UAEM | Universidad Autónoma del Estado de México

Universidad Autónoma del Estado de México

Centro Universitario UAEM Atlacomulco

“Simulación de circuitos electrónicos en C”

Presenta:

Segundo Antonio Elias Edgardo

Unidad de Aprendizaje:

Estructuras de datos

Fecha de entrega:

6 de septiembre de 2017

1 Introducción

En esta práctica se diseñará un algoritmo que simule un circuito en serie con elementos resistivos y diversas fuentes utilizando estructuras de datos normales y anidadas.

El algoritmo realizado calcula la resistencia y el voltaje total en el circuito ingresado en la consola para después arrojar una medición de voltaje entre los nodos que indique el usuario, la medida del voltaje es mostrada en la consola y adicionalmente se muestra la gráfica con los valores constantes del voltaje medido y el voltaje total del circuito.

Emulando el funcionamiento en modo consola del programa PSPICE la entrada del usuario es ingresada mediante etiquetas, que con la nomenclatura establecida indican el tipo de elemento del circuito que se añadirá y la posición delimitada por nodos de tal elemento.

Una vez ingresados los datos de cada elemento el usuario puede poner fin al ciclo indicando que requiere una medición de voltaje, ingresando el código del multímetro el programa pedirá la posición de las puntas de este y después de ello se indicará el voltaje entre los nodos indicados.

Como características extra se incluyeron diversas alertas que indican, por ejemplo, si de acuerdo con la potencia de las resistencias y la potencia que pasa por ellas la resistencia soportara o se quemara o si la fuente de voltaje puede suministrar el amperaje suficiente para el circuito.

Después de esta descripción general de la práctica se procede a mostrar el marco teórico.

2 Marco teórico

En esta práctica solo se utilizaron los conceptos de suma de resistencias en paralelo y en serie para resolver los circuitos resistivos en serie planteados por el usuario, el primero de ellos es la definición para la suma de resistencias en serie que se desarrolla como sigue:

$$\underline{Rt = R1 + R2 + R3 + R4 \pm \cdots + Rn}$$

Donde Rt es la resistencia total equivalente en el circuito y cada elemento de la sumatoria es una resistencia que forma parte del circuito.

Complementando esta definición también se hizo uso de la ley de Ohm para resolver el circuito resistivo y obtener variables como la intensidad total del circuito, esta se enuncia como sigue:

$$V = IR \qquad I = \frac{V}{R}$$

La primer formula expresa la ley de ohm en términos de voltaje utilizando la resistencia del circuito y el amperaje, mientras que la segunda que proviene solo de un despeje expresa la intensidad de corriente en términos de un cociente de su voltaje y su resistencia total.

Adicionalmente también se hizo uso de las estructuras anidadas vistas en clase para representar los elementos en el circuito, aquí se muestra un ejemplo de implementación:

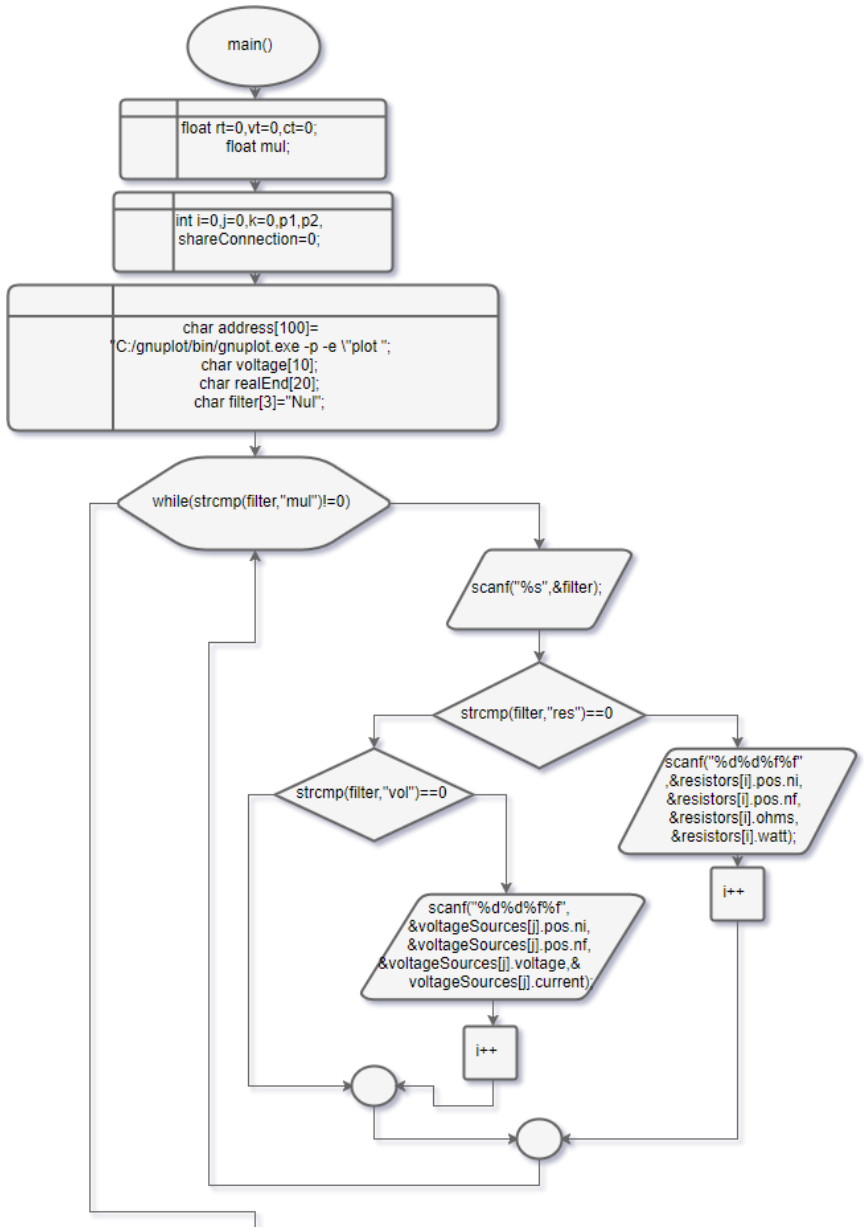
```
Struct coord. {

Int x, y;};

Struct circle {

Int radius;

Struct coord. CoordCircle;

} circle1;
```

