

**MATLAB**

**Student Name :**

**Student ID no :**

**Intake :**

**Module code : EE038-3-2-ESA**

**Lecturer Name : Dr Lau Chee Yong**

**Submission Date :**

**ACKNOWLEDGMENTS**

I would like to express my deepest appreciation to our module lecturer, who never hesitated to offer a helping hand when we needed it. Apart from that, I would want to offer my deepest appreciation to my friends and classmates for their aid and encouragement during this process, which has been beneficial..

**Table of contents**

**Contents Page no**

Acknowledgements ii

List of figures iv

Introduction 1 Design and Flow of routine 2

Format and Style of routine 6

Results and analysis 9

Discussion 12

Conclusion 13

References 14

Appendix 15

**List of Figures:-**

**Figure**   **Page No**

Figure 1 :- Half wave rectifier waveform…………………………………..1

Figure 2:- Full wave rectifier waverform………………………………..2

Figure 3:- Flow char of overall process …,,…………………………....4

Figure 4:- All functions…………..……………………………………...5

Figure 5:- Program GUI…………………………………………………9

Figure 6:- Full wave rectifier output…………………………..………...9

Figure 7:- Half wave rectifier output……………………………...……10

Figure 8:- Half wave rectifier output with capacitor filter on ………….10

Figure 9:- Error dialog…………………………………………………...11

**INTRODUCTION**

A rectifier is a device that converts alternating or oscillating current into a unidirectional or near direct current in one direction or the other. It exhibits a low resistance to current flow in one direction and a virtually infinite resistance to current flow in the other. On the most basic level, it transforms alternating voltage to direct current. Rectifiers can be molded into a variety of forms depending on their use, including semiconductor diodes, SCRs (silicon-controlled rectifiers), vacuum tube diodes, mercury-arc valves, and other similar devices. Diode rectifier circuits are widely used in electronic circuit design for signal discovery and power rectification. These circuits are found in a variety of devices such as radio signals or detectors, DC power supplies, and household appliances such as video game systems, laptop computers, televisions, and so on. For this assignment we will focus on half wave rectifier and full wave rectifiers.

When an alternating current (AC) supply is provided at the input, only the positive half cycle is visible across the load, while the negative half cycle is obscured by the rectifier, this process is known as half-wave rectifier. It is necessary to use a single diode in a single-phase supply, whereas three diodes are required in a three-phase supply. Because only half of the i/p waveforms reach the output, it is not capable of doing so. More filtering is necessary in the half-wave rectifier circuit to decrease the ripples of the alternating current frequency coming from the output. Please see the following link for additional information on the Half-wave Rectifier Circuit's Working Principle and Characteristics.

Diagram

Description automatically generated

Figure 1: Half wave rectifier waveform.

(Gao, 2015)

When an alternating current source is given to the i/p of the rectifier, the current flowing through the load flows in the same direction for both half cycles. When both polarities of the i/p waveform are changed to pulsating DC, this circuit produces a greater standard output voltage than before. This type of rectification may be performed by utilizing at the very least two crystal diodes, each of which conducts current in a separate direction.

Diagram

Description automatically generated

Figure 2: Full wave rectifier waveform.

The programming language used for the assignment is MATLAB is a high-performance programming language designed for technical computer applications. Computing, visualization, and programming are integrated in a simple-to-use environment where problems and answers are stated in mathematical notation that users are acquainted with. There are multiple ways to develop a Graphical User Interface program with MATLAB such as GUIDE, APP designer and Simulink. For this assignment we are using APP Designer. It is possible to create interactive applications using the MATLAB App Designer function, which is a part of the MATLAB program. App designer makes the GUI building easier by incorporating the drag and drop type of work, and creating auto functions for them, hence the user just need to implement the logic for their program rather than separate coding for GUI.

**DESIGN AND FLOW OF ROUTINES**

An algorithm was developed based on the requirements of the program; the algorithm was then implemented to flowchart. The flow chart shows the overall process of the MATLAB.

Figure3 shows the flow chart of the overall process, all the global parameters required for the input and output are declared in the properties block. The objects in the properties are initialized to have an initial state of the values necessary for the operation. There are several callback functions included in the program for voltage, frequency, resistor, capacitance, plot etc. These functions all respond to the main function. Hence the flow chart includes mainly of the main function. The script inside the main functions works only when the users turn on the switch from the GUI. Once the switch is on based on the preset variables for the input parameters the program plots the input signal and perform the necessary calculations required such as initializing the time variables.

