Circuit Analysis and Power Consumption's Cost

A Programming Project

A project presented for the course of
CS*204* - Engineering Programming *I*

Department of Electric Engineering
King Faisal University
January 17, 2022

Team Members

Student Name	ID
Name 1	999999
Name 2	999999
Name 3	999999

Contents

1 Abstract

For our Programming course final project, we coded a relatively large program to help Electrical Engineers do their most frequent calculations, in a fast and efficient manner.

This project helped us develop our team working and communication abilities. We also gained a huge experience in handling complexity of large programs with abstractions, as well as use abstractions created by other team members. we also improved our ability to write readable, fast, efficient and well-documented code, that is easy to use by others without needing to understand any of the inner code working or design.

2 Introduction

For this project, we implement a practical system used to help Electrical Engineers with frequently used calculations, in fast, efficient C code.

This Project helped us integrate and practice the concepts studied at this course, as well as other Electrical Engineering Courses, mainly Circuit I.

We are also an experienced a significant impact in our teamwork, as we divided the problem into abstract sub-problems, as well as use tool and functions created by others. This project aims to find the cost of power consumption in various ways and perform common circuit analysis.

First, we'll go over the most common issues that Electrical Engineers face. Second, we're looking for the most effective solutions to these issues. We'll also include the equations for each problem.

Finally, we will present our project's ultimate product.

3 Problem Statment

A huge amount of Electrical Engineers' time is wasted on analyzing circuits and calculating power usage and its cost. This includes conversions such as:

3.1 Calculating the component series and parallel:

1. Resistors in series
2. Inductor in series
3. Capacitors in series
4. Resistors in parallel
5. Inductor in parallel
6. Capacitors in parallel

3.2 Converting Components

1. Resistor Network Delta $\rightarrow$ Star
2. Inductor Network Delta $\rightarrow$ Star
3. Capacitor Network Delta $\rightarrow$ Star
4. Resistor Network Delta $\rightarrow$ Star
5. Inductor Network Delta $\rightarrow$ Star
6. Capacitor Network Delta $\rightarrow$ Star

Also, We found out that most engineers calculate the power cose using three different methods,

1. Voltage and Current

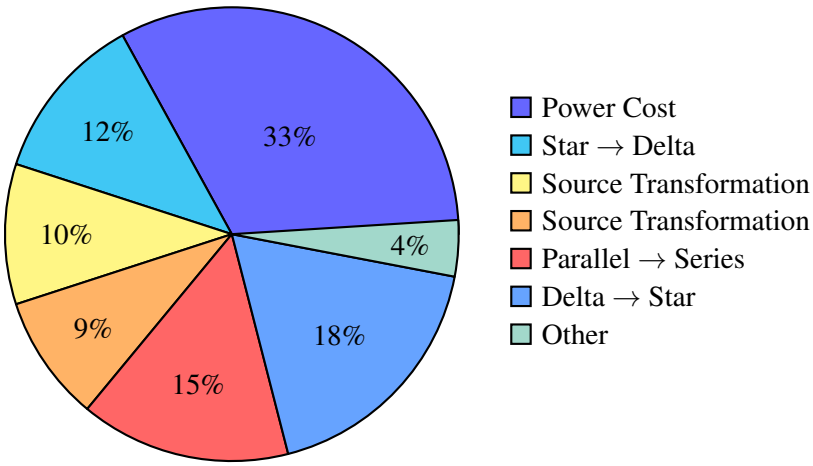


Figure 1: Most Performed Calculations

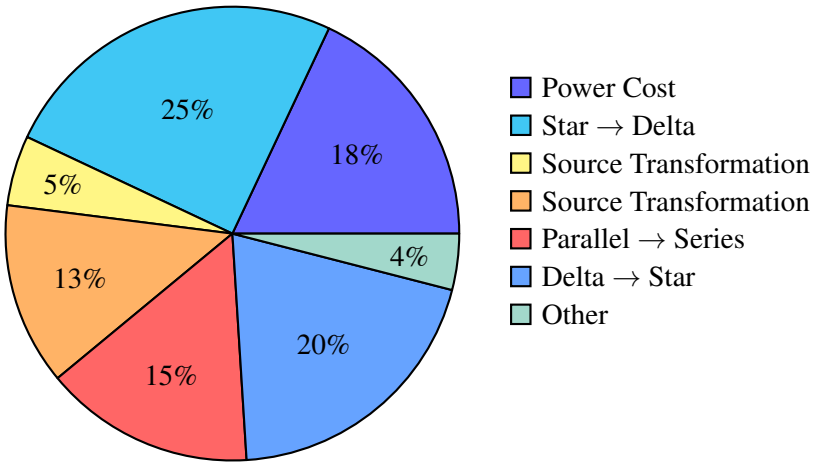


Figure 2: Hardest Calculations

2. Current and Resistance

3. Resistance and Voltage

It also calculates the monthly power bill for all of these methods with an optional custom Kilo watt per hour price.

Our C Program intends to efficient solve these problems easily to save Electrical Engineers' time.

4 Methodology

4.1 Collecting and Analyzing Data

Before choosing the calculations our program can perform, we the gathered data about what Electrical Engineers calculated often. The data was prepared and analyzed (Figure: 2 and Figure: 1), and the most often 4 calculations were added to the program.

From Figure: 1, it can be seen that the most performed calculation was Power Cost.

Another dataset was collected for the hardest calculations, but not necessarily the most often used, we ended up with 6 other calculations, which can be seen in Figure: 2. The most used calculations was: Delta → Star Transformation, and vice vera.

