Proyecto III – Circuit designer

Instituto Tecnológico de Costa Rica
Área Académica Ingeniería en Computadores
Algoritmos y Estructuras de Datos I (CE 1103)
Segundo Semestre 2020
Valor 15%

TEC | Tecnológico de Costa Rica

Objetivo General

- ➔ Diseñar e implementar un simulador en corriente continua de circuitos electrónicos resistivos con configuraciones en serie, paralelo o una combinación de ambas.

Objetivos Específicos

- ➔ Implementar grafos que permitan modelar un circuito electrónico compuesto de fuentes y resistencias.
- ➔ Implementar un simulador de circuitos electrónicos que pueda aplicar la Ley de Ohm para el cálculo de la tensión eléctrica en todos los nodos de un esquemático.
- ➔ Diseñar un algoritmo capaz de poder tomar un grafo y convertirlo a texto plano (*netlist*) para poder ser exportado y cargado luego al simulador.
- ➔ Aplicar algoritmos de ordenamiento en un set de datos.
- ➔ Investigar acerca de programación orientada a objetos en Python.
- ➔ Utilizar UML para modelar la solución de un problema.

Descripción del problema

Un circuito electrónico resistivo es todo aquel circuito cuyos elementos pueden ser resistencias o fuentes de poder. Adicionalmente, una resistencia es un componente electrónico que se opone al paso de la corriente y cuyo comportamiento puede ser modelado por la Ley de Ohm, cuya descripción matemática está dada por

$$V = IR$$

donde V corresponde a la tensión eléctrica (también conocida a nivel técnico como *voltaje*), I corresponde a la corriente (también conocida a nivel técnico como *amperaje*) y R es la resistencia.

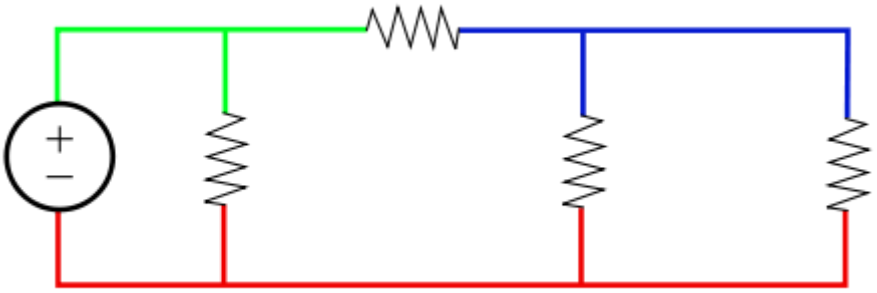
En este proyecto se propone realizar un programa implementado en Python, capaz de simular circuitos electrónicos resistivos con configuraciones en serie, paralelo o una combinación de ambas configuraciones.

Sistema en modo diseño

En modo diseño, el programa debe darle la capacidad al usuario de elegir entre dos elementos para crear el circuito: resistencias y fuentes de poder; además de permitirle al usuario diseñar en su interfaz la topología del circuito que desea simular. De esta manera, el usuario puede seleccionar alguno de los dos componentes electrónicos (fuentes o resistencias) y especificar **cuál va a ser el valor que ese componente va a tener y qué nombre tendrá**. Por ejemplo, si el usuario selecciona una resistencia, el sistema debe ser capaz de permitirle especificar cuál va a ser el valor de esa resistencia, medido en ohms (Ω) y qué nombre tendrá esa resistencia (por ejemplo, R1, R2, Resistencia principal, etcétera). Si selecciona una fuente, se tiene que poder especificar el valor de tensión de esa fuente, medido en *volts (V)*.

Para lograr brindar la mayor capacidad de personalización para el circuito, es necesario permitirle al usuario rotar cada elemento del circuito y elegir manualmente los nodos donde se conecta cada terminal de un elemento específico. Esto significa que el usuario debe poder colocar resistencias y fuentes ya sea de manera horizontal como vertical.

El concepto de nodo electrónico, se define como un punto donde dos o más componentes tienen una conexión común, en la siguiente imagen las líneas de igual color corresponden a nodos.