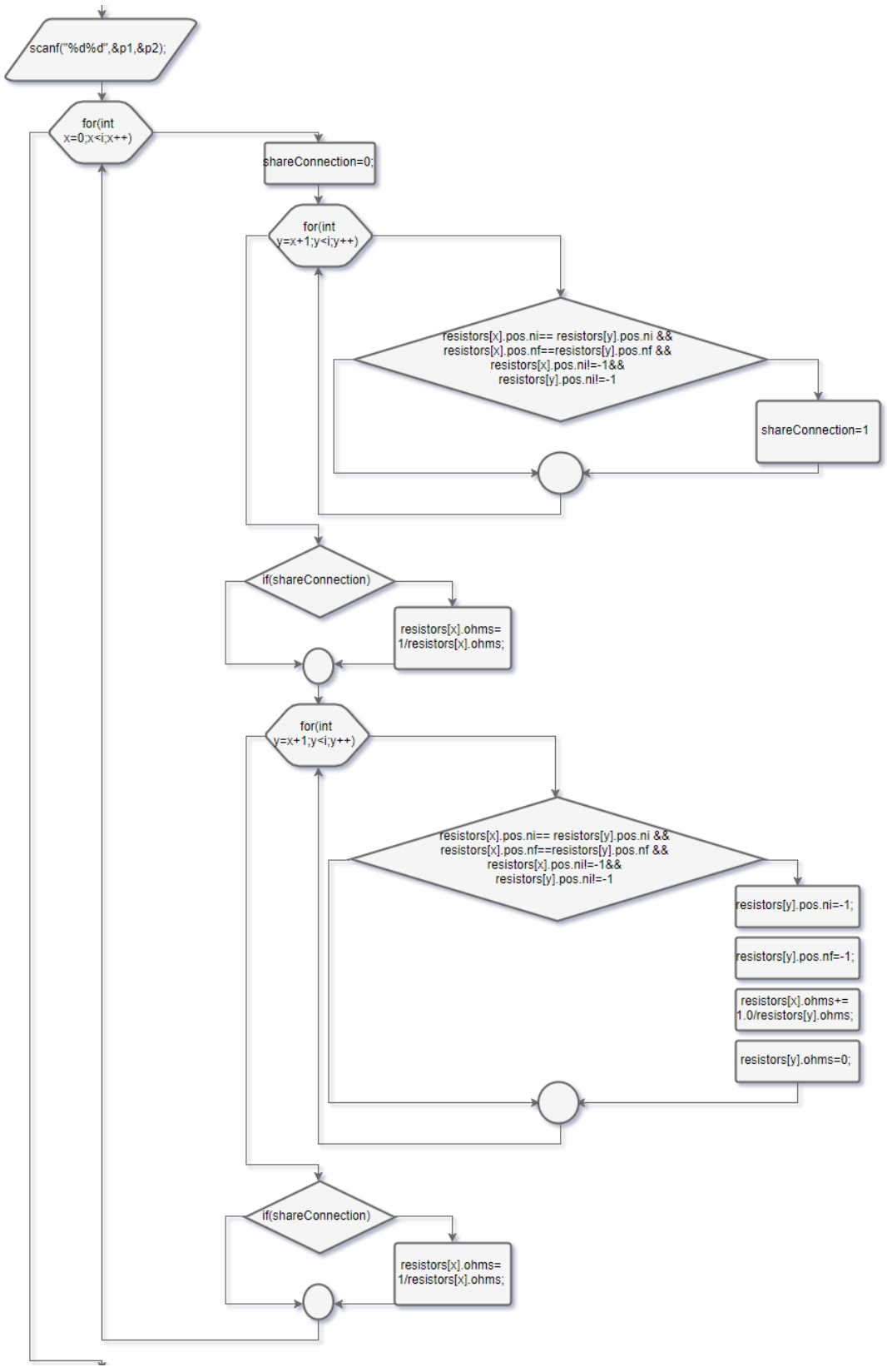
Como referencia el programa creado acepta entradas que siguen la estructura, se muestra el orden en la primera fila y los ejemplos en las filas siguientes:

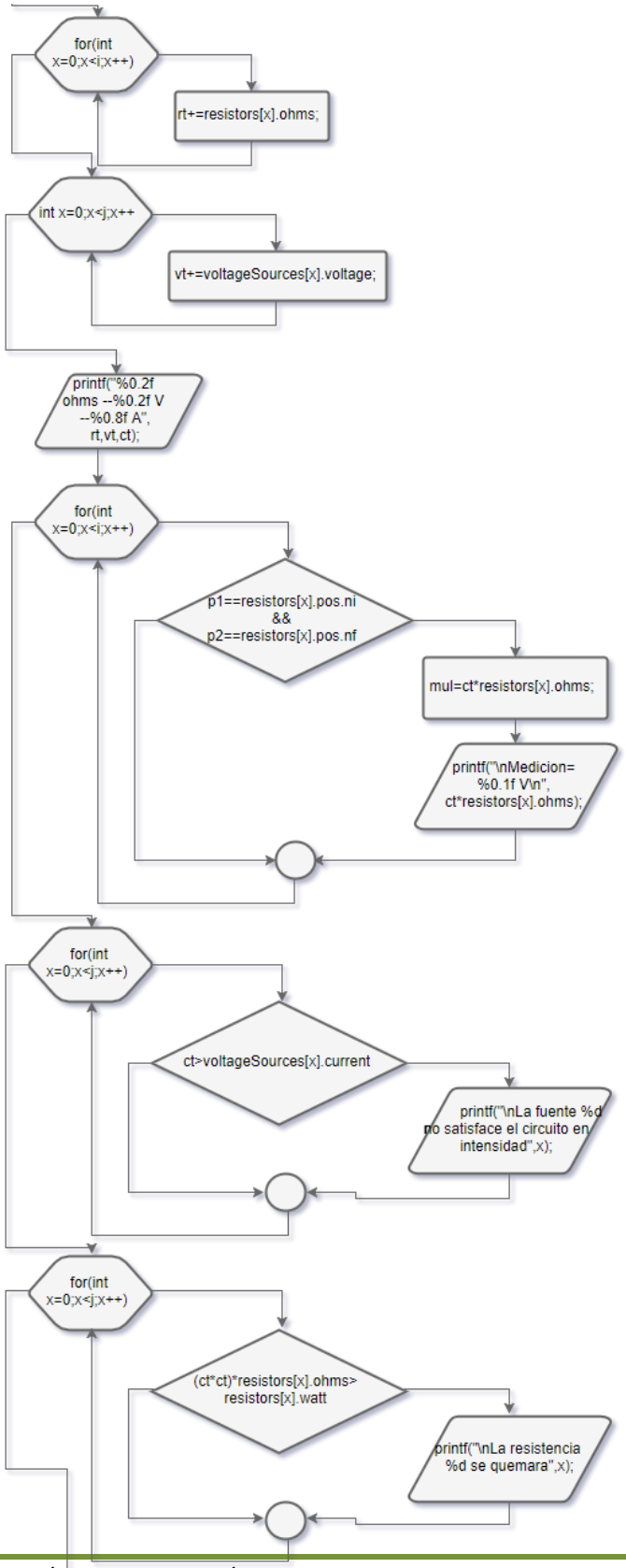
Código	Nodo inicial	Nodo final	Valor	Valor adicional (potencia, amperaje para fuentes)
Res	1	0	10e3	.25 A
Vol	2	3	10V	10ª
Mul	3	4		