5 Circuit Theory, Calculations and Equations

5.1 Calculating Power

There are many ways to calculating power (and by extension, it's cost), according to the types input given. Our program intends to be used by a wide variety of engineers, regradless of the data type they work with, so we added all possible methods (to the extent of our knowledge) to calculate power.

5.1.1 Voltage and Current

Given Voltage and Current, instantaneous power can be found using Equation: 1 [1].

$$P = VI \tag{1}$$

Which is short form of Equation: 2

$$Power = Voltage \times Current \tag{2}$$

5.1.2 Current and Resistance

Given Current and Resistance, instantaneous power can be found using Equation: 3 [3].

$$P = I^2R \tag{3}$$

Which is short form of Equation: 4

$$Power = Current^2 \times Resistance \tag{4}$$

5.1.3 Resistance and Voltage

Given Voltage and Current, instantaneous power can be found using Equation: 5 [2].

$$P = \frac{V^2}{R} \tag{5}$$

Which is short form of Equation: 6

$$Power = \frac{Voltage^2}{Resistance} \tag{6}$$

5.2 Calculating Equivelent Value of Component in Series

5.2.1 Resistors in Series

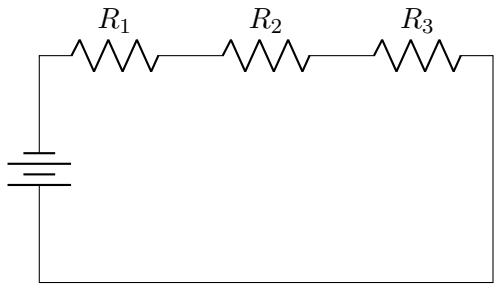


Figure 3: Resistors in series

$$R_{eq} = R_1 + R_2 + R_3 \tag{7}$$

To find the equivelent resistance of resistors in series (Figure: 3), add them together, as shown in Equation: 7.

5.2.2 Inductors in Series

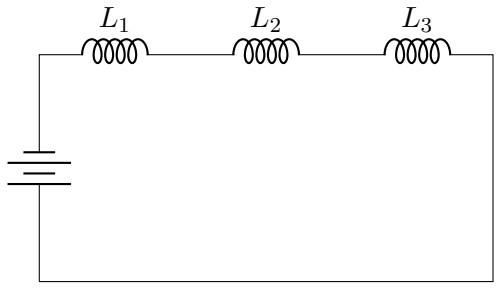


Figure 4: Inductors in series

$$L_{eq} = L_1 + L_2 + L_3 \tag{8}$$

To find the equivelent inductance of inductors in series (Figure: 4), add them together, as shown in Equation: 8.

5.2.3 Capacitors in Series

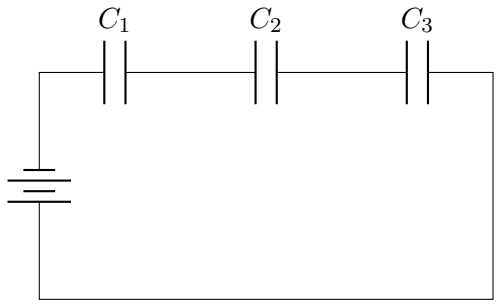


Figure 5: Capacitors in series

$$\frac{1}{C_{eq}} = \frac{1}{C_1} + \frac{1}{C_2} + \frac{1}{C_3} \tag{9}$$

Unlike Resistors and Inductors, Capacitors values don't simply add up.

To find the equivelent capacitance of capcitors in series Figure: 5, add their inverse together, and invert again, as showin in Equation: 9.

5.3 Calculating Equivelent Value of Component in Series

5.3.1 Resistors in Parallel

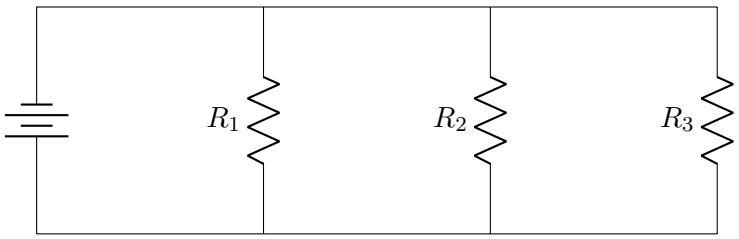


Figure 6: Resistors in Parallel

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} \tag{10}$$

To find the equivelent resistance of resistors in parallel Figure: 6, add their inverse together, and invert again, as showin in Equation: 10.

5.3.2 Inductors in Parallel

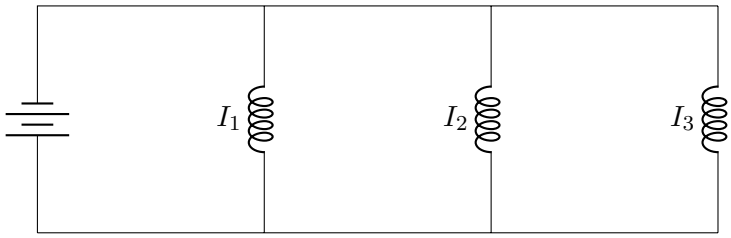


Figure 7: Inductors in Parallel

$$\frac{1}{L_{eq}} = \frac{1}{L_1} + \frac{1}{L_2} + \frac{1}{L_3} \tag{11}$$

To find the equivelent inductance of inductors in parallel Figure: 7, add their inverse together, and invert again, as showin in Equation: 11.

