PLATAFORMA PARA LA EMULACIÓN DE TARJETAS DE DESARROLLO.

Daniela Villamizar Tapias

Juan David Londoño Correa

Universidad del Quindío
Facultad de Ingeniería
Programa de Ingeniería Electrónica
Armenia, Colombia
2022

PLATAFORMA PARA LA EMULACIÓN DE TARJETAS DE DESARROLLO.

Daniela Villamizar Tapias

Juan David Londoño Correa

Tesis presentada como requisito parcial para optar al título de:

Ingeniero Electrónico

Director:

Luis Miguel Capacho Valbuena

Universidad del Quindío
Facultad de Ingeniería
Programa de Ingeniería Electrónica
Armenia, Colombia
2022

Agradecimientos

Agradecemos a nuestra alma mater, por la formación que nos brindó desde el primer semestre y todo el acompañamiento en cada uno de los procesos; así mismo agradecemos a cada uno de los docentes pues nos transmitieron sus conocimientos con el fin de formar profesionales con grandes habilidades y principios. Gracias a todos los compañeros que nos acompañaron en todo el proceso y a nuestras familias, pues fueron nuestro apoyo para poder culminar nuestros estudios; por último y no menos importante, agradecemos a nuestro director de tesis, el ingeniero Luis Miguel Capacho Valbuena, por su disposición y acompañamiento a lo largo de este proyecto.

Resumen

Las tarjetas de desarrollo se pueden definir como el cerebro de un objeto físico o espacio determinado, pues estos dispositivos cumplen la función de un computador, que de una u otra forma se pueden programar según la necesidad de cada usuario; por ello, estas tarjetas han venido a ser la base y/o la mejor ruta para plasmar y procesar mediante un entorno tecnológico y sistematizado, el mundo real. Existe una gran variedad de plataformas donde se puede sacar provecho de las funciones y tareas que las tarjetas de desarrollo pueden brindar, pues mediante su emulación a través de una interfaz, se logra simular la tarjeta como en su estado físico; sin embargo, la mayoría de estas plataformas, no cuentan con una arquitectura basada en el código abierto, donde usuarios externos pueden realizar sus aportes de documentación en aras de enriquecer poco a poco las funciones y/o características de la plataforma de una manera minimalista, con el fin de lograr un proyecto más robusto. Partiendo de lo anterior, surge la idea de crear una plataforma capaz de plasmar las tarjetas de desarrollo con cada una de sus herramientas y funciones que estas brindan de forma física, la cual a su vez será de gran ayuda para la implementación de proyectos y/o procesos experimentales. Es preciso mencionar que, a diferencia de otros proyectos previamente implementados con una finalidad similar, en este proyecto se pretende brindar la posibilidad de que algún usuario externo, pueda adicionar elementos y características nuevas, retroalimentando las funciones, tarjetas y herramientas de la plataforma. Cabe denotar que, se propone una arquitectura para el proyecto, propicia para los usuarios que deseen agregar cualquier tarjeta de desarrollo, pues este viene a ser el enfoque principal; esto con el fin de crear un entorno de desarrollo libre que a la postre seguirá creciendo poco a poco.

Palabras clave: QEMU, tarjetas de desarrollo, JSON, Raspberry Pi3, Python, emulación, código abierto.

Objetivos

Objetivo General

- Implementar una plataforma para la emulación de tarjetas de desarrollo que permita integrar de forma minimalista, las funciones y herramientas disponibles para dichas tarjetas.

Objetivos Específicos

- Diseñar una arquitectura modular que permita agregar nuevas características a la plataforma.
- Implementar la estructura de la plataforma, de acuerdo a la arquitectura propuesta.
- Validar el funcionamiento de la plataforma mediante la emulación de una tarjeta Raspberry Pi 3.

CONTENIDO

Listado de Imágenes

Capítulo 1

1. Introducción

Se entiende por emulación, el proceso digital que permite replicar un dispositivo físico con sus respectivas características y las funciones que este tiene, teniendo presente el tipo de arquitectura. De esta manera, un emulador brinda la posibilidad de plasmar mediante un entorno gráfico digital, a través de un software determinado, la esencia de un dispositivo sin importar su tipo de hardware.

Actualmente, la gran mayoría de las plataformas que se enfocan en la emulación de ciertas tarjetas de desarrollo, no cuentan con la posibilidad de anexar documentación con el fin de enriquecer la plataforma como tal, pues su arquitectura se enfoca en tarjetas de desarrollo específicas, dado que si se piensa en la idea de crear un proyecto capaz de emular cualquier tarjeta de desarrollo con cada una de sus características y funciones, se torna bastante complejo, teniendo en cuenta que cada tarjeta de desarrollo cuenta con cualidades diferentes. De esta manera, se debe pensar en un proyecto que logre basarse en un sistema de retroalimentación, donde diferentes personas puedan realizar aportes que nutren la documentación de la plataforma inicialmente propuesta.

Partiendo de lo anterior, se plantea crear este proyecto, con el fin de dar solución a este tipo de plataformas que no cuentan con una arquitectura de código abierto, lo cual impide que se agreguen nuevas herramientas de forma minimalista. Por ello, la plataforma que se diseña centra su enfoque en la creación de una interfaz, con una arquitectura base, planteando como idea principal, la emulación de tarjetas de desarrollo; así mismo, con el fin de poder mostrar un sistema más visual, se propone distribuir la interfaz de manera que los usuarios logren validar en un principio, la funcionalidad de la misma, donde podrán interactuar con el espacio gráfico a través de algunos componentes. Por ello, la interfaz cuenta con la posibilidad de visualizar en un entorno gráfico, una tarjeta de muestra y algunos componentes; también cuenta con un espacio para la edición de texto, donde se pueden realizar los códigos pertinentes para la programación de las tarjetas de desarrollo y a su vez, cuenta con la ayuda de una terminal donde se pueden realizar diversas operaciones.

Para desarrollar el proyecto, este trabajo se estructuró en seis (6) capítulos, los cuales son:

Capítulo 1: Marco Introductorio, donde se introduce al tema analizando el problema hallado, definiendo para su resolución un objetivo general y varios específicos. Capítulo 2: Marco Teórico Referencial, aquí se contemplan los antecedentes de estudios previos al proyecto, asimismo se puntualizan algunas consideraciones teóricas convenientes para el abordaje del mismo. Capítulo 3: Desarrollo e Implementación, se describen los aspectos más relevantes de cómo se desarrolló el proyecto y cuáles fueron las partes involucradas en su implementación. Capítulo 4: Resultados y análisis, el cual consiste en analizar los resultados obtenidos y validar el funcionamiento de la aplicación. Capítulo 5: Conclusiones, donde se exponen las conclusiones obtenidas a lo largo del proyecto. Capítulo 6: Trabajos Futuros, se plantean posibilidades para ampliar el alcance de la herramienta y poder vincular más funciones.

Capítulo 2

2. Marco Teórico

Los sistemas digitales son la base o el área de estudio en el cual se va a desenvolver este proyecto, teniendo en cuenta que esta rama de la electrónica tiene multitud de variantes que año tras año se han potenciado gracias al sin fin de proyectos y procesos experimentales que se han llevado a cabo, con el fin de perfeccionar cada uno de los conceptos y mecanismos que esto conlleva.

Un claro ejemplo de ello es el proyecto Wyliodrin STUDIO, el cual fue lanzado hace aproximadamente 2 años al mundo digital, cabe aclarar que este tiene por naturaleza o esencia convertirse en un proyecto de código abierto, pues se espera que poco a poco, pueda ir creciendo paulatinamente con el fin de convertirse en una gran plataforma multifuncional. “Wyliodrin STUDIO" es una plataforma educativa para el desarrollo de software y hardware para IoT y sistemas Linux embebidos. La aplicación se ha construido como un marco extensible. La arquitectura principal es una colección de complementos que agregan funcionalidad. Este enfoque se ha elegido porque los diferentes dispositivos tienen formas muy diferentes de conectarse e interactuar con la computadora y / o el navegador. El sistema de complementos permite que Wyliodrin STUDIO sea muy flexible y ampliable. Agregar características como dispositivos o idiomas compatibles y eventos nuevos y muy diferentes funcionalidades es una cuestión de escribir un nuevo complemento” [1].

Partiendo del proyecto mencionado anteriormente y su ideología, ha surgido la idea de concebir, diseñar, implementar y operar una plataforma con características similares, que a su vez permita por medio de su arquitectura, alimentar y/o agregar diferentes cualidades que a la postre harán de esta una plataforma óptima y polifuncional. Para comprender mejor el enfoque de esta idea de proyecto, es necesario tener claro como primera instancia, algunos de los conceptos y elementos en los que se desenvolverá el desarrollo de este; como se describe en las siguientes secciones.

2.1 Tarjetas de desarrollo

Con el paso de los años, se ha hecho notorio el aumento de las tarjetas de desarrollo y su gran variedad en el mercado, permitiendo así tener varias opciones a la hora de diseñar un proyecto de electrónica y/o ramas similares.