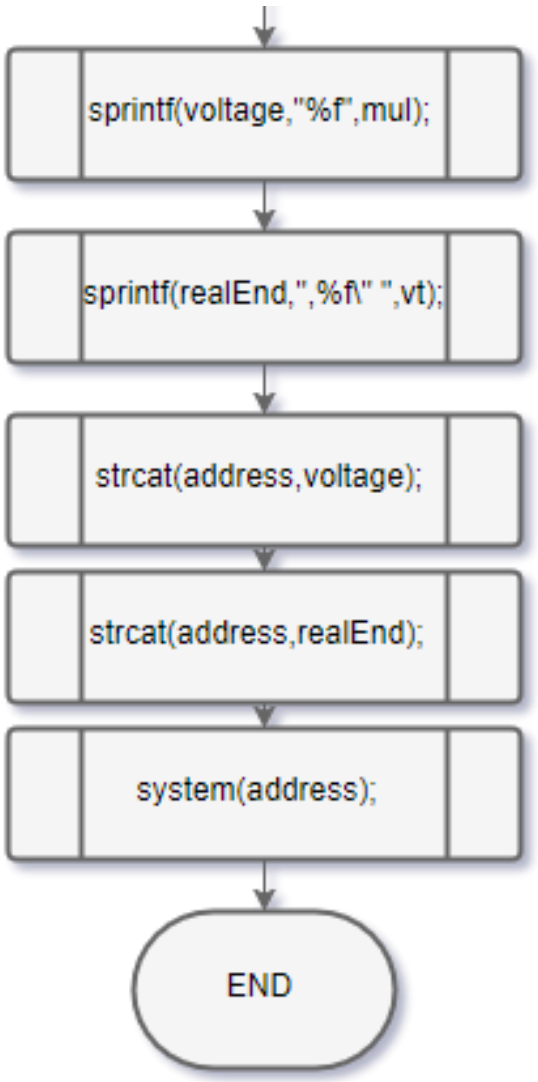
3 Desarrollo

A continuación, se mostrarán los diagramas de flujo de la práctica realizada


main()
float rt=0,vt=0,ct=0;
float mul;
int i=0,j=0,k=0,p1,p2,
shareConnection=0;
char address[100]=
"C:/gnuplot/bin/gnuplot.exe -p -e \"plot ";
char voltage[10];
char realEnd[20];
char filter[3]="Nul";
while(strcmp(filter,"mul")!=0)
scanf("%s",&filter);
strcmp(filter,"res")==0
strcmp(filter,"vol")==0
scanf("%d%d%f%f"
,&resistors[i].pos.ni,
&resistors[i].pos.nf,
&resistors[i].ohms,
&resistors[i].watt);
i++
scanf("%d%d%f%f",
&voltageSources[j].pos.ni,
&voltageSources[j].pos.nf,
&voltageSources[j].voltage,&
voltageSources[j].current);
i++


scanf("%d%d",&p1,&p2);
for(int x=0;x<i;x++)
shareConnection=0;
for(int y=x+1;y<i;y++)
resistors[x].pos.ni== resistors[y].pos.ni &&
resistors[x].pos.nf==resistors[y].pos.nf &&
resistors[x].pos.ni!=-1&&
resistors[y].pos.ni!=-1
shareConnection=1
if(shareConnection)
resistors[x].ohms=
1/resistors[x].ohms;
for(int y=x+1;y<i;y++)
resistors[x].pos.ni== resistors[y].pos.ni &&
resistors[x].pos.nf==resistors[y].pos.nf &&
resistors[x].pos.ni!=-1&&
resistors[y].pos.ni!=-1
resistors[y].pos.ni=-1;
resistors[y].pos.nf=-1;
resistors[x].ohms+=
1.0/resistors[y].ohms;
resistors[y].ohms=0;
if(shareConnection)
resistors[x].ohms=
1/resistors[x].ohms;


for(int x=0;x<i;x++)
rt+=resistors[x].ohms;
int x=0;x<j;x++
vt+=voltageSources[x].voltage;
printf("%0.2f ohms --%0.2f V --%0.8f A", rt,vt,ct);
for(int x=0;x<i;x++)
p1==resistors[x].pos.ni && p2==resistors[x].pos.nf
mul=ct*resistors[x].ohms;
printf("\nMedicion= %0.1f V\n", ct*resistors[x].ohms);
for(int x=0;x<j;x++)
ct>voltageSources[x].current
printf("\nLa fuente %d no satisface el circuito en intensidad",x);
for(int x=0;x<j;x++)
(ct*ct)*resistors[x].ohms> resistors[x].watt
printf("\nLa resistencia %d se quemara",x);


sprintf(voltage,"%f",mul);
sprintf(realEnd,",%f\" ",vt);
strcat(address,voltage);
strcat(address,realEnd);
system(address);
END

Estructura atrib		
Tipo de dato	Nombre	Uso
Entero	Ni	Nodo inicial
Entero	Nf	Nodo final
Estructura resistor		
Tipo de dato	Nombre	Uso
Float	Ohms	Valor de la resistencia
Float	Watt	Valor de la potencia
Struct atrib	Pos	Nodos de referencia para posición
Estructura voltageSource		
Tipo de dato	Nombre	Uso
Float	Voltaje	Voltaje de la fuente
Float	Current	Amperaje de la fuente
Struct atrib	pos	Nodos de referencia para posición

main.c

```
#include <stdio.h>
#include<math.h>
#include <stdio.h>
#include <string.h>
struct atrib{
    int ni;
    int nf;
};

struct resistor{
    float ohms;
    //float current;
    float watt;
    struct atrib pos;
}resistors[20];
```

