Interfaz RoboAct-pantalla multitáctil para interacción con el usuario en la creación de dramatizaciones de teatro robótico

Daniel Andrés Crovo Pérez

Director:

Eduardo Gerlein PhD.

Pontificia Universidad Javeriana

Facultad de Ingeniería

Departamento de Electrónica

Mayo de 2018

Lista de Figuras

Lista de Tablas

1. Introducción.

El uso de robots como herramientas didácticas en las aulas de clase ha demostrado ser de gran utilidad sobre todo en áreas pertinentes a la tecnología y ciencias relacionadas a la ingeniería[1]. Con el fin de explorar aplicaciones de robótica en un contexto educativo, surgen plataformas como RoboAct y Quemes, la primera implementa el modelo de control autónomo para la toma de decisiones de un robot y un modelo de cooperación para múltiples robots, en el contexto del teatro robótico[2], y la segunda inicialmente se planteó como un desarrollo de una plataforma robótica que permitiera la cooperación, comunicación y coordinación de sistemas multiagentes, y posteriormente se replanteó para darle un enfoque pedagógico, haciendo uso de robots móviles que se desplazan en una cuadrícula que representa la ciudad en la cual se ejecutan las tareas asignadas a los robots [3].

Los grandes avances en robótica han abierto la posibilidad de expandir las áreas de aplicación de ésta, por ejemplo, las artes y, especialmente las artes escénicas, ya que estas se presentan como una forma efectiva para llegar al público con el objetivo de compartir y comunicar algo en específico, o simplemente despertar ciertas emociones en la audiencia. Gracias a esto se puede observar el gran potencial que representa el teatro robótico y específicamente las aplicaciones pedagógicas de este. El uso de robótica con fines educativos se ha enfocado sobre todo en la enseñanza de áreas afines a la ciencia, ingeniería y tecnología debido a su inherente relación con estas. Sin embargo, el teatro desarrollado con robots se presenta como una aplicación de la robótica para un contexto pedagógico capaz de ampliar las áreas del conocimiento a las cuales se ha aplicado tradicionalmente, mostrando una gran capacidad para comunicar otro tipo de conocimientos y educar por medio del contenido de una obra teatral. Además, debido al interés natural que generan los robots, el teatro robótico se genera un entorno de aprendizaje con estrategias didácticas novedosas [1]. Varios desarrollos a nivel de hardware y software se han realizado en la Pontificia Universidad Javeriana, generando avances importantes en el diseño e implementación de sistemas para realizar dramatizaciones de teatro robótico, esto ha generado la oportunidad de desarrollar numerosos trabajos de grado en las carreras de Ingeniería Electrónica e Ingeniería de Sistemas.

Este trabajo de grado hace parte de un proyecto de investigación que busca explorar nuevas aplicaciones de robótica en un contexto educativo, por ejemplo, el teatro robótico como herramienta para enseñar temáticas no técnicas, esto se logra por medio del mensaje trasmitido a la audiencia a través de los libretos ejecutados por robots en una dramatización, fomentando en los usuarios habilidades como creatividad, resolución de problemas, comunicación y colaboración[4]. Siguiendo esta línea y, debido a la reciente adquisición por parte del grupo de investigación SIDRE de la pantalla multitáctil 3M C556PW de 55’ y 60 puntos, se propuso una integración entre la pantalla, RoboAct y Quemes, donde: la pantalla se usa de escenario, de interfaz de interacción con el usuario y para adquirir la información de puntos de contacto activados, con el fin de localizar los robots que se mueven sobre esta; RoboAct permitirá ejecutar libretos teatrales para cada robot, además, gestionará la comunicación y cooperación entre estos; Quemes se usa como plataforma robótica, utilizando los robots móviles como actores en una dramatización. Integrar la pantalla con las 2 plataformas presenta una forma muy interesante de implementar un sistema para crear dramatizaciones de teatro robótico, generando ventajas que abarcan desde la interacción con el usuario hasta la forma de adquirir la información de localización y orientación de los robots actores, fundamental en el contexto de los sistemas multiagentes y más específicamente en el caso de RoboAct, dado que cada agente necesita tener conocimiento de su entorno con el fin de ejecutar tareas, cumplir metas comunes y tener cooperación.

Este documento muestra el proceso de diseño e implementación del sistema y está organizado de la siguiente forma: en el capítulo 2 se enuncia el objetivo principal del proyecto, enunciando de manera general lo que se realizó, además, el objetivo general y específicos; en el capítulo 3 se describe de manera más detallada RoboAct, Quemes y los principios de funcionamiento de la pantalla, también, se enuncian antecedentes de teatro robótico y localización de robots en ambientes interiores; el capítulo 4 describe el proceso de

desarrollo del trabajo de grado, y cada sección corresponde a la solución de un objetivo específico exceptuando el objetivo específico 5, de esta forma la sección 4.1 corresponde al objetivo específico 1, la 4.2 al 2, la 4.3 al 4 y la 4.4 al 3; el capítulo 5 muestra el proceso de validación y pruebas del sistema y los subsistemas; el capítulo 6 corresponde al análisis de resultados; el capítulo 7 contiene las conclusiones y recomendaciones obtenidas.

2. Objetivo del proyecto

En este trabajo de grado se diseñó y posteriormente se implementó en Java una aplicación, que se ejecuta en un computador, la cual que permite a al usuario (niños entre 9 y 14 años) configurar por medio de una interfaz gráfica, los elementos básicos para crear dramatizaciones de teatro robótico, tales como: escenarios, escenas, objetos virtuales, actores virtuales y reales (robots Quemes), trayectorias de desplazamiento para cada tipo de actor. La aplicación integra la pantalla 3M C556PW y las plataformas de la siguiente manera:

- RoboAct, que se encargará de gestionar la cooperación y comunicación entre los robots actores, así como de la ejecución de los libretos por parte de estos, para esto se implementó un protocolo de comunicación por medio de *Sockets*, con el fin de enviar información a RoboAct de la localización y orientación de cada robot ubicado sobre la pantalla, así como también la localización de los actores virtuales. RoboAct se ejecuta paralelamente en el mismo computador que la aplicación desarrollada en este trabajo de grado.
- Quemes, los robots móviles que se desplazan sobre la pantalla. Se implementó un protocolo de comunicación entre la aplicación y cada robot por medio de Bluetooth. Esto es necesario para realizar el control de desplazamiento de los robots que deben seguir las trayectorias definidas por el usuario.

2.1. Objetivo general

Desarrollar un sistema de integración entre las plataformas RoboAct, Quemes y la pantalla multitáctil 3M C556PW para creación de obras de teatro robótico.

2.2. Objetivos específicos

1. Diseñar una interfaz física que permita el reconocimiento de objetos ubicados sobre la pantalla multitáctil mediante el uso de firmas digitales de tres puntos.
2. Diseñar e implementar una aplicación con las siguientes funcionalidades: cálculo de posición y orientación de los objetos ubicados sobre la pantalla con una latencia entre 114.8 ms y 600 ms, identificación del tipo de objeto ubicado sobre la pantalla, creación de trayectorias para robots y objetos virtuales sobre la pantalla, generar escenarios para visualizarlos en la pantalla.
3. Implementar un protocolo que permita realizar la comunicación entre el sistema y la plataforma RoboAct.
4. Desarrollar una interfaz gráfica para la interacción con usuario en la definición de rutas de robots y objetos y la generación de escenarios.
5. Validar el sistema con usuarios en términos de casos de usabilidad y capacidad de integración con la plataforma RoboAct.

3. Marco Teórico

3.1. Teatro Robótico

El teatro robótico es una de las múltiples incursiones de la robótica en las artes siendo un campo de investigación relativamente reciente con un gran crecimiento a futuro gracias a los avances tecnológicos en robótica, sensores, inteligencia artificial y el desarrollo de la teoría de los sistemas multi-agentes[2], permitiendo autonomía en los robots e interacción entre ellos. En concreto, se podría definir el teatro robótico como el uso de robots en el contexto de dramatizaciones teatrales, ya sea que los robots estén interactuando con humanos o con ellos mismos, con el fin de dar una interpretación teatral a un guion.

Se han realizado interpretaciones teatrales como se muestra en[5] desarrollando el libreto de una obra de 18 minutos donde los dos personajes principales Xavier (humano) y PR2 (robot) discuten cómo es posible hallar campos de interés en común. El trabajo que se presenta en [6] explora tres tecnologías principales: una plataforma de desarrollo de software integrado para control de escenarios inteligentes, un sistema de visión estéreo que rastrea múltiples características de la audiencia participante en tiempo real y un robot intérprete autónomo e interactivo con expresiones naturales y expresivas, con un enfoque lúdico y de entretenimiento.

3.1.1. RoboAct

RoboAct surge de la necesidad de generar un modelo guía para el diseño e implementación de robots actores, el cual se enfoca en aspectos como la autonomía, la cooperación y la expresión emocional del robot. El modelo propuesto es basado en el paradigma multi-agentes, el cual permite analizar y modelar un problema como un conjunto de agentes racionales que interactúan y cooperan entre sí para solucionarlo[2].

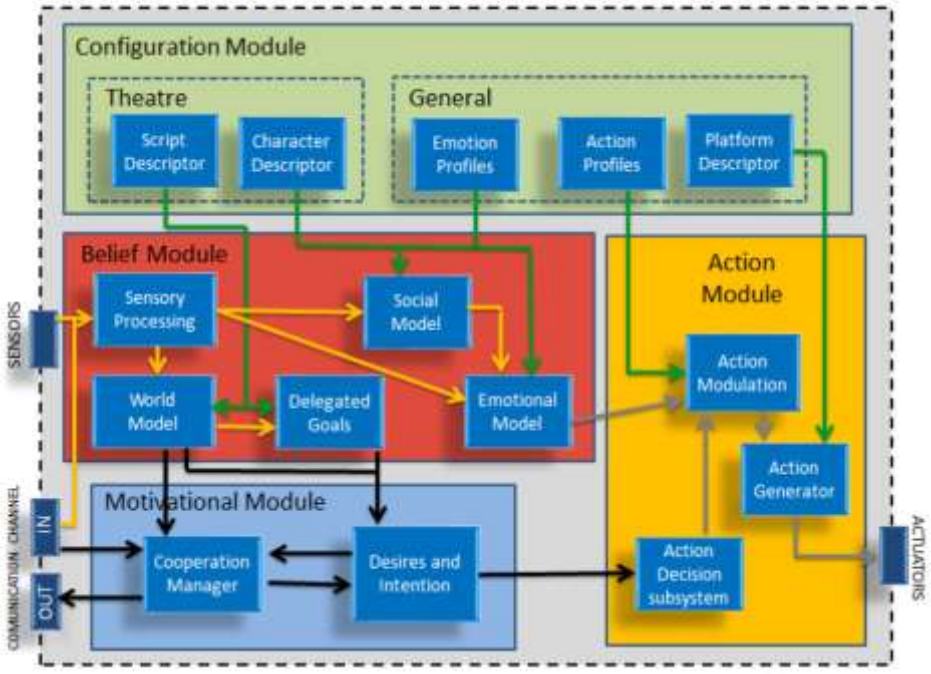

Figura 3.1.1 Modelo del agente actor, fuente [2].

RoboAct implementa un modelo para robots que participan en obras teatrales (robot actor), cada actor se modela como un agente racional y cooperativo como se puede observar en la figura 3.1.1, la particularidad del modelo se encuentra principalmente en el módulo de configuración que contiene los descriptores de libreto y de personaje, haciendo posible que el agente se desempeñe en el contexto del teatro robótico. El proceso deliberativo del robot actor se basa en la arquitectura BDI (creencias, deseos e intenciones) que permite un razonamiento del agente con capacidades y recursos limitados, además, dado que el modelo BDI tiene como objetivo generar un comportamiento similar al humano, resulta muy pertinente utilizarlo para el enfoque de teatro robótico. Como RoboAct se planteó para robots o agentes físicos, la forma de conocer e interactuar con su entorno es a través de los sensores, actuadores y el canal de comunicaciones, aclarando que los sensores y actuadores sí representan un componente físico del robot, mientras que, el canal de comunicaciones representa una infraestructura de comunicación entre agentes que, en ocasiones puede ser un componente físico y en otras no.

Finalmente, ya que varios robots actores estarán interactuando entre sí, se hace necesario el uso de un *framework* que permita desarrollar sistemas multi-agentes, específicamente, RoboAct es implementado en BESA[7] desarrollado en la Pontificia Universidad Javeriana.

3.1.2. Quemes

Quemes es el resultado de un proyecto de investigación de los grupos SIDRE y TAKINA del departamento de sistemas de la Pontificia Universidad Javeriana, en un principio se planteó como una propuesta para desarrollar y profundizar la teoría de sistemas multi-agentes y de robótica cooperativa, posteriormente se fue desarrollando para enfocarse como una herramienta pedagógica[8]. 'La plataforma Quemes es una herramienta pedagógica que permite a los usuarios ver reflejado el diseño y la programación orientada a eventos bajo el paradigma de los sistemas multiagentes de manera gráfica e intuitiva'[9, p47]. Al igual que en RoboAct se utiliza BESA como *framework* de desarrollo de sistemas multiagentes.

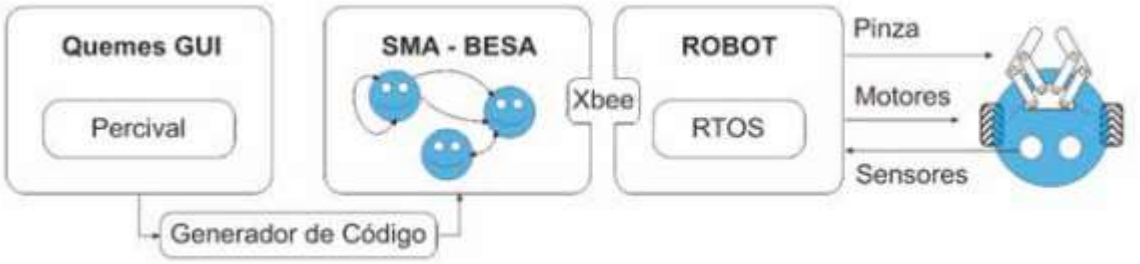

Figura 3.1.2 Arquitectura de la plataforma Quemes. fuente [10]

En la figura 3.1.2 se ilustra la arquitectura inicial de la plataforma Quemes, esta consiste en 3 capas:

- Interfaz gráfica, permite al usuario programar el comportamiento de cada robot y establecer vínculos y cooperación entre estos.
- Control inteligente, implementado en BESA y se encarga de gestionar el sistema multiagentes que representa la programación hecha por el usuario, en un principio se utilizaron módulos XBee para establecer la comunicación con el robot, actualmente esto se hace por medio de Bluetooth.
- Robot, se construyen a partir de: 2 servomotores, una tarjeta Arduino, 5 sensores infrarrojos, 1 módulo Bluetooth HC-05 y los componentes relacionados al diseño mecánico. Pueden realizar 5 movimientos (adelante, atrás, girar a la derecha e izquierda y sobre su propio eje) estos se logran por medio de un PWM implementado en la tarjeta Arduino de cada robot. Ya que el entorno en el

cual se desenvuelven los robots es una cuadricula hecha con cartulina y cinta negra tienen implementado un seguidor de línea a partir de los sensores infrarrojos, de esta forma se controla la trayectoria que seguirá el robot de acuerdo a lo programado por el usuario.

3.2. Localización en robots móviles

En aplicaciones multi-robot, la localización de cada uno de los miembros del equipo es uno de los factores determinantes para garantizar el buen funcionamiento del sistema [11]. Debido a la necesidad de realizar desplazamientos para lograr ejecutar una tarea orientada a cumplir una meta global, es indispensable tener conocimiento de la ubicación de cada uno de los robots que componen el sistema, más aún cuando se requiere cooperación para lograr cumplir con el objetivo en común.

Existen varias técnicas para estimar la localización de un robot en ambientes interiores que utilizan señales inalámbricas y la información obtenida a a partir de los sensores del robot. Los métodos más utilizados son: tiempo de llegada (TOA), diferencia de tiempo de llegada (TDOA), ángulo de llegada (AOA), indicador de potencia de la señal recibida (RSSI)[12]. Tales métodos matemáticos de localización pueden ser implementados utilizando señales inalámbricas como WiFi y Bluetooth. Es pertinente aclarar que para implementar cualquiera de los métodos anteriores es necesario poseer transmisores y receptores de señal en el sistema. Tal y como se discutió anteriormente, el uso de hardware adicional incrementará el costo y complejidad de los robots negando su uso en contextos de escuelas oficiales de Colombia a las cuales podría estar orientado el proyecto en un futuro.

El uso de señales WiFi para localizar robots en ambientes interiores ha sido investigado ampliamente [13][7][14]. Principalmente, se usan los sensores WiFi-instalados en los robots para obtener el RSSI de la señal proveniente de los puntos de acceso instalados en el entorno donde se ubica el robot, posteriormente los datos son procesados digitalmente, aplicando diferentes algoritmos para calcular la posición, tales como Localización y Mapeo Simultáneo (SLAM por sus siglas en inglés) y métodos Monte Carlo.

Para el caso de señales Bluetooth también es posible realizar la medición del RSSI y posteriormente determinar la ubicación del robot. Particularmente en [8], se propone un método de localización partiendo del uso de 3 faros bluetooth, realizando un mapeo entre el RSSI y la distancia entre el robot y el faro. A continuación, utilizan trilateración para calcular la posición del robot respecto a los 3 faros, finalmente se usa un filtro de partículas que concretamente implementa un método de Monte Carlo secuencial para reducir el error. Los resultados experimentales de este artículo muestran que el error medio observado es de 0.427m con una desviación estándar de 0.229m. Este error no resultaría aceptable en el contexto del presente trabajo de grado, teniendo en cuenta que la pantalla tiene dimensiones de 1.21m x 0.680m.