5.3.3 Capacitors in Parallel

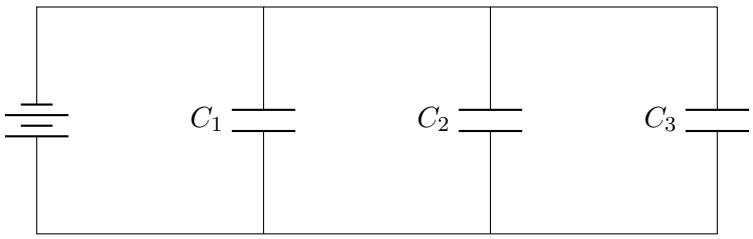


Figure 8: Capacitors in Parallel

$$C_{eq} = C_1 + C_2 + C_3 \tag{12}$$

To find the equivelent capacitance of capacitors in parallel Figure: 8, add their inverse together, and invert again, as showin in Equation: 12.

5.4 Converting Components Network

This was the hardest calculation as collected from our data, see Figure :2.

5.4.1 Resistors Delta → Star

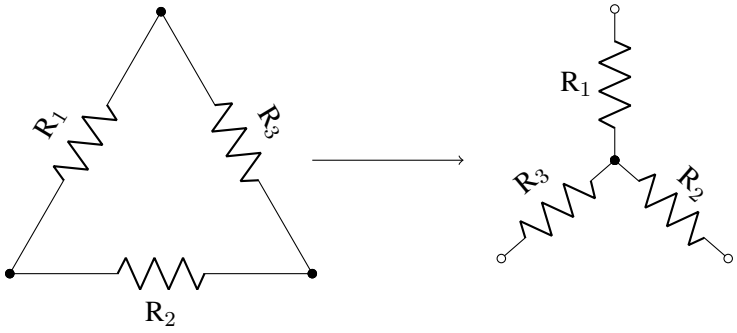


Figure 9: Resistor Delta → Star Transformation

$$R_1 = \frac{R_b \times R_c}{R_a + R_b + R_c} \tag{13}$$

$$R_2 = \frac{R_a \times R_c}{R_a + R_b + R_c} \tag{14}$$

$$R_3 = \frac{R_a \times R_b}{R_a + R_b + R_c} \tag{15}$$

The transformation from Delta → Star Netowrk of Resistors can be done using the equations above. Note that $R_a + R_b + R_c$ can be calculated once, which can improve speed and efficiency the program.

5.4.2 Inductors Delta → Star

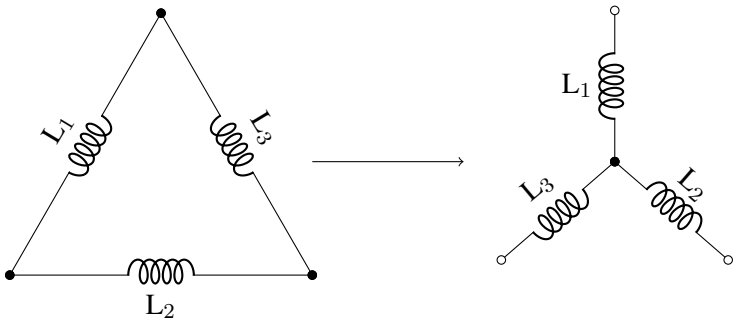


Figure 10: Inductor Delta → Star Transformation

$$L_1 = \frac{L_b \times L_c}{L_a + L_b + L_c} \tag{16}$$

$$L_2 = \frac{L_a \times L_c}{L_a + L_b + L_c} \tag{17}$$

$$L_3 = \frac{L_a \times L_b}{L_a + L_b + L_c} \tag{18}$$

The transformation from Delta → Star Netowrk of Inductors can be done using the equations above. Note that $L_a + L_b + L_c$ can be calculated once, which can improve speed and efficiency the program.

Also note that the same equation for Resistors can work for Inductors.

5.4.3 Capacitors Delta → Star

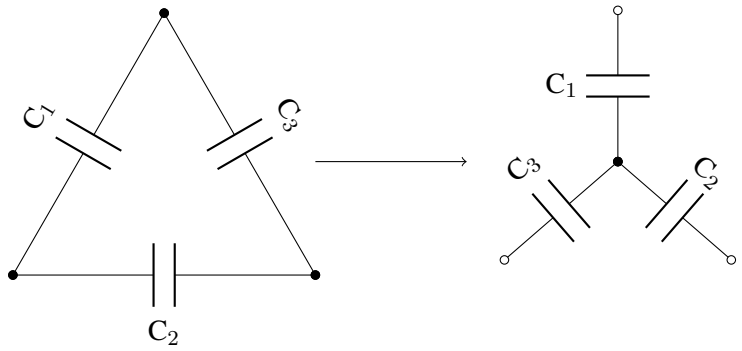


Figure 11: Capacitor Delta → Star Transformation

$$C_1 = \frac{C_a \times C_b + C_b \times C_c + C_c \times C_a}{C_a} \tag{19}$$

$$C_2 = \frac{C_a \times C_b + C_b \times C_c + C_c \times C_a}{C_b} \tag{20}$$

$$C_3 = \frac{C_a \times C_b + C_b \times C_c + C_c \times C_a}{C_c} \tag{21}$$

The transformation from Delta → Star Netowrk of Capacitors can be done using the equations above. Note that $C_a \times C_b + C_b \times C_c + C_c \times C_a$ can be calculated once, which can improve speed and efficiency the program.