Las tarjetas de desarrollo tienen como componente principal un microcontrolador, el cual se encarga de correr o ejecutar de forma rápida una serie de instrucciones de un programa suministrado; este a su vez cuenta con elementos como puertos, conectores, reguladores que permiten la programación del componente, suministrar un voltaje adecuado, entre otras cualidades que permitirán a la postre un óptimo funcionamiento del controlador y así mismo proporcionar acceso a las entrada y salidas del microcontrolador para la conexión de sensores y actuadores.

Las tarjetas de desarrollo brindan una gran variedad de características, según su enfoque: Diseño electrónico, dispositivos de IoT (Internet de las cosas), dispositivos portables (wearables), dispositivos con inteligencia artificial.

2.1.1. Tarjetas de desarrollo más comunes

- **Arduino*:*** *"Es una plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un microcontrolador reprogramable y una serie de pines hembra. Estos permiten establecer conexiones entre el microcontrolador y los diferentes sensores y actuadores de una manera muy sencilla (principalmente con cables DuPont)"* [2].

- **Raspberry Pi:** *"La Raspberry Pi es una computadora de bajo costo y con un tamaño compacto, del porte de una tarjeta de crédito, puede ser conectada a un monitor de computador o un TV, y usarse con un mouse y teclado estándar. Es un pequeño computador que corre un sistema operativo Linux capaz de permitirles a las personas de todas las edades explorar la computación y aprender a programar lenguajes como Scratch y Python. Es capaz de hacer la mayoría de las tareas típicas de un computador de escritorio, desde navegar en internet, reproducir videos en alta resolución, manipular documentos de ofimática, hasta reproducir juegos"* [3].

- **ESP8266:** *"El ESP8266 es un chip de bajo costo Wi-Fi con un stack TCP/IP completo y un microcontrolador, fabricado por Espressif, una empresa afincada en Shanghái, China"* [4].

- **ESP32:** *"El módulo ESP32 es una solución de Wi-Fi/Bluetooth todo en uno, integrada y certificada que proporciona no solo la radio inalámbrica, sino también un procesador integrado con interfaces para conectarse con varios periféricos."* [5].

- **FPGA:** *"FPGA es el acrónimo de **Field Programable Gate Arrays** y no es más que una serie de dispositivos basados en semiconductores a base de matrices de **bloques lógicos configurables o CLB**, donde además se conectan a través de lo que en el sector se denomina como interconexiones programables. Su principal característica y ventaja es que pueden ser reprogramados para un trabajo específico o cambiar sus requisitos después de haberse fabricado. El inventor de esta tecnología fue Xilinx, el cual ha evolucionado dicha tecnología hasta convertirla en un nuevo concepto a tener en cuenta en ciertos entornos de trabajo"* [6].

2.1.2 Raspberry Pi 3

La Raspberry Pi es básicamente un mini computador, capaz de prestar una gran variedad de funciones a los usuarios que deseen aprender y experimentar en el mundo de la programación y la informática. Este dispositivo, como cualquier otro ordenador, cuenta con un SoC, CPU, RAM, y varios puertos de entrada y salida, para diversas aplicaciones. Esta tarjeta se puede conectar a dispositivos como: televisores, teclados, mouses, etc, con los cuales los usuarios pueden interactuar y poner en marcha sus proyectos. A grandes rasgos, con esta tarjeta de desarrollo, los usuarios pueden usarla como media center para interactuar con otros dispositivos; como emulador, para plasmar en un entorno digital, mecanismos físicos; sirve incluso como un mini ordenador de software libre, basado en Linux y así mismo se puede llegar a emplear para proyectos más complejos como la domótica y la actual tecnología IoT.

Esta tarjeta cuenta con una gran variedad de pines que funcionan como entradas y salidas, como se logra apreciar en la Figura 1.

Figura 1. Asignación de pines Raspberry Pi 3

2.2 QEMU

Este será el emulador con el cual se podrá validar el funcionamiento de distintas tarjetas de desarrollo, sin embargo, a lo largo del proyecto se va a trabajar específicamente con la tarjeta Raspberry Pi 3.

La función principal de este programa, es ejecutar máquinas virtuales que puedan emular varios sistemas y/o dispositivos, dentro de un sistema operativo en específico, ya sea en Linux, Windows o algún otro. Este a su vez puede ejecutarse en un sin fin de microprocesadores. QEMU emula el sistema computador completo, incluyendo el procesador y varios periféricos. Puede ser usado para proveer varios almacenamientos de web virtual, en una sola computadora [8].

2.2.1 Características QEMU

- Soporta emulación de IA-32 (x86) PC, AMD64 PC, MIPS R4000, Sun's SPARC sun4m, Sun's SPARC sun4u, ARM development boards (Integrator/CP y Versatile/PB), SH4 SHIX board, PowerPC (PReP y Power Macintosh), y arquitecturas ETRAX CRIS.
- Soporte para otras arquitecturas en *host* y sistemas emulados (ver página principal para una lista completa)
- Aumento de velocidad — algunas aplicaciones pueden correr a una velocidad cercana al tiempo real.
- Implementa el formato de imagen de disco Copy-On-Write. Se puede declarar una unidad virtual multi-gigabyte, la imagen de disco ocupará solamente el espacio actualmente utilizado.
- Implementa la superposición de imágenes. Se puede mantener una foto del sistema huésped, y escribir cambios en un archivo de imagen separado. Si el sistema huésped se colapsa, es sencillo volver a la foto del sistema huésped.
- Soporte para ejecutar binarios de Linux en otras arquitecturas.
- Es posible salvar y restaurar el estado de la máquina (por ejemplo programas en ejecución.)
- Emulación de tarjetas de red virtuales.
- Soporte SMP.
- El Sistema Operativo huésped no necesita ser modificado o parcheado.

- Mejoras en el rendimiento cuando se usa el módulo del kernel KQEMU (no soportado desde la version 0.12).
- Las utilidades de línea de comandos permiten un control total de QEMU sin tener que ejecutar X11.
- Control remoto de la máquina emulada a través del servidor VNC integrado [8].

2.3 QT Creator

Qt es una aplicación multiplataforma y un marco de interfaz gráfica de usuario (GUI), un kit de herramientas que se utiliza para desarrollar software que se puede ejecutar en diferentes plataformas de hardware y sistemas operativos. Qt facilita el desarrollo de software con GUI de aspecto nativo (para el sistema operativo en el que se ejecuta) utilizando C ++ estándar, por lo que también se clasifica como un kit de herramientas de widgets [9].

2.4 Formato JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

Este formato está desarrollado sobre dos estructuras universales, que todos los lenguajes de programación las soportan y son [10]:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un *objeto*, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Capítulo 3

3. Desarrollo e Implementación

En este capítulo se mencionan los requerimientos generales del proyecto, se describe así mismo la arquitectura de la plataforma y la manera en que se desarrollaron cada uno de los elementos claves que la conforman.

3.1 Requerimientos

Este proyecto se plantea como propósito general, poder brindar a los usuarios la posibilidad de interactuar en un entorno de fácil manejo, en el cual pueden desempeñar diversas funciones, plasmadas en una serie de herramientas que se derivan de un corto listado de componentes; así mismo, tiene como enfoque principal una arquitectura base con la posibilidad de ampliar el catálogo de componentes, tarjetas y funciones de la misma. Partiendo de lo anterior, lo que se espera es que los usuarios puedan:

- Tener un entorno de trabajo simple.
- Distribuir los espacios de trabajo de forma que sea fácil su uso.
- Tener a disposición un menú que integre todas las funciones básicas de la plataforma.
- Tener un editor de texto.
- Tener una terminal.
- Agregar nuevos componentes.
- Administrar proyectos de la aplicación.
- Ejecutar las funciones de cada componente.

3.2 Arquitectura

El desarrollo de este proyecto se centra en una arquitectura base para la emulación de tarjetas de desarrollo, donde podrán encontrar una serie de tarjetas y componentes con cada una de sus funciones. De esta manera, los usuarios que deseen contribuir con esta arquitectura agregando nuevos componentes y funciones, deberán basar su documentación en la estructura planteada y agregar los archivos correspondientes. (Ver inciso 4.2 Crear un Componente).

Además, esta arquitectura fue implementada pensando en la comodidad de los usuarios en cuanto al manejo de la misma, diseñando controles rápidos que les permitan realizar una serie de funciones básicas dentro de la aplicación como, abrir o guardar un proyecto, elegir un componente y visualizarlo en el espacio grafico, poder escribir código fuente en cualquier lenguaje de programación en el editor de texto de la aplicación y que, además podrán aprovechar todas sus características para ajustarlo a su gusto y tener a disposición una terminal. (Ver 3.3 Interfaz Gráfica).