Una vez el usuario finaliza de diseñar su circuito utilizando la interfaz del programa, este puede empezar a simular el mismo.

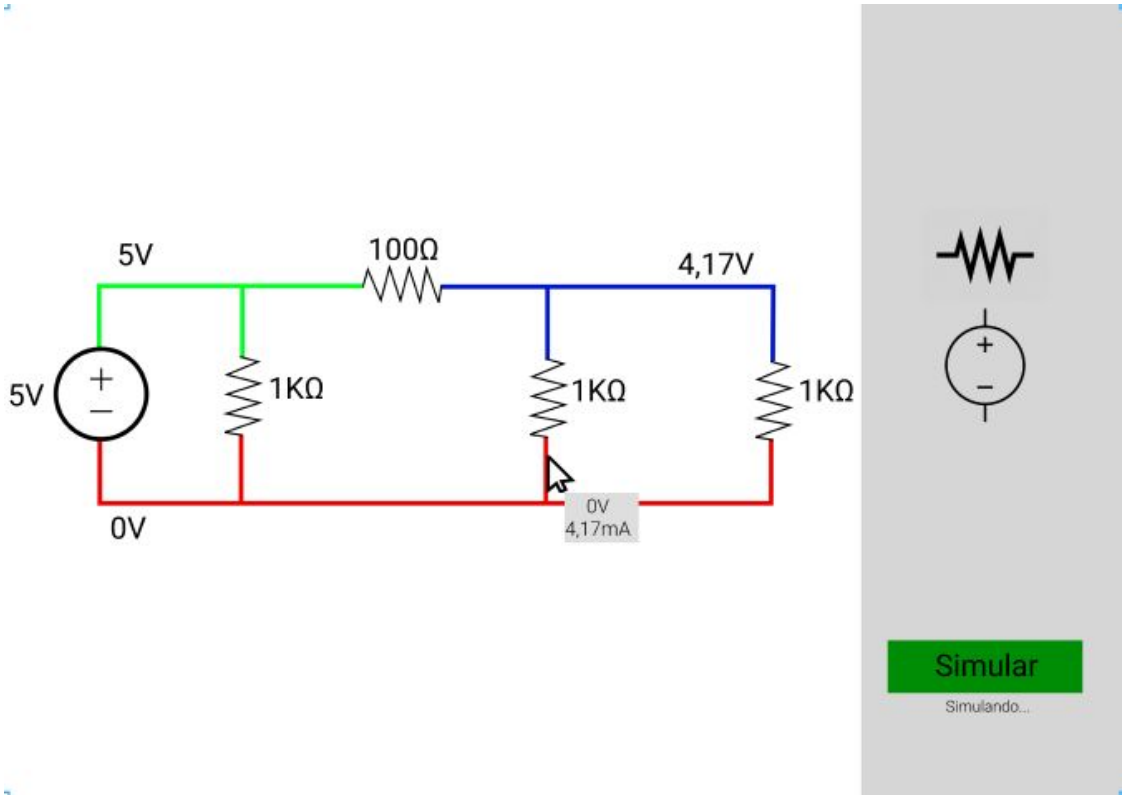
Sistema en modo simulación

En modo simulación, el usuario puede pasar el *mouse* por encima de un nodo de su diseño para enfatizar el voltaje (medida en *volts (V)*) y la corriente (medida en *mili Ampers (mA)*) que atraviesa dicho nodo.