For loop is initialized to work for provided number of sample for each time derivative voltage calculation. When the number of samples exceeds the executions stop there, else the program keeps the executions on. There are two different nested loops for the execution inside the for loop one for when the user selects the calculations to be performed with capacitor and one without the filter capacitor. If the user selects the smoothing capacitors operations. The programs then check for the type of rectifier selected, if the halfwave rectifier is selected then the parameters required for the output of the voltage calculation as per formula is computed and the required output is plotted. While the similar operation is performed if the user selects for the full wave rectifier. Else if the user turned off the smoothing capacitors the similar operation is performed the only difference is in the calculation where the required formula for the voltage without the capacitor filter is plotted in the output.

Figure 4 shows the all the functions used in the development of the MATLAB program.

Diagram

Description automatically generated

Figure 3:- Flow chart of overall process.

Diagram

Description automatically generated

Figure 4:- All function.

**FORMAT AND STYLE OF ROUTINES.**

After the completion of the flowchart the process were implemented in MATLAB. The Graphical User Interface(GUI) was designed using the App designer, and several functional components were used for the program. Three Axes components were used for plotting the input signal, output voltage, and comparison between the output and input signal. Switches were used for performing those operation which required On and Off situation such as for turning on the capacitor effect. Drop down component for the selection the full wave rectifier and half wave rectifier selection. Edit text field were used for the input of selected parameters as well as for the Output of Ripple Voltage peak to peak.

properties (Access = private)

voltage\_input\_amp = 10; %input voltage amplitude

Input\_signal;

frequency = 50; %default frequency

resistance = 1000;%default resistance

capacitor = 1200e-6; %default capacitance

voltage\_output;%output voltage variable

voltage\_rippleptp;%ripple voltage

omega;

% below are the time variables

t;

t1;

t2;

timevalue;

mainstate=0;%hold the state

dropdown= 0;%for the type of rectifier

capcaitorfilterselection=0; % Description

end

The above line of codes is for the creation of the properties which includes all the variables used in the program. Some of the variables were initialized which are required for the initial calculation of the program. There were variables which were used to hold the current state of a execution.

function CIRCUITDropDownValueChanged(app, event)

value = app.CIRCUITDropDown.Value;

if value == "HALF WAVE"

app.dropdown = 1;

else

app.dropdown = 0;

end

main(app,app.mainstate)

end

The above line of code shows the creation of a component function which in this case is for the drop-down selection. The component included two options one for half wave and other was full wave, if the user selects the half wave the variable which holds the current state for the drop-drown selection changes as per the conditions. After the conditions main function is called to execute the whole operation based on the current selection of the parameters.

function VoltageEditFieldValueChanged(app, event)

value = app.VoltageEditField.Value;

if value <=0

errordlg('ENTER A POSITIVE VALUE','INPUT ERROR');

else

app.voltage\_input\_amp = value;

end

main(app,app.mainstate)

end

The above line of code shows the execution of the edit text field component function which is for the voltage input. The user input is stored in a value variable if the value is less than or equal to zero and error dialog is displayed to the user about the input error. Else the input voltage variable is updated with the value variable. At last, the main function is called, this algorithm is implemented for all the Edit text field component.

methods (Access = private)

function main(app,state)% main function requiers the state paramenter for exectuib

if state == 1

app.t = (0:1/1000:0.1);%timpe variable

app.omega = 2 \* pi \* app.frequency;%omega calculation

app.Input\_signal = app.voltage\_input\_amp \* sin(app.omega \*app.t);%input signal

plot(app.InputAxes, app.Input\_signal)% plotting the input signal in the first axis

tau = app.resistance \* app.capacitor; % tau parameter for capacitor filter

The above line of code is of the main function which takes in the app module and a state variables as parameters. If the state variable is equal to 1 which means that the simulation is on then the mode code will get executed else the program will just be on the previous state. t variable is for the simulation of time which is a 1\*101 size of double array, the starting value is 0 and the last variable is 0.1 where an increment of 1/1000 is provided after each indices. The other statements are for the required variables for the execution of rectification .