Por último, este proyecto tiene un código fuente estructurado de tal forma que se cuenta con una clase principal, encargada de agrupar las demás clases y establecer un intercambio de información necesaria para el funcionamiento del mismo. Para dar un orden al código fuente y siguiendo la idea principal de la arquitectura base planteada, cada uno de los elementos principales del proyecto tiene una clase la cuál realizará funciones específicas para cada una de las partes del proyecto, cómo se puede observar en la Figura 2.

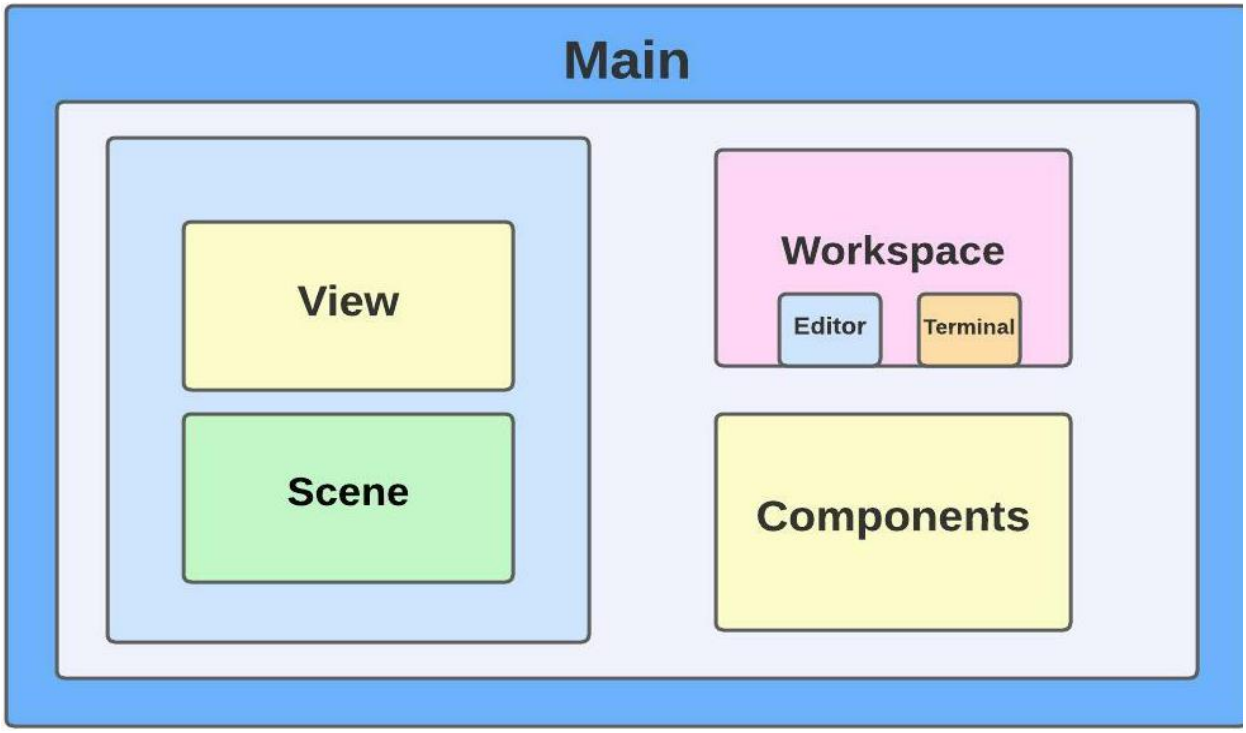

Figura 2. Arquitectura del código fuente.

3.3 Interfaz Gráfica

Para dar inicio al desarrollo de la plataforma, se procedió inicialmente a crear un proyecto en Qt creator, por medio del sistema operativo Linux, echando mano de las bibliotecas del PyQt5, de donde se importaron las clases y métodos necesarios para la escritura del código fuente. Una vez creado el proyecto en Qt, se procedió a organizar la distribución de los widgets que conformarían el entorno gráfico, teniendo en cuenta que, la idea central a la hora de generar el espacio gráfico, era brindar simplicidad al usuario, logrando que cada herramienta y/o funciones del espacio, fueran fáciles de manejar y comprender por cualquier usuario, basándose siempre en una arquitectura minimalista. De esta manera, se optó por conformar este entorno gráfico, con un menú para cada una de las funciones básicas de la interfaz e inmediatamente bajo este menú, se encuentra la barra de herramientas; se determinó así mismo un espacio para la visualización de las tarjetas de desarrollo y los componentes con sus respectivas características de "hardware". Este espacio se encuentra acompañado de una terminal en la parte inferior donde los usuarios pueden obtener muchas ayudas a la hora de ejecutar sus proyectos y también validar la correcta emulación de los mismos. Por otro lado, se implementó un editor de texto, basados en la documentación de QScintilla, al cual se le acoplaron varias características, con el fin de brindar al usuario un espacio donde pueda realizar la escritura de los códigos o documentación en general de sus proyectos y experimentos. Lo anterior se puede comprender mejor, según las Figuras 3 y 4.

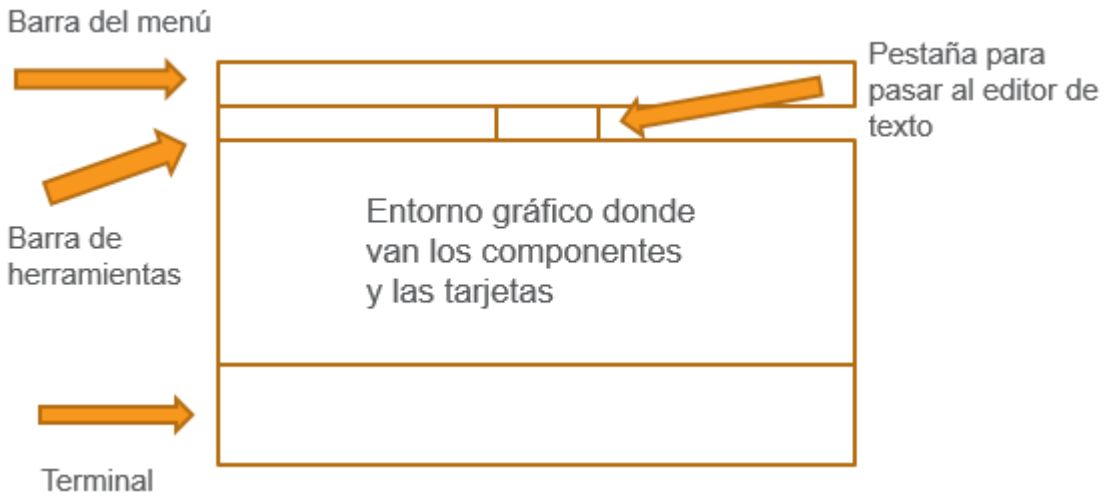

Figura 3 Bosquejo interfaz gráfica. Tomado de fuente propia

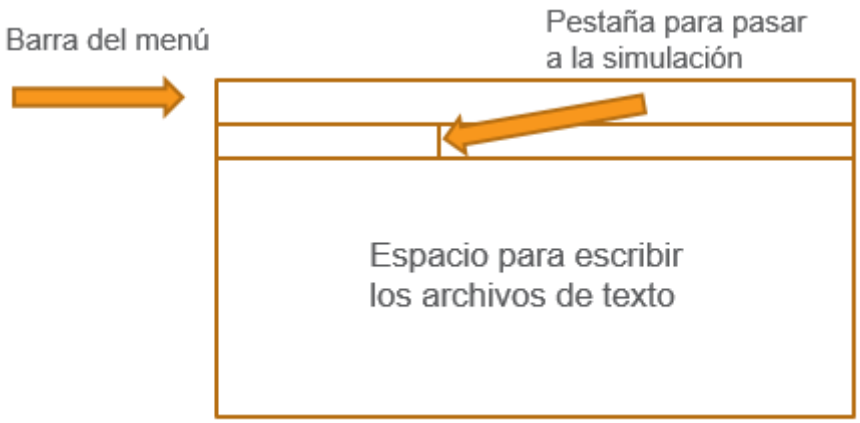

Figura 4 Bosquejo editor de texto. Tomado de fuente propia

3.3.1 Menú

Este menú, incorpora las funciones básicas de cualquier aplicación, agrupadas en cinco pestañas: File, Edit, Components, Settings, Help. Para el caso de la pestaña File, allí se pueden encontrar acciones como la creación de un nuevo proyecto, guardar los cambios realizados, abrir un proyecto existente y salir de la interfaz; en la pestaña Edit, se encuentran las acciones como copiar, pegar, cortar y borrar; en la pestaña de Components, se encuentra el listado de las categorías de las tarjetas y los componentes y de allí se pueden seleccionar todos los dispositivos; para el caso de la pestaña de Settings, al dar click sobre esta, se despliega un menú con las modificaciones y características que se le pueden generar al editor de texto como el tamaño y tipo de fuente, el tamaño de tabulación, autotabulación, guías de indentación, entre otros; por último, se tiene la pestaña de Help, donde se podrá visualizar la ventana de ayudas.