Also note that this equation is very different from the equation used for Resistors and Inductors.

5.4.4 Resistors Star → Delta

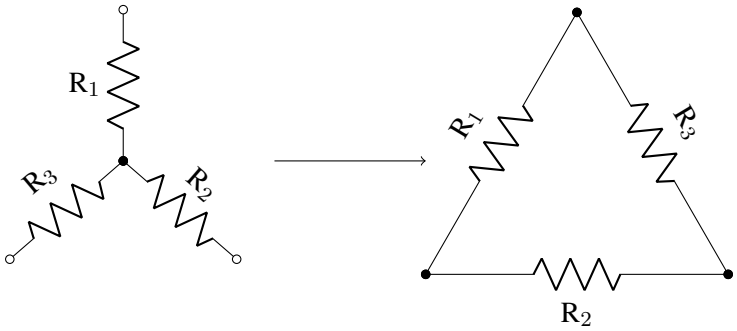


Figure 12: Resistor Star → Delta Transformation

$$R_1 = \frac{R_a \times R_b + R_b \times R_c + R_c \times R_a}{R_a} \tag{22}$$

$$R_2 = \frac{R_a \times R_b + R_b \times R_c + R_c \times R_a}{R_b} \tag{23}$$

$$R_3 = \frac{R_a \times R_b + R_b \times R_c + R_c \times R_a}{R_c} \tag{24}$$

The transformation from Star → Delta Netowrk of Resistors can be done using the equations above. Note that $R_a \times R_b + R_b \times R_c + R_c \times R_a$ can be calculated once, which can improve speed and efficiency the program.

5.4.5 Inductors Star → Delta

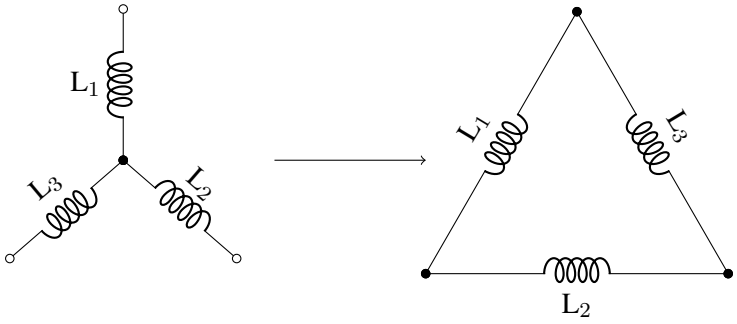


Figure 13: Inductor Star → Delta Transformation

$$L_1 = \frac{L_a \times L_b + L_b \times L_c + L_c \times L_a}{L_a} \tag{25}$$

$$L_2 = \frac{L_a \times L_b + L_b \times L_c + L_c \times L_a}{L_b} \tag{26}$$

$$L_3 = \frac{L_a \times L_b + L_b \times L_c + L_c \times L_a}{L_c} \tag{27}$$

The transformation from Star → Delta Netowrk of Inductors can be done using the equations above. Note that $L_a \times L_b + L_b \times L_c + L_c \times L_a$ can be calculated once, which can improve speed and efficiency the program.