El método de localización propuesto en [9] utiliza un sistema RFID con una red de 198 *IC Tags* pasivos ubicados en forma de malla cuadricular sobre el piso del ambiente en el cual se desplaza el robot. El robot posee además una antena que detecta los *IC Tags* cuando está cerca de estos, enviando la información a un computador para realizar los cálculos de la posición estimada del robot. En este artículo se logra obtener un error medio de 6.0cm para el eje x y 5.3cm para el eje y. Aunque estos valores de error son menores que para el caso de señales Bluetooth, la implementación de este sistema supone un incremento en el costo del trabajo de grado por la cantidad de *hardware* adicional que se requiere para implementar el sistema de estimación de la localización.

3.3. Pantalla Multitáctil

La pantalla táctil utilizada en el proyecto es la fabricada por 3M cuya referencia es c5567pw. Esta pantalla utiliza tecnología de capacitancia proyectada (3M PCT), la cual, a comparación de las infrarrojas y las ópticas, tiene un mejor desempeño alcanzando un tiempo de reconocimiento de 10 ms o menor. Una de las tecnologías notorias de 3M en este producto es su algoritmo de rechazo de palmas, el cual evita tener en cuenta contactos indeseados. Con la capacidad de registrar hasta 60 contactos al mismo tiempo, esta pantalla cuenta con 18,600 puntos sensibles de contacto.

Una de las principales ventajas de utilizar la tecnología capacitiva es que, a diferencia de las pantallas resistivas, soporta múltiples contactos simultáneos, algo bastante relevante para el contexto de este trabajo de grado. Esta tecnología utiliza la capacitancia de los objetos, para determinar posición del contacto en dos dimensiones o coordenadas haciendo un procesamiento de la variación del campo eléctrico que se presenta en la pantalla. Esto evita la necesidad de generar presión sobre la pantalla, permitiendo que con toques suaves la pantalla reconozca la ubicación de la pulsación.

Al tocar la pantalla, como se observa en la figura 3.3.1, la capacitancia del cuerpo y la diferencia de cargas entre la pantalla y el cuerpo genera un flujo de carga entre estos, produciendo así una perturbación del campo electroestático de la pantalla, que los sensores capacitivos son capaces de percibir, este cambio es procesado por el controlador de la pantalla haciendo uso de diferentes algoritmos para determinar la ubicación en dos dimensiones de la perturbación. Lo anterior permite que materiales conductores puedan ser utilizados para accionar un punto de contacto siempre y cuando, la diferencia de cargas sea suficiente para generar un flujo, por ejemplo, un conductor conectado a tierra es suficiente, algo que resulta ser muy conveniente para el diseño de la interfaz de reconocimiento, explicada en el siguiente capítulo.

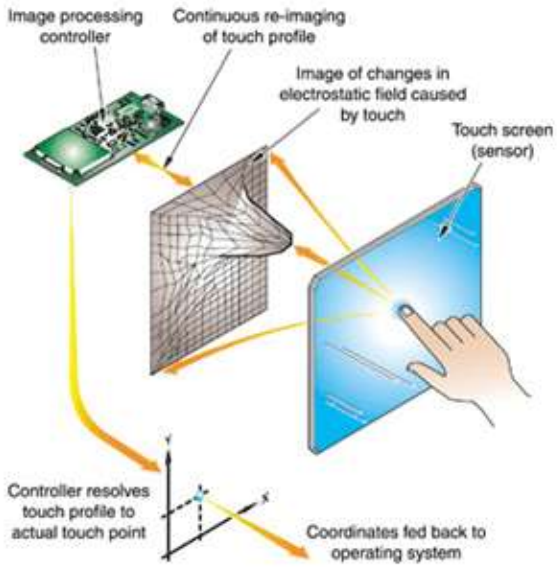

Figura 3.3.1 Funcionamiento de pantalla táctil capacitiva, fuente [14]

4. Desarrollo

Este capítulo ilustra el proceso de diseño e implementación de la aplicación desarrollada en el trabajo de grado, en la figura 4.1 se muestra un diagrama en bloques general de todo el sistema, incluyendo el flujo de información entre los diferentes componentes y módulos, así como bajo qué protocolos se establecieron las comunicaciones.

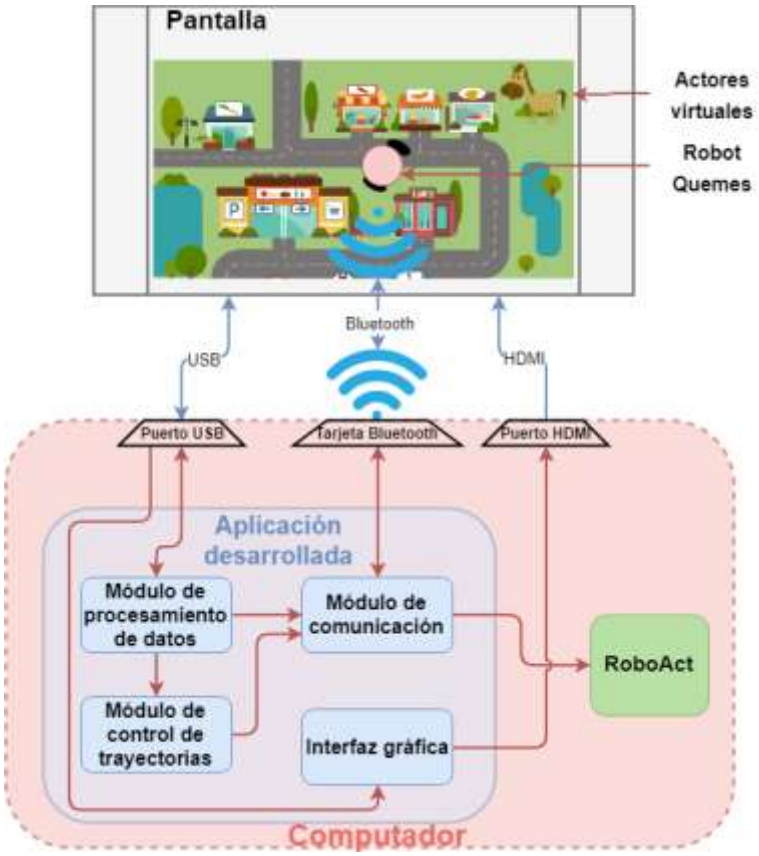

Figura 4.1 Diagrama en bloques general, fuente propia.

El sistema general está conformado por 4 componentes: la pantalla, robots Quemes, RoboAct y la aplicación desarrollada, los últimos 2 se ejecutan paralelamente en un computador. A continuación, se da una breve explicación de las funciones principales de cada uno de los componentes y de los módulos de la aplicación.

Pantalla: Se conecta al computador por medio de HDMI y USB, con el primero se logra la visualización de todo el sistema operativo y específicamente de la interfaz gráfica con el fin de permitir la interacción con el usuario, con el segundo se obtienen la información de los puntos de contacto presionados en la pantalla.

RoboAct: Gestiona todo lo relacionado al teatro robótico, para esto, se comunica con la aplicación por medio de Sockets solicitando el envío de información cada vez que lo requiera.

Robots Quemes: Como fue mencionado antes, la plataforma quemes tiene 3 capas, sin embargo, en este trabajo de grado sólo se hizo uso de una de estas, los robots, a los cuales se les instaló una interfaz capacitiva con el fin de generar puntos de contacto en la pantalla, permitiendo así ser reconocidos y localizados por el sistema, lo cual es fundamental teniendo en cuenta que los robots se desplazan sobre esta. Para controlar las

trayectorias de movimiento de los robots se implementó un protocolo de comunicación entre la aplicación y estos a través de Bluetooth.

Interfaz gráfica: Es la que permite al usuario interactuar con el sistema, en esta se muestran las opciones de configuración para crear una obra de teatro robótico

Módulo de procesamiento de datos: Tiene la función de realizar el procesamiento de la información entregada por los sensores capacitivos de la pantalla al computador por medio de la conexión USB. Este implementa el algoritmo para identificar los robots ubicados sobre la pantalla, calcula la ubicación y orientación relativa a la pantalla y envía la información a los módulos de comunicación y control de trayectorias.

Módulo de control de trayectorias: Recibe la información del módulo de procesamiento de datos e implementa el algoritmo para controlar el desplazamiento de los robots con el fin de que sigan la trayectoria definida por el usuario, posteriormente envía la acción de control al módulo de comunicaciones para que sea transmitida por Bluetooth a los robots.

Módulo de comunicación: Implementa los protocolos de comunicación entre la aplicación, RoboAct y los robots, los cuales transmiten la información recibida por los módulos de procesamiento de datos y control de trayectorias.

4.1. Interfaz para reconocimiento de objetos ubicados sobre la pantalla

Para lograr reconocer los robots ubicados sobre la pantalla sin implementar sistemas de localización de robots en ambientes interiores como los mencionados en la sección 3.2 y, haciendo uso de los 60 puntos de contacto que puede reconocer esta, se hace necesario el diseño de una interfaz física que se pueda instalar fácilmente sobre los objetos y robots, haciendo que estos generen un punto de contacto sobre la pantalla para procesarlos e identificar el objeto que fue ubicado.

4.1.1. Firmas Digitales.

Se define una firma digital como un identificador único asignado a cada robot u objeto, que pueden ser implementadas de diferentes formas, en este trabajo de grado, las firmas digitales se plantearon como un conjunto de 3 puntos de contacto generados en la pantalla que garanticen:

- Identificación del objeto ubicado sobre la pantalla.
- Estimación de la localización del objeto ubicado en la pantalla.
- Estimación de orientación del objeto.

Además de los requerimientos anteriores, se encontraron 2 limitantes fundamentales para el diseño de éstas: el espacio disponible para instalar la interfaz física y la distancia mínima a la cual dos puntos de contacto pueden ser activados simultáneamente, que es una restricción generada a nivel de software, mas no por la resolución de la pantalla y de los sensores capacitivos de esta, específicamente, la pantalla deja de reconocer puntos que son activados simultáneamente si están a una separación menor de 1.2 cm.

Se puede observar en la figura 4.1.1 el procedimiento abordado para diseñar el conjunto de firmas. Primero se definió el método de reconocimiento, la pantalla proporciona información sobre el conjunto de puntos de contacto que están siendo activados en esta como un punto (x, y), por lo tanto, los tres puntos de contacto generados por medio de la interfaz física forman un triángulo que está contenido en un eje coordenado, que para el contexto de la pantalla es en x de 0 a 1920 y, en y de 0 a 1080. Se decidió entonces realizar el

reconocimiento de las firmas, determinando el área del triángulo que estas generan. Para calcular el área del triángulo del área de la figura 4.1.1 se tiene la ecuación 4.1, donde *a* es la base y *h* la altura.

$$Area = \frac{a \cdot h}{2} \tag{4.1}$$

Por esto se hace necesario tener en cuenta varias consideraciones respecto al diseño de las distancias en cada una de las firmas digitales:

- Se deja fija la distancia *a*, para facilitar la estimación de la orientación, en la sección 4.2.3 se hará más evidente el porqué. Esto conlleva a que los puntos p1 y p2 son fijos para todo el conjunto de firmas.

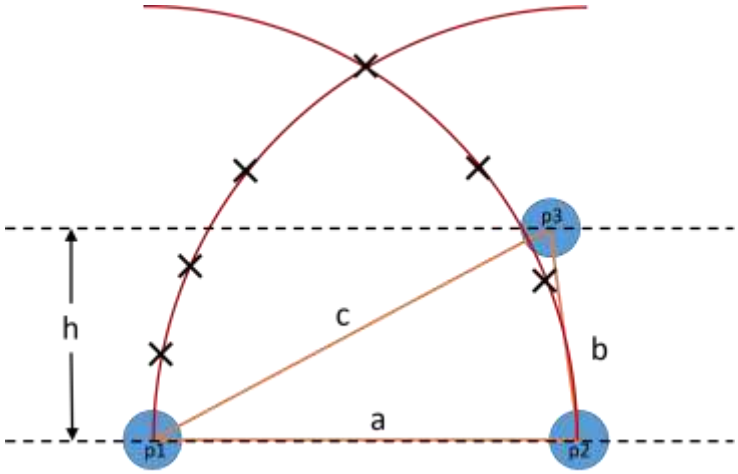

Figura 4.1.1 Puntos de contacto generadores de firma digital, fuente propia.

- No se puede repetir la distancia *h*, porque teniendo *a* fijo para todas las firmas, generaría áreas iguales, haciendo imposible distinguir los objetos con esas dos mismas firmas. La línea punteada en la figura 4.1.1 ilustra la ubicación del punto p3 que no puede ser usada por la siguiente firma.
- El triángulo no puede ser equilátero, si lo fuera, no podría determinarse la orientación. Lo anterior es lo mismo que $a \neq b \neq c$.
- El triángulo no puede ser isósceles, si lo fuera, no podría determinarse la orientación. Lo anterior es lo mismo que $a \neq c$. En la figura 4.1.1 se ilustra las líneas circulares en rojo sobre las cuales no podría estar ubicado el punto p3.

El primer diseño de las firmas fue planteado como se ilustra la figura 4.1.1, definiendo *a* como la distancia mayor, incluso se realizó la impresión 3D del prototipo, sin embargo, esta opción genera problemas cuando dos objetos están muy cerca, ya que podrían tener puntos que, como se ilustra en la figura 4.1.2, estén separados también por una distancia *a*. esto podría hacer que se reconozcan erróneamente los puntos que pertenecen a un triángulo.

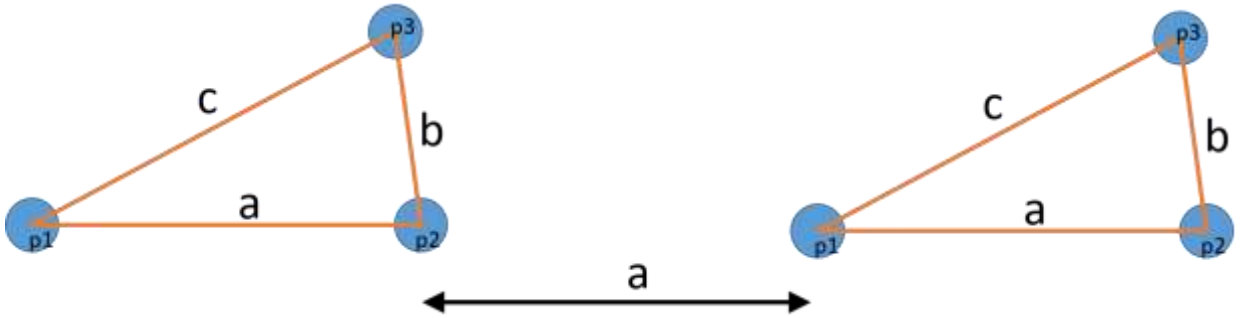

Figura 4.1.2 Problema al tener dos objetos cercanos, fuente propia.

Por lo anterior se decide hacer que *a* sea menor que *c* y *d*, reuniendo las consideraciones expuestas anteriormente, se llega al diseño final de las firmas digitales, expuesto en la figura 4.1.3.

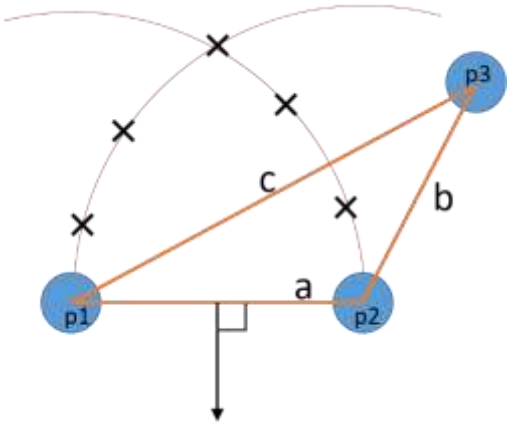

Figura 4.1.3Diseño final firmas digitales, fuente propia.

Además, en la figura 4.1.3 se incluye un vector perpendicular a la recta que pasa por los puntos p1 y p2, este vector indica la dirección del objeto al cual se le instaló la interfaz que tiene esa firma digital, para el caso de los robots, por ejemplo, el vector indicará la dirección en la cual normalmente se mueve hacia adelante. Conociendo la dirección del objeto, resulta viable estimar la orientación relativa a la pantalla del objeto, calculando ángulos entre vectores.

4.1.2. Diseño e impresión 3D.

Una vez diseñado el mecanismo de identificación de las firmas digitales, se procedió a diseñar la interfaz física. En el capítulo 3.3 se explicó resumidamente el funcionamiento de los sensores táctiles de la pantalla, por lo tanto, es necesario implementar un mecanismo que genere un punto de contacto sobre esta, para que esto ocurra se debe ubicar sobre la pantalla algún elemento que permita un flujo de carga hacia afuera de la

misma, generando una perturbación del campo eléctrico que está fijo, similar a como funciona en contacto con los dedos de los humanos.

Figura 4.1.4 Estructura para las firmas digitales (corte transversal), fuente propia.

El montaje propuesto se muestra en la figura 4.1.4. y consiste en una placa (color rojo) fabricada por una impresora 3D, que tiene diferentes orificios separados a las distancias definidas en la sección anterior, en los cuales se insertan 3 tornillos metálicos, la cabeza de los tornillos es la que hace contacto con la pantalla, ya que resultados experimentales mostraron que la superficie de contacto de la parte inferior de los tornillos no es suficiente para generar un punto de contacto en la pantalla. La parte inferior de los tornillos se introduce totalmente en la placa, para que haga contacto con una lámina metálica (color gris), que para este caso fue aluminio, y la lámina se conecta por medio de un cable a un terminal de tierra en las tarjetas de los robots (Arduino), con el fin de garantizar un flujo de carga de la pantalla a tierra. Inicialmente el diseño contemplaba poner un caucho conductor, como el usado en los *stylus* capacitivos, sin embargo, luego de realizar pruebas con robots en movimiento, se evidenció que estos obstaculizaban el desplazamiento de los robots.