3.3.2 Terminal

Este es un espacio de la plataforma con el cual los usuarios podrán visualizar el momento en que la tarjeta de desarrollo empieza su emulación y el momento en que termina; así mismo, como cualquier otra terminal, esta servirá de apoyo o ayuda para los usuarios, con tareas y comandos específicos. Para la creación de esta, se integró el proyecto QTermWidget el cuál proporciona un widget que por sus características, puede ser utilizado como una consola terminal integrada, y se agregó al Widget correspondiente en el espacio gráfico diseñado.

3.3.3 Editor de Texto

Este es un espacio de la plataforma que está destinado para la programación de las tarjetas de desarrollo por parte de los usuarios; este por su parte fue concebido mediante un editor de texto libre llamado Scintilla. Para la implementación de este editor, se optó por usar la clase QScintilla, echando mano de su gran variedad de características, las cuales le permiten a los usuarios acomodar el editor a su gusto, a la hora de desarrollar o construir sus propios códigos en cualquier lenguaje de programación. Algunas de las características más relevantes que ofrece la clase QScintilla, son: la posibilidad de auto indentar, subrayar el código, modificar el tamaño y el tipo de fuente, acomodar el tamaño de la tabulación, las líneas guías, la sangría y las márgenes, entre otros.

3.4 Componentes

Un componente es un elemento gráfico que tiene una serie de características y funciones con los que los usuarios pueden realizar sus diseños y/o códigos para sus respectivas emulaciones. Un componente puede ser a una tarjeta de desarrollo o un elemento básico que puedan integrarse a la tarjeta para hacer uso de todas las características de esta.

3.4.1 Características de un Componente

Partiendo del objetivo principal del proyecto, el cual se basa en lograr elaborar la base de la plataforma en código abierto, con el fin de que otros usuarios puedan realizar sus aportes a la documentación de la interfaz; se plantea la posibilidad de agregar nuevas tarjetas y elementos básicos. Partiendo de ello, se establecieron una serie de características y archivos para cada componente que vaya a ser agregado al proyecto, así mismo, todos estos requerimientos deben ir dentro de una carpeta que tendrá por nombre el mismo del componente que se desee agregar y debe ser ubicada en la ruta **home/development_cards/components**. Los requerimientos son:

- **Imagen:** Una imagen alusiva al componente a agregar en formato .svg.

- **Archivo Json:** Cada componente debe tener un archivo de formato .json el cual debe conservar la estructura planteada en el inciso **3.3** con una serie de características necesarias para poder ejecutar funciones de la aplicación.

- **Archivo de emulación:** Los componentes también deben tener un archivo python donde se especificarán las funciones que tendrán en la aplicación.

Cabe destacar que, en caso de que un usuario desee agregar una nueva característica al archivo de formato .json o una nueva función al archivo python del componente nuevo, el usuario así mismo deberá modificar los archivos de los componentes ya existentes, ya que, cómo está planteada la arquitectura de la aplicación, todos los elementos se deben agregar con las mismas funciones y características para el funcionamiento de la misma.

3.4.2 Lectura de Componentes

Partiendo de lo mencionado en el inciso anterior, se debe dar claridad de la manera en que estos archivos JSON deben ser leídos. De esta manera, se creó una clase con el fin de realizar la lectura de la carpeta donde se encuentran los archivos de cada uno de los componentes. Esta clase llamada **Components,** fue diseñada para hacer la lectura de estos archivos una vez iniciada la aplicación e ir guardando su información en un diccionario llamado **jsonList**, esto con el fin de poder acceder a dicha información en el momento en que se requiera, para realizar otras funciones de la aplicación, cómo agregar los componentes existentes al menú de la aplicación para poder hacer uso de ellos. Así mismo, se creó una clase llamada **ComponentsFunctions** para realizar la lectura y ejecución de las funciones correspondientes a cada uno de los componentes. Las funciones que deben tener cada uno de los componentes son una función **start()** que, iniciará la emulación de la tarjeta de desarrollo y dará un estado inicial a los elementos básicos existentes, además de esto se debe tener una función **stop()** la cuál permite detener la emulación para realizar algún cambio, también habrán dos funciones de **read()** y **write(),** que se estarán ejecutando constantemente una vez iniciada la emulación de la aplicación.

3.5 Estructura Archivo JSON

La estructura con la que deben contar cada uno de los archivos .json referentes a los nuevos componentes que se quieran agregar, debe ser así:

- **name:** En este **valor** se debe agregar un **string** relacionado al nombre del elemento a agregar.
- **category:** En este **valor** se debe agregar un **string** con categoría a la cual va a pertenecer la nueva tarjeta o el nuevo elemento electrónico, separados cómo board (tarjetas de desarrollo) y basic (elementos electrónicos).
- **image:** En este **valor** se debe agregar un **string** con el nombre de la imagen que ha de representar el nuevo componente con su extensión .SVG.
- **code_functions:** En este **valor** se debe agregar un **string** con el nombre del archivo python, es decir, el archivo .py
- **state:** en este caso, este **objeto** debe contener los diferentes estados del nuevo componente o la nueva tarjeta, los cuales se visualizan por medio de una imagen en formato svg alusiva a cada estado, sin embargo, en el caso de no tener diferentes estados, esta etiqueta se debe dejar vacía.
- **connections:** Este **arreglo** por su parte, contiene unos **valores**:

    1. **type:** Este **valor** representa el nombre de las entradas y salidas del componente y/o la tarjeta.
    2. **pinNumber:** Éste representa el **número** pin correspondiente.
    3. **iD:** Este **valor** representa la identificación que se le asigna a cada pin en específico.
    4. **coordinates:** Estos **números** representan las coordenadas de las entradas y salidas en la imagen del componente o la tarjeta.

Cabe aclarar que, dependiendo de la complejidad y las características del nuevo componente, se pueden agregar más valores. Así mismo, se debe tener presente que, cada uno de los **objetos**, **arreglos** y **valores** deben estar escritos en minúsculas, con el fin de evitar confusiones en la lectura del archivo.

3.6 Crear o Abrir un Proyecto

Una vez se abre la aplicación, los usuarios se encuentran con dos opciones, pues pueden elegir crear un nuevo proyecto o abrir uno ya existente, por medio de dos botones iniciales de la interfaz. En caso de elegir crear un nuevo proyecto, el usuario podrá pasar a visualizar todo el entorno gráfico con sus características y herramientas como se ha descrito anteriormente, donde podrá empezar a interactuar con las tarjetas y los componentes; si por el contrario el usuario escoge abrir un proyecto ya existente, será dirigido por medio de una ruta predeterminada, donde podrá elegir el proyecto con la extensión .ebe, la cual indica la extensión con la que se diferencian los proyectos de la plataforma (EmuBoardsElectronics) y así poder continuar trabajando en su proyecto.

A medida que el usuario trabaja dentro de la plataforma, realizando modificaciones y añadiendo o quitando componentes con cada una de sus características, internamente se está creando un archivo de texto plano, donde se van guardando cada una de estas modificaciones en los componentes, pues así, se garantiza que el usuario al abrir nuevamente la interfaz, pueda encontrar su proyecto tal y como lo había dejado en la última modificación. La estructura de este archivo plano se puede visualizar en la Figura 5:

```
{
    "button0": [
        {
            "name": nombre del componente,
            "item": objeto QGraphicsSvgItem creado,
            "route_image": ruta de la imagen del componente,
            "posX": posición en x,
            "posY": posición en y
        }
    ]
}
```

Figura 5. Estructura archivo de texto plano para guardar proyecto. Tomado de fuente propia

Con esta estructura se pueden guardar las características de los componentes, de modo tal que al abrir nuevamente el proyecto, todos los componentes se puedan hallar en las mismas posiciones y con las mismas características.

Capítulo 4

4. Resultados y Análisis

El código fuente del proyecto se encuentra disponible en un repositorio de GitHub (https://github.com/Daniela0308/Emu-Boards-Electronics), bajo la licencia de código abierto GPLv3. Esto se hizo pensando en que el proyecto pueda ser modificado para aumentar las características y de esta forma a través del trabajo colaborativo, crear una plataforma robusta y que pueda ser usada en cualquier espacio de desarrollo o espacios educativos sin problemas de licencia.

4.1 Entorno Gráfico

Este es el espacio principal para la visualización gráfica de las tarjetas de desarrollo y cada uno de los elementos asignados. Así mismo, se puede visualizar la barra de herramientas, correr o detener el programa, y una serie de opciones que permitirán al usuario desenvolverse fácilmente con la plataforma de manera minimalista, garantizando de esta manera un impacto visual agradable para los usuarios. En la Figura 6 se puede apreciar el resultado de este espacio en el que se van a desenvolver los usuarios.