5.4.6 Capacitors Star → Delta

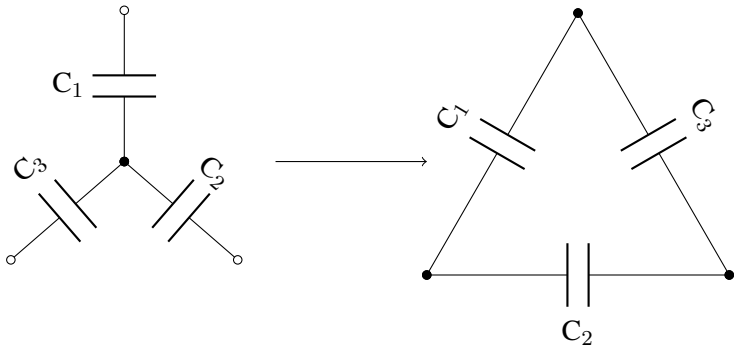


Figure 14: Capacitor Star → Delta Transformation

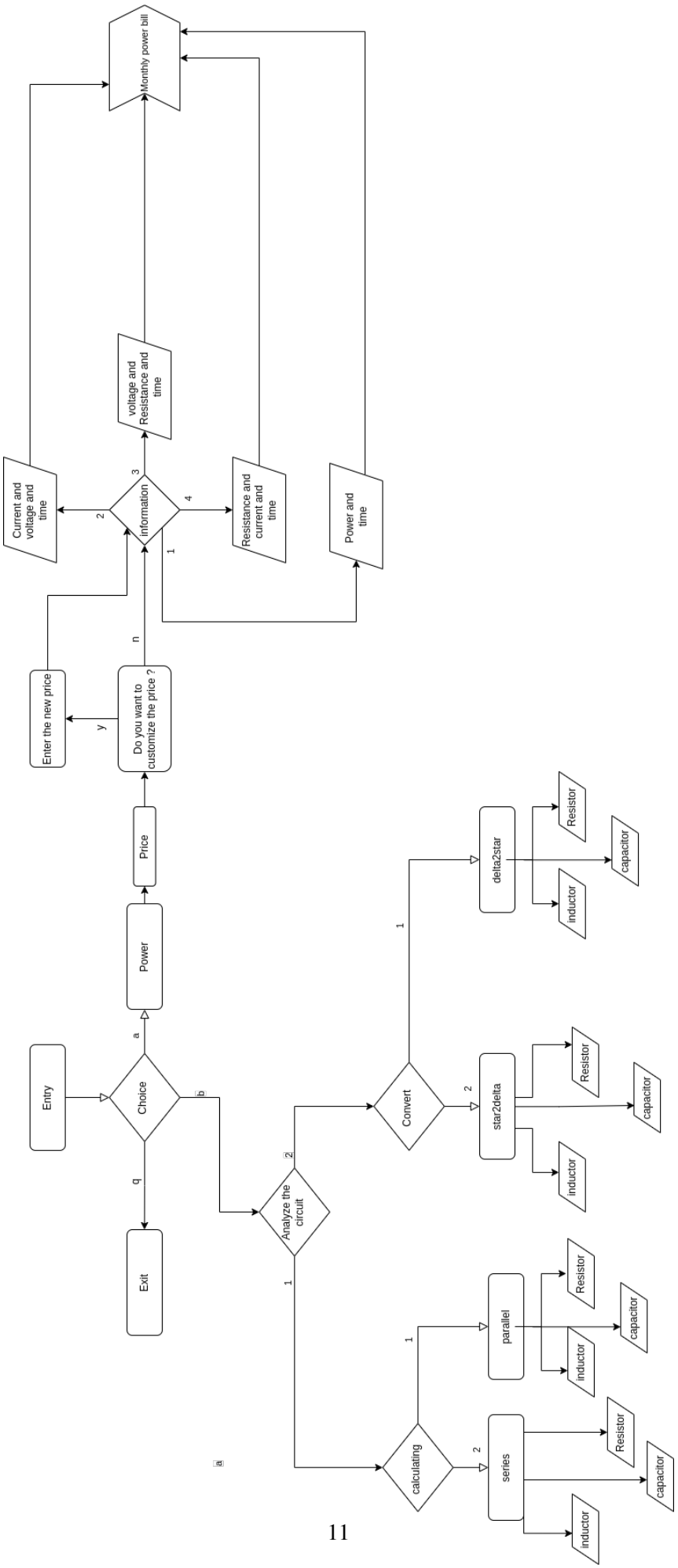
$$C_1 = \frac{C_b \times C_c}{C_a + C_b + C_c} \tag{28}$$

$$C_2 = \frac{C_a \times C_c}{C_a + C_b + C_c} \tag{29}$$

$$C_3 = \frac{C_a \times C_b}{C_a + C_b + C_c} \tag{30}$$

The transformation from Star → Delta Netowrk of Capacitors can be done using the equations above. Note that $C_a + C_b + C_c$ can be calculated once, which can improve speed and efficiency the program.