for l = 1:100 %program will run from 1 to 100 with an incremnt of 1 so total of 100 samples

if app.dropdown == 1% half wave recitification

%ripple voltage formula implementaiton for half wave

app.VoltagePeaktoPeakEditField.Value = (app.voltage\_input\_amp)/(app.frequency \* app.resistance \*app.capacitor);

app.t2 = 24;

%output voltage calculation based on the

%conditons

if sin(app.omega\*app.t(l)) <=0

app.voltage\_output(l) = 0;

else

app.voltage\_output(l) = app.voltage\_input\_amp \* sin(app.omega \*app.t(l));

end

The above line of code shows the for and nested if condition statements that were implemented for the execution of the rectifier output. For statement is for the number of sample the program will run and the output array index will be stored on the for statement variable. If condition was used to check for the condition whether the simulation is turned for half wave rectification or for the full wave rectification. Rest of the code are based on the formulas for each type of rectifier.

**RESULTS AND ANALYSIS**

Graphical user interface, application

Description automatically generated

Figure 5:- Program GUI.

Figure 5 show the developed program GUI, where all the required parameters are included in the program.

Graphical user interface

Description automatically generated

Figure 6:- Full wave rectifier output.

Figure 6 shows the output when the simulation is turned on, the three axis shows the plot of the input signal, output voltage as well as the comparison between the input and output signal. This is for the full wave rectifier output since the selection from the drop down menu is full wave.

Graphical user interface, chart

Description automatically generated with medium confidence

Figure 7: Half wave rectifier output

Figure 7 shows the output when the simulation is turned on, the three axis shows the plot of the input signal, output voltage as well as the comparison between the input and output signal. This is for the full wave rectifier output since the selection from the drop-down menu is half wave.

Graphical user interface

Description automatically generated

Figure 8:- Half wave rectifier with capacitor filter on.

Figure 8 shows the half wave rectifier output but the time when the capacitor is turned on, hence it can be seen in the graph that the time the voltage hits the max amplitude the capacitor starts to discharge, and it keeps discharging until it reaches the intersection point of the output voltage. There are three phases for the rectifier output which are implemented.

Graphical user interface, application

Description automatically generated

Figure 9:- Error dialog.

Figure 9 shows the error dialog generated when -12v was provided as the input to the voltage, this algorithm is implemented for all the input.

**DISCUSSION**

After the implementation of the GUI and the inclusion of all required parameters mentioned in the assignment question, knowing that the capacitor filter inclusion will take time it was left for later. Inclusion of capacitor requires the calculation for the charging and discharging of RC circuit, the behaviour of capacitor is similar for half wave and full wave rectifier. Implementation of algorithm based on the theory of rectifiers with capacitor was throwing a lot of errors, sometimes when there was no error the output turned out to be not as expected. At last switch case was implemented for the three cases for the RC circuit implementation, first stage when the output voltage goes from 0 to the max amplitude of the voltage, the second stage is when the capacitor discharges until the intersection point of output voltage and the capacitor voltage, the last stage is again when the capacitor charges to the max amplitude, this cycle repeats for the required samples.

Output of voltage with and without capacitor separately was not working for the analyzation of the output, hence an axes was included which output the voltage for both the situation. Inclusion of drop-down menu for the selection of the type of rectifier was also throwing multiple error when the condition was compared with the text, but after initializing the conditions with an integer variable solved the error.

To output the parameters after each new input in any stage of the program a main function was implemented which gets called whenever there is any input from any of the components in the program. All the callback functions call the main function by providing app and state variable as the arguments, which solves the issue of simulation switch. Without the implementation of state variable, the output was plotted even when the switch was off.

**Conclusion**

In conclusion it can be said that the required parameters of the assignment have been achieved. The assignment uses App designed for the development of the GUI, drop down list was implemented for the type of rectifier selection. Editfield text component was implemented for allowing the user to input the voltage and frequency. A switch was implemented for enabling and disabling the effect of the capacitor for smoothing performance. The algorithm also allowed user to plot the output voltage at any point of time during the simulation. Error handling was implemented for every possible errors users can do such as entering negative or zero values for resistance and capacitance value.