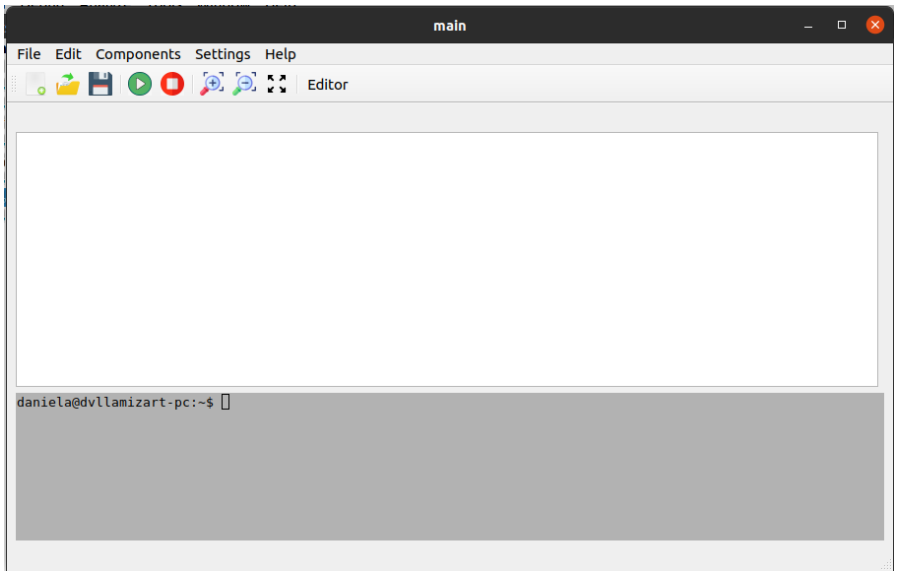

Figura 6 Implementación final de la interfaz gráfica. Tomado de fuente propia

4.1.1 File

En esta parte de la interfaz, se pueden ver los accesos directos con los que cuentan los usuarios, para darle manejo a sus proyectos. El resultado de este se puede apreciar en la Figura 7.

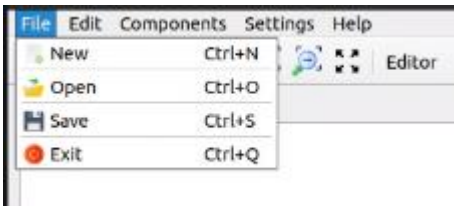

Figura 7 Pestaña File. Tomado de fuente propia

4.1.2 Edit

Como se puede apreciar en la Figura 8, en esta pestaña el usuario encuentra las acciones que le podrán facilitar el manejo de su proyecto, pues con este tipo de funciones, se puede ahorrar trabajo y organizar mejor su espacio de trabajo.

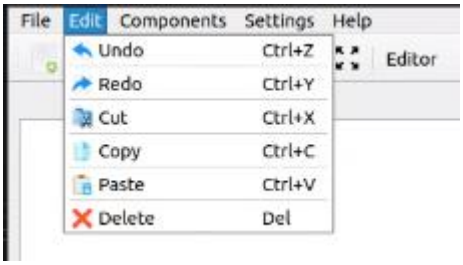

Figura 8 Pestaña Edit. Tomado de fuente propia

4.1.3 Components

Inicialmente, el usuario podrá ver en esta pestaña, los componentes que se encuentran a disposición, según el tipo o categoría al que pertenecen, sin embargo, este listado irá cambiando a medida que se agreguen nuevas carpetas con los archivos necesarios para la creación de nuevas tarjetas o componentes. Un esquema de lo que se encontrará el usuario inicialmente, es cómo se logra apreciar en la Figura 9.

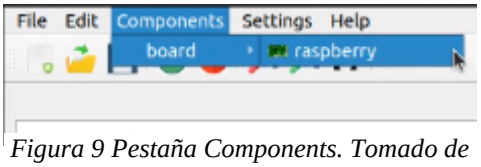

Figura 9 Pestaña Components. Tomado de fuente propia

4.1.4 Settings

Al dar click sobre esta pestaña, el usuario despliega una nueva ventana, donde se encontrará con las características del editor de texto, las cuales se pueden modificar según la necesidad requerida, de modo tal que se le facilite al usuario la escritura de los archivos de texto. Un ejemplo de ello es como se puede ver en la Figura 10, donde las guías de indentación se encuentran activadas; el tamaño de los espacios al tabular es de 4; el color de la línea visible es gris oscuro; el tipo de fuente es de letra cursiva y su tamaño es de 16.

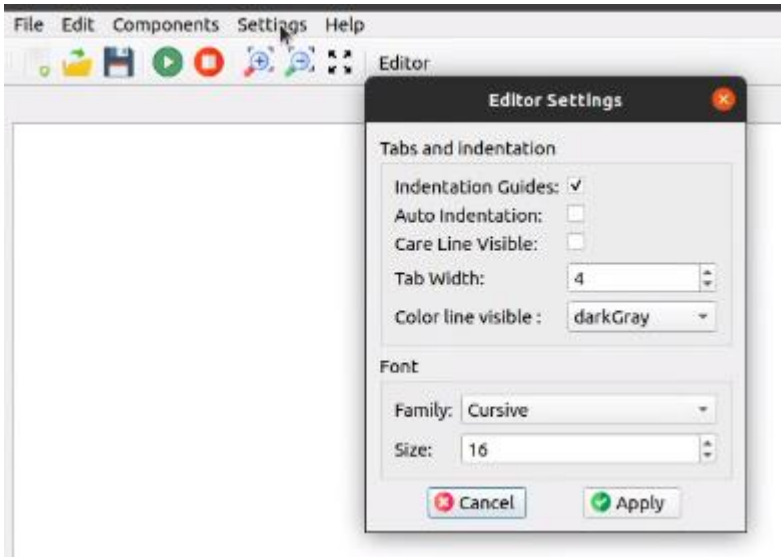

Figura 10 Ventana Settings para el editor de texto. Tomado de fuente propia

4.1.5 Barra de Herramientas

Esta barra de herramientas, cuenta básicamente con 9 íconos, como se observa en la Figura 11.

- Crear un nuevo proyecto
- Abrir un proyecto
- Guardar un proyecto
- Correr la emulación
- Detener la emulación
- Aumentar escala del zoom
- Disminuir escala del zoom
- Zoom predeterminado
- Cambio de página entre entorno gráfico y editor de texto

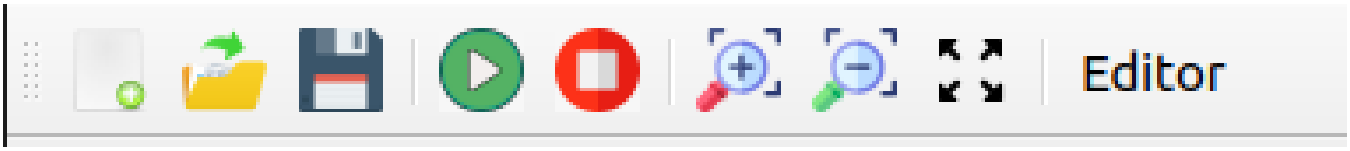

Figura 11 Barra de herramientas. Tomado de fuente propia

4.1.6 Ventana Del Editor

En este punto el usuario puede realizar la escritura de todos sus archivos de texto. Como se aprecia en la Figura 12, donde se visualizan unas líneas escritas que cuentan con ciertas características asignadas al pulsar el botón de Settings.

Figura 12 Editor de texto. Tomado de fuente propia

4.2 Crear un Componente

Cómo se mencionó en el Capítulo 3, Desarrollo e implementación, para crear un componente se deben crear una serie de archivos que cumplen con unas características específicas, esto con el fin de poder aprovechar cada una de las funciones de cada componente y manipularlo dentro de la aplicación. Para la validación de esto, se mostrarán los pasos realizados para crear la tarjeta de desarrollo Raspberry Pi que, como se mencionó desde un principio esta será la tarjeta con la que se validará el correcto funcionamiento de la aplicación.

