# Answer 1A

**Output to an input step response :**

**Chart, line chart

Description automatically generated**

**MATLAB Code:**

close all; clear; clc;

Given Parameters

% Resistance 1 [ohms]

R1 = 100000;

% Resistance 2 [Ohms]

R2 = 100000;

% Inductance [Henrys]

L = 4.7e-3;

% Capacitance [Farads]

C = .001e-9;

% Conductace [1 / Ohms]

% G = 1 / R

G1 = 1 / R1;

G2 = 1 / R2;

Transfer Function

% as^2 + bs + c = 0

% Normalizing the polynomial: s^2 + (b/a)s + (c/a) = 0

a = (G1 + G2);

b = ((G1 \* G2 \* L) + C) / (L \* C);

c = G2 / (L \* C);

% Numerator of tranfer function

numerator = [(G1 \* G2 ) / C, 0];

% Denominator of transfer function

denominator = [a, b, c];

% Constructing transfer function

Gs = tf(numerator, denominator);

% Creating new figure

figure(1)

% Ploting the step responce of the system

step(Gs);

% Plot parameters

hold on

grid on

grid minor

% Plot descriptors

xlabel('{\emph Time}','fontsize',14,'Interpreter','latex');

ylabel('{\emph Aplitude }','fontsize',14,'Interpreter','latex');

title('{Step Response}','fontsize',16,'Interpreter','latex');

legend('Output', 'fontsize', 10, 'Interpreter', 'latex');

# Answer 1B

**Characteristic parameters of the system:**

Natural frequency:

Damping ratio:

Where:

Since , the system is underdamped.

**MATLAB Code:**

Characteristic Parameters

% Natural frequency [Hz]

wn = sqrt(c/a);

% x = 2 \* zeta \* wn

x = b / a;

% Daming ratio

zeta = x / (2 \* wn);

Note that this section of code is dependent on the previous section of code.

# Answer 1C

Chart

Description automatically generated

**System Response:**

The poles are complex, with a negative real portion. There is a fast rise time, but it oscillates and is slow to converge. Despite this, it eventually converges and is therefore stable.

**MATLAB Code:**

System Response

% Solving for roots of characteristic polynomial

characteristicRoots = roots(denominator);

% Creating new figure

figure(2)

pzplot(Gs)

hold on

grid on

% Plot descriptors

set(gca, 'fontName', 'Times');

xlabel('\sigma Real Axis', 'fontName', 'Times', 'fontSize', 14);

ylabel('j\omega\_{n} Imaginary Axis', 'fontName', 'Times', 'fontSize', 14);

title('{Pole-Zero Map}','fontsize',16,'Interpreter','latex');

Note that this section of code is dependent on the previous section of code.

# Answer 2A

**Output to an input step response for three parameter combinations:**

**Diagram

Description automatically generated with medium confidence**

**MATLAB Code:**

close all; clear; clc;

Given Parameters

% Mass [kg]

M = 100;

% K [N/m]

K = [1.5 15 150];

% fv [kg/s]

fv = [3 270 78];

Constructing Transfer Function and Plotting Output

% Creating new figure

figure(1)

% Numerator of transfer function

% Constant for all cases

numerator = 1/M;

for ii = 1:length(K)

% Denominator of transfer function for all 3 cases

denominator = [1 fv(ii)/M K(ii)/M];

% Generating transfer function

Gs = tf(numerator, denominator);

% Creating subplot

subplot(3, 1, ii)

% Plotting the step response of the system

step(Gs)

hold on

% Plot parameters

grid on

grid minor

% Plot descriptors

xlabel('{\emph Time}','fontsize',14,'Interpreter','latex');

ylabel('{\emph Aplitude }','fontsize',14,'Interpreter','latex');

titleText = sprintf('Step Response - %d', ii);

title(titleText,'fontsize',16,'Interpreter','latex');

leg = sprintf('System Parameter Combination - %d', ii);

% Placing legend in southeast corner for cobination 2

if ii == 2

legend(leg, 'fontsize', 10, 'location', 'southeast','Interpreter', ...

'latex')

% Placing legend in northeast corner for combination 1 & 3

else

legend(leg, 'fontsize', 10, 'location', 'northeast','Interpreter', ...