La figura 4.1.5 muestra el primer prototipo que fue probado, se realizó el diseño 3D, se imprimió y se realizó el montaje propuesto, adhiriendo la placa con una cinta doble faz, la altura de la pieza se definió en 2mm con el fin de que exista una separación de 4.8 mm entre la cabeza de los tornillos y la estructura del robot, ya que la separación entre la estructura y el piso es de 5mm. Haciendo pruebas con las diferentes firmas generadas se evidenció un problema respecto a la separación de los puntos p3. La pantalla tiene un margen de error en el reconocimiento del punto, cuando la placa se mantiene sin movimiento, los puntos generados por los tornillos poseen una variación pequeña en pixeles, esta variación se ve magnificada en el cálculo del área. Por lo anterior y por cambios en el diseño físico de los robots se decide cambiar el diseño 3D de la placa, teniendo en cuenta que la cantidad de firmas se verá reducida drásticamente, por la separación entre cada posible punto p3.

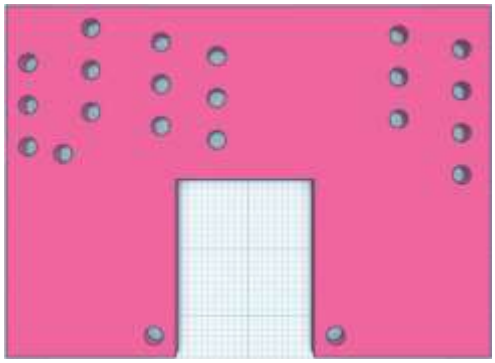

Figura 4.1.5 Primer prototipo 3D, fuente propia.

Se realizaron medidas con el fin de establecer la distancia mínima en la cual dos puntos de contacto pueden ser reconocidos sin problema, esta es 2.54 cm. Además, debido a los cambios que se realizan en el soporte de los robots, se hace posible incluir el espacio para la pieza, obteniendo el diseño que se observa en la figura 4.1.6, la altura de la pieza en este caso es de 4mm.

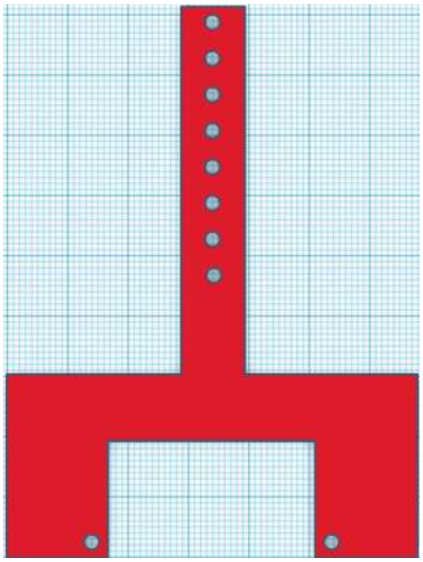

Figura 4.1.6 Segundo prototipo 3D, fuente propia.

En la figura 4.1.7 se puede ver el prototipo con los tornillos ensamblados.

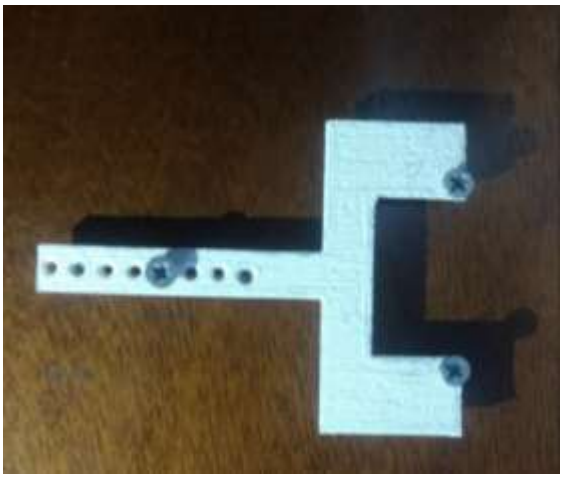

Figura 4.1.7Impresión del prototipo, fuente propia.

Una restricción de uso de este diseño es la distancia de separación entre las firmas, en este caso si se usa el último hueco para insertar el tornillo, las firmas deberán estar separadas a una distancia de 9 cm. Teniendo en cuenta las dimensiones de los robots, esto supondrá que deben estar separados 5 cm con el fin de no confundir las firmas.

4.2. Desarrollo de aplicación.

En este capítulo se exponen los pasos más importantes del proceso que se tuvo para desarrollar la aplicación, partiendo del diseño de la arquitectura general del sistema hasta la programación.

4.2.1. Arquitectura y diseño de clases.

El proceso de diseño de la aplicación parte del análisis de los requerimientos enunciados a continuación:

- Interfaz de usuario que permita: crear escenas
- Adquisición de los datos de puntos de contacto activados por las interfaces capacitivas al colocarlas sobre la pantalla.
- Identificar el conjunto de 3 puntos que pertenecen a cada firma digital cuando las interfaces capacitivas son ubicadas sobre la pantalla.
- Estimar la ubicación y orientación de los robots colocados sobre la pantalla.
- Enviar información a RoboAct de la escena (orientación y ubicación de los robots y objetos virtuales).
- Control de desplazamiento de los robots acorde a la trayectoria definida por el usuario.

El criterio principal de diseño fue emplear un módulo que dé solución a cada requerimiento, posteriormente se define el flujo de información entre los módulos y las fuentes externas a la aplicación teniendo en cuenta los protocolos de comunicación que manejan la pantalla y los robots, obteniendo así la arquitectura planteada en la figura 4.2.1. Esta se puede fragmentar en tres subsistemas de la siguiente forma: los bloques en azul representan los sistemas que interactúan y tienen comunicación por medio de diferentes protocolos con la aplicación desarrollada en este trabajo de grado; los bloques en rojo procesan la información; los bloques en verde constituyen la interfaz de usuario.

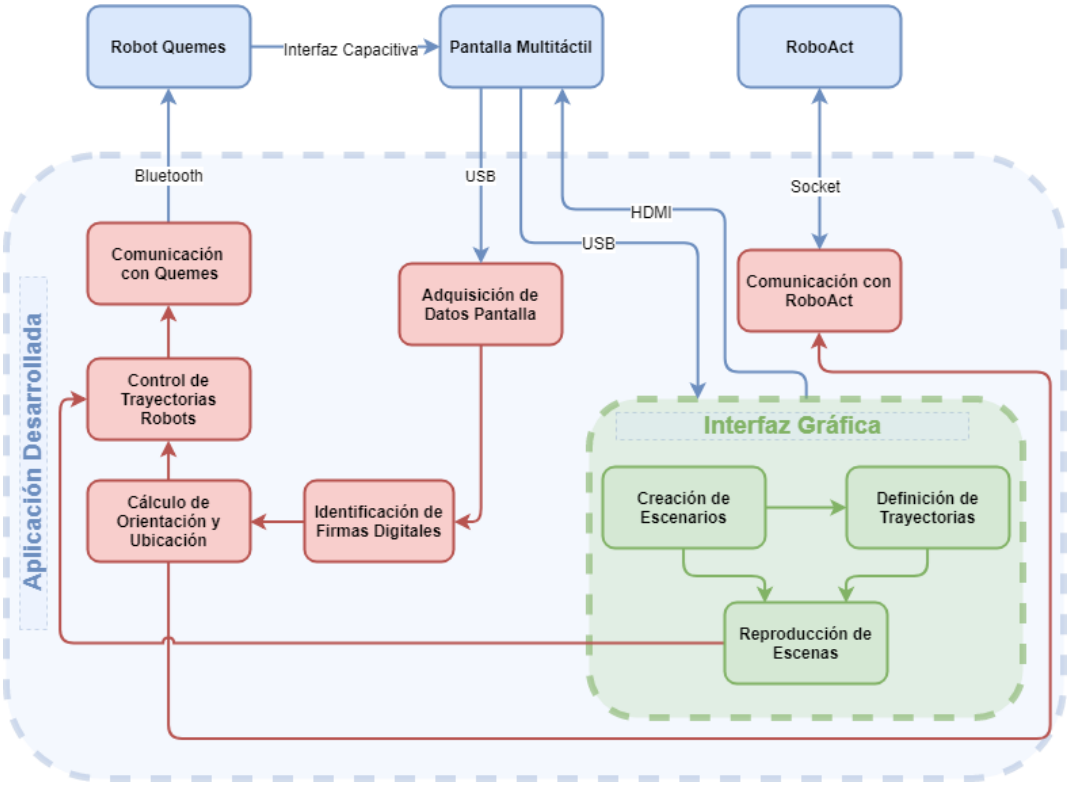

Figura 4.2.1 Arquitectura del sistema, fuente propia.

A continuación, se da una explicación del principio de funcionamiento cada módulo y de la información que intercambia,

Adquisición de datos de pantalla: Recibe la información que entrega la pantalla por el puerto USB de esta, respecto al estado de los sensores capacitivos. Se encarga de realizar la adquisición periódicamente para poder obtener información en tiempo real, respetando el tiempo mínimo que tarda la pantalla en actualizar la información de los sensores.

Identificación de firmas digitales: A partir de una lista de puntos presionados en la pantalla, que es obtenida en el módulo de adquisición, y de la información almacenada del área de cada firma, identifica cada conjunto de 3 puntos que representan una firma digital y a qué robot está asociada.

Cálculo de ubicación y orientación: Calcula la orientación en grados y la ubicación como punto en coordenada (x, y), de cada firma digital y relaciona estos datos al robot asociado a esta.

Control de trayectorias: Implementa el algoritmo que controla el movimiento del robot para seguir las trayectorias definidas por los usuarios, recibe la ubicación del robot y envía la señal del control al módulo de comunicación con Quemes.

Comunicación con Quemes: Implementa el protocolo de comunicación entre la aplicación y Quemes a través de Bluetooth. Se encarga de establecer el enlace cuando la aplicación se está iniciando y de enviar la señal de control al robot.

Comunicación con RoboAct: Utiliza un Socket para establecer un canal de comunicación entre la aplicación y RoboAct. Cuando RoboAct solicita que se le envíe información de la escena, el módulo se la envía con el formato de mensaje que se definió.

Creación de escenarios: Por medio de un menú de interacción implementado en la interfaz gráfica, permite al usuario crear un escenario con una imagen de fondo, objetos y actores virtuales (imágenes de árboles, animales, etc.) y robots. En este se implementan los códigos que permiten por medio de reconocimiento de gestos multitáctiles para modificar el escenario, por ejemplo: arrastrar para cambiar la posición de un objeto virtual, aumentar el tamaño colocando dos dedos sobre la imagen del actor y separando los dedos, etc.

Definición de trayectorias: Permite definir la trayectoria de desplazamiento para actores virtuales o robots, se implementa en la interfaz gráfica, pues el usuario dibuja con los dedos la trayectoria o escoge un punto inicial y final y posteriormente asocia la trayectoria a cada actor o robot.

Reproducción de escenas: Implementa el algoritmo que permite reproducir las escenas definidas por el usuario, debe proyectar las imágenes de fondo, objetos y actores conforme a cuál escena se va a reproducir. Tiene que indicar al módulo de control de trayectorias cuando se inicia la reproducción con el fin de que el movimiento de los robots y actores virtuales empiece sincronizadamente.

Teniendo una descripción general de la función de cada módulo y de cómo se comunica con los otros, se procedió a realizar un diseño de la arquitectura enfocado a el proceso de programación y, teniendo en cuenta que la aplicación es implementada en Java, se hace un diseño orientado a objetos, como lo ilustra la figura 4.2.2 que muestra el diagrama de clases. Se define a un objeto como la entidad fundamental de un programa escrito orientado a objetos, en el cual estos interactuarán entre sí, y una clase es una plantilla o modelo de diferentes tipos de objetos, concretamente una definición de un tipo de objeto[15]. En la metodología de diseño orientado a objetos existe el concepto de encapsulamiento, en el cual se agrupan las clases que desempeñan tareas relacionadas o comunes bajo un mismo paquete, de esta forma se decidió modelar el sistema con 5 paquetes, a continuación, se da una explicación de estos y de las clases principales:

Procesamiento.

Este paquete contiene todas las clases que tienen como función el procesamiento de datos de la pantalla, realización de cálculos, gestión de la aplicación, almacenamiento de las escenas, actores y trayectorias. Se

decidió establecer una clase por cada elemento presente en la creación de la obra como el escenario, actores y trayectorias. La clase *Sistema* almacena los diferentes objetos instanciados de las clases *Actor*, *Trayectoria* y *Escena.* También tiene 3 atributos de tipo *ComunicacionQ* para establecer la comunicación entre el sistema y los robots Quemes. Además, *Sistema* implementa todos los métodos necesarios para crear, añadir, eliminar los elementos anteriormente mencionados, así como también, los dos métodos que reconocen e identifican las firmas digitales y el que actualiza la ubicación y orientación de cada robot.

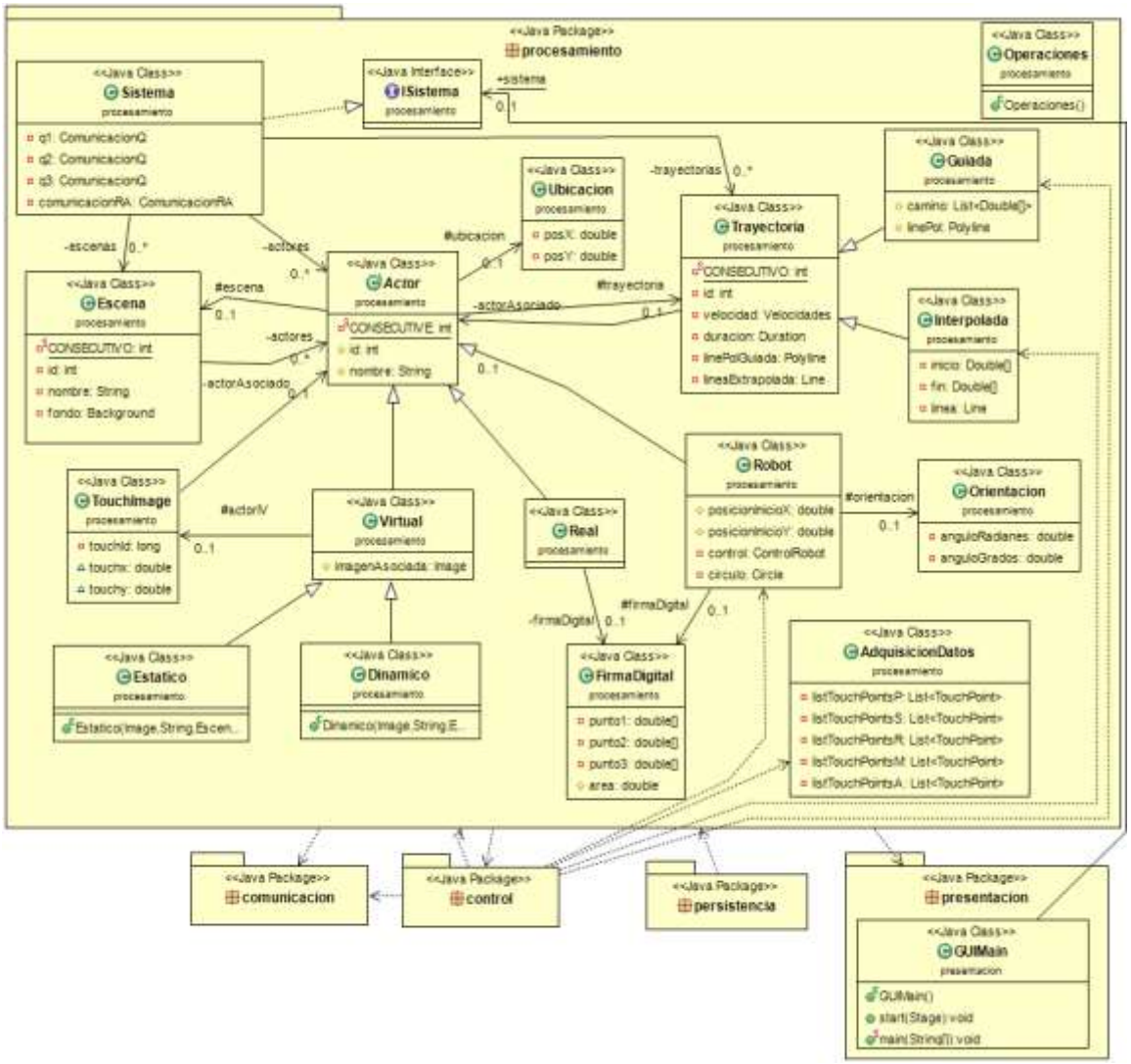

Figura 4.2.2 Diagrama de clases reducido, fuente propia.

La clase *Trayectoria* contiene la información de la ruta que deberá seguir el actor durante la reproducción de la escena, esta se puede generar de dos formas, por esto se modelan dos subclases, *Guiada* e *Interpolada.* Para generar una trayectoria interpolada, se le pide al usuario ingresar un punto de inicio y fin, estos se almacenan como atributos de la clase y se forma una línea recta entre los dos puntos. En la trayectoria guiada se le pide al usuario que la dibuje y posteriormente se almacenan como una lista conteniendo todos los puntos de contacto que fueron activados mientras se dibuja la trayectoria.

Se define al actor como el elemento que hace parte de la dramatización, ya sea virtual o un robot y que está contenido en una escena. Se decidió modelar el actor como la clase abstracta *Actor* ya que se hace necesario que las clases que heredan de ésta tengan diferentes implementaciones para los métodos relacionados con la ubicación y orientación. Las subclases *Virtual* y *Robot* son abstracciones de los diferentes tipos que puede

tener una escena. Para el reconocimiento y almacenamiento de las firmas digitales se modeló la clase *FirmaDigital* la cual almacena 3 puntos de contacto generados por la interfaz física, además, el área del triángulo generado por estos. La clase *Escena* se diseñó con el objetivo de que contenga toda la información necesaria para poder reproducir la dramatización que creó el usuario, contiene todos los actores creados por el usuario y el fondo.

