APPENDIX

I. Arduino Program

```
#include "max6675.h"


int thermoCLK = 31;

int thermoCS = 35;

int thermoSO = 39;


int thermo1CLK=30;

int thermo1CS= 34;

int thermo1SO= 38;


int relayPin=47;

MAX6675 thermocouple(thermoCLK, thermoCS, thermoSO);

MAX6675 thermocouple1(thermo1CLK, thermo1CS,thermo1SO);

int vccPin = 3;

int gndPin = 2;


void setup() {

  Serial.begin(9600);

  // use Arduino pins

  pinMode(vccPin, OUTPUT); digitalWrite(vccPin, HIGH);
```

```
  pinMode(gndPin, OUTPUT); digitalWrite(gndPin, LOW);
  pinMode(relayPin,OUTPUT);digitalWrite(relayPin,LOW);

  // wait for MAX chip to stabilize
  delay(500);
}

void loop() {

   Serial.print("A");
   Serial.println(thermocouple.readCelsius());
   Serial.print("R");
   Serial.println(thermocouple1.readCelsius());
   if(Serial.available()) {
  char serialListener = Serial.read();
  if (serialListener == 'H'){
  digitalWrite(relayPin,HIGH);
  }
  else if (serialListener == 'L'){
  digitalWrite(relayPin,LOW);
  }
   }

   delay(1000);
}
```

II.GUI Program

```
#----- Imports -----
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
from matplotlib.ticker import MaxNLocator
from matplotlib.dates import DateFormatter
import tkinter as tk
import numpy as np
import serial as ser
import re
from csv import writer
from datetime import datetime


#----- Global variables -----
temp_rod = np.array([])
temp_ambient = np.array([])
ts_rod = np.array([])
ts_ambient = np.array([])


cond = False
temp_reached = False
temp_settings = False
csv_appended = False
```

```
temp_start = 0.0

temp_step = 0.0

temp_number = 0

temp_array = []

temp_smoothed = 0.0


#----- CSV functions and initialization -----


filename = 'experiment_data.csv'
with open(filename, 'w', newline='') as write_obj:
          csv_writer = writer(write_obj, delimiter=';')
         csv_writer.writerow(['Timestamp', 'Temperature'])


#--- Function: Append to CSV ---
def append_csv(timestamp, temperature):
                    global filename

        with open(filename, 'a+', newline='') as write_obj:
           csv_writer = writer(write_obj, delimiter=";")
           csv_writer.writerow([timestamp, temperature])


#----- END: CSV functions -----


#----- Heater functions -----


#--- Function: Turn on heater ---
```

```
def heater():
                            print('ON')
                           ser.write(b'H')


#--- Function: Turn off heater ---
def heateroff():
                            print('OFF')
                           ser.write(b'L')


#----- END: Heater functions -----


#----- Plot functions -----


#--- Function: Start plotting data ---
def plot_start():
                        global cond, temp_settings


                      if not temp_settings:
       label_validation.grid(row=8, column=0, columnspan=2)
    label_validation['text'] = 'Please set up the temperature
settings before starting to plot.'