6 Code Structure and Design



7 Project's Code

```c
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

char scan_char();
void power_choice();
void analyze_circuit_choice();
double scan_number();
double power_Kwh();
double current_voltage();
double Voltage_Resitance();
double Resistance_Current();
double Convert_components();
double Calculating_components();
double series();
double parallel();
void delta2star(double a, double b, double c, const char *unit);
void star2delta(double a, double b, double c, const char *unit);

int main() {
MAIN_CHOICE:
  printf("Enter your choice to run the program.\n");
  printf("a) Calculate Monthly Power bill.\n");
  printf("b) Analyze the circuit.\n");
  printf("q) Terminate the program.\n");

  int choice = scan_char();

  switch (choice) {
    case 'a':
      power_choice();
      break;
    case 'b':
      analyze_circuit_choice();
      break;
    case 'q':
      exit(0);
      break;
    default:
      printf("Invalid\n");
      goto MAIN_CHOICE;
  }
  printf("\nDo you wish to continue? y Note:y=yes\n                                    "
      "Note:Press any key to exit.\n");
  if (scan_char() != 'y')
    exit(0);
  system("cls");
  goto MAIN_CHOICE;
}

/*
 * `power_choice`: Gives the user the price of their energy usage.
```

```
 *                A choice for an optional price per KWh is given.
 *
 * This function takes zero arguments
 *
 * It can call only ONE of these functions at a time.
 * The function choosed is acording the inputs the user has.
 * - power_Kwh
 * - current_voltage
 * - voltage_resistance
 * - Resistance_current
 */

void power_choice() {
  double price = 0.18, energy = 0;
  char c;
  printf("\nThe Price is 0.18SAR per Kwh\n");
  printf("Do you want to customize the price ? y/n     Note: y=yes n=no\n");

PRICE_CHOICE:
  c = scan_char();
  if (c == 'y') {
    price = scan_number("\nEnter the new price");
  } else if (c != 'n') {
    printf("Invalid Choice!\n");
    goto PRICE_CHOICE;
  }

POWER_CHOICE:
  printf("_____________________________________________________\n");
  printf("What information do you have?\n");
  printf("1) House Power Consumption in watts and hours\n");
  printf("2) Current and Voltage and hours\n");
  printf("3) Voltage and Resitance and hours\n");
  printf("4) Resistance and Current and hours\n");

  switch (scan_char()) {
    case '1':
      energy = power_Kwh();
      break;
    case '2':
      energy = current_voltage();
      break;
    case '3':
      energy = Voltage_Resitance();
      break;
    case '4':
      energy = Resistance_Current();
      break;
    default:
      printf("Invalid choice\n");
      goto POWER_CHOICE;
  }
  printf("The price for the monthly bill is [%.2lf ]SAR\n",
```

```
      price * energy * 30);
  printf("_____________________________________________________\n");
}

/*
 *`analyze_circuit_choice`: Allows the user to perform a circuit analysis
 *                          of their choosing.
 *
 * This function takes zero arguments
 *
 * It can call only ONE of these functions at a time.
 * The function choosed is acording the inputs the user has.
 Convert
 * - delta2star
 * - star2delta
 calculat
 * -series
 * -parallel
 */

void analyze_circuit_choice() {

  int choice;
  printf("\nEnter your choice here:\n");
  printf("1)Calculating components series or parallel.\n2)Convert the "
      "components from Delta -> Star or Star -> Delta .\nChoice>");

  scanf("%d", &choice);

  switch (choice) {
    case 1:
      Calculating_components();
      break;

    case 2:
      Convert_components();
      break;
  }
}

/**
 * `delta2star`: Calculates the delta -> star transformation for any
 *               given component.
 *
 * This function takes 4 arguments
 * - `a`: 1st Component
 * - `b`: 2nd Component
 * - `c`: 3rd Component
 * - `unit`: string used in the printf
 *
 * This function doesn't call any other function. (apart from printf)
 */
```

```c
void delta2star(double a, double b, double c, const char *unit) {
  double sum = a + b + c;
  double r1 = b * c / sum;
  double r2 = a * c / sum;
  double r3 = a * b / sum;

  printf("A = %lf %s \n", r1, unit);
  printf("B = %lf %s \n", r2, unit);
  printf("C = %lf %s \n", r3, unit);
}

/**
 * `star2delta`: Calculates the star -> delta transformation for any
 *               given component.
 *
 * This function takes 4 arguments
 * - `a`: 1st Component
 * - `b`: 2nd Component
 * - `c`: 3rd Component
 * - `unit`: string used in the printf
 *
 * This function doesn't call any other function. (apart from printf)
 */
void star2delta(double a, double b, double c, const char *unit) {
  double product = a * b + b * c + a * c;
  double r1 = product / a;
  double r2 = product / b;
  double r3 = product / c;

  printf("A = %lf %s \n", r1, unit);
  printf("B = %lf %s \n", r2, unit);
  printf("C = %lf %s \n", r3, unit);
}


/**
 * `scan_number`: Scans for a number with a custom prompt and returns it.
 *
 * This function takes one argument.
 * - `prompt`: the string used to prompt the user.
 *
 */
double scan_number(const char *prompt) {
  double a;
  printf("%s >", prompt);
  scanf(" %lf", &a);
  return a;
}


/**
 * `scan_char`: Scans for a character with a custom prompt
 *              and returns it.
```

```
