 2) {
                    Capacitor = Compo2.text;
                    Capacitor = Capacitor.Substring(0, Capacitor.Length - 3);
                    float.TryParse(Capacitor, out CapacitorValor2);
                    CapacitorValor2 = CapacitorValor2 * 0.000001f; }
                if (C[j] == 3) {
                    Capacitor = Compo3.text;
                    Capacitor = Capacitor.Substring(0, Capacitor.Length - 3);
                    float.TryParse(Capacitor, out CapacitorValor3);
                    CapacitorValor3 = CapacitorValor3 * 0.000001f; }
                if (C[j] == 4) {
                    Capacitor = Compo4.text;
                    Capacitor = Capacitor.Substring(0, Capacitor.Length - 3);
                    float.TryParse(Capacitor, out CapacitorValor4);
```

```
            CapacitorValor4 = CapacitorValor4 * 0.000001f; }
        if (C[j] == 5) {
            Capacitor = Compo5.text;
            Capacitor = Capacitor.Substring(0, Capacitor.Length - 3);
            float.TryParse(Capacitor, out CapacitorValor5);
            CapacitorValor5 = CapacitorValor5 * 0.000001f; } }

    //Extrae los valores de las resistencias seleccionadas
    for (int j = 0; j < R.Length; j++) {
        if (R[j] == 1) {
            Resistencia = Compo1.text;
      Resistencia = Resistencia.Substring(0, Resistencia.Length - 4);
            float.TryParse(Resistencia, out ResistorValor1); }
        if (R[j] == 2) {
            Resistencia = Compo2.text;
      Resistencia = Resistencia.Substring(0, Resistencia.Length - 4);
            float.TryParse(Resistencia, out ResistorValor2); }
        if (R[j] == 3) {
            Resistencia = Compo3.text;
      Resistencia = Resistencia.Substring(0, Resistencia.Length - 4);
            float.TryParse(Resistencia, out ResistorValor3); }
        if (R[j] == 4) {
            Resistencia = Compo4.text;
      Resistencia = Resistencia.Substring(0, Resistencia.Length - 4);
            float.TryParse(Resistencia, out ResistorValor4); }
        if (R[j] == 5) {
            Resistencia = Compo5.text;
      Resistencia = Resistencia.Substring(0, Resistencia.Length - 4);
            float.TryParse(Resistencia, out ResistorValor5); } }

    //Extrae los valores de los inductores seleccionados
    for (int j = 0; j < I.Length; j++) {
        if (I[j] == 1) {
            Inductor = Compo1.text;
            Inductor = Inductor.Substring(0, Inductor.Length - 3);
            float.TryParse(Inductor, out InductorValor1);
            InductorValor1 = InductorValor1 * 0.001f; }
        if (I[j] == 2) {
            Inductor = Compo2.text;
            Inductor = Inductor.Substring(0, Inductor.Length - 3);
            float.TryParse(Inductor, out InductorValor2);
            InductorValor2 = InductorValor2 * 0.001f; }
        if (I[j] == 3) {
            Inductor = Compo3.text;
            Inductor = Inductor.Substring(0, Inductor.Length - 3);
            float.TryParse(Inductor, out InductorValor3);
            InductorValor3 = InductorValor3 * 0.001f; }
        if (I[j] == 4) {
            Inductor = Compo4.text;
            Inductor = Inductor.Substring(0, Inductor.Length - 3);
            float.TryParse(Inductor, out InductorValor4);
```

```
                InductorValor4 = InductorValor4 * 0.001f; }
            if (I[j] == 5) {
                Inductor = Compo5.text;
                Inductor = Inductor.Substring(0, Inductor.Length - 3);
                float.TryParse(Inductor, out InductorValor5);
                InductorValor5 = InductorValor5 * 0.001f; } }

        //Llama a la escena de “Alambrado del circuito”
        SceneManager.LoadScene("EscenaAlambrado"); } }
```

M Formatos de encuestas de facilidad de uso

M.1 Encuesta para alumnos

Encuesta de uso de la app RACE: Realidad Aumentada en Circuitos Eléctricos

Esta encuesta fue preparada con fines informativos sobre la facilidad de uso de la app RACE: Realidad Aumentada en Circuitos Eléctricos, por parte de la comunidad dentro de la Facultad de Ingeniería. Las preguntas son basadas en la System Usability Scale (SUS) creada por John Brooke que permite evaluar productos y servicios de hardware, software, dispositivos móviles, sitios web, aplicaciones entre otros. La información obtenida por esta encuesta no será utilizada con otros fines más que los planteados. Dado que las condiciones del producto están fuera del control de la Facultad de Ingeniería, la UNAM no asume responsabilidad en conexión con el uso de esta información. Nada de lo aquí expresado será divulgado a externos.