'latex')

end

end

# Answer 2B

**Characteristic parameters of the system:**

Where:

Case 1:

Since , the system is underdamped.

Case 2:

Since , the system is overdamped.

Case 3:

Since , the system is underdamped.

**MATLAB Code:**

Characteristic Parameters

% Initializing vectors

wn = zeros(1, length(K));

x = zeros(1, length(K));

zeta = zeros(1, length(K));

for ii = 1: length(K)

% Natural frequency

wn(ii) = sqrt(K(ii) / M);

% x = 2 \* zeta \* wn

x(ii) = fv(ii) / M;

zeta(ii) = x(ii) / (2 \* wn(ii));

end

Note that this section of code is dependent on the previous section of code.

# Answer 2C

Chart

Description automatically generated

Case 1:

The poles are complex, with a negative real portion. There is a fast rise time, but it oscillates and is slow to converge. Despite this, it eventually converges and is therefore stable.

Case 2:

The poles are negative on the real axis and have no imaginary part. This response is slow to converge but is still considered steady.

Case 3:

The poles are complex, with a negative real portion. There is a fast rise time, but it oscillates and is slow to converge. Despite this, it eventually converges and is therefore stable.

**MATLAB Code:**

System Response

% Creating new figure

figure(2)

% Initializing arrays

characteristicRoots = zeros(length(K), 2);

% Plotting poles

for ii = 1:length(K)

% Denominator of transfer function for all 3 cases

denominator = [1 fv(ii)/M K(ii)/M];

% Generating transfer function

Gs = tf(numerator, denominator);

% Solving for roots of characteristic polynomial

polynomialRoots = roots(denominator);

% Storing roots

characteristicRoots(ii, 1) = polynomialRoots(1, 1);

characteristicRoots(ii, 2) = polynomialRoots(2, 1);

% Creating pole - zero plot

pzplot(Gs);

hold on

end

% Turning grid on

grid on

% Plot descriptors

set(gca, 'fontName', 'Times');

xlabel('\sigma Real Axis', 'fontName', 'Times', 'fontSize', 14);

ylabel('j\omega\_{n} Imaginary Axis', 'fontName', 'Times', 'fontSize', 14);

title('{Pole-Zero Map}','fontsize',16,'Interpreter','latex');

legend('Poles - Case 1', 'Poles - Case 2', 'Poles - Case 3', ...

'fontsize', 10, 'Interpreter', 'latex', 'location', 'northwest');

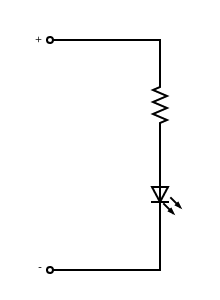
Note that this section of code is dependent on the previous section of code.

# Answer 3A

Resistors are current limiting resistors for the LEDs in the circuit. Their purpose is to limit the forward current into the diodes since they have a certain operating region. The forward current (typical and surge) as well as the forward voltage (typical and maximum) for 5mm red and green diodes can be seen in the table below.

|  |  |  |
| --- | --- | --- |
|  | **Red LED** | **Green LED** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Using KVL, the output voltage is represented by the following equation:



Rearranging this and solving for the voltage across the resistor, the equation becomes:

From Ohm’s law, the voltage across the resistor is represented by:

Substituting the equations and solving for , the equation becomes:

The values for are as follows:

|  |  |
| --- | --- |
| **Resistance** | |
| **Red LED** | **Green LED** |
|  |  |

Resistors serve as pull down resistors. They ensure that when the switch is open, despite any leakage current from the Arduino micro controller, the voltage will be zero. The leakage current is assumed to be 30% of the supply voltage. Ohm’s law on these resistors is as follows:

Rearranging and solving for the pull-down resistance, the equation becomes:

With and , the pull-down resistance is:

This resistance is the maximum allowable resistance for the desired voltage. Because of this, a resistance value less that should be used in the circuit.