 *
 * This function takes zero arguments.
 */
char scan_char() {
  char a;
  printf("Choice >");
  scanf(" %c", &a);
  return a;
}

/**
 * `power_Kwh`: returns the Energy after prompting the user for:
 * - power
 * - time
 *
 * This function takes zero arguments.
 */
double power_Kwh() {
  double power = scan_number("\nPower in w");
  double time = scan_number("Time in hours");
  double Energy = (power * time) / 1000;
  printf("\nThe power counsumed per day is    [%.2lf]Kwh \n", Energy);
  return Energy;
}

/**
 * `current_voltage`: returns the Energy after prompting the user for:
 * - current
 * - voltage
 * - time
 *
 * This function takes zero arguments.
 */
double current_voltage() {
  double current = scan_number("\nCurrent in Ampere");
  double voltage = scan_number("Voltage in Volts");
  double time = scan_number("Time in hours");
  double Energy = (current * voltage * time) / 1000;
  printf("\nThe power counsumed per day is    [%.2lf]Kwh \n", Energy);
  return Energy;
}

/**
 * `voltage_resitance`: returns the Energy after prompting the user for:
 * - voltage
 * - resistance
 * - time
 *
 * This function takes zero arguments.
 */
double Voltage_Resitance() {
  double Resitance = scan_number("\nResitance in ohm");
  double voltage = scan_number("Voltage in Volts");
```

```
  double time = scan_number("Time in hours");
  double Energy = ((powf(voltage, 2) / Resitance) * time) / 1000;
  printf("\nThe power counsumed per day is    [%.2lf]Kwh \n", Energy);
  return Energy;
}

/**
 * `resistance_current`: returns the Energy after prompting the user for:
 * - resistance
 * - current
 * - time
 *
 * This function takes zero arguments.
 */
double Resistance_Current() {
  double Resitance = scan_number("\nResitance in ohm");
  double current = scan_number("Current in Ampere");
  double time = scan_number("Time in hours");
  double Energy = ((powf(current, 2) / Resitance) * time) / 1000;
  printf("\nThe power counsumed per day is    [%.2lf]Kwh \n", Energy);
  return Energy;
}

/*`Convert components`: returns the the new value of the component after
 *prompting the user for: Component A Component B Component C
 */
double Convert_components() {
ANALYZE_CIRCUIT_CHOICE:
  printf("_____________________________________________________\n");
  printf("1) Convert Resistor  Network Delta -> Star.\n");
  printf("2) Convert Capacitor Network Delta -> Star.\n");
  printf("3) Convert Inductor  Network Delta -> Star.\n");
  printf("4) Convert Resistor  Network Star  -> Delta.\n");
  printf("5) Convert Capacitor Network Star  -> Delta.\n");
  printf("6) Convert Inductor  Network Star  -> Delta.\n");
  printf("\n");
  char choice = scan_char();
  printf("_____________________________________________________\n");
  printf("Note:Enter the value of components.\n");
  double a = scan_number("Component A");
  double b = scan_number("Component B");
  double c = scan_number("Component C");

  switch (choice) {
    case '1':
      delta2star(a, b, c, "Ohm");
      break;
    case '2':
      star2delta(a, b, c, "Farad");
      break;
    case '3':
      delta2star(a, b, c, "Henry");
      break;
```

```
    case '4':
      star2delta(a, b, c, "Ohm");
      break;
    case '5':
      delta2star(a, b, c, "Farad");
      break;
    case '6':
      star2delta(a, b, c, "Henry");
      break;
    default:
      printf("Invalid chice\n");
      goto ANALYZE_CIRCUIT_CHOICE;
  }
}

/*`Calculating components`: returns the the equevlant componernt after
  prompting the user for:

 *Component A
 *Component B
 *Component C
 Depend on your choice resister|inductor|capactor.
 */
double Calculating_components() {
ANALYZE_CIRCUIT_CHOICE:
  printf("_____________________________________________\n");
  printf("1) Calculate Resistor in series.\n");
  printf("2) Calculate Capacitor in series.\n");
  printf("3) Calculate Inductor  in series.\n");
  printf("4) Calculate Resistor  in parallel.\n");
  printf("5) Calculate Capacitor in parallel.\n");
  printf("6) Calculate Inductor  in parallel.\n");
  printf("\n");
  char choice = scan_char();
  printf("_____________________________________________\n");
  printf("Note:Enter the value of components.\n");
  double a = scan_number("Component A");
  double b = scan_number("Component B");
  double c = scan_number("Component C");

  switch (choice) {
    case '1':
      series(a, b, c, "Ohm");
      break;
    case '2':
      parallel(a, b, c, "Farad");
      break;
    case '3':
      series(a, b, c, "Henry");
      break;
    case '4':
      parallel(a, b, c, "Ohm");
      break;
```

```
    case '5':
      series(a, b, c, "Farad");
      break;
    case '6':
      parallel(a, b, c, "Henry");
      break;
    default:
      printf("Invalid chice\n");
      goto ANALYZE_CIRCUIT_CHOICE;
  }
}

/* It is sum the component a,b and c then, return it to the main function with
 * the result  */
double series(double a, double b, double c) {
  double sum;
  sum = a + b + c;
  printf("%.2lf", sum);
  return sum;
}

/* It is sum the inverse for each component a,b and c then, return it to the
 * main function with the result  */
double parallel(double a, double b, double c) {
  double sum;
  sum = 1 / ((1 / a) + (1 / b) + (1 / c));
  printf("%.2lf", sum);
  return sum;
}
```