*Obligatorio

1. Selecciona la opción que describa tu situación académica *

- ○ Estudiante cursando actualmente la materia de Ecuaciones Diferenciales
- ○ Estudiante que cursó con anterioridad la materia de Ecuaciones Diferenciales

2. Por favor contesta las siguientes preguntas, seleccionando la casilla que mejor exprese tu experiencia con el uso de la app *

	Fuertemente de acuerdo	De acuerdo	Ni de acuerdo, ni en desacuerdo	En desacuerdo	Fuertemente en desacuerdo
Creo que me gustaría usar esta app frecuentemente	☐	☐	☐	☐	☐
Encuentro la app innecesariamente compleja	☐	☐	☐	☐	☐
Creo que la app fue fácil de usar	☐	☐	☐	☐	☐
Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar esta app	☐	☐	☐	☐	☐
Las funciones de esta app están bien integradas	☐	☐	☐	☐	☐
Creo que la app es muy inconsistente	☐	☐	☐	☐	☐
Imagino que la mayoría de la gente aprendería a usar esta app de forma muy rápida	☐	☐	☐	☐	☐

El uso de la app es muy confuso	☐	☐	☐	☐	☐
Me siento confiado con el uso de esta app	☐	☐	☐	☐	☐
Necesité aprender muchas cosas antes de ser capaz de usar esta app	☐	☐	☐	☐	☐

3. ¿Consideras haber cumplido con los objetivos planteados en la práctica desarrollada dentro de la app? *

- ◯ Sí
- ◯ No

4. ¿La tecnología de Realidad Aumentada, te ayudó a comprender y desarrollar con facilidad las actividades de la práctica? *

- ◯ Sí
- ◯ No

5. ¿En qué modalidad prefieres realizar la misma práctica? *

- ◯ De forma presencial (De manera tradicional, utilizando herramientas de laboratorio y el formato impreso)
- ◯ Utilizando la app RACE
- ◯ Ambas

6. Con respecto a la pregunta anterior, describe brevemente la razón de tu preferencia *

_____

7. Escribe cualquier comentario u observación extra que desees agregar sobre tu experiencia utilizando la app *

_____

M.2 Encuesta para profesores

Encuesta de Uso de la app RACE: Realidad Aumentada en Circuitos Eléctricos (profesores)

Esta encuesta fue preparada con fines informativos sobre la facilidad de uso de la app RACE: Realidad Aumentada en Circuitos Eléctricos, por parte de la comunidad dentro de la Facultad de Ingeniería. Las preguntas son basadas en la System Usability Scale (SUS) creada por John Brooke que permite evaluar productos y servicios de hardware, software, dispositivos móviles, sitios web, aplicaciones entre otros. La información obtenida por esta encuesta no será utilizada con otros fines más que los planteados. Dado que las condiciones del producto están fuera del control de la Facultad de Ingeniería, la UNAM no asume responsabilidad en conexión con el uso de esta información. Nada de lo aquí expresado será divulgado a externos

*Obligatorio

1. Seleccione la opción que describa su situación académica *

- ◯ Profesor de la materia de Ecuaciones Diferenciales