                                else:
                     root.after(100, main_loop)
                                cond = True


#--- Function: Stop plotting data ---
```

```
def plot_stop():
                            global cond
                            cond = False


#--- Function: Plot rod temperature ---
def plot_rod(xdata, ydata):
                ax.set_title('Rod Temperature')
                    ax.set_xlabel('Time')
                ax.set_ylabel('Temperature (°C)')
                    ax.set_xlim(auto=True)
                     ax.set_ylim(0,100)
            ax.xaxis.set_major_locator(MaxNLocator(5))
     ax.xaxis.set_major_formatter(DateFormatter('%H.%M.%S'))
                ax.plot(xdata, ydata, color='blue')

#--- Function: Plot ambient temperature ---
def plot_ambient(xdata, ydata):
              ax1.set_title('Ambient Temperature')
                    ax1.set_xlabel('Time')
               ax1.set_ylabel('Temperature (°C)')
                    ax1.set_xlim(auto=True)
                     ax1.set_ylim(0,100)
           ax1.xaxis.set_major_locator(MaxNLocator(5))
     ax1.xaxis.set_major_formatter(DateFormatter('%H.%M.%S'))
                ax1.plot(xdata, ydata, color='blue')
```

```
#--- Function: Check serial data and plot ---
def plot_data():
  global cond, temp_rod, ts_rod, a, b, temp_ambient, ts_ambient


                        if cond:
                    ts = datetime.now()
                read = str(s.readline())
                    a=read[3:][:-5]


            a_bool = re.search(r'A',read,re.MULTILINE)
            r_bool = re.search(r'R',read,re.MULTILINE)
                        if r_bool:
                    if(len(temp_rod) < 50):
                ts_rod = np.append(ts_rod, ts)
            temp_rod = np.append(temp_rod,float(a))
                        else:
                ts_rod[0:49] = ts_rod[1:50]
                    ts_rod[49] = ts
                temp_rod[0:49] = temp_rod[1:50]
                    temp_rod[49] = float(a)


                        ax.clear()
                    plot_rod(ts_rod, temp_rod)
                        canvas.draw()


                        if a_bool:
```

```
                if(len(temp_ambient) < 50):
            ts_ambient = np.append(ts_ambient, ts)
          temp_ambient = np.append(temp_ambient,float(a))
                            else:
             ts_ambient[0:49] = ts_ambient[1:50]
                    ts_ambient[49] = ts
           temp_ambient[0:49] = temp_ambient[1:50]
                 temp_ambient[49] = float(a)

                            ax1.clear()
            plot_ambient(ts_ambient, temp_ambient)
                           canvas1.draw()


def temperature_save():
global  temp_start,  temp_step,  temp_number,  temp_array,
temp_reached, temp_settings

                        temp_reached = False
                          temp_array = []
       label_validation.grid(row=8, column=0, columnspan=2)

                                try:
            temp_start = float(entry_tempstart.get())
             temp_step = float(entry_tempstep.get())
          temp_number = float(entry_tempdpoints.get())
```

```
                except Exception as e:
                    temp_settings = False
    label_validation['text'] = 'Input not number (integer, float)!'
                else:
            end = temp_start - temp_step * temp_number
            label_str = 'Will notify on temperatures: '
            for i in np.arange(end, temp_start, temp_step):
                temp_array.append(i)
                label_str += str(round(i, 2)) + ', '
            temp_array = temp_array[::-1]
            label_validation['text'] = label_str[:-2]
            temp_settings = True


#----- Notification functions -----

#--- Function: Check if temperature reached desired interval ---
def notification_check():
global temp_rod, temp_reached, temp_settings, temp_smoothed, csv_appended, temp_array


            if temp_reached and temp_settings:
                if len(temp_array) > 0:
                if temp_smoothed < temp_array[0]:
                notification_show(temp_array[0])
```

```
                temp_array = np.delete(temp_array, 0)
                        if not csv_appended:
                  append_csv(datetime.now(), round(temp_smoothed,
2))
                        csv_appended = True


#--- Function: Show notification if desired starting temperature
reached ---
def notification_temp():
     label_notification.grid(row=10, column=0, columnspan=2)
label_notification['text'] = 'Desired starting temperature
reached.'