Por otro lado, teniendo en cuenta que hay varias tareas que deben ejecutarse paralelamente para lograr interactuar efectivamente con el usuario, el uso de hilos se hace indispensable. Un hilo es un subproceso en el cual se puede ejecutar una secuencia de tareas [16] y en una CPU se puede hacer uso de esos subprocesos con el fin de realizar varias secuencias de tareas paralelamente. Por lo anterior, se decide que cada clase que modele algún proceso que deba ser ejecutado al mismo tiempo que el proceso principal debe ser un hilo también. En el paquete procesamiento la única clase que se diseñó como un hilo fue *AdquisicionDatos*, Esta clase implementa el manejo de los eventos generados por la pantalla al accionar un punto de contacto, estos se generan de diferentes formas: al presionar, mover o dejar estacionario un punto de contacto.

Persistencia:

Este paquete sólo contiene la clase *ManejoArchivos*, la cual implementa los métodos necesarios para guardar y cargar archivos. Se utilizó para generar los archivos csv resultantes del protocolo de pruebas implementado.

Comunicación:

Este paquete contiene las clases referentes a los protocolos de comunicación que se implementaron, que para este caso fueron 2, *ComunicacionQ* y *ComunicacionRA*, la primera se encarga de la comunicación por medio de Bluetooth con los robots Quemes, y la segunda con la plataforma RoboAct.

Control:

Contiene sólo una clase *ControlRobot* en la que se implementa el control de seguimiento de trayectoria de los robots y su funcionamiento es explicado en el capítulo 4.5.

Presentación:

Como se puede observar en la figura 4.2.3 este paquete contiene las clases que conforman la interfaz gráfica, dónde la clase principal es *GUIMain* es la que tiene el *Main Thread,* por lo tanto, es la clase ejecutable. Al ser esta la clase principal tiene como atributo un objeto de tipo *Sistema* que se instancia al abrir la aplicación, además, tiene otros 3 atributos que son necesarios para poder visualizar la interfaz gráfica que se diseñó y también inicializa el protocolo de comunicación con RoboAct al abrir el *socket* por el cual se realizará la comunicación. La clase *GUIControlFxml* tiene los métodos que implementan la interacción con el usuario, por esto es en ésta donde se realiza la adquisición de datos que vienen del usuario, por ejemplo, la definición de escenarios y trayectorias, estos se explicarán a mayor detalle en el capítulo 4.3.

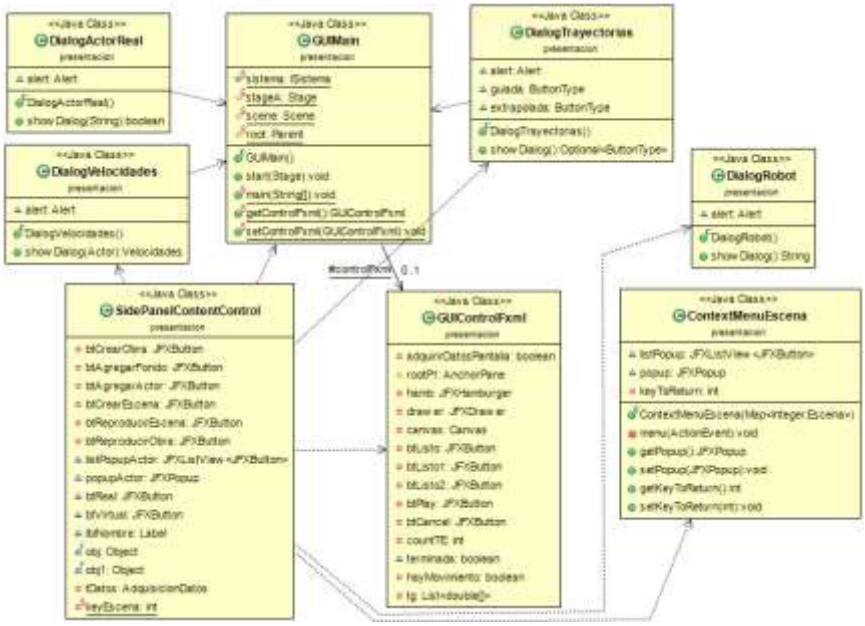

Figura 4.2.3 Diagrama de clases paquete presentación, fuente propia.

4.2.2. Identificación de firmas ubicadas sobre la pantalla.

La clase *AdquisicionDatos* tiene almacenada en forma de listas los puntos de contacto accionados, a continuación, se ilustra un ejemplo de cómo entrega la pantalla la información de los puntos:

Id	X	Y	Estado
1	320.0	560.0	*Pressed*
2	24.0	460.0	*Moved*
3	720.0	46.0	*Released*
4	240.0	245.0	*Stationary*

Figura 4.2.4 Formato de datos de contacto, fuente propia.

Cada columna representa la información relevante para procesar contenida en cada objeto de tipo punto de contacto, el estado del punto representa cómo fue generado por parte del usuario: el estado *Pressed* se genera al oprimir con los dedos la pantalla, *Moved* se genera al deslizar el dedo sobre la pantalla, *Released* es generado cuando se levanta el dedo de la pantalla después de haberla oprimido, *Stationary* se genera cuando se deja el dedo oprimiendo la pantalla. Con el fin de identificar el conjunto de 3 puntos que pertenecen a una firma digital, es necesario realizar un procesamiento de esta información y considerando que, se diseñan las firmas digitales para que no sea posible confundirlas cuando estas estén a la mínima distancia de separación, se puede identificar a cada firma como se explica continuación:

- Dependiendo de la clase que esté utilizando este método recibe una lista con los puntos presionados, estacionarios o en movimiento. Por ejemplo, para realizar la identificación de los robots mientras estos están desplazándose se usa específicamente la lista de puntos en movimiento, en cambio, al momento de iniciar la reproducción de la escena cuando aún los robots no han empezado su trayectoria, se utiliza la lista con los puntos presionados.
- Partiendo del punto ubicado en la primera posición de la lista, se calcula la distancia entre éste y todos los demás puntos de la lista por medio de la ecuación 4.2.

$$\overrightarrow{\|D\|} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4.2}$$

- Se crea un objeto de tipo *IdDistancias* en el cual se almacena, el índice de los 2 puntos y su distancia, el primer punto es fijo, considerando el paso anterior. *IdDistancias* es una clase auxiliar que se declara en la clase *Sistema.*
- Se genera una lista con los objetos de tipo *IdDistancias* y se organiza de menor a mayor distancia.
- Los dos primeros objetos de la lista tienen los 3 puntos que están más cerca entre sí, estos son puntos de una firma digital
- Se crea un objeto de tipo *FirmaDigital* a partir de los 3 puntos hallados en el paso anterior y se agrega a un mapa de firmas digitales.
- Se elimina de la lista de puntos los 3 puntos hallados.
- Se repite el procedimiento hasta encontrar todas las firmas digitales.
- Se retorna un mapa de firmas digitales, en este ya están presentes todas las firmas digitales que están generando puntos de contacto en la pantalla

Teniendo identificado a que firma digital pertenece cada conjunto de 3 puntos ya es posible reconocer cual firma corresponde a cada actor real, para lograrlo, se realiza el cálculo del área del triángulo que genera cada conjunto de puntos accionados con la ecuación 4.3[17], donde (x_1, y_1) , (x_2, y_2) y (x_3, y_3) son las coordenadas de los puntos leídos en la pantalla. Posteriormente, se recorre el mapa de actores que está almacenado en la clase *Sistema* para comparar el área de la firma digital que se calculó al momento de registrar el robot, con el área calculada a partir de los puntos leídos. Durante el proceso de desarrollo de este algoritmo, se evidenció que cada vez que se retira y vuelve a colocar el objeto en la pantalla, el valor del área varía aproximadamente un ±8%, por lo tanto, en el momento de realizar la comparación de áreas se hace necesario incluir el error y generar un intervalo de valores válidos de área para que la firma pueda ser reconocida.

$$A = \frac{1}{2}|x_1 \cdot y_2 + x_2 \cdot y_3 + x_3 \cdot y_1 - x_1 \cdot y_3 - x_2 \cdot y_1 - x_3 \cdot y_2\,| \tag{4.3}$$

4.2.3. Estimación de ubicación y orientación.

Ubicación:

Dado que los objetos son reconocidos con tres puntos de contacto, se hace necesario definir un punto en el eje coordenado (*x, y*) que represente la ubicación actual del objeto. Se decide entonces, tomar el baricentro o punto medio del triángulo, definido como el punto M desde el cual la distancia de este a cualquier línea es igual a un tercio de la suma algebraica de las distancias desde los vértices a la misma línea [18], o como muestra la ecuación 4.2

$$PM = \frac{Pa + Pb + Pc}{3} \tag{4.4}$$

Donde Pa, Pb y Pc son puntos cualesquiera que pasen por las líneas a, b y c respectivamente (remitirse a la figura 4.1.1).

Orientación:

Como ilustra la figura 4.2.1, la línea que genera los puntos fijos P1 y P2 de las firmas digitales es paralela a la línea del límite de la parte trasera del robot, por lo tanto, un vector perpendicular a ésta indicará la orientación del robot. Se decide entonces generar un vector normal a la recta P1P2 para luego calcular el ángulo entre el vector normal y el eje *x*, obteniendo así la orientación del robot relativa al eje *x* de la pantalla, a continuación, se explica el algoritmo implementado para realizar este cálculo.

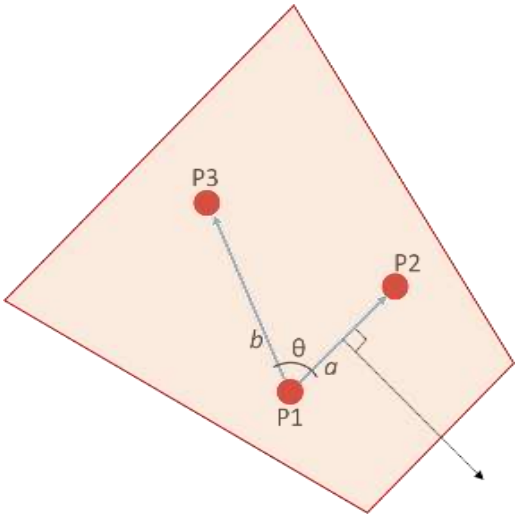

Figura 4.2.5 Vista para determinar orientación, fuente propia.

En la clase Robot se implementa el método que se encarga de integrar los diferentes métodos necesarios para calcular la orientación del robot, el algoritmo es el siguiente:

- Se calculan las magnitudes de los 3 vectores que se generan entre el origen del eje coordenado y los puntos de la firma digital asociada al robot.
- Se halla la distancia mínima entre los tres puntos con el fin de conocer cuáles son los dos puntos fijos P1 y P2.
- Se determina cuál de estos dos tiene la menor magnitud para calcular el determinante entre los vectores P1P2 y P1P3. La ecuación 4.5 define el determinante entre dos vectores $\vec{a}, \vec{b}$ y la 4.6 ilustra una definición geométrica del determinante en R^2. Se puede deducir que el signo del determinante depende del ángulo θ, por lo tanto, si el punto P3 está al lado izquierdo del vector P1P2 el signo del producto determinante será positivo, si está al lado derecho será positivo y si está contenido en la recta P1P2 será 0.

$$\det(\vec{a}, \vec{b}) = \begin{vmatrix} a_x & a_y \\ b_x & b_y \end{vmatrix} = a_x \cdot b_y - a_y \cdot b_x = \|\vec{a}\| \cdot \|\vec{b}\| \cdot \sin\theta \tag{4.5}$$

$$\det(\vec{a}, \vec{b}) = \|\vec{a}\| \cdot \|\vec{b}\| \cdot \sin\theta \tag{4.6}$$

- Ya que el punto P3 está ubicado en la dirección opuesta de la orientación del robot como se detalla en la figura 4.2.1 se define la variable *offset*, la cual es sumada al cálculo final de la orientación y puede tener solo dos valores, 0 si P3 está a la derecha o π si está a la izquierda.
- Teniendo el lado de P3, se genera un vector normal a la recta P1P2, para lograrlo se calcula el vector distancia, el cual es el vector que resulta de la resta P2-P1 y se transpone. Considerando que un vector normal es aquel que es perpendicular a la recta y que existen infinitos de estos, se decidió rotar el vector distancia 90° en el sentido contrario a las manecillas del reloj, para lo anterior se hace

uso de la matriz de rotación enunciada en la ecuación 4.5, donde x' y y' son las componentes del vector rotado un ángulo θ, luego este se normaliza.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \quad (4.7)$$

- Habiendo generado el vector normal a la recta P1P2, se calcula el ángulo entre este y el eje x, para lo cual se calcula la función inversa de la tangente. el ángulo obtenido representa la orientación del robot relativa al eje *x* de la pantalla.

4.3. Desarrollo de interfaz gráfica.

Teniendo en cuenta que los usuarios finales de esta aplicación son niños con edades entre 9 y 14 años, existe un gran reto a la hora de diseñar una interfaz gráfica de usuario que sea fácil de usar, en esta sección se muestra el desarrollo y las decisiones que se tomaron en el diseño e implementación de la interfaz. Es pertinente aclarar que, se decidió utilizar la librería JavaFx para implementar la interfaz gráfica y no las librerías más comúnmente utilizadas, ya que estas no tienen soporte para el manejo de eventos *multitouch*, que en el contexto de este trabajo de grado resulta obligatorio.

Los requerimientos de diseño de la interfaz gráfica son:

- Agradable y fácil de usar para el usuario.
- Permitir crear escenas
- Agregar un fondo a la escena.
- Agregar objetos y actores virtuales para configurar la escena.
- Manipular las imágenes de los actores virtuales y objetos, cambiar el tamaño, rotar y cambiar de posición, esto debe poder hacerse con los dedos presionando sobre la imagen.
- Definir trayectorias de desplazamiento para los robots y actores virtuales, ya sea dibujándola o eligiendo el punto inicial y final.
- Reproducir las diferentes escenas creadas.

4.3.1. Características principales.

Se decidió hacer un diseño gráfico sencillo y lo más minimalista posible para que su uso sea fácil e intuitivo, por esto y, partiendo de los requerimientos de diseño de la interfaz gráfica, se plantearon 2 componentes principales como se observa en la figura 4.3.1, el menú desplegable y el área donde se muestra el escenario. Para implementarlos, se programaron dos archivos en el formato xml con la ayuda de la herramienta *Scene Builder*, en estos archivos se almacena la información referente a la composición gráfica de cada uno de los elementos que componen la interfaz gráfica, estos archivos se vinculan con las clases *GUIControlFxml* y *SidePanelContentControl*, en las cuales se programa toda la lógica.

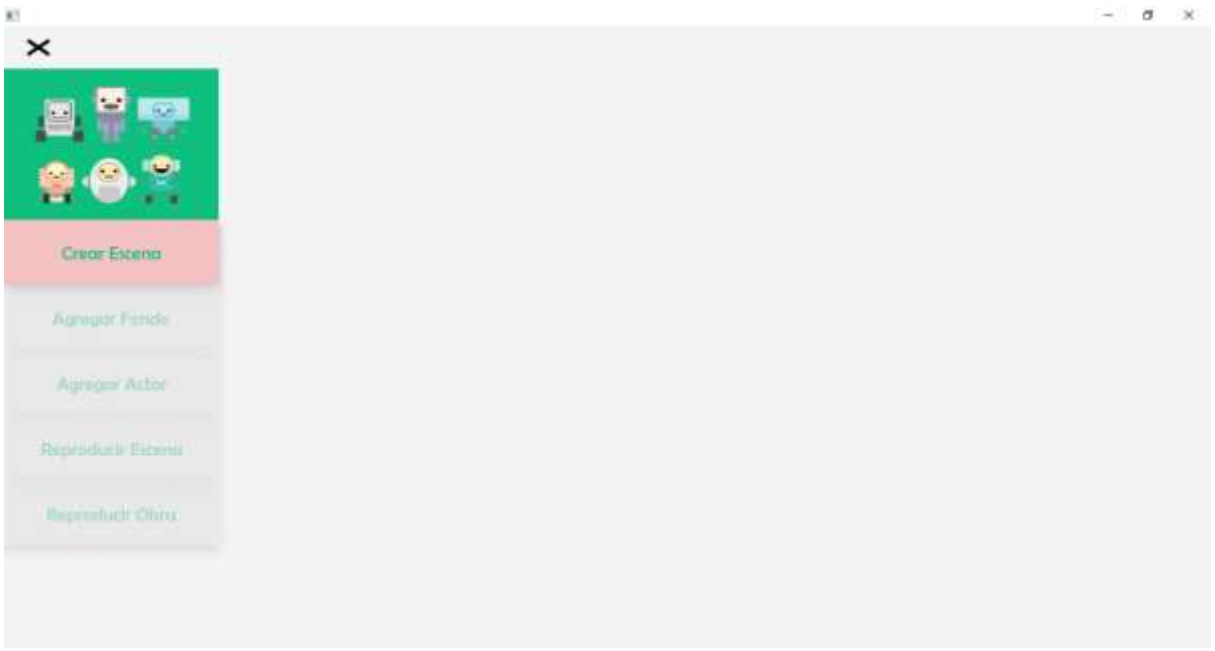

Figura 4.3.1Pantalla de inicio interfaz gráfica, fuente propia.

Se define el fondo de la aplicación de tipo *AnchorPane,* y sobrepuesto a este se declara un objeto de tipo *Canvas* ya que esta permite fácilmente obtener los puntos de contacto, así como también proyectar diferentes tipos de figuras sobre esta, y con el fin de poder visualizar las imágenes que se cargaran en el fondo se hace que tenga un color transparente, en adelante se le llamará lienzo a este objeto. Se puede observar que el menú despegable tiene varias opciones para elegir, sin embargo, ya que se decidió que la creación del escenario se hiciera de manera secuencial, cada botón se va activando consecutivamente. Además, la interfaz gráfica hace uso de cuadros de diálogos, estos interactúan con el usuario para definir los parámetros de los actores:

- El dialogo de velocidades muestra las opciones de velocidad de desplazamiento de los actores virtuales que son, lenta, media y rápida
- El dialogo de actor real permite escoger entre un objeto real estático y un robot.
- El dialogo de robot muestra la opción de seleccionar cuál de los 3 robots disponibles se quiere utilizar.
- El dialogo de trayectorias permite escoger entre los dos tipos de trayectorias para que el actor siga.