- ◯ Profesor de la División de Ciencias Básica exceptuando la materia de Ecuaciones Diferenciales
- ◯ Otros: _____

2. Por favor conteste las siguientes preguntas, seleccionando la casilla que mejor exprese su experiencia con el uso de la app *

	Fuertemente de acuerdo	De acuerdo	Ni de acuerdo, ni en desacuerdo	En desacuerdo	Fuertemente en desacuerdo
Creo que me gustaría usar esta app frecuentemente	☐	☐	☐	☐	☐
Encuentro la app innecesariamente compleja	☐	☐	☐	☐	☐
Creo que la app fue fácil de usar	☐	☐	☐	☐	☐
Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar esta app	☐	☐	☐	☐	☐
Las funciones de esta app están bien integradas	☐	☐	☐	☐	☐
Creo que la app es muy inconsistente	☐	☐	☐	☐	☐
Imagino que la mayoría de la gente aprendería a usar esta app de forma muy rápida	☐	☐	☐	☐	☐

El uso de la app es muy confuso	☐	☐	☐	☐	☐
Me siento confiado con el uso de esta app	☐	☐	☐	☐	☐
Necesité aprender muchas cosas antes de ser capaz de usar esta app	☐	☐	☐	☐	☐

3. ¿Considera que el alumno puede cumplir exitosamente con los objetivos planteados en la práctica desarrollada dentro de la app? *

- ◯ Sí
- ◯ No

4. ¿En qué modalidad preferiría impartir la misma práctica? *

- ◯ De forma presencial (De manera tradicional, utilizando herramientas de laboratorio y el formato impreso)
- ◯ Utilizando la app RACE
- ◯ Ambas

5. Con respecto a la pregunta anterior, describa brevemente la razón de su preferencia *

_____

6. Escriba cualquier comentario u observación extra que desee agregar sobre su experiencia utilizando la app *

_____

ANEXO

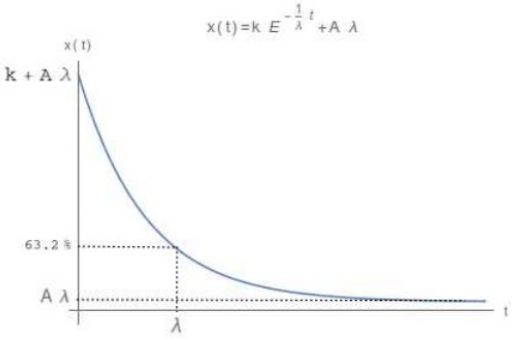
1 Formato de la práctica “Aplicación de las ecuaciones diferenciales en circuitos eléctricos”

Proyecto PAPIME 111218 Semestre 2019-1

Práctica

Aplicación de las Ecuaciones diferenciales en circuitos eléctricos

1 Objetivos



1 Determinar el valor de la inductancia en un circuito RL.

2 Verificar experimentalmente el valor de la resistencia que se necesita para que un circuito RLC en serie sea críticamente amortiguado, y además corroborar el rango de valores que aquél puede tener para que el sistema tenga una respuesta subamortiguada o sobreamortiguada.

3 Comprobar la respuesta en estado permanente en el circuito RLC en serie con fuente de alimentación sinusoidal.

2 Introducción

El modelo matemático de un sistema de primer orden es una ecuación diferencial que puede escribirse como:

$$\frac{d\,x(t)}{d\,t}+\frac{1}{\lambda}x(t)=A$$

considerando a A un valor constante, cuya respuesta estará dada por:

$$x(t)=ke^{-\frac{1}{\lambda}t}+A\lambda$$

donde λ es una constante, que determinará el tiempo en el cuál se alcanza el 63.2% de valor final ($A\lambda$) considerando su valor inicial en $k+A\lambda$. Como se puede apreciar en la gráfica.