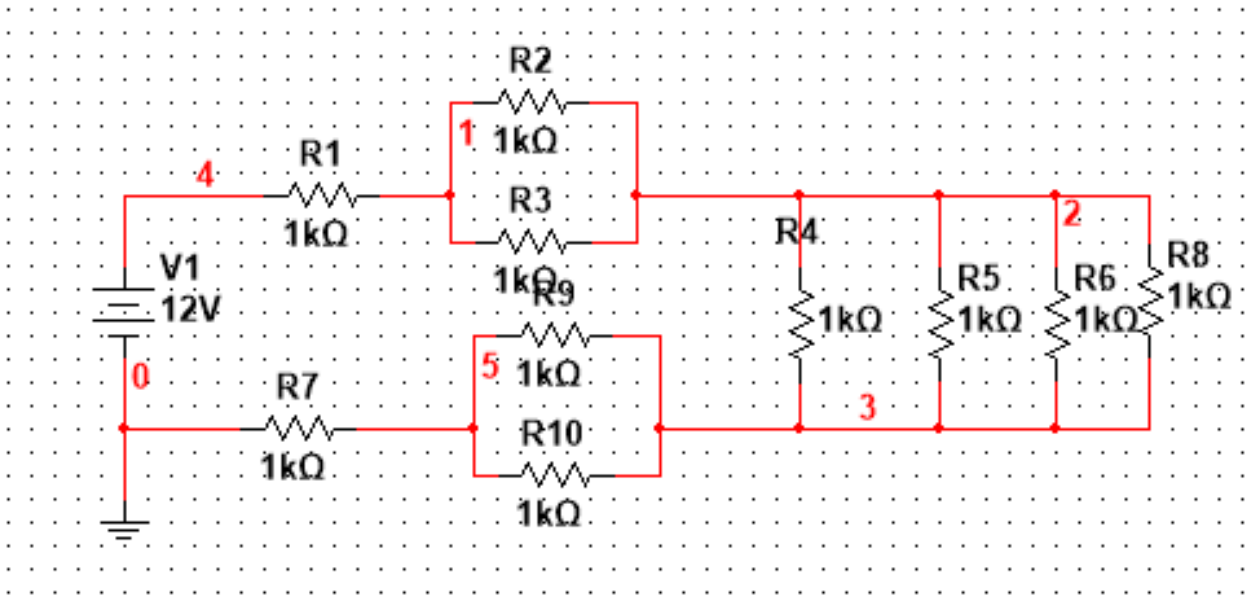
```
struct voltageSource{
    float voltage;
    float current;
    struct atrib pos;
}voltageSources[5];


/*void printResistors(struct resistor resistors[20])
{
    printf("I\t N1 \t N2 \t Value \tPow \n\n");
    for(int i=0;i<20;i++)
    {
        struct resistor r=resistors[i];
        printf("%d\t %d\t %d\t %.2f\t %.2f \n",i,r.pos.ni,r.pos.nf, r.ohms,r.watt);
    }
}*/
int main()
{
    int i=0,j=0,k=0,p1,p2;
    float rt=0,vt=0,ct=0;
    float mul;
    char address[100]="C:/gnuplot/bin/gnuplot.exe -p -e \"plot ";
    char voltage[10];
    char realEnd[20];
    char filter[3]="Nul";
    while(strcmp(filter,"mul")!=0)
    {
        scanf("%s",&filter);
        if(strcmp(filter,"res")==0)
        {

scanf("%d%d%f%f",&resistors[i].pos.ni,&resistors[i].pos.nf,&resistors[i].ohms,&resistor
s[i].watt);
            i++;
        }
        else if(strcmp(filter,"vol")==0)
        {

scanf("%d%d%f%f",&voltageSources[j].pos.ni,&voltageSources[j].pos.nf,&voltageSou
rces[j].voltage,&
                voltageSources[j].current);
            j++;
        }
    }
```

```
    scanf("%d%d",&p1,&p2);
    int shareConnection=0;
    for(int x=0;x<i;x++)
    {
        shareConnection=0;
        struct resistor raux=resistors[x];
        for(int y=x+1;y<i;y++)
        {
            if(resistors[x].pos.ni==                resistors[y].pos.ni                &&
resistors[x].pos.nf==resistors[y].pos.nf && resistors[x].pos.ni!=-1
              &&resistors[y].pos.ni!=-1)
              {
                  shareConnection=1;
              }
        }
        if(shareConnection) resistors[x].ohms=1/resistors[x].ohms;
        for(int y=x+1;y<i;y++)
        {
            if(resistors[x].pos.ni==                resistors[y].pos.ni                &&
resistors[x].pos.nf==resistors[y].pos.nf && resistors[x].pos.ni!=-1
              &&resistors[y].pos.ni!=-1)
            {
              //  printf("Resistencia en paralelo\n");
                resistors[y].pos.ni=-1;
                resistors[y].pos.nf=-1;
                resistors[x].ohms+=1.0/resistors[y].ohms;
                resistors[y].ohms=0;

            }
        }
        if(shareConnection)
        {
            resistors[x].ohms=1.0/resistors[x].ohms;

        }
    }
    for(int x=0;x<i;x++)
    {
        rt+=resistors[x].ohms;
    }
    for(int x=0;x<j;x++)
    {
        vt+=voltageSources[x].voltage;
    }
```

```
    ct=vt/rt;
    //printResistors(resistors);a
    printf("%0.2f ohms --%0.2f V --%0.8f A",rt,vt,ct);
    for(int x=0;x<i;x++)
    {
        if(p1==resistors[x].pos.ni&&p2==resistors[x].pos.nf)
        {
            mul=ct*resistors[x].ohms;
            printf("\nMedicion= %0.1f V\n",ct*resistors[x].ohms);
        }
    }
    for(int x=0;x<j;x++)
    {
        if(ct>voltageSources[x].current)
        {
            printf("\nLa fuente %d no satisface el circuito en intensidad",x);
        }
    }
    for(int x=0;x<i;x++)
    {

        if(   (ct*ct)*resistors[x].ohms>resistors[x].watt )
        {
            printf("\nLa resistencia %d se quemara",x);
        }
    }

    sprintf(voltage,"%f",mul);
    sprintf(realEnd,",%f\" ",vt);
    strcat(address,voltage);
    strcat(address,realEnd);
    printf("\n");
    puts(address);
    system(address);

    return 0;
}
```