Debido a que el cálculo de la corriente y la tensión de un circuito no es trivial, y requiere conocimiento en electrónica que actualmente el estudiante no necesariamente posee, **se permite que dichos valores sean generados de manera aleatoria** respetando las siguientes restricciones:

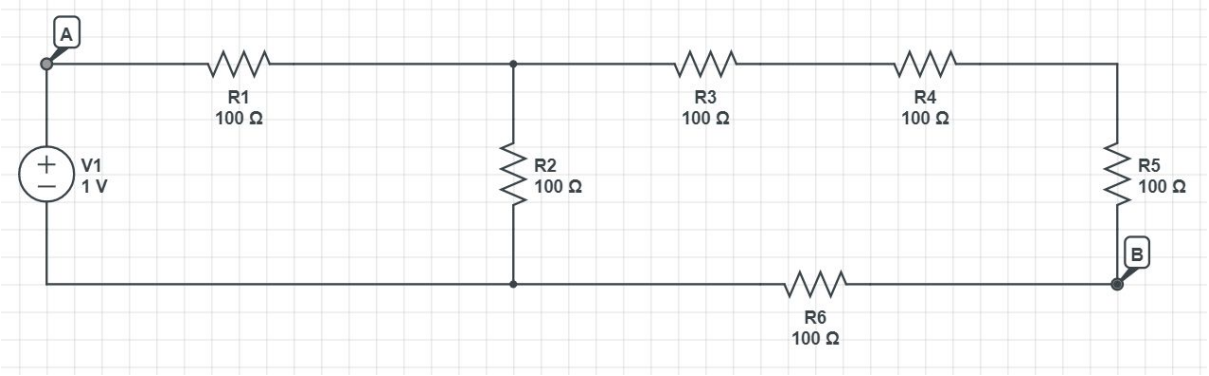
1. Las tensiones en los nodos deben de ir desde los 0V hasta los 10V.
2. Las corrientes en los nodos deben de ir desde los 0mA hasta los 1000mA.

Adicionalmente, **se premiará con 10pts adicionales en el proyecto a los grupos que logren calcular las corrientes y las tensiones de los nodos basado en el valor de la fuente y aplicando la ley de Ohm.** En tutoría se explicará una forma para calcular dichos valores.



Por otra parte, cuando el sistema está en modo simulación, se debe mostrar el *camino de mayor tensión* y el *camino de menor tensión,* partiendo de dos nodos cualesquiera que el usuario puede seleccionar. Esto se debe implementar utilizando el algoritmo de Dijkstra.

Por ejemplo, si el usuario tiene un circuito como el siguiente, puede seleccionar dos nodos cualesquiera y el sistema debe mostrarle cuál es el camino de mayor tensión y cuál es el camino de menor tensión.



Finalmente, cuando el sistema está en modo simulación, se debe mostrar en la interfaz gráfica el nombre de las resistencias de manera ordenada (recuerde que todas las resistencias tienen asociado un nombre o un *label* que permite identificarlas en el circuito). Para esto, deben implementar dos algoritmos de ordenamiento **a excepción de Selection Sort y Bubble Sort.** Uno de los ordenamientos se hace ascendente y el otro de forma descendente.

Sistema en modo importación y exportación

El circuito debe representarse en tiempo de ejecución como un grafo y se le debe dar la posibilidad al usuario de exportar dicho grafo a un archivo de texto (que en electrónica se denomina *netlist file*), con un formato propuesto por el estudiante, este archivo de texto debe de poderse importar al programa posteriormente y el programa debe de ser capaz de generar el mismo circuito (o un equivalente electrónico) que el inicialmente exportado.

Documentación requerida

1. Internamente, el código se debe documentar utilizando el formato estándar de Python.
2. Dado que el código se deberá mantener en GitHub, la documentación externa se hará en el Wiki de GitHub. El Wiki deberá incluir:
    a. Breve descripción del problema
    b. Diagrama de clases en formato JPEG o PNG
    c. Descripción de las estructuras de datos desarrolladas.
    d. Descripción detallada de los algoritmos desarrollados.
    e. Corridas de ejemplo (*screenshots*) para las funcionalidades más importantes del simulador. El estudiante debe definir qué funcionalidades va a agregar a esta sección, pero se recomienda que cada *screenshot* venga acompañado de una breve descripción.
    f. Problemas encontrados: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.
    g. Lecciones aprendidas, donde se describan los retos enfrentados por el equipo y la sugerencia de cómo gestionarlos en el futuro.