Para un sistema de segundo orden la ecuación diferencial puede escribirse como:

$$\frac{d^2x(t)}{dt^2}+2\xi\omega_0\frac{dx(t)}{dt}+\omega_0^2x(t)=\omega_0^2f(t)$$

en la cual al parámetro ξ se le denomina factor de amortiguamiento, y al parámetro ω_0 se le conoce como frecuencia angular natural de oscilación. La función $f(t)$ es la entrada o función de excitación del sistema y $x(t)$ es la salida o respuesta del mismo.

La ecuación característica que corresponde al modelo matemático anterior es:

$$s^2 + 2\xi\omega_0 s + \omega_0^2 = 0$$

y cuyas raíces son los valores característicos:

$$s_{1,2} = -\xi\omega_0 \pm \omega_0\sqrt{\xi^2 - 1}\,.$$

Dependiendo del valor de ξ, dichos valores pueden ser reales, imaginarios o complejos, dando los siguientes comportamientos en la respuesta del sistema:

si $\xi = 0$, entonces $s_{1,2} = \pm j\omega_0$ (valores imaginarios), y el sistema será no amortiguado (caso teórico ideal);

si $0 < \xi < 1$, entonces $s_{1,2} = -\xi\omega_0 \pm j\omega_0\sqrt{1-\xi^2}$ (valores complejos conjugados), y el sistema será subamortiguado;

si $\xi = 1$, entonces $s_{1,2} = -\xi\omega_0$ (valores reales negativos iguales), y el sistema será críticamente amortiguado;

finalmente, si $\xi > 1$, entonces $s_{1,2} = -\xi\omega_0 \pm \omega_0\sqrt{\xi^2 - 1}$ (valores reales negativos diferentes), y el sistema será sobreamortiguado.

Es importante el análisis cualitativo de los diferentes tipos de sistemas de segundo orden, de manera de poder reconocerlos a partir de la gráfica de su respuesta.

Para el caso particular de los sistemas de segundo orden subamortiguados, presentan varios parámetros de interés en su respuesta, como lo son la frecuencia de oscilación, su inverso el periodo de oscilación, el sobrepaso, el tiempo de sobrepaso, el tiempo de levantamiento y el tiempo de asentamiento.

1 Frecuencia de oscilación, f, es el número de oscilaciones que tiene la respuesta del sistema por unidad de tiempo:

$$f = \frac{\omega}{2\pi} \qquad \text{donde} \quad \omega = \omega_0\sqrt{1-\xi^2}\,.$$

2 Periodo de oscilación, T, es el tiempo que transcurre en una oscilación completa de la respuesta del sistema:

$$T = \frac{1}{f}\,.$$

3 Sobrepaso, Sp, es el valor máximo de la respuesta, considerando la respuesta permanente unitaria ($v_f = 1$):

$$Sp = \exp\left(-\frac{\xi\pi}{\sqrt{1-\xi^2}}\right).$$

4 Tiempo de sobrepaso, tp, es el tiempo necesario para que la respuesta alcance su valor máximo:

$$tp = \frac{\pi}{\omega_0 \sqrt{1-\xi^2}}.$$

5 Tiempo de levantamiento, tl, es el tiempo necesario para que la respuesta alcance su valor permanente por primera vez:

$$tl = \frac{\pi - \phi}{\omega_0 \sqrt{1-\xi^2}}, \text{ donde } \phi = \arccos \xi.$$

6 Tiempo de asentamiento, ta, es el tiempo que transcurre para que la respuesta oscile entre el 95% y el 105% de su valor permanente:

$$ta = \frac{3}{\xi \omega_0}.$$

En la Figura 1 se ilustran los parámetros mencionados en los párrafos anteriores.