4 Resultados

El programa se hicieron pruebas del programa con el siguiente circuito:



Resultados obtenidos

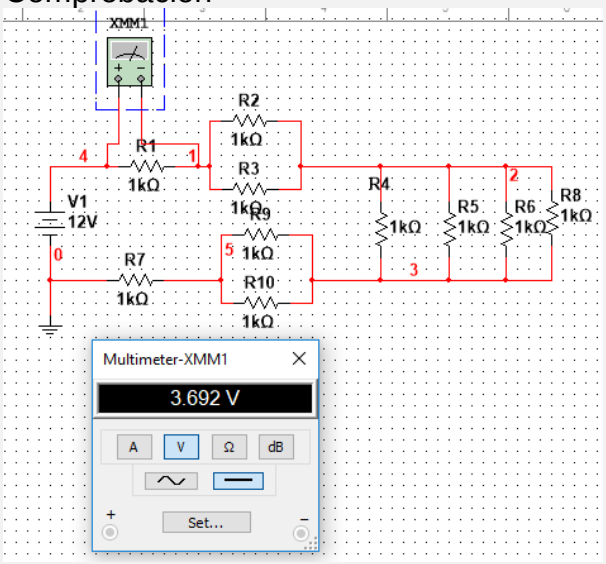
Entrada

```
res 4 1 1e3 .25
res 1 2 1e3 .25
res 1 2 1e3 .25
res 2 3 1e3 .25
res 2 3 1e3 .25
res 2 3 1e3 .25
res 2 3 1e3 .25
res 3 5 1e3 .25
res 3 5 1e3 .25
res 5 0 1e3 .25
vol 4 0 12 10
mul 4 1
```

Salida

```
3250.00 ohms --12.00 V --0.00369231 A
Medición= 3.7 V
```

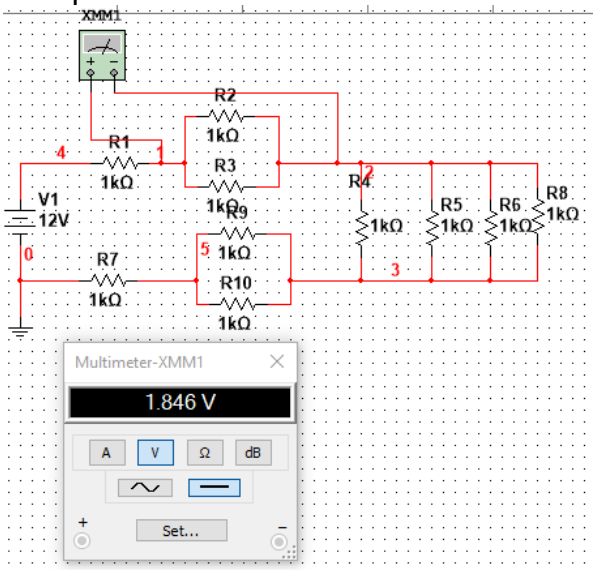
Comprobación



Entrada
res 4 1 1e3 .25
res 1 2 1e3 .25
res 1 2 1e3 .25
res 2 3 1e3 .25
res 2 3 1e3 .25
res 2 3 1e3 .25
res 2 3 1e3 .25
res 3 5 1e3 .25
res 3 5 1e3 .25
res 5 0 1e3 .25
vol 4 0 12 10
mul 1 2