Every potential error that users can make, such as entering negative or zero values for resistance and capacitance values, has been addressed in the implementation of error handling. Furthermore, the results and analysis section demonstrate that the developed system generated the values in the manner that was anticipated. Comments was included in the program to make it easier for other users to understand.

**REFRENCES**

|  |  |  |
| --- | --- | --- |
| **Resources** | **In-Text Citation** | **End-Text Referencing (Reference List)** |
| **Article:**  **Four.**  **Author** | MATLAB App Designer is a feature that allows MATLAB code to be packaged into an interactive software. Harun, N. H., Hambali, H. A., Hassan, M. G., & Karim, K. N. (2017). Learn by example: MATLAB app design. | Harun, N. H., Hambali, H. A., Hassan, M. G., & Karim, K. N. (2017). Learn by example: MATLAB app design. |
| **Article:**  **Six Authors** | The voltage waveform of half-wave rectifier and filter….. LU, Y. X., HUANG, H., & PU, X. W. (2015). Voltage Analysis of Half-wave Rectifier and Filter Circuit. *Physical Experiment of College*. | LU, Y. X., HUANG, H., & PU, X. W. (2015). Voltage Analysis of Half-wave Rectifier and Filter Circuit. *Physical Experiment of College*. |
| **Article:**  **Three autors** | Lata, K., & Jamadagni, H. S. (2009, October). Formal verification of full-wave rectifier: A case study. In *2009 IEEE 8th International Conference on ASIC* (pp. 1306-1309). IEEE. | Lata, K., & Jamadagni, H. S. (2009, October). Formal verification of full-wave rectifier: A case study. In *2009 IEEE 8th International Conference on ASIC* (pp. 1306-1309). IEEE. |

**APPENDIX**

classdef hwfw < matlab.apps.AppBase

% Properties that correspond to app components

properties (Access = public)

UIFigure matlab.ui.Figure

APUUNIVERSITYLabel matlab.ui.control.Label

POWERCONVERTERCIRCUITSIMULATIONLabel matlab.ui.control.Label

CIRCUITDropDownLabel matlab.ui.control.Label

CIRCUITDropDown matlab.ui.control.DropDown

SelectTypeofrectifierCircuitLabel matlab.ui.control.Label

EnterInputVoltageandFrequencyLabel matlab.ui.control.Label

VoltageEditFieldLabel matlab.ui.control.Label

VoltageEditField matlab.ui.control.NumericEditField

FrequencyEditFieldLabel matlab.ui.control.Label

FrequencyEditField matlab.ui.control.NumericEditField

CapacitoreffectSwitchLabel matlab.ui.control.Label

CapacitoreffectSwitch matlab.ui.control.RockerSwitch

InputAxes matlab.ui.control.UIAxes

OutputAxes matlab.ui.control.UIAxes

ResistorValueEditFieldLabel matlab.ui.control.Label

ResistorValueEditField matlab.ui.control.NumericEditField

CapacitorValueEditFieldLabel matlab.ui.control.Label

CapacitorValueEditField matlab.ui.control.NumericEditField

ComparisonAxes matlab.ui.control.UIAxes

SimulationSwitchLabel matlab.ui.control.Label

SimulationSwitch matlab.ui.control.Switch

VoltagePeaktoPeakEditFieldLabel matlab.ui.control.Label

VoltagePeaktoPeakEditField matlab.ui.control.NumericEditField

OHMLabel matlab.ui.control.Label

uVLabel matlab.ui.control.Label

end

properties (Access = private)

voltage\_input\_amp = 10; %input voltage amplitude

Input\_signal;

frequency = 50; %default frequency

resistance = 1000;%default resistance

capacitor = 1200e-6; %default capacitance

voltage\_output;%output voltage variable

voltage\_rippleptp;%ripple voltage

omega;