**MATLAB Code:**

close all; clear; clc

% LED Values

LED(1).color = 'RED';

LED(1).FWDcurrent = [20 \* 10 ^ -3, 1];

LED(1).FWDvoltage = [2 3];

LED(2).color = 'GREEN';

LED(2).FWDcurrent = [30 \* 10 ^ -3, 1];

LED(2).FWDvoltage = [2 3];

Vout = 5;

Vc = 5;

Ileak = 1 \* 10 ^ -3;

Solving for resistances R2 & R3

% Initializing arry

R = zeros(length(LED(1).FWDcurrent), ...

length(LED(1).FWDvoltage), ...

length(LED));

% R(:,:,1) = R2

% R(:,:,2) = R3

for ii = 1:length(LED)

for jj = 1:length(LED(1).FWDcurrent)

for kk = 1:length(LED(1).FWDvoltage)

R(jj, kk, ii) = (Vout - LED(ii).FWDvoltage(kk)) / ...

LED(ii).FWDcurrent(jj);

end

end

end

Solving for pull-down resistance (R4 & R5)

RpullDown = (.3 \* Vc) / Ileak;

# Answer 3B

**IO:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Classification** | **Signal Type** | **Pin** |
| Switch 1 | Input | Digital | 7 |
| Switch 2 | Input | Digital | 8 |
| Motor | Output | Analog | 6 |
| Red LED | Output | Digital | 2 |
| Green LED | Output | Digital | 3 |

Both switches are digital inputs that are either HIGH (depressed) or LOW (not depressed). They dictate the operation of the DC motor.

The DC motor is an analog output. In this case since no duty cycle is specified, the motor is off with a PWM of zero and on with a PWM of 255.

Both LEDs are digital outputs. An LED that is on receives a HIGH signal, and an LED that is off receives a LOW signal.

# Answer 3C

**Arduino Sketch:**

# define switch\_1 7

# define switch\_2 8

# define RED\_LED 2

# define GREEN\_LED 3

# define MOTOR 6

// Initializing both pushbottons

int switch\_1\_state;

int switch\_2\_state;

void setup() {

 // Defining IO

 pinMode(switch\_1, INPUT);

 pinMode(switch\_2, INPUT);

 pinMode(RED\_LED, OUTPUT);

 pinMode(GREEN\_LED, OUTPUT);

 pinMode(MOTOR, OUTPUT);

}

void loop() {

 // Determining the state of both swithces

 switch\_1\_state = digitalRead(switch\_1);

 switch\_2\_state = digitalRead(switch\_2);

 // IF the first swithc is pressed

 if (switch\_1\_state == HIGH && switch\_2\_state == LOW) {

   // Turn on the motor

   digitalWrite(MOTOR, HIGH);

   // Turn red LED on and green LED off

   digitalWrite(RED\_LED, HIGH);

   digitalWrite(GREEN\_LED, LOW);

   // Set state of swich 1 to off

   switch\_1\_state == LOW;

 }

 // If swith 2 is pressed

 else if (switch\_2\_state == HIGH && switch\_1\_state == LOW) {

   // Turn the motor off

   analogWrite(MOTOR, 0);

   // Turn the red LED off and green LED on

   digitalWrite(RED\_LED, LOW);

   digitalWrite(GREEN\_LED, HIGH);

   // Set the state of switch 2 back to 0

   switch\_2\_state == LOW;

 }

 // If both buttons are pressed simultaneously

 else{

   // Turn the motor off

   analogWrite(MOTOR, 255);

   // Turn the red LED off and the green LED on

   digitalWrite(RED\_LED, LOW);

   digitalWrite(GREEN\_LED, HIGH);

   // Set the state of both swithces to 0

   switch\_1\_state == LOW;

   switch\_2\_state == LOW;

 }

}

**Flow Diagram:**

**Diagram

Description automatically generated**

# Answer 3D

In this system, there are two pushbuttons, two pull-down resistors, two LEDs (red and green), two current limiting resistors, a 5V voltage source, an Arduino Uno microcontroller, and a DC motor. The Arduino input pin has a leakage current of around 1mA causing a voltage in circuit (around 30% of the supply voltage); including these two pull-down resistors ensures that when the switch is open, there is zero voltage in the circuit. The current limiting resistors, ensure that the LEDs are within viable operating conditions.

In terms of the operation of the system, the two pushbuttons act as switches for the motor. When the first switch is pressed, the motor starts. When the second pushbutton is pressed, the motor stops. In the case that both pushbuttons are simultaneously pressed, the motor will not start. Whenever the motor is operational, the red LED is on, and the green LED is off; this is reversed when the motor is not operational. The code is based on the state of the switches with conditional statements included to satisfy the execution of the tasks.