Salida
3250.00 ohms --12.00 V --0.00369231 A
Medición= 1.8 V

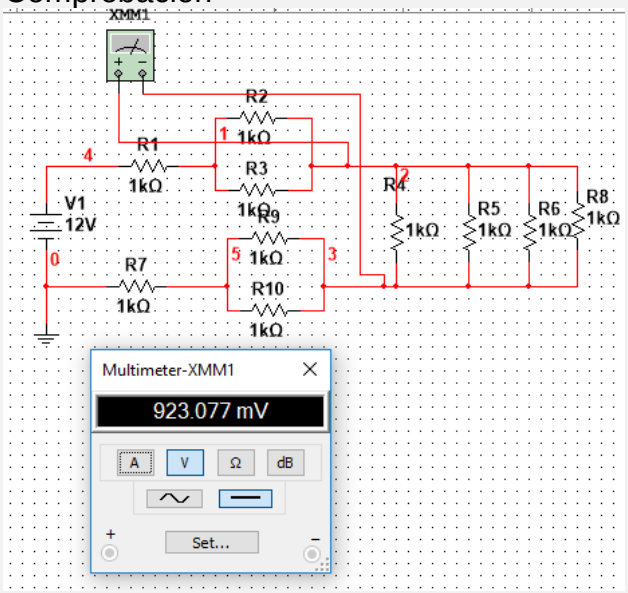
Comprobación

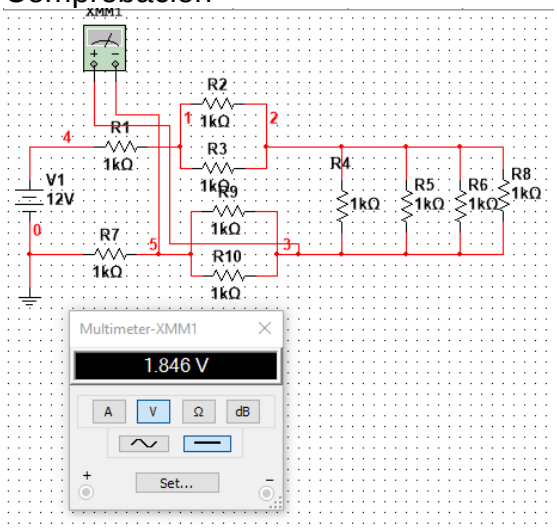
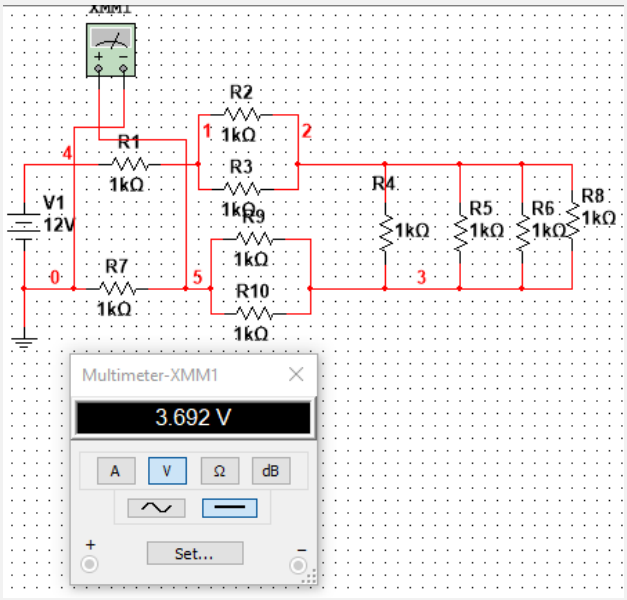


Entrada
res 4 1 1e3 .25
res 1 2 1e3 .25
res 1 2 1e3 .25
res 2 3 1e3 .25
res 2 3 1e3 .25
res 2 3 1e3 .25
res 2 3 1e3 .25
res 3 5 1e3 .25
res 3 5 1e3 .25
res 5 0 1e3 .25
vol 4 0 12 10
mul 2 3

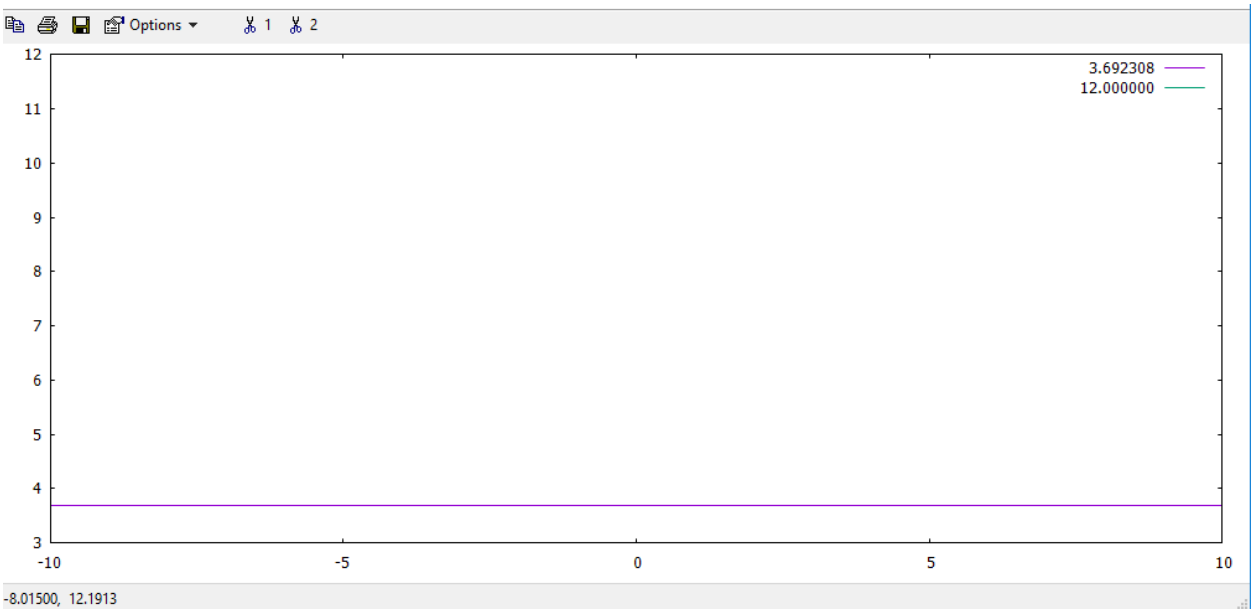
Salida
3250.00 ohms --12.00 V --0.00369231 A
Medición= 0.9 V