button_temp.grid(row=11, column=0, columnspan=2, padx=10,
pady=(5,10), sticky='nesw')

#--- Function: Show notification if desired temperature interval
reached ---
def notification_show(temp):
     label_notification.grid(row=10, column=0, columnspan=2)
label_notification['text'] = 'Take measurement data for
temperature: ' + str(temp) + '°C'
button_notification.grid(row=11, column=0, columnspan=2,
padx=10, pady=(5,10), sticky='nesw')


#--- Function: Reset notification and remove reached temperature
interval from array ---
def notification_reset():
```

```
                global temp_array, csv_appended


                    csv_appended = False


                label_notification.grid_forget()
                button_notification.grid_forget()



#--- Function: Reset notification when desired starting
temperature reached ---
def notification_tempreset():
                label_notification.grid_forget()
                  button_temp.grid_forget()


#----- END: Notification functions -----


#----- Main loop function -----
def main_loop():
global temp_reached, temp_array, temp_rod, temp_smoothed,
temp_step


                          plot_data()
            temp_smoothed = np.average(temp_rod[-5:])
val_tempsmoothed['text'] = str(round(temp_smoothed, 2)) + ' °C'
                    notification_check()
```

```
                    if not temp_reached:
                if temp_smoothed > temp_array[0] + temp_step:
                    temp_reached = True
                    notification_temp()
                        print('LOW')
            append_csv(datetime.now(), round(temp_smoothed, 2))


                    root.after(100, main_loop)

#----- END: Main loop function -----


#----- Main GUI code -----


#--- Initialize window ---
root = tk.Tk()
root.title('Real Time Plot')
root.configure(background = 'light blue')
root.columnconfigure(0, weight=2)
root.columnconfigure(1, weight=8)
root.rowconfigure(0, weight=1)
root.resizable(False, False)


frame_buttons = tk.Frame(root)
frame_buttons.grid(row=0, column=0, sticky='nws')
frame_buttons.configure(background = 'light blue')
```

```
frame_plots = tk.Frame(root)

frame_plots.grid(row=0, column=1, sticky='nes')

frame_plots.configure(background = 'light blue')


#--- Initialize widgets ---


#- Heater buttons -

label_heater = tk.Label(frame_buttons, text= "Heater Controls",
font = ('calibri', 14), background = 'light blue')

label_heater.grid(row=0, column=0, columnspan=2)


button_heateron    =    tk.Button(frame_buttons,    text="On",
font=('calibri', 12), command =lambda:heater(), width=20)

button_heateron.grid(row=1, column=0, padx=10, pady=(5,10))


button_heateroff    =    tk.Button(frame_buttons,    text="Off",
font=('calibri', 12), command=lambda:heateroff(), width=20)

button_heateroff.grid(row=1, column=1, padx=10, pady=(5,10))


#- Plotting buttons -

label_plot = tk.Label(frame_buttons, text= "Plot Controls", font
= ('calibri', 14), background = 'light blue')

label_plot.grid(row=2, column=0, columnspan=2)


button_plotstart    =    tk.Button(frame_buttons,    text="Start",
font=('calibri', 12), command =lambda:plot_start(), width=20)

button_plotstart.grid(row=3, column=0, padx=10, pady=(5,10))
```

```
button_plotstop = tk.Button(frame_buttons, text="Stop", font=('calibri', 12), command=lambda:plot_stop(), width=20)
button_plotstop.grid(row=3, column=1, padx=10, pady=(5,10))


#- Notification buttons -
label_notification = tk.Label(frame_buttons, text= "Notification Controls", font = ('calibri', 14), background = 'light blue')
label_notification.grid(row=4, column=0, columnspan=2)


label_tempstart = tk.Label(frame_buttons, text= "Temperature start (°C)", font = ('calibri', 12), background = 'light blue')
label_tempstart.grid(row=5, column=0, ipadx=5, ipady=5, sticky='e')


entry_tempstart = tk.Entry(frame_buttons, width=20)