4.3.2. Creación de escenarios.

En la clase *SidePanelContentControl* se implementan todos los métodos necesarios para lograr la interacción con el usuario en la creación de escenarios, la figura 4.3.2 ilustra un diagrama de flujo del algoritmo diseñado, que se realizó partiendo de los requerimientos de la interfaz gráfica y del menú desplegable implementado.

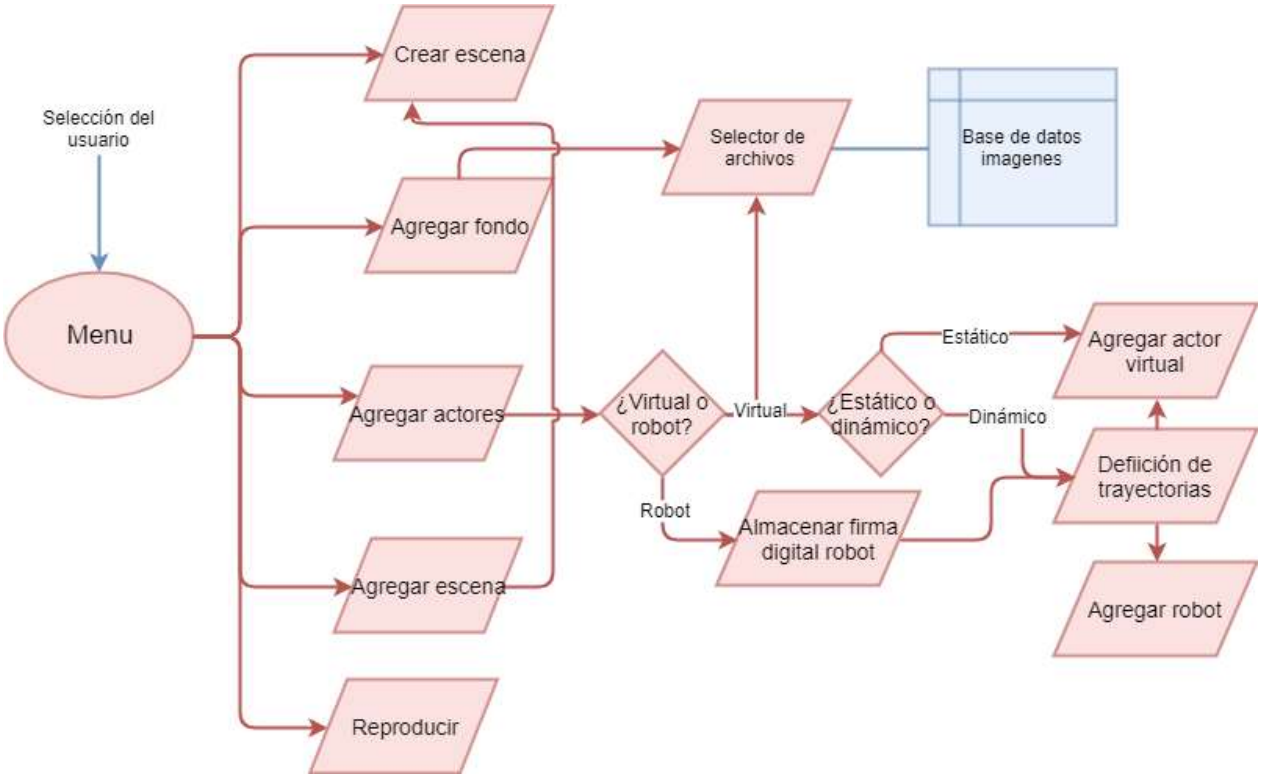

Figura 4.3.2 Diagrama de flujo interfaz gráfica, fuente propia.

Cuando el usuario presiona alguno de los botones del menú desplegable se invoca el método que corresponde a la selección del usuario como se observa en la figura anterior, ya que se fuerza al usuario a presionar los botones secuencialmente, el funcionamiento del algoritmo es el siguiente:

Crear escena: Se crea un objeto de tipo escena con un identificador único y consecutivo.

Agregar fondo: Permite definir el fondo de la escena por medio de un seleccionador de archivos que muestra las opciones de imágenes para los fondos.

Agregar actores: Si el actor es virtual se escoge alguna de las imágenes por medio del seleccionador de archivo, seguidamente se decide si va a tener movimiento en la dramatización (estático o dinámico), para los estáticos el proceso termina en este punto, para los dinámicos se debe definir la trayectoria a seguir. Si el actor es de tipo robot, se almacena la firma digital asociada a este, para lo cual el usuario debe seguir el procedimiento indicado en la pantalla, posteriormente se define la trayectoria que seguirá el robot. Como paso final se almacena toda la información del actor en el objeto de tipo escena.

Agregar escena: Permite crear una nueva escena, deja de proyectar todos los elementos de la escena que se estaba configurando anteriormente, seguidamente crea una nueva para que el usuario a repetir los pasos de configuración de escena.

Reproducir escena: Despliega un menú para escoger cuál escena reproducir, cuando se selecciona, carga todos los elementos configurados por el usuario para esa escena, agregando un botón para iniciar la reproducción.

Un ejemplo de creación de escenarios por medio la interfaz gráfica diseñada se observa en la figura 4.3.3.

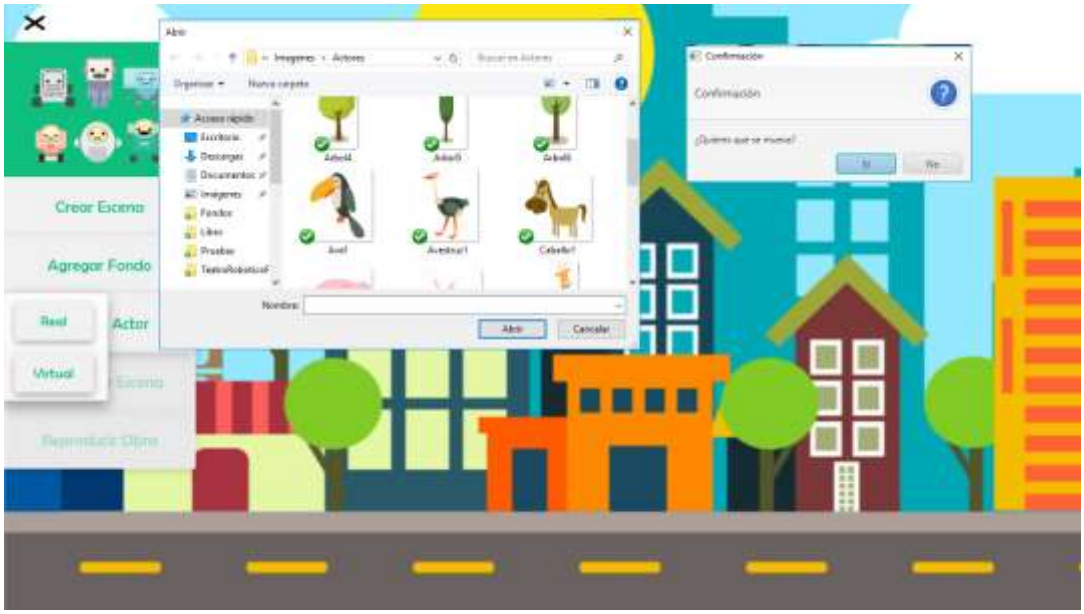

Figura 4.3.3Menú para agregar actor virtual estático, fuente propia.

4.3.3. Definición de trayectorias.

Los métodos para definir las trayectorias se implementaron en la clase *GUIControlFxml* la forma de adquirir la trayectoria es igual independientemente del tipo de actor, sin embargo, se crearon métodos para cada tipo ya que, en el caso de los actores virtuales el método también ajusta la posición de la imagen del actor con las coordenadas del inicio de la trayectoria. Como ya se han explicado, existen dos tipos de trayectorias que puede definir, estas se muestran en el dialogo de trayectorias, en el cual el usuario debe elegir entre dibujarla, para las trayectorias guiadas, o escoger un punto de inicio y final, para las trayectorias extrapoladas, después de seleccionar la opción en el dialogo y si el actor es virtual, se muestra el dialogo de velocidades en el cual el usuario pude optar por 3 velocidades, si el actor es de tipo robot el dialogo de velocidades no se muestra. Los procedimientos anteriores son iguales en los dos tipos de trayectorias, culminando en el paso en el cual se le pide al usuario definir la trayectoria, el algoritmo implementado para cada tipo de trayectoria se explica a continuación.

Trayectoria Guiada:

Se le pide al usuario que dibuje la trayectoria que quiere que siga el actor y, simultáneamente, se invoca el método que implementa el algoritmo de la figura 4.3.4, este tiene una lista de en la que se almacenan los puntos que componen la trayectoria, cada punto se definió como un vector de tamaño 2 en el cual la posición 0 corresponde a la coordenada en *x* y la 1 a la coordenada *y*, el tipo de dato del vector es *double*.

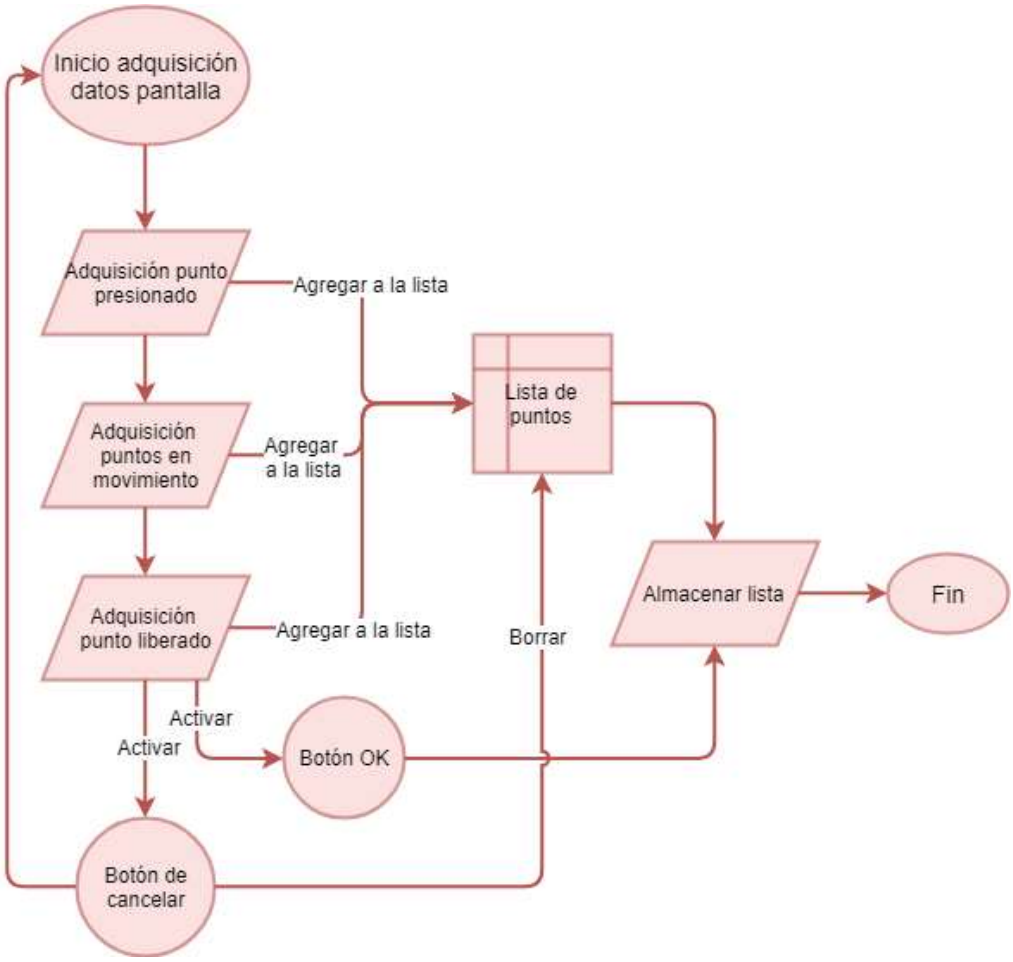

Figura 4.3.4 Algoritmo de definición ruta guiada, fuente propia.

El método agrega 3 manejadores de eventos, estos eventos se generan cuando se presiona, mueve o libera un punto en el lienzo. En el momento que el usuario empieza a dibujar la trayectoria, se acciona el evento de punto presionado, este representa el punto de inicio, y el manejador de este tipo de evento agrega la coordenada de este punto a la lista de puntos. El siguiente evento que se acciona es el de punto en movimiento pues el usuario ya está moviendo su dedo sobre el lienzo, el manejador de este tipo de evento agrega a la lista de puntos la coordenada de cada punto accionado, simultáneamente, se dibuja un ovalo centrado en la coordenada y con un diámetro de 8 pixeles, esto con el fin de proyectar la trayectoria mientras se está definiendo. Finalmente, se acciona el evento de punto liberado, cuando el usuario retiro su dedo de la pantalla, el manejador de este tipo de evento agrega la coordenada del punto final a la lista de puntos. Habiendo dibujado la trayectoria, el usuario tiene dos opciones que se muestran con dos botones como se observa en la figura 4.3.3, uno para confirmar y otro para eliminar o cancelar la trayectoria dibujada, en el caso que el usuario escoja el botón de confirmar: se remueven los manejadores de eventos del lienzo, de lo contrario van a estar "escuchando" y ejecutando sus tareas cada vez que alguno de los eventos sean accionados sobre el lienzo; se invoca el método *crearTrayectoriaGuiada* en el cual se guarda la trayectoria en el mapa de la trayectorias en la clase Sistema y en los atributos del actor ; si el actor es virtual la imagen de este se centra en el punto de inicio de la trayectoria; finalmente se configuran los botones para que no sean visibles.

La figurea 4.3.5 muestra un ejemplo de una trayectoria guiada en la interfaz gráfica.

Figura 4.3.5 Definición de trayectoria guiada

Trayectoria extrapolada

Para este caso, el usuario debe simplemente tocar la pantalla dos veces, primero en donde quiere que inicie la trayectoria del actor y luego en donde quiere que finalice, después del segundo toque se muestran los botones de confirmación y cancelación y se dibuja una línea recta iniciando en el primer punto y finalizando en el segundo como se puede ver en la figura 4.3.4.

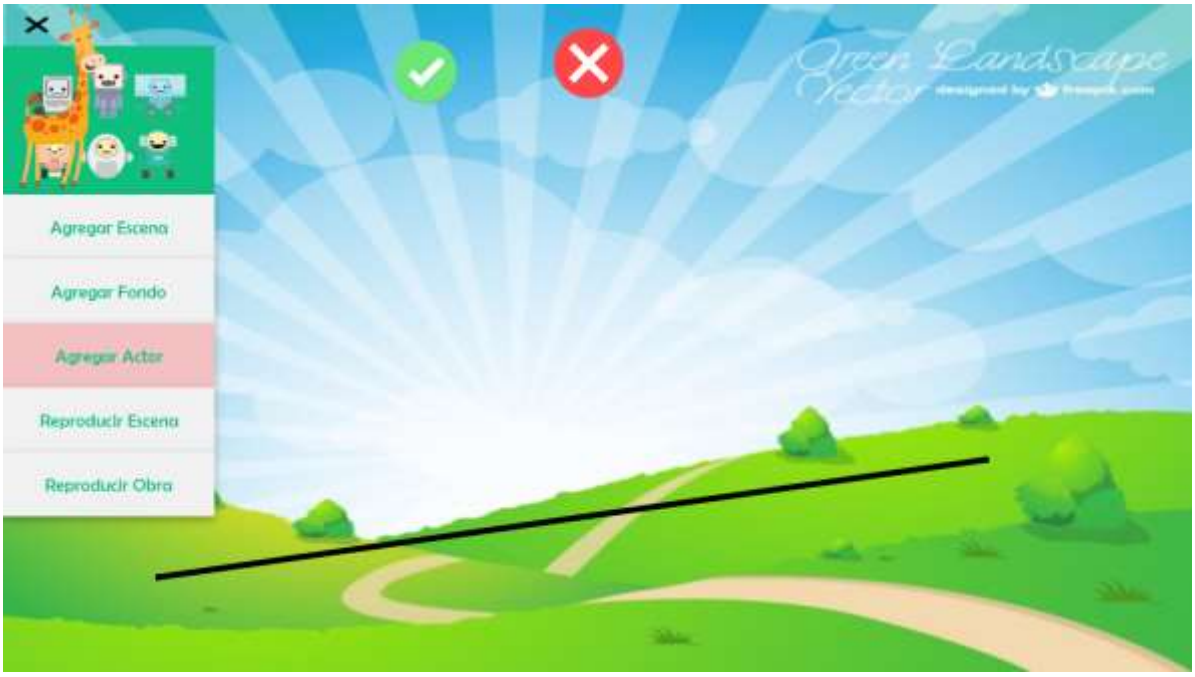

Figura 4.3.6 Definición trayectoria extrapolada

Lo anterior se implementó en el método *adquirirTrayectoriaExtrapolada*, el cual tiene una lista de dos vectores de tamaño 2 y de tipo *double* como en el anterior caso, también tiene un contador que empieza

desde 0 cada vez que el método es invocado. Este método agrega 2 manejadores de eventos que se generan cuando se presiona o libera un punto de contacto sobre el lienzo. Cuando se presiona un punto, el contador se incrementa en 1, y se guardan las coordenadas del punto que fue presionado, también se pinta un círculo negro centrado en la coordenada del punto. Cuando un punto deja de estar presionado, se valida si el contador es igual a 2, esto implica que ya se escogió el punto inicial y final, entonces si la condición anterior es verdadera, se dibuja una línea recta y se vuelven visibles los botones de confirmación y cancelación, si el usuario escoge el botón de cancelar se borra el dibujo y se pide que vuelva a realizar el procedimiento para definir la trayectoria, y si el usuario escoge el botón de confirmar: se remueven los manejadores de eventos del lienzo, se centra la imagen del actor en el punto inicial sólo si este es virtual, se invoca el método *crearTrayectoriaExtrapolada* en el cual se guarda la trayectoria en el mapa de la clase *Sistema* y en los atributos del actor, se reinicia el contador y los botones se configuran para que no sean visibles.