% below are the time variables

t;

t1;

t2;

timevalue;

mainstate=0;%hold the state

dropdown= 0;%for the type of rectifier

capcaitorfilterselection=0;

v; % Description

end

methods (Access = private)

function main(app,state)% main function requiers the state paramenter for exectuib

if state == 1

app.t = (0:1/1000:0.1);%timpe variable

app.omega = 2 \* pi \* app.frequency;%omega calculation

app.Input\_signal = app.voltage\_input\_amp \* sin(app.omega \*app.t);%input signal

plot(app.InputAxes, app.Input\_signal)% plotting the input signal in the first axis

tau = app.resistance \* app.capacitor; % tau parameter for capacitor filter

for l = 1:100 %program will run from 1 to 100 with an incremnt of 1 so total of 100 samples

if app.dropdown == 1% half wave recitification

%ripple voltage formula implementaiton for half wave

app.VoltagePeaktoPeakEditField.Value = (app.voltage\_input\_amp)/(app.frequency \* app.resistance \*app.capacitor);

app.t2 = 24;

%output voltage calculation based on the

%conditons

if sin(app.omega\*app.t(l)) <=0

app.voltage\_output(l) = 0;

else

app.voltage\_output(l) = app.voltage\_input\_amp \* sin(app.omega \*app.t(l));

end

else

%ripple voltage formula implementation for full

%wave

app.VoltagePeaktoPeakEditField.Value = (app.voltage\_input\_amp)/(app.frequency \* 2 \* app.resistance \*app.capacitor);

app.t2 = 14;

%conditonal execution for output

if sin(app.omega\*app.t(l)) <=0

app.voltage\_output(l) = -app.voltage\_input\_amp \* sin(app.omega \*app.t(l));

else

app.voltage\_output(l) = app.voltage\_input\_amp \* sin(app.omega \*app.t(l));

end

end

end

app.timevalue = 1;%state variable for three case

if(app.capcaitorfilterselection == 1)

app.t1=6;

for l=1:100

%switch case for all the three phases of rectifier

%output

switch app.timevalue

case 1

app.v(l) = app.voltage\_output(l);

if app.v(l) == app.voltage\_input\_amp

app.timevalue =2;

end

case 2

app.v(l) = app.voltage\_input\_amp \* (exp(-(app.t(app.t1))/tau));

if(app.t1 == app.t2)

app.timevalue = 3;

end

app.t1= app.t1+1;

case 3

app.v(l) = app.voltage\_output(l);

if(app.v(l) == app.voltage\_input\_amp)

app.timevalue = 2;

app.t1=6;

end

end

% app.voltage\_output(l) = v(l);

end

end

%plotting all the axis

if app.capcaitorfilterselection == 1

plot(app.OutputAxes, app.v)

else

plot(app.OutputAxes, app.voltage\_output)

end

plot(app.ComparisonAxes, app.voltage\_output)

hold(app.ComparisonAxes, "on")

plot(app.ComparisonAxes, app.v)

hold(app.ComparisonAxes,"off")

else

end

end

end

% Callbacks that handle component events

methods (Access = private)

% Value changed function: CIRCUITDropDown

function CIRCUITDropDownValueChanged(app, event)

% this function is for the selection of the type of rectigier

value = app.CIRCUITDropDown.Value;

if value == "HALF WAVE"

app.dropdown = 1;

else

app.dropdown = 0;

end

main(app,app.mainstate)

end

% Value changed function: SimulationSwitch

function SimulationSwitchValueChanged(app, event)

%this function is for the simulation execution

value = app.SimulationSwitch.Value;

if value == "On"

app.mainstate = 1;

main(app,app.mainstate)

else

app.mainstate = 0;

main(app,app.mainstate)

end

end

% Value changed function: VoltageEditField

function VoltageEditFieldValueChanged(app, event)

%this function holds the user voltage function

value = app.VoltageEditField.Value;

if value <=0

errordlg('ENTER A POSITIVE VALUE','INPUT ERROR');

else

app.voltage\_input\_amp = value;

end

main(app,app.mainstate)

end

% Value changed function: FrequencyEditField

function FrequencyEditFieldValueChanged(app, event)

value = app.FrequencyEditField.Value;

if value <=0

errordlg('ENTER A POSITIVE VALUE','INPUT ERROR');

else

app.frequency = value;

end

main(app, app.mainstate)

end

% Value changed function: CapacitoreffectSwitch

function CapacitoreffectSwitchValueChanged(app, event)

value = app.CapacitoreffectSwitch.Value;

if value == "On"

app.capcaitorfilterselection = 1;

else

app.capcaitorfilterselection = 0;

end

main(app, app.mainstate)

end

% Value changed function: VoltagePeaktoPeakEditField

function VoltagePeaktoPeakEditFieldValueChanged(app, event)