Como primera instancia, se creó un archivo de formato JSON con las características requeridas para crear la tarjeta de desarrollo Raspberry Pi, como se puede observar en la Figura 13.

```
{
    "name" : "nombre del componente",
    "category" : "categoria a la que pertecene, puede ser board o basic",
    "image" : "nombre de la imagen .svg que se va a visualizar",
    "code_functions" :"nombre del archivo .py que tiene las funciones del componente",
    "state" : {  "objeto que contiene los diferentes estados, en caso de no tener se deja vacío"
            "on" : "imagen de visualización del primer estado",
            "off" : "imagen de visualización del primer estado"
    },
    "connections" : [ "arreglo que contiene información de los conectores de cada componente"
        {
            "type" : "nombre del conector",
            "coordinates" : { "coordenada al rededor del conector"
                "x" : "número de la posición",
                "y" : "número de la posición",
                "h" : "número de la posición",
                "w" : "número de la posición"
            }
        },
        {
            "type" : "cathode",
            "coordinates" : {
                "x" : "número de la posición",
                "y" : "número de la posición",
                "h" : "número de la posición",
                "w" : "número de la posición"
            }
        }
    ]
}
```

Figura 13 Estructura formato JSON para Raspberry Pi 3. Tomado de fuente propia

Después de tener el archivo .json con las características que va a tener la tarjeta de desarrollo o el componente, se debe crear un archivo de emulación python que debe contener las funciones mencionadas en el inciso **3.4.2**, como se observa en la Figura 14.

```
def start():
    global process
    #global processId

    arg = [str(Path.home())+'/qemu-system-aarch64', '-m', '1024', '-M', 'raspi3', '-kernel',
    str(Path.home())+'/kernel8.img', '-dtb', str(Path.home())+'/bcm2710-rpi-3-b-plus.dtb', '-sd',
    str(Path.home())+'/2020-08-20-raspios-buster-armhf.img', '-append',
    'console=ttyAMA0 root=/dev/mmcblk0p2 rw rootwait rootfstype=ext4', '-nographic',
    '-device', 'usb-net,netdev=net0', '-netdev', 'user,id=net0,hostfwd=tcp::5555-:22']

    process = subprocess.Popen(arg, stderr=subprocess.PIPE, stdin=subprocess.PIPE)


def stop():
    process.kill()

def read(arg_term, arg_module, arg_graphics):
    pass

def write():
    return True
```

Figura 14 Archivo de emulación de la Raspberry Pi 3. Tomado de fuente propia

Estos dos archivos junto una imagen alusiva al componente de extensión .svg, se ponen en una carpeta que debe llevar el nombre del componente que se va a crear, y se agrega en la ruta del proyecto **home/development_cards/components** que, según la estructura del proyecto fue la ruta asignada para ubicar los archivos necesarios para crear un componente. Esto se puede observar en las Figuras 15 y 16.

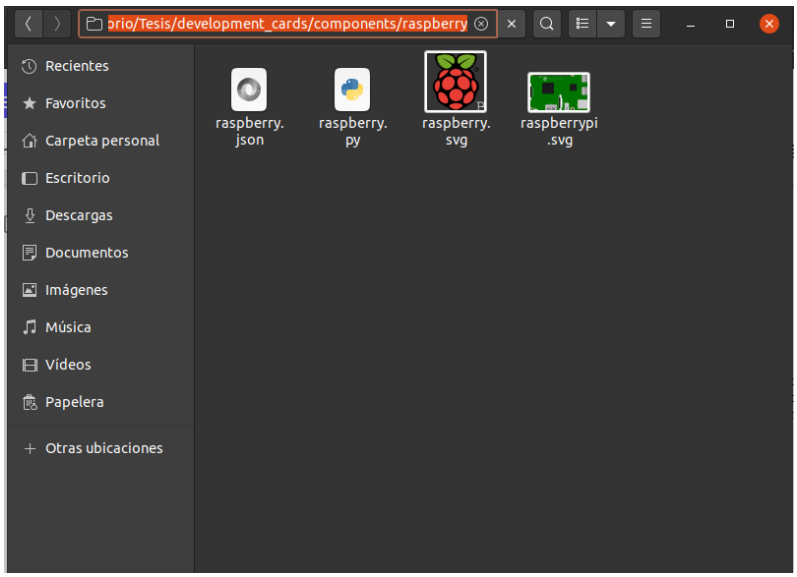

Figura 15 Archivos requeridos de la Raspberry Pi 3. Tomado de fuente propia

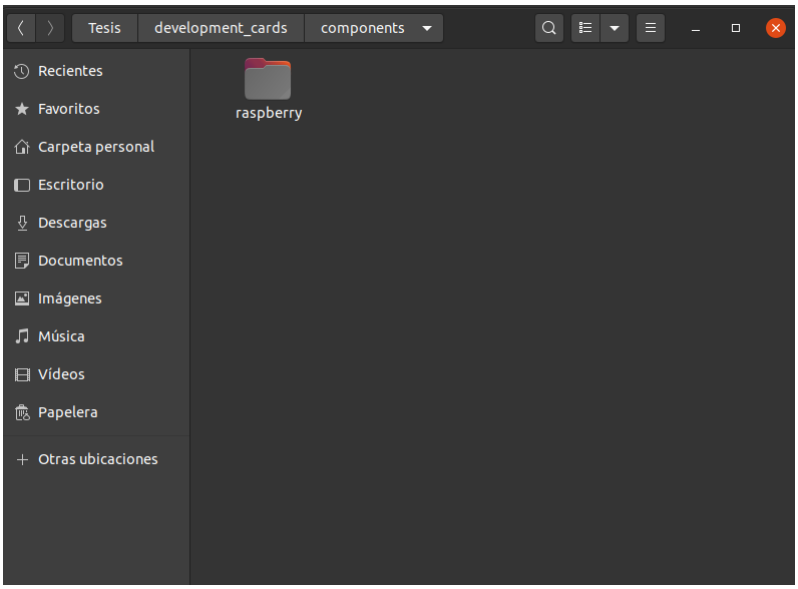

Figura 16 Carpeta Raspberry Pi 3. Tomado de fuente propia

Cuando el programa inicie, va a realizar la lectura de las carpetas existentes en la ruta **home/development_cards/components** y creará la tarjeta de desarrollo Raspberry Pi o el componente que se quiere crear, para validar esto podemos revisar la Figura 8, donde se observa que la tarjeta ya se encuentra disponible en la barra de componentes para poder trabajar con ella. Ahora, se puede utilizar la tarjeta de desarrollo para realizar su respectiva emulación, entonces solo se debe dar click en el menú donde se encuentra la tarjeta y elegir un lugar dentro del espacio gráfico para ponerla, nuevamente dando un click.Una vez se da click, lo que se genera es la creación de un nuevo ítem el cual a su vez corresponde a un nuevo componente, con el que posteriormente por medio de las acciones y/o señales ejecutadas con el cursor a partir del mouse, se puede desplegar en cualquier lugar

del recuadro blanco y se puede movilizar de un punto a otro, según las coordenadas de los ejes verticales y horizontales X y Y. De igual forma, cada componente se puede desplegar más de una vez, es decir, se pueden poner varios componentes del mismo tipo, teniendo en cuenta que cada uno de ellos será nombrado y enumerado por medio de una etiqueta, según su tipo y el orden en que se hayan sacado de la barra de componentes. Otro factor importante a tener en cuenta, es que los componentes se podrán eliminar con una tecla específica del keyboard o teclado. Lo anterior se puede apreciar en la Figura 17.

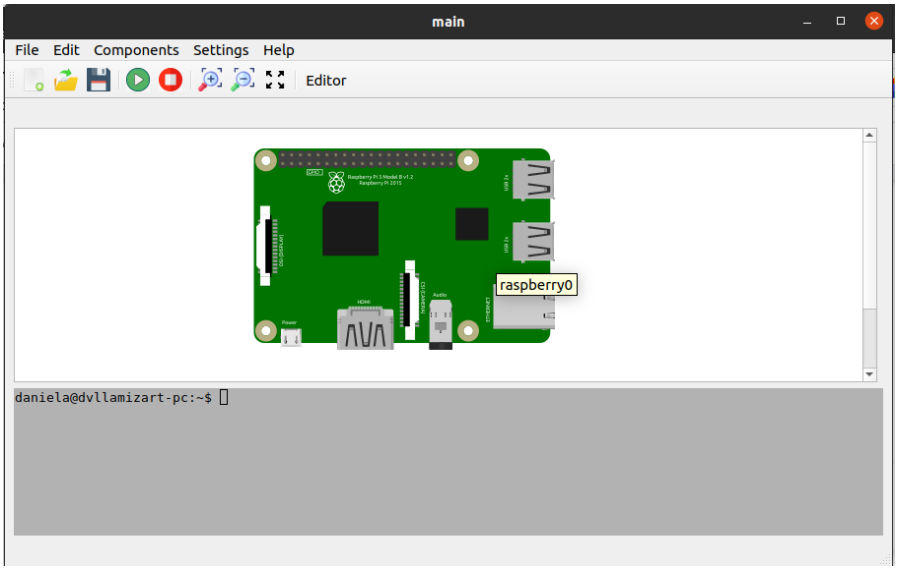

Figura 17 Visualización de Raspberry Pi 3 en el espacio gráfico. Tomado de fuente propia

Partiendo de lo explicado anteriormente, a continuación se muestra la inclusión de dos nuevos componentes, que son el led y el button, los cuales se agregan de la misma manera que fue agregada la tarjeta Raspberry Pi 3, teniendo en cuenta los archivos que deben ir dentro de cada una de las carpetas. La Figura 18 muestra la inclusión de estas dos nuevas carpetas, las cuales a su vez, nutren el listado de los componentes que se pueden seleccionar al dar click en el botón de Components, cómo se logra observar en la figura 19.