4.4. Implementación protocolos de comunicación.

Inicialmente se había planteado que la aplicación iba a tener sólo comunicación con la plataforma RoboAct, pues esta era la responsable de implementar el control de trayectorias de los robots, sin embargo, fue necesario realizar cambios teniendo en cuenta que este trabajo de grado hace parte de un proyecto más grande y que no fue posible que el control fuera realizado por RoboAct, debido a diferencia de tiempo en los desarrollos de RoboAct y el de este trabajo de grado. Por lo anterior se reestructuraron las funciones de los módulos y se hizo obligatorio diseñar un módulo de comunicación entre los robots Quemes y la aplicación.

4.4.1. Comunicación con robots Quemes.

Los robots Quemes tienen un módulo Bluetooth HC-05, con el cual se puede establecer una comunicación serial de manera sencilla, estos ya tienen programado en Arduino una función para recibir y enviar mensajes de tipo *int*. La programación del protocolo en java se basó en el ya existente entre quemes y su aplicación, sin embargo, se realizaron ajustes para que el robot fuera capaz de interpretar los comandos que reciba por mensaje. Es importante aclarar, que el robot debe primero establecer el enlace con el computador antes de poder realizar la conexión con la aplicación.

Se implementó el protocolo en la clase *ComunicacionQ* y teniendo en cuenta que la comunicación con el robot tiene que realizarse paralelamente a la ejecución de la aplicación, se debe implementar el protocolo por medio de un hilo, por esto, esta clase hereda de la superclase *Thread*. Cada robot tiene un nombre con el cual se identifica, este se muestra en el momento de realizar el enlace con el computador, el nombre debe ser indicado cuando se vaya a crear un objeto de esta clase, esto con el fin de que cada objeto creado sólo establezca una conexión con un robot, el nombre se almacena como un atributo. El método principal es el que ejecuta el hilo, en este se realiza una búsqueda de los dispositivos Bluetooth disponibles que guarda en una lista, luego busca el dispositivo que corresponde al nombre guardado para que por medio de la dirección Bluetooth del dispositivo se establezca la conexión, de esta manera, deja el puerto de entrada y salida abiertos para que puedan recibir y enviar mensajes. Los puertos de entrada y salida son de tipo *DataInputStream* y *DataOutputStream* estas clases modelan un flujo de bytes que entran o salen de un puerto de conexión, los datos se envían o reciben un byte a la vez.

Para enviar los mensajes a los robots se utiliza un método que escribe en el puerto de salida el byte que recibió por parámetro, en este caso los 8 bits que representen el número entero del mensaje. Se necesitan 4 mensajes diferentes, por lo tanto, se escogieron los números 10, 11, 12 y 13 ya que estos no están siendo utilizados por los robots, los mensajes representan los siguientes comandos que son implementados en Arduino:

- 10 avanzar hacia adelante
- 11 girar a la izquierda.
- 12 girar hacia la derecha.
- 13 detenerse.

4.4.2. Comunicación con RoboAct.

Como se mencionó anteriormente, no fue posible establecer la comunicación directamente con RoboAct, sin embargo, se dejó establecido el protocolo de comunicación, definiendo los mensajes que se intercambiarán, y el formato de información transmitida entre la aplicación y RoboAct. Además, teniendo en cuenta que la comunicación se realiza a través de un s*ocket*, el protocolo funcionará con cualquier programa que sea capaz de manejar la transferencia de datos por medio de *sockets*, lo cual RoboAct pude hacer.

Durante la reproducción de una escena se hace pertinente conocer por parte de RoboAct la información de los actores que están en esta, ya sean virtuales o reales. Por lo tanto, en el paquete comunicación se implementó la clase ComunicacionRA que hereda de la superclase *Thread* pues es necesario que se ejecute en un hilo propio. Esta clase se modeló como un servidor que constantemente está escuchando en el *socket* del servidor esperando que un cliente se conecte y le haga alguna solicitud, siendo RoboAct el cliente, la figura 4.41 ilustra el diagrama del algoritmo que se implementó para el protocolo.

La clase *socket* tiene dos métodos que retornan los objetos de tipo *DataInputStream* y *DataOutputStream* los cuales contienen el flujo de datos de entrada y salida del socket, así cuando el servidor acepta una solicitud de conexión entrante, puede leer línea a línea los mensajes que están entrando al socket, particularmente, RoboAct envía el mensaje “1”, para solicitar que se le envíe la información, entonces el servidor si recibe un uno en el buffer de entrada, envía los datos por medio del método writeUTF de la clase *DataOutputStream*, el cual escribe en la salida del socket datos de tipo *String* codificados en UTF-8. El mensaje enviado RoboAct contiene la información de los actores, se decidió utilizar para separar la información de cada actor el símbolo “#”, incluyéndolo en el inicio seguido del tipo de actor y en el final de la trama, el final de la trama también se puede indicar con un salto de línea, usando el símbolo “;” se separan los diferentes datos que pertenecen a un actor, la figura 4.4.2 ilustra un mensaje de este tipo.

El método que genera el mensaje de salida se implementó en la clase *Sistema*, y este recorre el mapa de actores extrayendo la información de estos mientras va generando el mensaje de tipo *String*, para los actores virtuales los datos que se envían son:

- Tipo de actor.
- Id.
- Nombre de la escena a la que pertenece
- Posición en x.
- Posición en y.

Y para los actores reales:

- Tipo de actor.
- Id.
- Nombre de la escena a la que pertenece
- Posición en x.
- Posición en y.
- Orientación en grados.

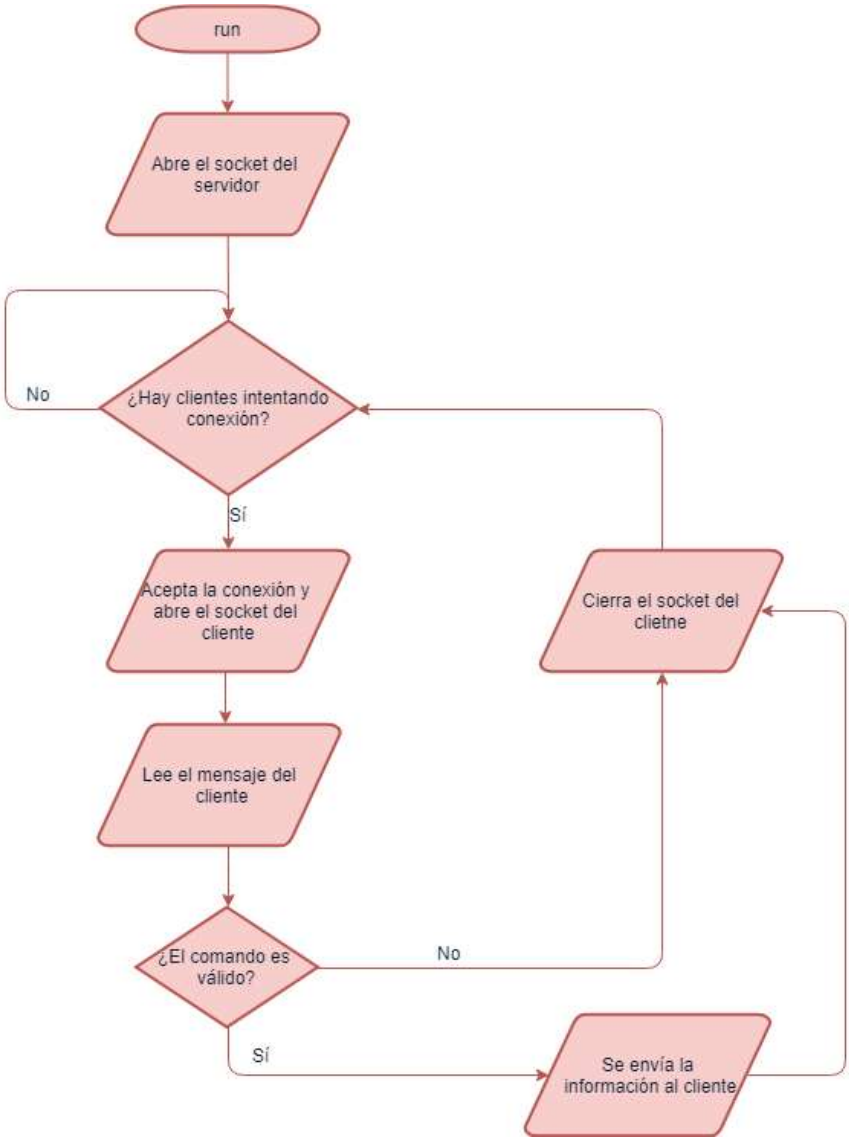

Figura 4.4.1 Algoritmo de comunicación con RoboAct, fuente propia.

4.5. Control de trayectorias para robots Quemes.

Como primer paso se realizaron pequeñas modificaciones al código en Arduino con el que ya vienen los Quemes, con el fin de que puedan interpretar los comandos enviados desde esta aplicación por medio del protocolo implementado en *ComunicacionQ*. Se agregaron funciones para que el robot se mueva hacia la izquierda, derecha y hacia adelante, modificando las velocidades de los motores. Además, se agrega la función para detener, en la cual se configura la velocidad de los motores en 0. En la función *loop* de Arduino, se declaró un *switch* que recibe como parámetro el mensaje y para cada caso, ya sea 10, 11, 12 o 13 invoca la función a la cual corresponde el mensaje, tal y como se definió en la sección 4.4.1. la función sigue ejecutándose hasta que llegue otro nuevo mensaje y sea diferente. El control se implementó en la clase *ControlRobot* del paquete control. Como el control del robot debe ejecutarse paralelamente, esta clase extiende de la superclase *Thread*, y en sus atributos contiene: el objeto de tipo *ComunicacionQ* que establece la comunicación Bluetooth con el robot, un objeto de tipo *AdquisicionDatos* y un objeto de tipo actor que es el actor asociado al cual se le realizará el control. Además, se utilizó la clase *Timer* y *TimerTask* haciendo uso de estas es posible ejecutar tareas sincronizadamente y separadas por un periodo de tiempo fijo, esto con el fin de actualizar la ubicación de cada robot y ejecutar la acción de control periódicamente.

Específicamente, se decidió actualizar la ubicación cada 10 ms y realizar la acción de control cada 50 ms. Se implementó un control para cada tipo de trayectoria, sin embargo, el principio de funcionamiento es el mismo, el cual se basa en saber hacia qué lado de la trayectoria está y dependiendo de esto se envía el mensaje pertinente al robot por Bluetooth. Cuando el usuario selecciona la escena que quiere reproducir, se instancian los objetos tipo *ControlRobot* de los robots y luego en el momento en el que el usuario presiona el botón de reproducir, se invoca el método *run* del control.

Trayectoria Extrapolada:

En la figura 4.5.1 se ilustra un diagrama de flujo del algoritmo implementado para realizar el control de trayectoria extrapolada. Cuando se crea un objeto nuevo de tipo *ControlRobot* y si la trayectoria del robot es extrapolada, se almacenan el punto inicial y final y se guardan como atributos. En el método *run* se realiza la implementación de las 2 tareas que se ejecutan de manera periódica y síncrona, actualizar ubicación y control, en la primera se invoca un método de la clase *Sistema* que haciendo uso del reconocimiento de las firmas digitales actualiza las ubicaciones de todos los robots ubicados sobre la pantalla, esta acción se ejecuta en primer lugar con un periodo de 10 ms, aunque realizando pruebas de funcionamiento se logró llegar a los 6 ms. La tarea control implementa el algoritmo de la figura 4.5.1 desde el punto en el que se halla el lado en el que está el robot, invocando una función de la clase *Operaciones* que ya fue explicada en el capítulo 4.2.3, los vectores que se le ingresan a esta corresponden al puno inicial y final, conociendo el lado se invocan las funciones para enviar el mensaje al robot por Bluetooth.

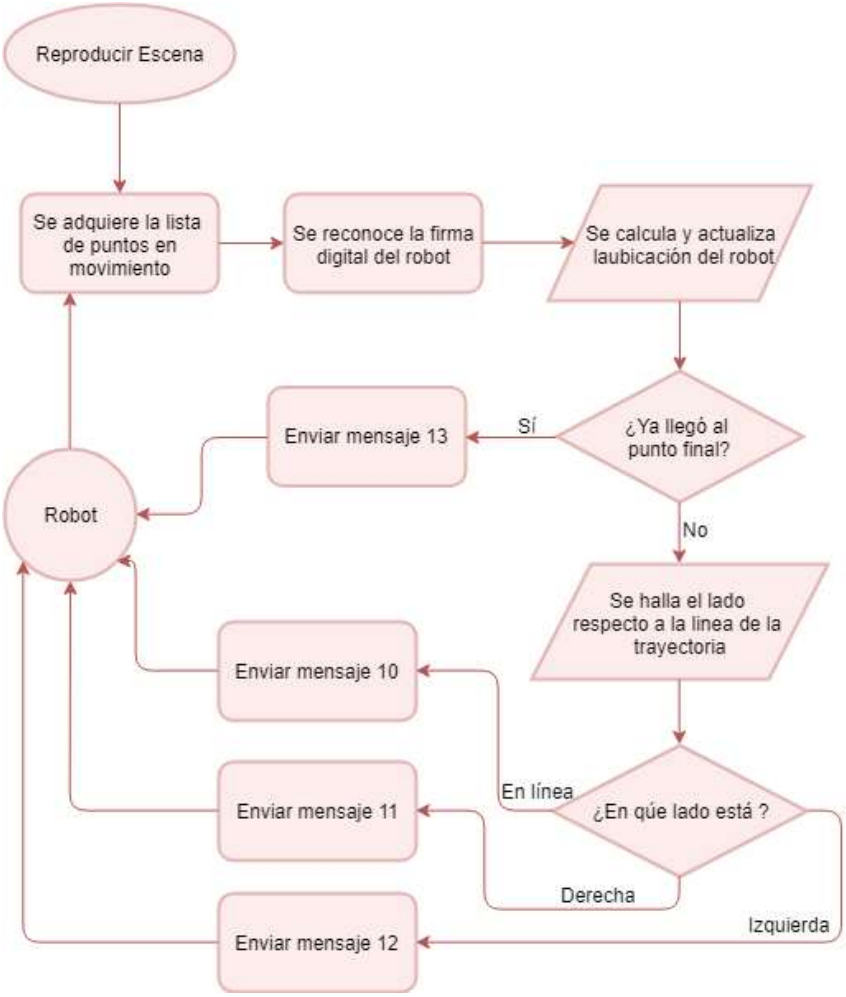

Figura 4.5.1 Algoritmo control trayectoria extrapolada

Además, se agrega una condición adicional para enviar el mensaje 10 que corresponde a avanzar hacia delante y es que si el robot está a menos de 30 pixeles de distancia del punto de inicio se debe enviar este mensaje.

Trayectoria guiada:

Para este tipo de trayectoria el control se hace más complicado si se quiere implementar una solución como para el caso anterior, no se encontró una forma matemática de conocer el lado de un punto respecto a una curva, por lo tanto, se decidió realizar una segmentación de la curva y aproximarla por medio de líneas rectas como se observa en la figura 4.5.2, para realizar este procedimiento se deja un tamaño de segmento fijo, el cual fue 50 puntos, este valor se fijó con el fin de tener entre 5 y 9 líneas rectas que aproximen a la trayectoria. Para escoger este valor se realizó un promedio de longitud de las trayectorias en puntos experimentalmente. Dividiendo el tamaño de la lista de puntos que conforman la trayectoria sobre el tamaño del segmento se tiene la cantidad de líneas necesarias para realizar la aproximación.

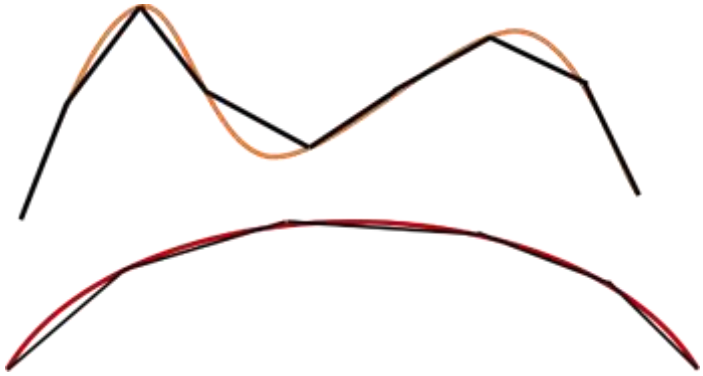

Figura 4.5.2 Aproximación de curvas por segmentos de línea recta

Como el objetivo es poder utilizar el mismo algoritmo de la trayectoria extrapolada, este se debe ejecutar secuencialmente para cada segmento de línea, para esto se genera una lista de puntos con el mismo tamaño que la cantidad de segmentos, en la cual se almacena el punto inicial seguido del punto final de la recta, este proceso se repite hasta llegar al final de la trayectoria, es pertinente aclarar que si la longitud de la trayectoria no es divisible entre 50, la última recta será del mismo tamaño que el residuo de la división, esto con el fin de no obtener excepciones de tipo *NullPointerException.*

Figura 4.5.3 Proyección en la aplicación de la segmentación de curvas

Teniendo la lista de puntos iniciales y finales, se implementa el mismo algoritmo de control que, en la trayectoria extrapolada, con la diferencia de que los puntos iniciales de las rectas cambian conforme el robot se llega a los puntos finales. Más concretamente, al usuario oprimir el botón de reproducción, se halla el lado en el que el robot está respecto a la primera línea recta y se aplica el control respecto a esta, cuando este llega al punto final de la primera e inicial de la segunda, se actualizan los puntos iniciales y finales, y se halla el lado en el que está respecto a la segunda línea recta y se aplica el control respecto a esta y así sucesivamente hasta llegar al final de la trayectoria. Esta forma de realizar el control tiene una limitación, y es que, para garantizar el buen seguimiento de la trayectoria, los ángulos entre las líneas rectas consecutivas no pueden ser menores de 120°.