# Answer 4A

**IO:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Classification** | **Signal Type** | **Pin** |
| Potentiometer | Input | Analog | A1 |
| Thermistor | Input | Analog | A0 |
| Heater | Output | Analog | 3 |
| Red LED | Output | Digital | 2 |
| Green LED | Output | Digital | 4 |

The potentiometer is an analog input that has a range of 0 – 1023. In the code, these values are mapped to a range of 0 – 100 allowing for a comparison with the temperature received from the thermistor.

The thermistor is also an analog input. This contains the temperature in , which is then converted to . This value is also mapped from 50 – 100 to 0 – 100.

The heater is a digital output. Based on the if statement, the heater will either receive a HIGH or LOW command, dictating whether it is on or off.

The red and green LEDs also operate in the same manner. Based on certain conditions, the red and green LEDs will receive a HIGH command for on or a LOW command for off as they are digital outputs.

# Answer 4B

**Arduino Sketch:**

#define POTENTIOMETER A1  // Defining pin A1 as the potentioketer

#define THERMISTOR A0     // Defining A0 as the thermistor

#define HEATER 3          // Defining pin 3 as the heater

#define RED\_LED 2         // Defining pin 2 as the red LED

#define GREEN\_LED 4       // Defining pin 4 as the green LED

// Defining integers

int pot\_val;

int set\_point;

// Defining floats

float current\_temp;

float current\_temp\_F;

// Initializing the heater PWM

int heater\_PWM = 0;

void setup() {

 // Defining IO

 pinMode(POTENTIOMETER, INPUT);

 pinMode(THERMISTOR, INPUT);

 pinMode(HEATER, OUTPUT);

 pinMode(RED\_LED, OUTPUT);

 pinMode(GREEN\_LED, OUTPUT);

 // Initialiing the heater as off

 analogWrite(HEATER, heater\_PWM);

}

void loop() {

 // Reading the potentiometer value

 pot\_val = analogRead(POTENTIOMETER);

 // Mapping the pot values

 set\_point = map(pot\_val, 0, 1023, 0, 100);

 // Reading the thermistor and coverting to Farenheit

 current\_temp\_F = (analogRead(THERMISTOR) \* (9 / 5)) + 32;

 // Mapping the thermistor values

 current\_temp = map(current\_temp\_F, 50, 100, 0, 100);

 // If the current temperature is lower than the setpoint

 if (current\_temp < set\_point) {

   // Heater PWM is set to a duty cycle of 75%

// Max PWM value is 255

   heater\_PWM = 191;

   // Heater is written the PWM value

   analogWrite(HEATER, heater\_PWM);

   // Red LED is turned on

   digitalWrite(RED\_LED, HIGH);

   // Green LED is turned off

   digitalWrite(GREEN\_LED, LOW);

 }

 // If the current temperature is greater than or equal to the setpoint

 else if (current\_temp >= set\_point) {

   // The heater PWM is set to 0

   heater\_PWM = 0;

   // The heater is turned off (PWM = 0)

   analogWrite(HEATER, heater\_PWM);

   // Red LED is turned off

   digitalWrite(RED\_LED, LOW);

   // Green LED is turned on

   digitalWrite(GREEN\_LED, HIGH);

 }

}

**Flow Diagram:**

**Diagram

Description automatically generated**

# Answer 4C

The components involved in this system consisted of a potentiometer, thermistor, two LEDs (red and green), two current limiting resistors, a 5V voltage source, and an Arduino Uno microcontroller. The potentiometer is used to control the temperature setpoint. Since the Arduino reads analog data on a scale from 0 – 1023, these values must be mapped to 0 – 100. This value is compared with a temperature reading from a thermistor. The thermistor reads the temperature in degrees Celsius; this value is converted to degrees Fahrenheit and mapped from an input range of 50 - 100 to 0 – 100. This allows for an equal comparison of the temperature read from the thermistor and the set temperature.

When the temperature is less than the set temperature, the heater operates with a duty cycle of 75%; under this condition, the red LED is on, and the green LED is off. When the temperature is greater than or equal to the set temperature, the heater stops, and the red LED turns off while the green LED turns on.