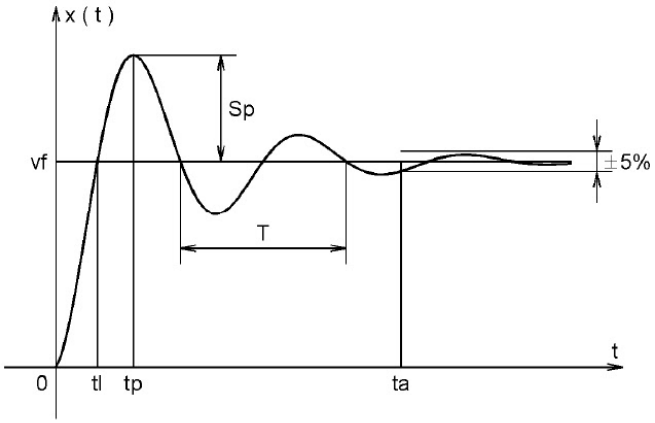


Figura 1 Parámetros de la respuesta de un sistema subamortiguado.

En el diseño de circuitos se presenta con frecuencia el problema de la obtención de los valores de los dispositivos eléctricos, en este caso del resistor, del inductor y del condensador, de tal forma que, dada la configuración del sistema, se obtenga una salida determinada.

Dado que las cantidades ξ y ω_0 quedan en función de los valores de los dispositivos que conforman al circuito, es posible calcular los valores de estos últimos si se establece que la respuesta del sistema es críticamente amortiguada, o bien, se conocen otros parámetros de diseño, como el periodo para el caso de la respuesta subamortiguada.

3 Equipo y material empleado

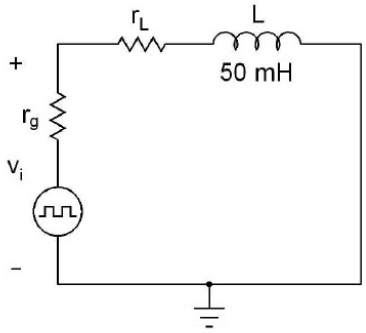
Un osciloscopio	resistores de 68, 180, 820, 1000 y 1800 Ω
un generador de funciones	un inductor (50 mH aprox.)

una fuente de poder
un multímetro
un condensador de 0.22 μF
una tableta de experimentación (protoboard).

4 Desarrollo

4.1 Medición de la inductancia

Conecte al inductor las terminales del generador de funciones, con una señal cuadrada de 5 V pico y con una frecuencia de 200 Hz.



La ecuación diferencial que modela el comportamiento del circuito es:

$$\frac{d\, i_L(t)}{d\, t}+\frac{R}{L}\, i_L(t)=\frac{V_i(t)}{L}$$

Verifique con el osciloscopio la señal de voltaje de dicho inductor, y determine el tiempo en que se obtiene una variación del 63.2% de la diferencia del valor final y el valor inicial. Dicho valor es la constante de tiempo del circuito RL, la cual debe ser igual a:

$$\tau=\frac{L}{R}\ .$$

Figura 2 Circuito RL en serie.

Dado que se conoce R, el cual es la suma de las resistencias internas del generador de funciones y del inductor, es posible determinar el valor de la inductancia L.

4.2 Circuito RLC serie, respuesta transitoria

Arme el circuito mostrado en la Figura 3 con un resistor con un valor de resistencia de R = 68 Ω. Note que las resistencias r_g y r_L son las resistencias internas del generador de funciones y del inductor, respectivamente. Aplique una señal cuadrada con una amplitud de 5 V pico y una frecuencia de 200 Hz.

Observe en el osciloscopio la señal de salida del circuito, v_C, y verifique el tipo de sistema al que corresponda (subamortiguado, sobreamortiguado, críticamente amortiguado).

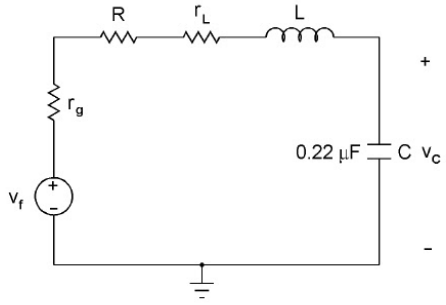


Figura 3 Circuito RLC en serie.