5. Validación de especificaciones y protocolo de pruebas

Teniendo en cuenta que se desarrollaron varios sistemas para lograr cumplir con los objetivos del trabajo de grado, se diseñaron 3 pruebas con las cuales se pude validar el correcto funcionamiento de cada sistema fundamental de la aplicación. Y una prueba de funcionamiento general, la cual fue ejecutada por usuarios. Para las pruebas de desempeño se programaron scripts que permitieran automatizar las pruebas con el fin de tomar varios datos en un tiempo reducido.

5.1. Pruebas de reconocimiento y ubicación de objetos.

Con el fin de establecer validar el funcionamiento del sistema que se diseñó e implemento para procesar puntos de contacto, identificar las firmas digitales y reconocer a cuál robot pertenece, se programó un script en el cual se ejecutaron diferentes pruebas y los resultados fueron guardados como archivos separados por comas, que son fácilmente procesados por Excel. Los resultados de las diferentes pruebas se muestran a continuación

Prueba con pieza estática periodo 10ms:

El objetivo de esta prueba es comprobar el funcionamiento del algoritmo para reconocer las firmas digitales, como para el reconocimiento de una firma se estableció un intervalo de valores validos de área que

representan una firma digital, en esta prueba se valida para los intervalos generados por un ±5%, 7% y 8% de error, por ejemplo, si el área es 100 los intervalos de valores válidos serán: [95,105], [93, 107], [92, 108]. Además, dado que el tiempo de procesamiento es un factor importante para el control de trayectorias de robots y la comunicación con RoboAct, se mide el tiempo de procesamiento de los datos del algoritmo.

La prueba consiste en ubicar la pieza con una firma digital fija, antes de comenzarla, se registra la firma y se calcula el área, esta corresponde al dato de área almacenada en la tabla 5.1, posteriormente se ejecuta el script, invocando los métodos necesarios para reconocer la firma, este se ejecuta durante 5 segundos, invocando el método cada 10 ms para un total de 500 datos, la pieza es retirada y colocada de nuevo durante este proceso, con el fin de observar la variación del área obtenida. En la tabla 5.1 el área almacenada representa el valor calculado cuando se registra la firma digital al iniciar la prueba, el área obtenida es el área que está siendo calculada cada 10 ms que es llamado el método de reconocimiento, el tiempo de procesamiento es el que tarda el algoritmo en entregar el resultado es medido en segundos, las 3 ultimas columnas representan si con el intervalo de valores válidos que se utilizó en el algoritmo la firma es reconocida. Para el análisis de resultados se calcula, desviación estándar, promedio, valor máximo y mínimo. También se halla el número de veces que se reconoció la firma y el porcentaje de acierto para cada intervalo.

Área Almacenada (pixeles²)	Área Obtenida (pixeles²)	Error Porcentual (%)	Tiempo de procesamiento (s)	Identifica (±5%)	Identifica (±7%)	Identifica (±8%)
980,0	1020,0	-4,08	6,20E-05	Si	Si	Si
980,0	1020,0	-4,08	1,02E-04	Si	Si	Si
980,0	1020,0	-4,08	6,86E-05	Si	Si	Si
980,0	1041,0	-6,22	6,90E-05	No	Si	Si
980,0	1041,0	-6,22	9,42E-05	No	Si	Si
980,0	1041,0	-6,22	9,98E-05	No	Si	Si
980,0	1041,0	-6,22	5,79E-05	No	Si	Si
980,0	1041,0	-6,22	1,26E-03	No	Si	Si
980,0	1041,0	-6,22	6,11E-05	No	Si	Si
980,0	1041,0	-6,22	6,48E-05	No	Si	Si
980,0	1064,5	-8,62	6,25E-05	No	No	Si
980,0	1064,5	-8,62	6,44E-05	No	No	Si
980,0	1064,5	-8,62	6,16E-05	No	No	Si
980,0	1064,5	-8,62	1,83E-04	No	No	Si
980,0	1064,5	-8,62	6,02E-05	No	No	Si
980,0	1064,5	-8,62	7,28E-05	No	No	Si
980,0	1051,0	-7,24	6,67E-05	No	Si	Si
980,0	1027,5	-4,85	8,96E-05	Si	Si	Si
980,0	1027,5	-4,85	7,42E-05	Si	Si	Si
980,0	1027,5	-4,85	6,34E-05	Si	Si	Si
980,0	1027,5	-4,85	8,26E-05	Si	Si	Si
980,0	1006,5	-2,70	7,00E-05	Si	Si	Si
980,0	1020,0	-4,08	6,07E-05	Si	Si	Si
980,0	1020,0	-4,08	5,93E-05	Si	Si	Si

980,0	1020,0	-4,08	6,86E-05	Si	Si	Si
980,0	1020,0	-4,08	6,67E-05	Si	Si	Si
Desviación Estándar	17.59666415	1.795577974	100.4E-6	-	-	-
Promedio	1033.966615	5.537142857	97.5E-6	-	-	-
Valor máximo	1064.5	2.704081633	1.3E-3	-	-	-
Valor mínimo	1006.5	-8.62244898	54.1E-6	-	-	-
Número de firmas Identificadas	-	-	-	281	416	500
Porcentaje de acierto	-	-	-	56,2%	83,2%	100%

Tabla 5.1Resultados prueba con pieza estática

Prueba con 3 firmas reconociendo una por solicitud:

Estas pruebas consistieron en registrar todas las firmas digitales, haciendo uso de las 3 piezas que hay disponibles y cambiando los tornillos de hueco. En la pantalla se ubican las 3 piezas y después de esto se ejecuta el script indicándole que la firma que se debe reconocer es la que tiene id 1, en esta prueba se generaron 4000 datos y el periodo fue de 10 ms. Con esta prueba se puede validar que efectivamente se reconoce una firma específica, y que no se confunda con otra.

Área almacenada (pixeles²)	Área obtenida (pixeles²)	Error Porcentual (%)	Tiempo
1008.00	982.50	2.530	6.1116E-05
1008.00	982.50	2.530	5.0852E-05
1008.00	982.50	2.530	4.3854E-05
1008.00	982.50	2.530	6.2515E-05
1008.00	982.50	2.530	5.9250E-05
1008.00	982.50	2.530	5.7851E-05
1008.00	982.50	2.530	5.1786E-05
1008.00	982.50	2.530	6.7647E-05
1008.00	982.50	2.530	6.7181E-05
1008.00	982.50	2.530	5.1319E-05
1008.00	982.50	2.530	6.4848E-05
1008.00	982.50	2.530	5.5984E-05
1008.00	982.50	2.530	6.7181E-05
1008.00	982.50	2.530	5.1786E-05
1008.00	982.50	2.530	7.2313E-05
1008.00	982.50	2.530	4.7120E-05
1008.00	982.50	2.530	3.9189E-05
1008.00	982.50	2.530	5.2718E-05
1008.00	982.50	2.530	5.5051E-05
1008.00	982.50	2.530	5.1319E-05
1008.00	982.50	2.530	4.1055E-05
1008.00	982.50	2.530	5.7850E-05

1008.00	982.50	2.530	6.1116E-05
1008.00	982.50	2.530	5.5985E-05
1008.00	982.50	2.530	1.5116E-04
Desviación Estándar	18.01	1.787	9.0689E-05
Media	997.35	1.02	7.2988E-05
valor máximo	1044.00	4.415	1.6534E-03
valor mínimo	963.50	-3.571	3.5456E-05

Tabla 5.2 Resultado Pruebas con 3 firmas

Prueba con robot en movimiento.

En esta prueba se coloca la pieza debajo del robot, por ahora se ajusta con cinta adhesiva, dado que los robots para los cuales se diseñó la pieza aún no han salido a producción. Antes de ejecutar el script se registra la firma digital de la pieza y se calcula el área. Posteriormente se ejecuta el script y se pone el robot a desplazarse sobre la pantalla, en esta prueba se generaron 4000 datos, y el método es invocado cada 10ms, la tabla 5.3 muestra los últimos datos registrados con los mismos cálculos estadísticos de la prueba anterior, para ver la totalidad de los datos referirse al anexo 3.

Área almacenada (pixeles²)	Área obtenida (pixeles²)	Error Porcentual (%)	Tiempo proceso (s)	Error porcentual absoluto	Ux	Uy
1937	1903	1,76	3,97E-01	1,76E+04	638	1173
1937	1950	-0,65	3,92E-01	6,45E+03	639	1183
1937	1895	2,17	4,25E-01	2,17E+04	640	1213
1937	1914	1,19	6,16E-01	1,19E+04	640	123
1937	1914	1,19	4,20E-01	1,19E+04	641	125
1937	1926	0,57	4,29E-01	5,68E+03	641	126
1937	1914	1,19	6,25E-01	1,19E+04	641	127
1937	1914	1,19	4,01E-01	1,19E+04	642	129
1937	1926	0,57	4,48E-01	5,68E+03	642	130
1937	1952	-0,77	7,93E-01	7,74E+03	642	131
1937	1919	0,96	8,72E-01	9,55E+03	643	134
1937	1872	3,36	4,20E-01	3,36E+04	643	1373
1937	1899	1,96	3,78E-01	1,96E+04	644	1387
1937	1919	0,96	4,01E-01	9,55E+03	644	140
1937	1853	4,34	5,41E-01	4,34E+04	645	1427
1937	1892	2,35	6,16E-01	2,35E+04	645	1447
1937	1906	1,6	4,01E-01	1,60E+04	646	146
1937	1964	-1,39	5,69E-01	1,39E+04	646	1473
1937	1899	1,96	4,20E-01	1,96E+04	647	1497
1937	1911	1,34	4,53E-01	1,34E+04	647	151
1937	1964	-1,39	3,92E-01	1,39E+04	648	1533
1937	1885	2,68	1,08E+00	2,68E+04	648	1543
1937	1929	0,41	4,39E-01	4,13E+03	650	1597

1937	1937	0	3,64E-01	0,00E+00	650	1597
Desviación Estándar	40.49	2.09	1.3266E-04	1.32	0.00	40.49
Media	1941.61	-0.28	6.4096E-05	1.65	1937.00	1941.61
Valor Máximo	2092.00	10.48	7.0199E-03	10.48	1937.00	2092.00
Valor Mínimo	1734.00	-8.00	3.0324E-05	0.00	1937.00	1734.00

Tabla 5.3 Resultados de pruebas con robot en movimiento

5.2. Prueba de cálculo de orientación.

Angulo Esperado (°)	Angulo Obtenido (°)	Error Porcentual (%)	Error Absoluto
0,00	0,00	0	0
10,00	11,04	-10,4094018	-1,04
20,00	20,56	-2,780226098	-0,56
30,00	31,43	-4,765218716	-1,43
40,00	40,16	-0,389999062	-0,16
50,00	50,91	-1,812282228	-0,91
60,00	59,74	0,427395273	0,26
70,00	70,71	-1,014219687	-0,71
80,00	76,29	4,633701255	3,71
90,00	91,33	-1,480244282	-1,33
100,00	103,39	-3,392497754	-3,39
110,00	116,57	-5,968228343	-6,57
120,00	119,58	0,351801099	0,42
130,00	135,94	-4,56860842	-5,94
140,00	139,69	0,224357257	0,31
150,00	148,57	0,953043743	1,43
160,00	159,44	0,347528262	0,56
170,00	164,98	2,951976842	5,02
180,00	178,60	0,776211682	1,40
190,00	188,13	0,984156656	1,87
200,00	193,71	3,146519498	6,29
210,00	203,55	3,070350632	6,45
220,00	217,15	1,296960592	2,85
230,00	228,81	0,515619637	1,19
240,00	234,78	2,17399707	5,22
250,00	248,46	0,616390367	1,54
260,00	251,57	3,244211086	8,43
270,00	268,64	0,505158345	1,36
280,00	276,79	1,146794795	3,21
290,00	295,35	-1,843508945	-5,35
300,00	300,26	-0,085479055	-0,26
310,00	312,09	-0,673923282	-2,09
320,00	320,71	-0,222060355	-0,71

330,00	329,74	0,077708231	0,26
340,00	335,22	1,404453108	4,78
350,00	346,91	0,883398018	3,09
Media		-0,268726794	0,811463649
Máximo		4,633701255	8,434948823
Mínimo		-10,4094018	-6,565051177

Tabla 5.4 Resultados pruebas orientación

En esta prueba se generó un script para automatizar las pruebas del sistema que calcula la orientación, la prueba consiste en proyectar la imagen de la figura 5.2.1 la cual tiene líneas separadas por un ángulo de 10°, luego se ubica la pieza sobre la pantalla alineándola con las líneas. Esta prueba genera 36 datos, y para registrar cada uno de estos se presiona un botón de confirmación cuando ya esté alineada la pieza. Los resultados de estas pruebas se muestran en la tabla 5.4.

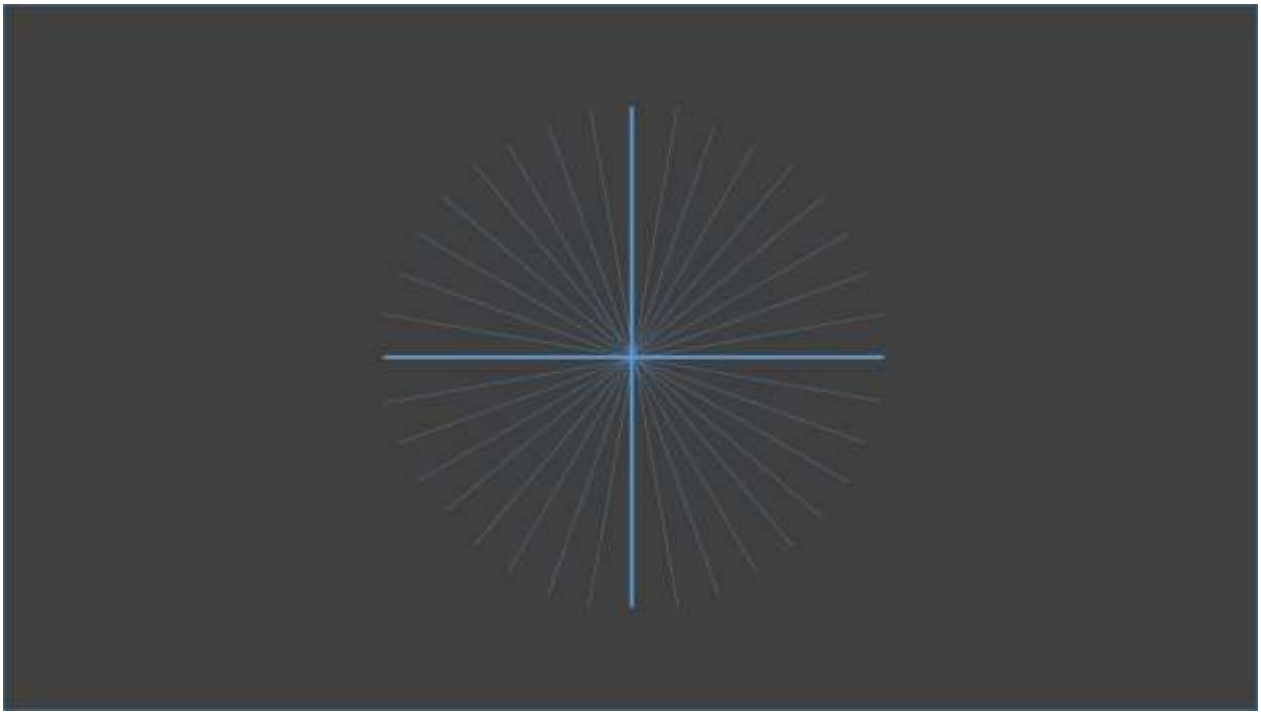

Figura 5.2.1 Imagen para alinear la pieza con las líneas separadas

5.3. Validación de protocolo de comunicación.

En esta prueba se pretende simular el comportamiento que tendría RoboAct en la comunicación con la aplicación, para lograrlo se programa un script en el cual se implementa un cliente por medio de sockets, se crea el socket y se conecta al mismo puerto y a la misma dirección que el protocolo de comunicación implementado en *ComunicacionRA* que es el servidor, se escogió hacer la conexión por el puerto 9000 y como dirección *localhost* pues se están ejecutando en el mismo computador.

El cliente hace la solicitud de conexión y envía un 1 para que se le envíe la información de los actores en tiempo real, esta acción se ejecutó por medio de un objeto de tipo *Timer* con el fin de que se haga la solicitud de información cada 10 ms, hasta recolectar 1000 datos, los cuales se almacenan en un archivo de tipo csv. La tabla 5.5 ilustra los datos recibidos por el cliente, del robot mientras se está desplazando sobre la pantalla, se puede observar que efectivamente la información es enviada correctamente.