3. **Planificación y administración del proyecto:** se utilizará Azure DevOps para la administración de proyecto. Debe incluir:
    - ➔ Lista de features e historias de usuario identificados de la especificación
    - ➔ Distribución de historias de usuario por criticalidad
    - ➔ Plan de iteraciones que agrupen cada bloque de historias de usuario de forma que se vea un desarrollo incremental
    - ➔ Descomposición de cada user story en tareas.
    - ➔ Asignación de las tareas entre los diferentes miembros del equipo.

Aspectos operativos y evaluación:

1. **Fecha de entrega: De acuerdo al cronograma del curso**
2. El proyecto tiene un valor de 15% de la nota del curso.
3. El trabajo es **en grupos de 3 personas**.
4. Es obligatorio utilizar GitHUB.
5. Es obligatorio integrar toda la solución.
6. El código tendrá un valor total de 85% y la documentación 15%.
7. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
8. Se evaluará que la documentación sea coherente, acorde a la dificultad/tamaño del proyecto y el trabajo realizado, se recomienda que realicen la documentación conforme se implementa el código.
9. La nota del código NO podrá exceder en 35 puntos la nota de la documentación, por lo cual se recomienda documentar conforme se programa.
11. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
12. Los estudiantes pueden seguir trabajando en el código hasta 15 minutos antes de la cita revisión oficial
13. Aun cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
    a. Si no se entrega documentación, automáticamente se obtiene una nota de 0.
    b. Si no se utiliza un manejador de código se obtiene una nota de 0.
    c. Si la documentación no se entrega en la fecha indicada se obtiene una nota de 0.
    d. Sí el código no compila o tiene un error de sintaxis se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
    f. La nota de la documentación debe ser acorde a la completitud del proyecto.
14. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto. El único requerimiento que se consultará durante la defensa del proyecto es el diagrama de clases, documentación interna y la documentación en el manejador de código.
15. Cada estudiante tendrá como máximo 15 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.
16. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
17. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.

ANEXO DEL PROYECTO

Objetivo General

➔ Elaborar un documento que evidencie la capacidad para la descripción de un problema complejo de ingeniería en términos de requerimientos de diseño y limitantes y la capacidad para la organización de las actividades individuales en el equipo de trabajo.

Objetivos Específicos

➔ Determinar los requerimientos de ingeniería para un problema complejo considerando partes involucradas, estado del arte, estándares, normas, entre otras.
➔ Establecer los lineamientos para el trabajo en equipo (metas, roles, reglas, cronogramas, bitácoras, entre otros).
➔ Identificar el rol y las responsabilidades como miembro de un equipo de trabajo.

Atributos de Acreditación

➔ Diseño (Inicial).
➔ Trabajo individual y en equipo (Inicial).

Descripción del Entregable

Cada grupo debe elaborar un documento que tenga la siguiente estructura:

1. Portada.
2. Tabla de contenidos.
3. Introducción.
4. Diseño.
    a. Listado de requerimientos utilizando el formato de historias de usuario.
    b. Estado del arte: Describir los conceptos teóricos asociados al proyecto con la respectiva referencia bibliográfica para cada concepto.
    c. Estándares o normas: Describir los estándares o normas que deben utilizarse para el desarrollo de la solución.
5. Trabajo individual y en equipo.
    a. Metas del proyecto: Describir los entregables asociados al proyecto.
    b. Roles: Describir los roles y responsabilidades utilizados en el desarrollo del proyecto.
    c. Reglas: Describir las reglas principales que han sido definidas para el trabajo en equipo.
    d. Cronograma: Plan de proyecto con las actividades planeadas, responsables de cada actividad y fechas de entregas estimadas.
    e. Bitácora: Deben describir las actividades realizadas como reuniones con el compañero de trabajo, investigaciones, consultas, entre otras. Se debe describir todo por más insignificante que sea. Esto demostrará el trabajo de cada uno de los miembros del equipo según el rol asignado.

Aspectos operativos y evaluación

1. El documento debe ser enviado en formato PDF y de entregarse junto con los demás entregables establecidos para el Proyecto #3.