8 Results and Outputs

8.1 Calculation Examples

Figure: 15 shows an example of calculating Power Consumption and its associated Energy Cost with *custom* price, given Power and Time.

Figure: 16 shows an example of calculating equivalent of resistors in series. *default* price, given Kwh.

Figure: 17 shows an example of Delta $\rightarrow$ Star conversion of a resistor.

8.2 Invalid Input handling

We made sure our program handles all possible invalid inputs, and provide helpful message to its users. Figure: 18 shows an example of an invalid input when making a calculation choice.

Figure: 19 shows an example of an invalid price choice

Figure: 20 shows the program prompting the user back when facing an invalid input, instead of crashing or ignoring it.

8.3 Terminating the program

Figure: 21 shows the program allowing the user to terminate it when done.

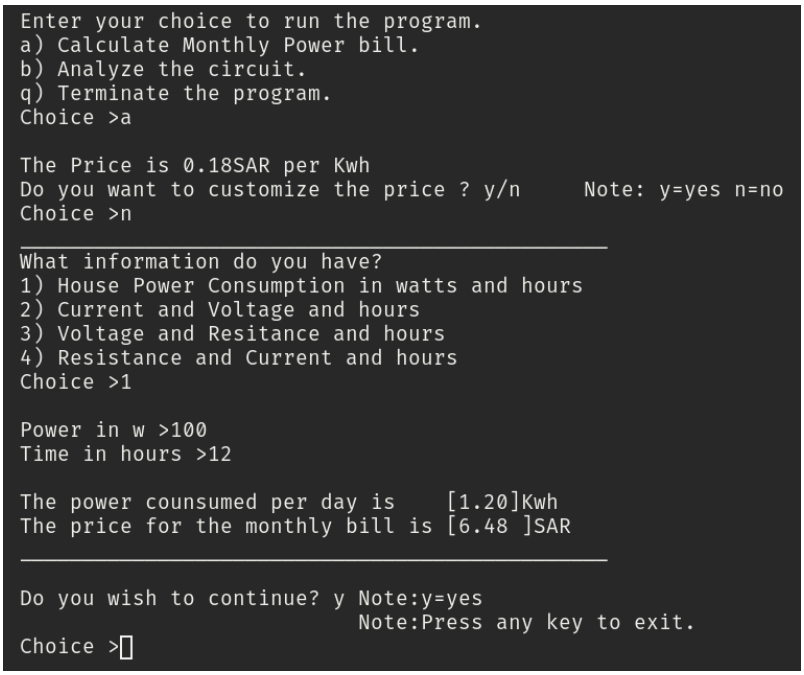


Figure 15: Power Consumption's Price given Power and Time

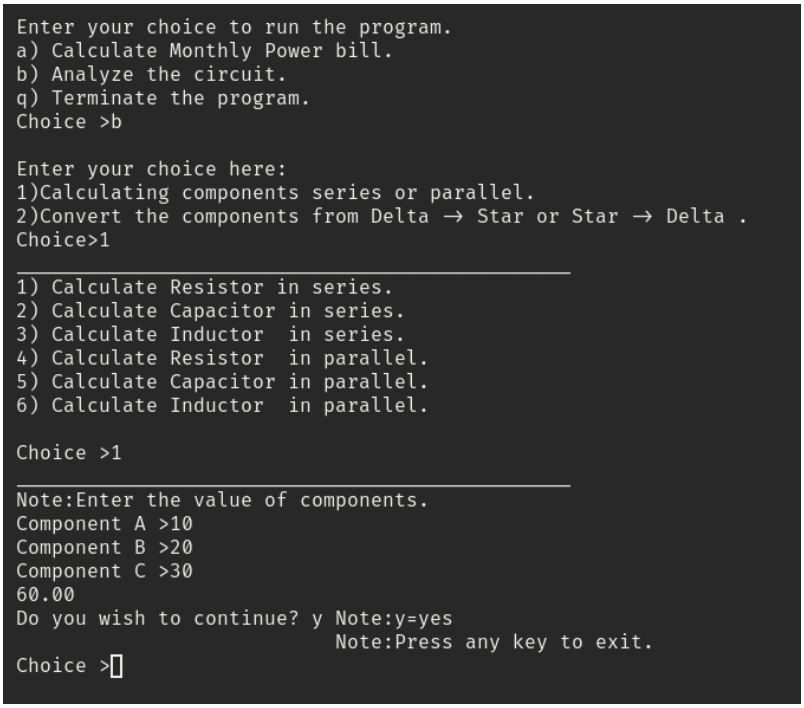


Figure 16: Equivalent values of resistors in series

```
Enter your choice to run the program.
a) Calculate Monthly Power bill.
b) Analyze the circuit.
q) Terminate the program.
Choice >b

Enter your choice here:
1)Calculating components series or parallel.
2)Convert the components from Delta → Star or Star → Delta .
Choice>2
_____________________________________________
1) Convert Resistor  Network Delta → Star.
2) Convert Capacitor Network Delta → Star.
3) Convert Inductor  Network Delta → Star.
4) Convert Resistor  Network Star  → Delta.
5) Convert Capacitor Network Star  → Delta.
6) Convert Inductor  Network Star  → Delta.

Choice >1
_____________________________________________
Note:Enter the value of components.
Component A >90
Component B >12
Component C >34
A = 3.000000 Ohm
B = 22.500000 Ohm
C = 7.941176 Ohm

Do you wish to continue? y Note:y=yes
                           Note:Press any key to exit.
Choice >
```

Figure 17: Delta → Star Conversion of a Resistor network

```
Enter your choice to run the program.
a) Calculate Monthly Power bill.
b) Analyze the circuit.
q) Terminate the program.
Choice >d
Invalid
Enter your choice to run the program.
a) Calculate Monthly Power bill.
b) Analyze the circuit.
q) Terminate the program.
Choice >
```

Figure 18: Invalid Selection

```
Enter your choice to run the program.
a) Calculate Monthly Power bill.
b) Analyze the circuit.
q) Terminate the program.
Choice >a

The Price is 0.18SAR per Kwh
Do you want to customize the price ? y/n     Note: y=yes n=no
Choice >aa
Invalid Choice!
Choice >Invalid Choice!
Choice >
```

Figure 19: Only two options allowed, y/n

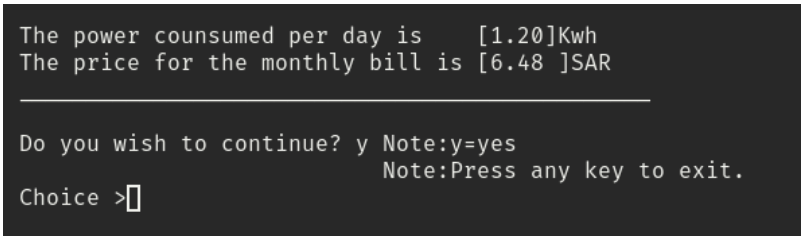


Figure 20: Prompting back the user

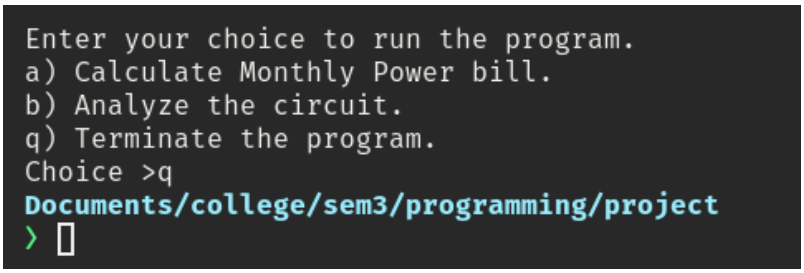


Figure 21: Terminating the program

List of Figures

References

[1] Matthew Sadiku Charles Alexander. *Fundamentals of Electric Circuits*. McGraw-Hill, 2003.

[2] Paul Horowitz and Winfield Hill. *The art of electronics*. 3rd ed. Cambridge, England: Cambridge University Press, Mar. 2015.

[3] Paul Scherz and Simon. *Practical electronics for inventors, fourth edition*. 4th ed. Columbus, OH: McGraw-Hill Education, Mar. 2016.