end

% Value changed function: ResistorValueEditField

function ResistorValueEditFieldValueChanged(app, event)

value = app.ResistorValueEditField.Value;

if value <=0

errordlg('Resistance cant be negative or Zero','INPUT ERROR');

else

app.resistance = value;

end

main(app, app.mainstate)

end

% Value changed function: CapacitorValueEditField

function CapacitorValueEditFieldValueChanged(app, event)

value = app.CapacitorValueEditField.Value;

if value <=0

errordlg('Please enter a positive value','INPUT ERROR');

else

app.capacitor = value \* 10^-6;

end

main(app, app.mainstate)

end

end

% Component initialization

methods (Access = private)

% Create UIFigure and components

function createComponents(app)

% Create UIFigure and hide until all components are created

app.UIFigure = uifigure('Visible', 'off');

app.UIFigure.Color = [0.302 0.7451 0.9333];

app.UIFigure.Position = [100 100 1025 612];

app.UIFigure.Name = 'MATLAB App';

% Create APUUNIVERSITYLabel

app.APUUNIVERSITYLabel = uilabel(app.UIFigure);

app.APUUNIVERSITYLabel.BackgroundColor = [0.0745 0.6235 1];

app.APUUNIVERSITYLabel.HorizontalAlignment = 'center';

app.APUUNIVERSITYLabel.FontName = 'Arial';

app.APUUNIVERSITYLabel.FontWeight = 'bold';

app.APUUNIVERSITYLabel.Position = [1 580 1025 33];

app.APUUNIVERSITYLabel.Text = 'APU UNIVERSITY';

% Create POWERCONVERTERCIRCUITSIMULATIONLabel

app.POWERCONVERTERCIRCUITSIMULATIONLabel = uilabel(app.UIFigure);

app.POWERCONVERTERCIRCUITSIMULATIONLabel.BackgroundColor = [0 0.4471 0.7412];

app.POWERCONVERTERCIRCUITSIMULATIONLabel.HorizontalAlignment = 'center';

app.POWERCONVERTERCIRCUITSIMULATIONLabel.FontWeight = 'bold';

app.POWERCONVERTERCIRCUITSIMULATIONLabel.Position = [1 551 1025 30];

app.POWERCONVERTERCIRCUITSIMULATIONLabel.Text = 'POWER CONVERTER CIRCUIT SIMULATION';

% Create CIRCUITDropDownLabel

app.CIRCUITDropDownLabel = uilabel(app.UIFigure);

app.CIRCUITDropDownLabel.BackgroundColor = [0 1 1];

app.CIRCUITDropDownLabel.HorizontalAlignment = 'right';

app.CIRCUITDropDownLabel.Position = [13 400 54 22];

app.CIRCUITDropDownLabel.Text = 'CIRCUIT';

% Create CIRCUITDropDown

app.CIRCUITDropDown = uidropdown(app.UIFigure);

app.CIRCUITDropDown.Items = {'FULL WAVE', 'HALF WAVE', ''};

app.CIRCUITDropDown.ValueChangedFcn = createCallbackFcn(app, @CIRCUITDropDownValueChanged, true);

app.CIRCUITDropDown.BackgroundColor = [0 1 1];

app.CIRCUITDropDown.Position = [82 400 100 22];

app.CIRCUITDropDown.Value = 'FULL WAVE';

% Create SelectTypeofrectifierCircuitLabel

app.SelectTypeofrectifierCircuitLabel = uilabel(app.UIFigure);

app.SelectTypeofrectifierCircuitLabel.BackgroundColor = [0 1 0];

app.SelectTypeofrectifierCircuitLabel.FontWeight = 'bold';

app.SelectTypeofrectifierCircuitLabel.Position = [13 430 185 22];

app.SelectTypeofrectifierCircuitLabel.Text = 'Select Type of rectifier Circuit';

% Create EnterInputVoltageandFrequencyLabel

app.EnterInputVoltageandFrequencyLabel = uilabel(app.UIFigure);

app.EnterInputVoltageandFrequencyLabel.BackgroundColor = [0 1 0];

app.EnterInputVoltageandFrequencyLabel.FontWeight = 'bold';

app.EnterInputVoltageandFrequencyLabel.Position = [19 347 205 22];

app.EnterInputVoltageandFrequencyLabel.Text = 'Enter Input Voltage and Frequency';