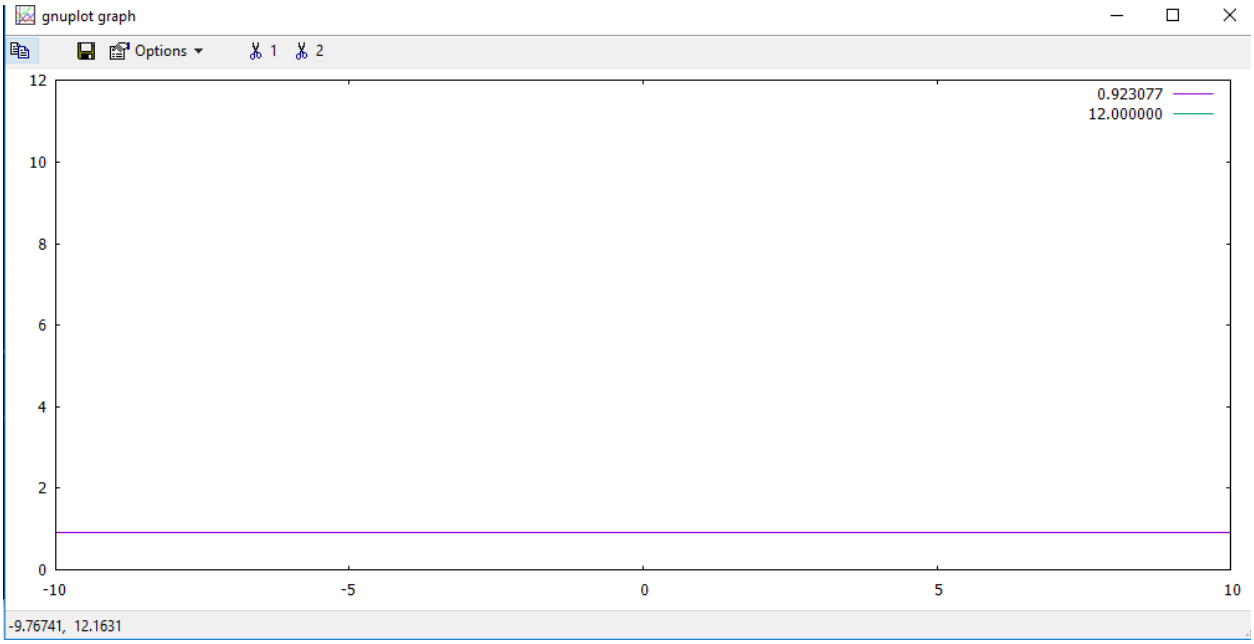
Comprobación



Entrada res 4 1 1e3 .25 res 1 2 1e3 .25 res 1 2 1e3 .25 res 2 3 1e3 .25 res 2 3 1e3 .25 res 2 3 1e3 .25 res 2 3 1e3 .25 res 3 5 1e3 .25 res 3 5 1e3 .25 res 5 0 1e3 .25 vol 4 0 12 10 mul 3 5	Salida 3250.00 ohms --12.00 V --0.00369231 A Medición= 1.8 V Comprobación 
Entrada res 4 1 1e3 .25 res 1 2 1e3 .25 res 1 2 1e3 .25 res 2 3 1e3 .25 res 2 3 1e3 .25 res 2 3 1e3 .25 res 2 3 1e3 .25 res 3 5 1e3 .25 res 3 5 1e3 .25 res 5 0 1e3 .25 vol 4 0 12 10 mul 5 0	Salida 3250.00 ohms --12.00 V --0.00369231 A Medición= 3.7 V Comprobación 

Ejemplos de graficas generadas por el programa





5 Conclusiones

El programa creado resuelve correctamente circuitos en serie, paralelos y mixtos además de indicar las posibles resistencias quemadas y las fuentes cuyo amperaje no satisface el circuito.

Adicionalmente el programa genera 2 graficas con los valores constantes de la medición y el voltaje total en el circuito, sin embargo, aún quedan algunos tipos de circuitos que el programa no es capaz de resolver, por ejemplo, circuitos con conexiones estrella-delta o circuitos resistencias en profundidad.

Por otra parte, el uso de estructuras anidadas permitió realizar comparaciones más simples entre los elementos y sus variables, lo que facilito la implementación de la reducción de resistencias.

También se introdujo el uso de la librería GNUPLOT para graficar los voltajes medidos, lo intuitivo de su interfaz y la posibilidad de crear una llamada desde C permitió añadir el complemento para graficar de manera sencilla, se espera que en las practicas posteriores se pueda usar esta herramienta de manera más profunda para visualizar datos.

Finalmente se concluye que la practica tuvo resultados satisfactorios de acuerdo con los parámetros de entrada establecidos, pero no se ignoran las areas de mejora, finalmente los conocimientos de la librería GNUPLOT serán de gran ayuda para practicas posteriores.

6 Referencias

Langsam, Y., Augenstein, M., & Tenenbaum, A. (1997). *Estructuras de datos con C y C++.* Mexico: Prentice Hall.

Marquez Fausto, T. G., Osorio Angel, S., & Olvera Perez, E. N. (2011). *Introduccion a la programacion estructurada en C.* Mexico: Prentice Hall.

Tenebaum, A., Langsam, Y., & Augenstein, M. (1993). *Estructuras de datos en C.* Mexico: Prentice Hall.