La ecuación diferencial que modela el comportamiento del circuito es:

$$\frac{d^2 V_c(t)}{d t^2} + \frac{R}{L}\frac{d V_c(t)}{d t} + \frac{1}{LC} V_c(t) = \frac{V_f(t)}{LC}$$

Posteriormente cambie el resistor por uno que tenga una resistencia de R = 820 Ω, y de igual manera verifique el tipo de sistema al que corresponde su respuesta.

Finalmente realice la misma operación, pero para un resistor con una resistencia de R = 1.8 kΩ.
Para el circuito cuya respuesta corresponda a un sistema subamortiguado, mida el periodo del transitorio, T, el tiempo de sobrepaso, tp, el tiempo de levantamiento, tl, y el tiempo de asentamiento, ta.

4.5 Circuito RLC serie, respuesta permanente con fuente sinusoidal.

Para el circuito del experimento anterior, cambie el tipo de señal por una sinusoidal, conecte en el segundo canal del osciloscopio la señal de entrada y anote la amplitud, frecuencia de ambas señales, así como el desfase entre ellas.

5 Informe

5.1 Resuelva la ecuación diferencial que modela el comportamiento del circuito RL de la figura 2, considerando que Vi=5 V, la solución queda en término de iL, obtenga la expresión del voltaje del inductor si se sabe que: $V_L(t) = L\frac{d i_L(t)}{d t}$

Dibuje la gráfica de VL y compárela con la señal que obtuvo en el osciloscopio para el cálculo de la inductancia, acotando los valores máximo y mínimo del voltaje en el inductor, así como el tiempo y su respectivo valor de voltaje en el punto en que se midió el 63.2% de la diferencia de los valores final e inicial.

Obtenga el valor de la inductancia medida.

5.2 Escriba la ecuación diferencial que modela al sistema eléctrico de la Figura 2, para los valores teóricos de R utilizados, obtenga la solución de dicha ecuación, considere la Vf como constante igual a 5. Indique para cuales el circuito tiene las respuestas:

1 subamortiguada, con $\xi = 0.2$;

2 críticamente amortiguada;

3 sobreamortiguada, con $\xi = 2$.

5.3 Para el caso del circuito con respuesta subamortiguada, obtenga los valores teóricos del periodo del transitorio, T, el tiempo de sobrepaso, tp, el tiempo de levantamiento, tl, y el tiempo

de asentamiento, ta, y compárelos con los valores medidos experimentalmente. Haga los comentarios que considere pertinentes.

5.4 Para el circuito RLC Considerando una fuente de alimentación de Vf=5 cos 400πt V, determine la respuesta en estado permanente del voltaje en el contensador, escriba la expresión en términos de una sola función sinusoidal utilizando las relaciones matemáticas pertinentes, y compárela con la obtenida en el osciloscopio. Verificar que se cumple el desfasamiento de la señal de salida con respecto a la señal de entrada.

$$\frac{d^2 V_c(t)}{d t^2} + \frac{R}{L}\frac{d V_c(t)}{d t} + \frac{1}{LC} V_c(t) = \frac{V_f(t)}{LC}$$

6 Conclusiones, sugerencias y comentarios

7 Bibliografía

Dorf, Svoboda, *Circuitos eléctricos*, Quinta edición, Alfaomega, México, 2003.

Desoer, Kuh, *Basic Circuit Theory*, McGraw-Hill, EUA, 1969.

Ogata, *Dinámica de Sistemas*, Prentice Hall Hispanoamericana, México, 1987.

Minami Yukihiro. *Prácticas 8 y 9 del laboratorio de Análisis de Circuitos, DIMEI* octubre de 2010.

Propuesta de conexión del circuito RLC en la tarjeta protoboard