Tipo de Actor	Id del Actor	Ubicación x	Ubicación y
Robot	1	964,00	554,00
Robot	1	964,00	554,00
Robot	1	964,00	554,00
Robot	1	964,00	554,00
Robot	1	964,00	554,00
Robot	1	964,00	554,00
Robot	1	964,00	554,00
Robot	1	964,00	554,00
Robot	1	964,00	554,00
Robot	1	964,00	554,00
Robot	1	964,00	554,00
Robot	1	964,00	554,00
Robot	1	964,00	554,00
Robot	1	964,00	554,00
Robot	1	964,00	554,00
Robot	1	963,00	552,67
Robot	1	963,00	552,67
Robot	1	961,00	551,00
Robot	1	961,00	551,00
Robot	1	959,33	549,67
Robot	1	959,33	549,67
Robot	1	959,33	549,67
Robot	1	957,00	547,67
Robot	1	957,00	547,67
Robot	1	957,00	547,67
Robot	1	953,33	545,00
Robot	1	953,33	545,00
Robot	1	953,33	545,00
Robot	1	949,67	542,67
Robot	1	949,67	542,67
Robot	1	949,67	542,67
Robot	1	949,67	542,67

Tabla 5.5 Resultados prueba de comunicación

5.4. Validación de la aplicación por parte de usuarios.

Para validar la aplicación ya integrada, se programaron una serie de pruebas con niños en un intervalo de edad entre 9 a 14 años. El objetivo de la prueba es validar el funcionamiento de toda la aplicación y ver qué tan fácil resulta la interacción con el usuario final. La prueba consiste en lo siguiente, primero hay una explicación de lo que puede hacer y lo que no la aplicación, se hace la aclaración de la limitante para definir la trayectoria guiada para robots que se mencionó en el capítulo 4.5, se les habló con un lenguaje no técnico

para que no se fueran a confundir. Habiendo ya explicado cómo funciona la aplicación, se le solicita a cada niño que cree 2 escenas diferentes, con la variedad de actores y robots que ellos quieran y que las reprodujeran colocando los robots en sus posiciones iniciales antes de oprimir el botón de reproducir el objetivo fue que interactuaran libremente con la aplicación por medio de la interfaz gráfica. Videos y fotos de estas pruebas se encuentran en el anexo 5.

Al terminar la prueba se les pide llenar una encuesta basada en el cuestionario Escala de Usabilidad del sistema (SUS), esta genera un único valor que representa el grado de conformidad de usabilidad global del sistema, se obtiene un valor entre 0 y 100. Se debe calificar cada afirmación en el intervalo de 1 a 5, sien 1 estar en total desacuerdo y 5 en total acuerdo, las preguntas son las siguientes

- ¿Creo que usaría esta aplicación frecuentemente?
- ¿Encuentro esta aplicación innecesariamente compleja?
- ¿Creo que la aplicación fue fácil de usar?
- ¿Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar esta aplicación?
- ¿Las funciones de esta aplicación están bien integradas?
- ¿Creo que la aplicación es muy inconsistente?
- ¿Imagino que la mayoría de la gente aprendería a usar esta aplicación en forma muy rápida?
- ¿Encuentro que la aplicación es muy difícil de usar’
- ¿Me siento confiado al usar esta aplicación?
- ¿Necesité aprender muchas cosas antes de ser capaz de usar la aplicación?

La puntuación SUS se calcula de la siguiente forma:

- Para las preguntas impares, se le resta 1 al puntaje obtenido, generando un rango de 0 a 4.
- Para las preguntas pares, se resta 5 menos el puntaje obtenido, generando un rango de –4 a 0.
- Sumar los nuevos valores obtenidos y multiplicarlo el resultado por 2.5.

Las pruebas escaneadas se encuentran en el anexo 6.

6. Análisis de resultados

Se logró cumplir con el objetivo general, la aplicación permite la configuración de los elementos necesarios para crear obras de teatro robótico haciendo uso de la pantalla multitáctil 3M C556PW, además se logró integrar con los robots Quemes, e implementar un protocolo de comunicación por medio de sockets con el fin de integrar RoboAct con el sistema, no fue posible realizar pruebas con RoboAct por cuestiones de diferencias de tiempo en los desarrollos, sin embargo, ya están definidos los parámetros de comunicación entre el sistema y RoboAct. A continuación, se detalla el análisis de los resultados obtenidos por cada desarrollo que se realizó en este trabajo de grado y de los resultados obtenidos en las pruebas realizadas.

Interfaz capacitiva:

El diseño de la interfaz capacitiva permitió que efectivamente se reconocieran los robots cuando estos están ubicados en la pantalla. Se presentaron problemas a la hora de instalarla en los robots Quemes disponibles en este momento, ya que se diseñaron para la versión nueva de estos. Esto genera ocasionalmente que se pierdan los puntos de contacto cuando el robot está en movimiento, debido a que no se logra un buen empalme entre la cabeza del tornillo y la pantalla.

Pruebas con pieza estática periodo 10ms y prueba con 3 firmas reconociendo una por solicitud:

En la tabla 5.1 se muestran datos que proporcionan información valiosa a la hora de analizar el desempeño del diseño de las firmas digitales, el algoritmo para discriminar firmas digitales a partir de los puntos de contacto y del método de identificación de la firma. En la tabla se calculó el error porcentual de cada medición tomando como el dato real el área almacenada, en el caso específico de esta prueba se evidencia que todos los valores de este son negativos, lo cual indica que la variación del área leída sólo fue en valores menores, esto no ocurre necesariamente así en todos los casos, también puede tener lecturas mayores al área almacenada, sin embargo, si se tienen en cuenta los valores extremos se puede deducir un rango en el cual el área está variando, por ejemplo, el valor mínimo del error porcentual es -8.62%, este da un indicio de un posible valor máximo del error porcentual.

La prueba con las 3 piezas simultáneamente puestas en la pantalla y escogiendo cuál de estas 3 es la que se decide reconocer y hallar su ubicación demuestra que efectivamente el sistema discrimina adecuadamente las firmas digitales, en la tabla se muestran los valores de error porcentual menores o mayores, y además se disminuye los valores extremos de estos -3.57% y 4.41%.

Prueba con la pieza en movimiento

En esta prueba se evidencia un aumento del error porcentual, los valores extremos que se obtienen son 10.48% y -8%, es un intervalo grande, y seguramente es debido al movimiento, ya que no se logra un buen contacto entre la interfaz y la pantalla en todos los casos, pero es pertinente resaltar que el promedio de este error fue tan solo de -0.28% para 4000 datos, un valor bastante bajo que indica que en la mayoría de los casos se reconoce efectivamente la firma digital.

Además, en la tabla 5.3 se incluye el tiempo que se tarda en realizar todo el procesamiento y es evidente que el rango entre el valor máximo y el mínimo es de dos órdenes de magnitud, sin embargo, el promedio de los 4000 datos es 79.9 microsegundos, un valor que es bastante aceptable teniendo en cuenta los requerimientos planteados.

Prueba de orientación:

El promedio del error absoluto y porcentual de los resultados obtenidos en esta prueba son bajos, y el error de medición se debe en una gran parte al tipio de prueba que se realizó, pues el ángulo se cuadró a simple vista, con la guía de la imagen, esto da pie que se cometan bastantes errores en la acomodación de la pieza.

Prueba comunicación con RoboAct:

En esta prueba se puede observar que efectivamente el cliente está recibiendo la información de los actores en tiempo real, cuando este la solicita, como en este caso, cada 10 ms se solicitaron los datos, y estos fueron recibidos adecuadamente.

Prueba de usabilidad del sistema.

En la tabla 6.1 se muestran los resultados obtenidos en las encuestas que se hicieron después de hacer las pruebas de interacción del usuario con la aplicación, este es una prueba parametrizada y tiene una escala para interpretar el resultado.

Usuario	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	Puntaje
Usuario 1	4	1	5	1	5	1	5	1	5	1	**97.5**
Usuario 2	4	3	5	1	5	3	5	1	5	2	**85.0**
Usuario 3	5	2	5	3	5	5	3	3	3	3	**62.5**
Usuario 4	4	2	5	2	2	2	4	2	5	1	**77.5**
Usuario 5	4	3	5	5	4	3	3	4	4	5	**50.0**
Usuario 6	4	3	5	3	4	3	5	2	4	4	**67.5**
										Prom	**73.3**

Tabla 6.1 Resultados usabilidad del sistema

Para interpretar el resultado se utilizó la escala propuesta por [19] y que se muestra en la figura 5.4.1. Teniendo en cuenta el promedio del puntaje obtenido, el sistema se encuentra en el percentil que representa una buena práctica de desarrollo de la interfaz de usuario, esto lo que infiere es que la aplicación tiene un grado de aceptación bueno por parte del usuario.

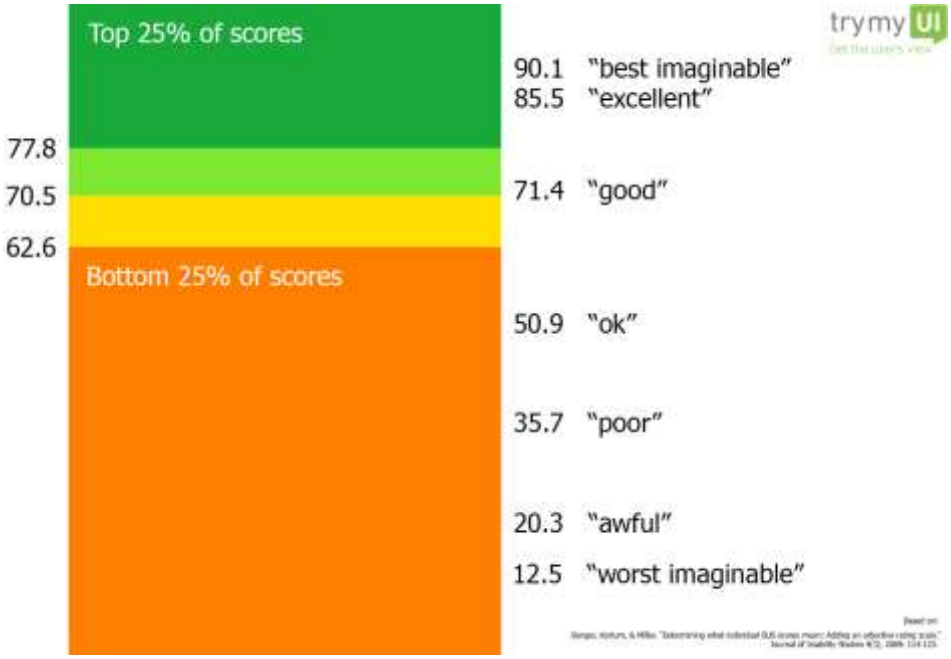

Figura 5.4.1Escala de calificación casos de usabilidad, fuente[20]

7. Conclusiones y Recomendaciones

Las interfaces físicas funcionan y cumplen su objetivo de permitir accionar los puntos de contacto de la pantalla, sin embargo, se puede lograr un mejor desempeño si se asegura que la superficie de la cabeza de los tornillos haga un buen contacto con la pantalla. Se logró identificar objetos por medio de las firmas digitales, pero no fue posible generar las 60 firmas, por limitaciones de espacio en el diseño mecánico de los robots y del reconocimiento de la pantalla, se recomienda para disminuir el error y evitar que una firma se confunda con otra incrementar en al menos 1mm la distancia entre los puntos que son opcionales, punto P3. Además como trabajo futuro, se puede investigar más respecto a los materiales a usar para activar los puntos de contacto, en este trabajo de grado se utilizaron tornillos de aluminio, sin embargo, existen materiales que pueden generar un mejor desempeño tanto en la generación del punto de contacto como para permitir un buen desplazamiento del robot y además protegiendo la pantalla de posibles rayones, uno de estos materiales es el filamento de grafeno conductivo que se utiliza en impresoras 3d, también permitiendo reducir la complejidad del ensamblado del prototipo.

Como trabajo futuro respecto a RoboAct, se debe utilizar el protocolo de comunicación diseñado para establecer la transferencia de información, permitiendo así que se gestione la ejecución de guiones y el sistema multiagentes, con el fin de que exista cooperación y comunicación entre los robots. Además, se sugiere implementar el control de trayectoria en RoboAct, para que este maneje toda la lógica concerniente a los robots.

La aplicación implementada permite calcular la posición y orientación de los objetos ubicados sobre la pantalla con una latencia mucho menor que 114.8 ms, como se pudo observar en los resultados obtenidos en las pruebas de desempeño, además permite al usuario crear escenarios cargando imágenes preestablecidas para el fondo, agregando actores virtuales y reales a los cuales se les pueden definir dos tipos de trayectoria todo esto para la definición del escenario y de la obra a reproducir. Como trabajo futuro se propone investigar en el reconocimiento de las firmas digitales, el cual se realizó comparando las áreas, pues este también limitó la distancia de separación de los huecos diseñados en la interfaz capacitiva. Además, si se logra deshabilitar la restricción de software que tiene la pantalla respecto a la distancia mínima para reconocer dos puntos, se podría logar generar un mayor número de firmas digitales que el actual.

La interfaz gráfica cumple con el objetivo de permitir interactuar al usuario con la aplicación, por medio de esta el usuario puede realizar todas las interacciones que se muestran en el párrafo anterior, como recomendación se puede volver un poco más sencilla, sobre todo en el momento de agregar un robot al escenario disminuyendo los pasos que se den realizar, así como también reorganizar la ubicación de los botones y diálogos teniendo en cuenta la altura promedio de los niños de edades entre 9 y 14 años, ya que la pantalla está puesta en una mesa de a aproximadamente 70cm. También, se recomienda ampliar la biblioteca de imágenes con el fin de tener mayor variedad para el usuario.

8. Bibliografía

[1] D. Ávila, M. Bermeo, and F. Merchán, “Plataforma de Dramatización Robótica Modular,” Pontificia Universidad Javeriana, 2016.

[2] A. de la Peña, “PI121-04-MCCoop-TeatroRob,” Pontifficia Universidad Javeriana, 2014.

[3] E. González, A. Pérez, C. Rodríquez, M. Manrique, Y. Arévalo, and C. Otálora, *Robótica Cooperativa: exepriencias de sistemas multiagente (SMA)*, 1st ed. Bogotá: Editorial Javeirana, 2012.

[4] F. A. Bravo, A. M. González, and E. González, “Interactive Drama With Robots for Teaching Non - Technical Subjects,” vol. 6, no. 2, pp. 48–69, 2017.

[5] S. Lemaignan, M. Gharbi, J. Mainprice, M. Herrb, and R. Alami, “Roboscopie: A theatre performance for a human and a robot [video abstract],” *Human-Robot Interact. (HRI), 2012 7th ACM/IEEE Int. Conf.*, p. 427, 2012.

[6] C. Breazeal *et al.*, “Interactive robot theatre,” *Proc. 2003 IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS 2003) (Cat. No.03CH37453)*, vol. 4, no. October, pp. 3648–3655, 2003.

[7] E. Gonzalez, C. Bustacara, and J. Avila, “Agents for Concurrent Programming,” 2003.

[8] G. B. Guio, “PA1710-7-RACUQuemes Rediseño arquitectónico de la capa de presentación de la plataforma Quemes para incorporar atributos de calidad que mejoren la usabilidad,” 2017.

[9] E. González Guerrero, J. Páez Rodríguez, and F. J. Roldan, “Uso de robots cooperativos para el desarrollo de habilidades de trabajo cooperativo en niños,” *Rev. Investig. UNAD*, vol. 12, no. 2, pp. 43–56, 2013.

[10] E. G. Guerrero, J. J. P. Rodríguez, and F. J. Roldán, “Robots Cooperativos, Quemes Para La Educación,” *Rev. Vínculos*, vol. 10, no. 2, pp. 47–62, 2014.

[11] Y.-H. Cheng, Q.-H. Meng, Y.-J. Liu, M. Zeng, L. Xue, and S.-G. Ma, “Fusing sound and dead reckoning for multi-robot cooperative localization,” no. 2, pp. 1474–1478, 2016.

[12] L. Ruiz and Z. Wang, “Real Time Multi Robot 3D Localization System Using Trilateration,” no. 0, pp. 1510–1515, 2016.

[13] M. Todescato, A. Carron, R. Carli, A. Franchi, and L. Schenato, “Multi-Robot Localization via GPS and Relative Measurements in the Presence of Asynchronous and Lossy Communication,” pp. 2527–2532, 2016.

[14] A. Carabantes, “Diferencia entre pantallas capacitivas y resistivas,” 2009. [Online]. Available: https://www.androidsis.com/wp-content/uploads/pantalla-capacitiva.jpg. [Accessed: 10-May-2018].

[15] C. T. Wu, *An Introduction to Object-Oriented Programming with Java*, 5th ed. Mc Graw Hill.

[16] B. Meyer, *Object-Oriented Software Construction*. Prentice Hall, 1997.

[17] R. Rhoad, G. Milauskas, and R. Whipple, *Geometry for Enjoyment and Challenge*. Houghton Mifflin School, 1991.

[18] R. A. Johnson, *Advanced Euclidean Geometry*. 1960.

[19] A. Bangor, P. Kortum, and J. Miller, “Determining what individual SUS scores mean: Adding an adjective rating scale,” *J. usability Stud.*, 2009.

[20] TrymyUI, “The System Usability Scale.” .

9. Anexos

1. **Anexo 1**: https://www.dropbox.com/s/pdz6ab93zn6oo80/PruebaReconocimientoF1_Estatica_10ms_format_Final.xlsx?dl=0

2. **Anexo 2:** https://www.dropbox.com/s/0ssjimq59dfp9cj/PruebaReconocimientoF1_Estatica_10ms_3Firmas2_Format_Final.xlsx?dl=0

3. **Anexo 3:** https://www.dropbox.com/sh/5cpl68txml969n0/AAAcx3xpY-Kr9qe8DQIvQnkba?dl=0}

4. **Anexo 4:** https://www.dropbox.com/s/w3v8i2ddq2ey66e/pruebaComFinal%20-%20copia.xlsx?dl=0

5. **Anexo 5:** https://www.dropbox.com/sh/6vgh5f8k1co616l/AABAUkEQFsncvyIWCMfNrQ4Pa?dl=0

6. **Anexo 6:** https://www.dropbox.com/sh/xjmwy5m067azjbw/AACimkmvnfd8YZZfVaD6DaqYa?dl=0