% Create VoltageEditFieldLabel

app.VoltageEditFieldLabel = uilabel(app.UIFigure);

app.VoltageEditFieldLabel.BackgroundColor = [1 0.4118 0.1608];

app.VoltageEditFieldLabel.HorizontalAlignment = 'right';

app.VoltageEditFieldLabel.FontWeight = 'bold';

app.VoltageEditFieldLabel.Position = [16 315 48 22];

app.VoltageEditFieldLabel.Text = 'Voltage';

% Create VoltageEditField

app.VoltageEditField = uieditfield(app.UIFigure, 'numeric');

app.VoltageEditField.ValueChangedFcn = createCallbackFcn(app, @VoltageEditFieldValueChanged, true);

app.VoltageEditField.FontWeight = 'bold';

app.VoltageEditField.BackgroundColor = [1 0.4118 0.1608];

app.VoltageEditField.Position = [79 315 116 22];

% Create FrequencyEditFieldLabel

app.FrequencyEditFieldLabel = uilabel(app.UIFigure);

app.FrequencyEditFieldLabel.BackgroundColor = [0.3922 0.8314 0.0745];

app.FrequencyEditFieldLabel.HorizontalAlignment = 'right';

app.FrequencyEditFieldLabel.FontWeight = 'bold';

app.FrequencyEditFieldLabel.Position = [17 272 66 22];

app.FrequencyEditFieldLabel.Text = 'Frequency';

% Create FrequencyEditField

app.FrequencyEditField = uieditfield(app.UIFigure, 'numeric');

app.FrequencyEditField.ValueChangedFcn = createCallbackFcn(app, @FrequencyEditFieldValueChanged, true);

app.FrequencyEditField.FontWeight = 'bold';

app.FrequencyEditField.BackgroundColor = [0.3922 0.8314 0.0745];

app.FrequencyEditField.Position = [98 272 100 22];

% Create CapacitoreffectSwitchLabel

app.CapacitoreffectSwitchLabel = uilabel(app.UIFigure);

app.CapacitoreffectSwitchLabel.HorizontalAlignment = 'center';

app.CapacitoreffectSwitchLabel.FontWeight = 'bold';

app.CapacitoreffectSwitchLabel.Position = [13 129 96 22];

app.CapacitoreffectSwitchLabel.Text = 'Capacitor effect';

% Create CapacitoreffectSwitch

app.CapacitoreffectSwitch = uiswitch(app.UIFigure, 'rocker');

app.CapacitoreffectSwitch.ValueChangedFcn = createCallbackFcn(app, @CapacitoreffectSwitchValueChanged, true);

app.CapacitoreffectSwitch.Position = [50 187 20 45];

% Create InputAxes

app.InputAxes = uiaxes(app.UIFigure);

title(app.InputAxes, 'Input')

xlabel(app.InputAxes, 'time')

ylabel(app.InputAxes, 'Voltage')

app.InputAxes.Position = [245 307 381 236];

% Create OutputAxes

app.OutputAxes = uiaxes(app.UIFigure);

title(app.OutputAxes, 'Output')

xlabel(app.OutputAxes, 'X')

ylabel(app.OutputAxes, 'Voltage')

app.OutputAxes.Position = [645 301 381 236];

% Create ResistorValueEditFieldLabel

app.ResistorValueEditFieldLabel = uilabel(app.UIFigure);

app.ResistorValueEditFieldLabel.BackgroundColor = [0.9294 0.6941 0.1255];

app.ResistorValueEditFieldLabel.HorizontalAlignment = 'right';

app.ResistorValueEditFieldLabel.FontWeight = 'bold';

app.ResistorValueEditFieldLabel.Position = [4 70 88 22];

app.ResistorValueEditFieldLabel.Text = 'Resistor Value';

% Create ResistorValueEditField

app.ResistorValueEditField = uieditfield(app.UIFigure, 'numeric');

app.ResistorValueEditField.ValueChangedFcn = createCallbackFcn(app, @ResistorValueEditFieldValueChanged, true);

app.ResistorValueEditField.FontWeight = 'bold';

app.ResistorValueEditField.BackgroundColor = [0.9294 0.6941 0.1255];

app.ResistorValueEditField.Position = [102 70 100 22];