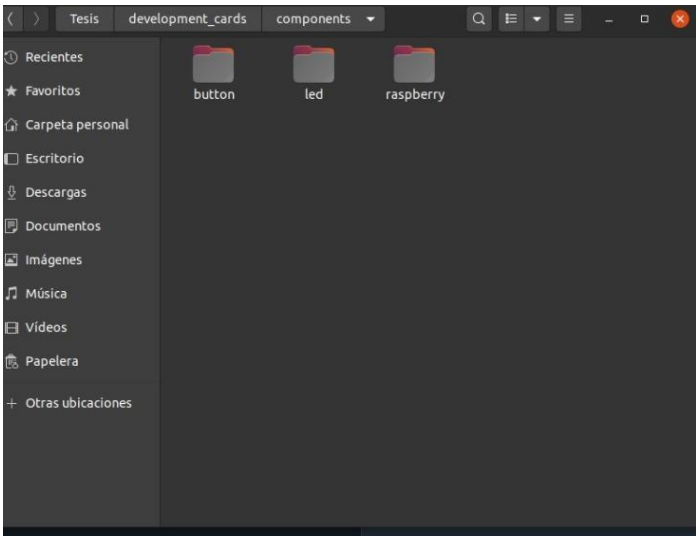

Figura 18 Carpetas de nuevos componentes. Tomado de fuente propia

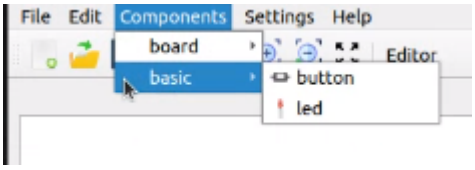

Figura 19 Componentes nuevos agregados en el menú. Tomado de fuente propia

Una vez agregados estos nuevos componentes, se puede hacer uso de ellos, logrando de esta manera seleccionarlos para ser desplegados en la pantalla gráfica junto con la tarjeta Raspberry Pi 3, los cuales se pueden mover con las acciones del cursor, como se ha explicado anteriormente. En la Figura 20 se pueden apreciar los componentes plasmados en el espacio gráfico.

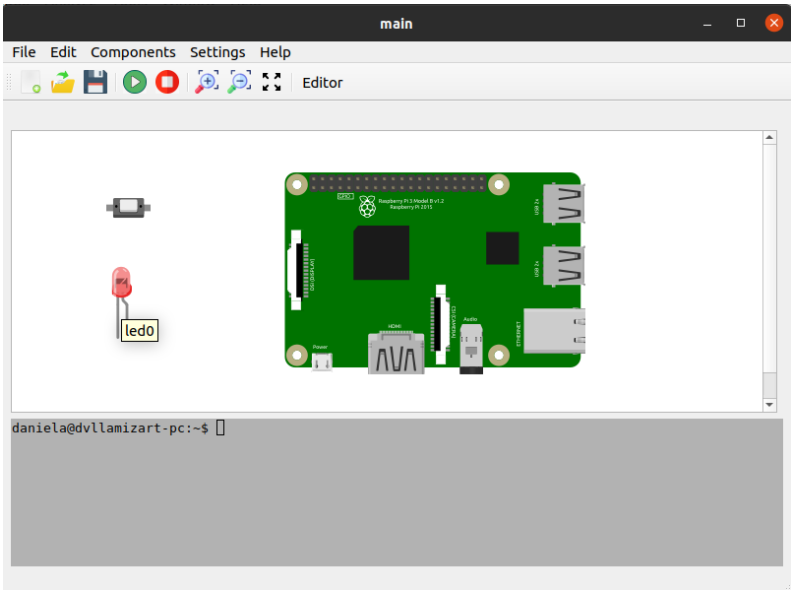

Figura 20 Nuevos componentes plasmados en el espacio gráfico. Tomado de fuente propia

4.3 Guardar un Proyecto

Luego de que el usuario haya realizado tareas en la interfaz gráfica y/o en el editor de texto, desplegando los componentes de su gusto y colocándolos en el lugar de su preferencia, este podrá guardar su proyecto con cada una de las características con que se encuentre la interfaz, dando click en el ícono de guardar o save, que se encuentra en la barra de herramientas, desplegando de esta manera una ventana que le mostrará las rutas de su computador o las carpetas donde podrá archivar el progreso actual de su proyecto, como se muestra en la Figura 21.

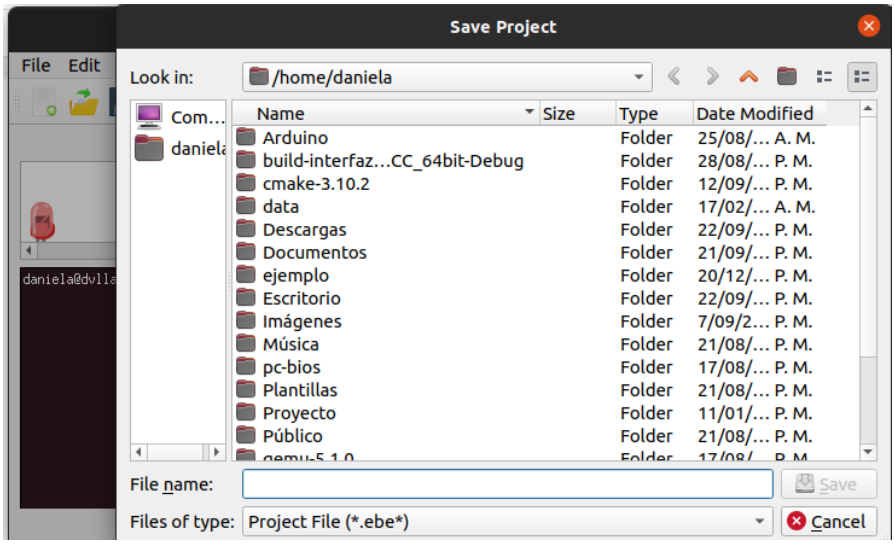

Figura 21 Guardar un proyecto. Tomado de fuente propia

En cuanto el usuario seleccione la ruta donde desea guardar su proyecto, deberá ponerle un nombre a este, seguido de la extensión .ebe, la cual indica que su proyecto es del tipo de la plataforma EmuBoardsElectronics. De esta manera su proyecto quedará guardado para luego poder ser abierto y continuar con el progreso de sus tareas, partiendo del último estado en que había quedado. En la Figura 22 se puede observar un ejemplo de ello.

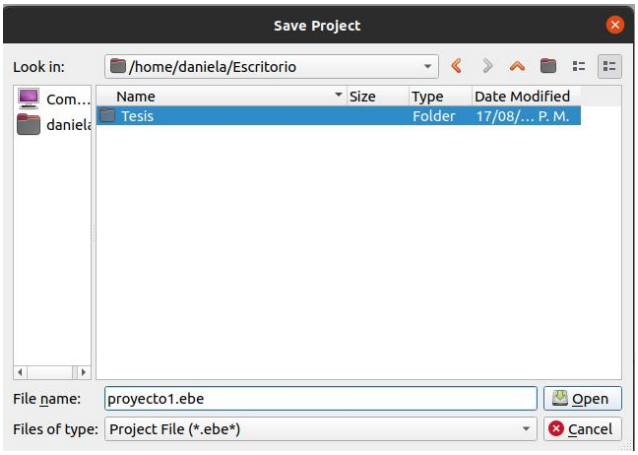

Figura 22 Nombre para guardar un proyecto y la extensión. Tomado de fuente propia

En la Figura 23 se logra apreciar la estructura que toma el archivo de texto plano que se crea una vez el usuario guarda su proyecto.

Figura 23 Estructura del archivo de texto plano al guardar un proyecto. Tomado de fuente propia

4.4 Emulación de Raspberry Pi 3

Para realizar la emulación de la tarjeta de desarrollo Raspberry Pi, se utilizó la herramienta QEMU la cuál permite emular un sistema operativo completo, en este caso, el sistema operativo Raspberry Pi OS con el que trabaja la tarjeta de desarrollo que estamos utilizando. En la aplicación existe un botón llamado RUN que tiene la función de iniciar la emulación de la tarjeta que se encuentre en el espacio de trabajo, cuando se da inicio a esta emulación lo

que hace la aplicación es llamar los archivos de emulación de los componentes que se estén utilizando en ese momento, en este caso, se llama la función **start()** de la tarjeta Raspberry Pi que contiene las líneas de código mostrada en la Figura 24.

```
7   def start():
8       arg = [str(Path.home())+'/qemu-system-aarch64', '-m', '1024', '-M', 'raspi3', '-kernel',
9       str(Path.home())+'/kernel8.img', '-dtb', str(Path.home())+'/bcm2710-rpi-3-b-plus.dtb', '-sd',
10      str(Path.home())+'/2020-08-20-raspios-buster-armhf.img', '-append',
11      'console=ttyAMA0 root=/dev/mmcblk0p2 rw rootwait rootfstype=ext4', '-nographic',
12      '-device', 'usb-net,netdev=net0', '-netdev', 'user,id=net0,hostfwd=tcp::5555-:22']
13
14      p1 = subprocess.Popen(arg, stderr=subprocess.PIPE, stdin=subprocess.PIPE)
15
```

Figura 24 Función start() Raspberry Pi 3. Tomado de fuente propia

Estás líneas de código están creando un proceso que será enviado a la consola-terminal de nuestra aplicación para poder realizar la emulación de la tarjeta de desarrollo. A este proceso, se le envían una serie de argumentos necesarios para realizar la respectiva emulación, indicando el tipo de arquitectura que va a utilizar la tarjeta, la tarjeta que se va a emular, la máquina de estado que se va a trabajar, la tarjeta de memoria, los estándares de entrada y salida, entre otras características cómo se observa en la línea número 8 de la Figura 24. Al correr este proceso se empezará a cargar el sistema operativo de la tarjeta y esta validación se puede observar en la terminal, como lo muestra la Figura 25.

```
daniela@dvllamizart-pc:~$ ./qemu-system-aarch64 -m 1024 -M raspi3 -kernel kernel8.img -dtb bc
m2710-rpi-3-b-plus.dtb -sd 2020-08-20-raspios-buster-armhf.img -append "console=ttyAMA0 root=
/dev/mmcblk0p2 rw rootwait rootfstype=ext4" -nographic -device usb-net,netdev=net0 -netdev us
er,id=net0,hostfwd=tcp::5555-:22
WARNING: Image format was not specified for '2020-08-20-raspios-buster-armhf.img' and probing
 guessed raw.
         Automatically detecting the format is dangerous for raw images, write operations on
block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the restrictions.
[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x410fd034]
[    0.000000] Linux version 5.4.51-v8+ (dom@buildbot) (gcc version 5.4.0 20160609 (Ubuntu/Li
naro 5.4.0-6ubuntu1~16.04.9)) #1333 SMP PREEMPT Mon Aug 10 16:58:35 BST 2020
[    0.000000] Machine model: Raspberry Pi 3 Model B+
[    0.000000] efi: Getting EFI parameters from FDT:
[    0.000000] efi: UEFI not found.
[    0.000000] Reserved memory: created CMA memory pool at 0x0000000038000000, size 64 MiB
[    0.000000] OF: reserved mem: initialized node linux,cma, compatible id shared-dma-pool
[    0.000000] percpu: Embedded 31 pages/cpu s89240 r8192 d29544 u126976
[    0.000000] Detected VIPT I-cache on CPU0
[    0.000000] CPU features: detected: ARM erratum 845719
[    0.000000] CPU features: detected: ARM erratum 843419
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 241920
[    0.000000] Kernel command line: console=ttyAMA0 root=/dev/mmcblk0p2 rw rootwait rootfstyp
e=ext4
[    0.000000] Dentry cache hash table entries: 131072 (order: 8, 1048576 bytes, linear)
[    0.000000] Inode-cache hash table entries: 65536 (order: 7, 524288 bytes, linear)
[    0.000000] mem auto-init: stack:off, heap alloc:off, heap free:off
[    0.000000] Memory: 880604K/983040K available (9532K kernel code, 1104K rwdata, 3352K roda
ta, 1088K init, 1201K bss, 36900K reserved, 65536K cma-reserved)
[    0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=4, Nodes=1
[    0.000000] ftrace: allocating 31763 entries in 125 pages
```

Figura 25 Emulación de Raspberry Pi 3. Tomado de fuente propia

Al finalizar este proceso se deben ingresar el usuario y contraseña para poder ingresar a la emulación, cómo usuario en este caso se trabajó con la palabra Pi y la contraseña es raspberry. Cabe mencionar que, esto puede modificarse en las configuraciones de la tarjeta. Entonces de esta manera, se podrá hacer uso de todas las funciones de la tarjeta de desarrollo por medio de un canal como lo es la terminal, sin necesidad de tener la tarjeta física. La Figura 26, muestra que la emulación fue un éxito debido a que, ya podemos manipular la tarjeta de desarrollo porque nos

encontramos dentro de su sistema operativo y como un breve accedemos a las características del hardware de la tarjeta para validar este proceso.

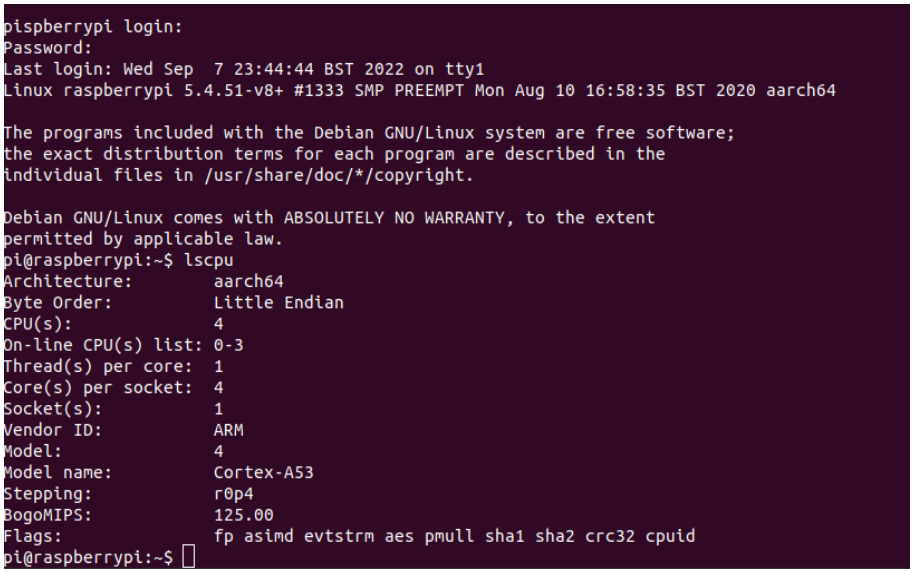

Figura 26 Validación de la emulación de la tarjeta. Tomado de fuente propia

Capítulo 5

5. Conclusiones

- Al haber diseñado este proyecto basado en un sistema de código abierto, se brinda la posibilidad de poder nutrir poco a poco el cuerpo de la plataforma, con el aporte que otros usuarios pueden hacer a la documentación general, con el fin de construir paulatinamente un trabajo más robusto; teniendo presente que, de no ser así, se tornaría bastante complejo para un pequeño equipo de trabajo el poder brindar una amplia variedad de funciones, detalles y características en la plataforma.
- Se creó un sistema con una arquitectura genérica con el fin de emular tarjetas de desarrollo. Para validar el trabajo realizado se hicieron una serie de pruebas donde se pudo verificar que la arquitectura diseñada funciona de la forma esperada y además se muestra como prueba final la emulación de la tarjeta Raspberry Pi 3.

Capítulo 6

6. Trabajos Futuros

Con el fin de continuar y mejorar este trabajo de grado, se proponen los siguientes trabajos futuros:

- Realizar la implementación de las conexiones entre componentes y validarlo por medio de su respectiva emulación.
- Dar mayor funcionalidad al editor de texto, para realizar códigos que puedan determinar el funcionamiento y desempeño de los componentes en el espacio de trabajo.
- Agregar nuevas funciones a los componentes.

Bibliografía

[1].*WyliodrinSTUDIO*. Wyliodrin.studio. (2022). Retrieved 23 August 2022, from https://wyliodrin.studio/

[2] *¿Qué es Arduino?*. Arduino.cl. (2022). Retrieved 23 August 2022, from https://arduino.cl/que-es-arduino/

[3] *¿Que es Raspberry Pi? - Raspberry Pi*. Raspberry Pi. (2022). Retrieved 23 August 2022, from https://raspberrypi.cl/que-es-raspberry/

[4] *ESP8266 - Wikipedia, la enciclopedia libre*. Es.wikipedia.org. (2022). Retrieved 23 August 2022, from https://es.wikipedia.org/wiki/ESP8266

[5] *El módulo ESP32*. www.digikey. (2022). Retrieved 23 August 2022, from https://www.digikey.com/es/articles/how-to-select-and-use-the-right-esp32-wi-fi-bluetooth-module#:~:text=El%20m%C3%B3dulo%20ESP32%20es%20una,para%20conectarse%20con%20varios%20perif%C3%A9ricos

[6] *Qué es un FPGA*. HardZone. (2022). Retrieved 23 August 2022, from https://hardzone.es/reportajes/que-es/fpga-caracteristicas-utilidad/

[7] *Raspberry Pi GPIO Pinout.* Pinout.xyz. (2022). Retrieved 23 August 2022, from https://pinout.xyz/

[8] *QEMU - EcuRed*. Ecured.cu. (2022). Retrieved 23 August 2022, from https://www.ecured.cu/QEMU

[9] *¿Qué es qt?*. Icy Science. (2022). Retrieved 23 August 2022, from https://es.theastrologypage.com/qt#menu-1

[10] *JSON*. Json.org. (2022). Retrieved 23 August 2022, from https://www.json.org/json-es.html