% Create CapacitorValueEditFieldLabel

app.CapacitorValueEditFieldLabel = uilabel(app.UIFigure);

app.CapacitorValueEditFieldLabel.BackgroundColor = [0.9294 0.6941 0.1255];

app.CapacitorValueEditFieldLabel.HorizontalAlignment = 'right';

app.CapacitorValueEditFieldLabel.FontWeight = 'bold';

app.CapacitorValueEditFieldLabel.Position = [4 28 95 22];

app.CapacitorValueEditFieldLabel.Text = 'Capacitor Value';

% Create CapacitorValueEditField

app.CapacitorValueEditField = uieditfield(app.UIFigure, 'numeric');

app.CapacitorValueEditField.ValueChangedFcn = createCallbackFcn(app, @CapacitorValueEditFieldValueChanged, true);

app.CapacitorValueEditField.FontWeight = 'bold';

app.CapacitorValueEditField.BackgroundColor = [0.9294 0.6941 0.1255];

app.CapacitorValueEditField.Position = [108 28 100 22];

% Create ComparisonAxes

app.ComparisonAxes = uiaxes(app.UIFigure);

title(app.ComparisonAxes, 'Comparsion')

xlabel(app.ComparisonAxes, 'X')

ylabel(app.ComparisonAxes, 'Y')

app.ComparisonAxes.Position = [451 5 517 271];

% Create SimulationSwitchLabel

app.SimulationSwitchLabel = uilabel(app.UIFigure);

app.SimulationSwitchLabel.HorizontalAlignment = 'center';

app.SimulationSwitchLabel.FontWeight = 'bold';

app.SimulationSwitchLabel.Position = [62.5 468 67 22];

app.SimulationSwitchLabel.Text = 'Simulation';

% Create SimulationSwitch

app.SimulationSwitch = uiswitch(app.UIFigure, 'slider');

app.SimulationSwitch.ValueChangedFcn = createCallbackFcn(app, @SimulationSwitchValueChanged, true);

app.SimulationSwitch.FontWeight = 'bold';

app.SimulationSwitch.Position = [72 505 45 20];

% Create VoltagePeaktoPeakEditFieldLabel

app.VoltagePeaktoPeakEditFieldLabel = uilabel(app.UIFigure);

app.VoltagePeaktoPeakEditFieldLabel.BackgroundColor = [0.3922 0.8314 0.0745];

app.VoltagePeaktoPeakEditFieldLabel.HorizontalAlignment = 'right';

app.VoltagePeaktoPeakEditFieldLabel.FontWeight = 'bold';

app.VoltagePeaktoPeakEditFieldLabel.Position = [189 198 125 22];

app.VoltagePeaktoPeakEditFieldLabel.Text = 'Voltage Peak to Peak';

% Create VoltagePeaktoPeakEditField

app.VoltagePeaktoPeakEditField = uieditfield(app.UIFigure, 'numeric');

app.VoltagePeaktoPeakEditField.ValueChangedFcn = createCallbackFcn(app, @VoltagePeaktoPeakEditFieldValueChanged, true);

app.VoltagePeaktoPeakEditField.FontWeight = 'bold';

app.VoltagePeaktoPeakEditField.Position = [329 198 100 22];

% Create OHMLabel

app.OHMLabel = uilabel(app.UIFigure);

app.OHMLabel.FontWeight = 'bold';

app.OHMLabel.Position = [223 70 33 22];

app.OHMLabel.Text = 'OHM';

% Create uVLabel

app.uVLabel = uilabel(app.UIFigure);

app.uVLabel.FontWeight = 'bold';

app.uVLabel.Position = [223 28 25 22];

app.uVLabel.Text = 'uV';

% Show the figure after all components are created

app.UIFigure.Visible = 'on';

end

end

% App creation and deletion

methods (Access = public)

% Construct app

function app = hwfw

% Create UIFigure and components

createComponents(app)

% Register the app with App Designer

registerApp(app, app.UIFigure)

if nargout == 0

clear app

end

end

% Code that executes before app deletion

function delete(app)

% Delete UIFigure when app is deleted

delete(app.UIFigure)

end

end

end