entry_tempstart.grid(row=5, column=1, ipadx=5, ipady=5, sticky='w')


label_tempstep = tk.Label(frame_buttons, text= "Temperature step (°C)", font = ('calibri', 12), background = 'light blue')
label_tempstep.grid(row=6, column=0, ipadx=5, ipady=5, sticky='e')


entry_tempstep = tk.Entry(frame_buttons, width=20)
entry_tempstep.grid(row=6, column=1, ipadx=5, ipady=5, sticky='w')
```

```
label_tempdpoints = tk.Label(frame_buttons, text= "Number of data points", font = ('calibri', 12), background = 'light blue')
label_tempdpoints.grid(row=7, column=0, ipadx=5, ipady=5, sticky='e')


entry_tempdpoints = tk.Entry(frame_buttons, width=20)
entry_tempdpoints.grid(row=7, column=1, ipadx=5, ipady=5, sticky='w')


label_validation = tk.Label(frame_buttons, font = ('calibri', 12), background = 'light blue', wraplength=300)


button_tempsave = tk.Button(frame_buttons, text="Save", font=('calibri', 12), command =lambda:temperature_save())
button_tempsave.grid(row=9, column=0, columnspan=2, padx=10, pady=(5,10), sticky='nesw')
```



```
label_notification = tk.Label(frame_buttons, font = ('calibri', 12), background = 'orange', wraplength=300)


button_notification = tk.Button(frame_buttons, text="Confirm", font=('calibri', 12), command =lambda:notification_reset())


button_temp = tk.Button(frame_buttons, text="Confirm", font=('calibri', 12), command =lambda:notification_tempreset())
```

```
label_tempsmoothed = tk.Label(frame_buttons, text= "Temperature smoothed (°C)", font = ('calibri', 12), background = 'light blue')

label_tempsmoothed.grid(row=13, column=0, ipadx=5, ipady=5, sticky='e')


val_tempsmoothed = tk.Label(frame_buttons, font = ('calibri', 12), background = 'light blue')

val_tempsmoothed.grid(row=13, column=1, ipadx=5, ipady=5, sticky='w')


root.update()


#--- Initialize plots ---

fig = Figure()

ax = fig.add_subplot(111)

plot_rod([], [])

canvas = FigureCanvasTkAgg(fig, master=frame_plots)

canvas.get_tk_widget().grid(row=0, column=0, padx=10, pady=10, sticky='nesw')

canvas.draw()


fig1 = Figure()

ax1 = fig1.add_subplot(111)

plot_ambient([],[])

canvas1 = FigureCanvasTkAgg(fig1, master=frame_plots)

canvas1.get_tk_widget().grid(row=0, column=1, padx=10, pady=10, sticky='nesw')
```

```
canvas1.draw()


#--- Start serial port ---

s = ser.Serial('COM3',9600)


root.mainloop()
```

III. Experiment Data

Experiment 1		
Type of rod	**Aluminium**	
Initial length of rod	**0.7 meters**	
Initial Temperature (Ti)	**Final Temperature (Tf)**	**Change in Length (Δl) (in mm)**
76	**71**	**0.078**
71	**66**	**0.075**
66	**61**	**0.079**
60	**55**	**0.079**
55	**50**	**0.074**
50	**45**	**0.071**
44	**39**	**0.081**
Average Coefficient of linear thermal expansion	**2.171 * 10-5 m/(moC)**	
Experiment 2		
Type of rod	**Aluminium**	
Initial length of rod	**0.7 meters**	
Initial Temperature (Ti)	**Final Temperature (Tf)**	**Change in Length (Δl) (in mm)**
76	**71**	**0.077**
71	**66**	**0.078**
66	**61**	**0.076**
60	**55**	**0.07**

55	**50**	**0.081**
50	**45**	**0.078**
44	**39**	**0.07**
Average Coefficient of linear thermal expansion	**2.19 *10-5 m/(moC)**	
Experiment 3		
Type of rod	**Aluminium**	
Initial length of rod	**0.7 meters**	
Initial Temperature (Ti)	**Final Temperature (Tf)**	**Change in Length (Δl) (in mm)**
76	**71**	**0.008**
71	**66**	**0.071**
66	**61**	**0.077**
60	**55**	**0.066**
55	**50**	**0.066**
50	**45**	**0.069**
44	**39**	**0.077**
Average Coefficient of linear thermal expansion	**2.19* 10-5 m/(moC)**	
Experiment 4		
Type of rod	**Aluminium**	
Initial length of rod	**0.7 meters**	
Initial Temperature (Ti)	**Final Temperature (Tf)**	**Change in Length (Δl) (in mm)**
76	**71**	**0.066**
71	**66**	**0.069**
66	**61**	**0.071**
60	**55**	**0.08**
55	**50**	**0.071**
50	**45**	**0.072**
44	**39**	**0.076**
Average Coefficient of linear thermal expansion	**2.042* 10-5 m/(moC)**	
Experiment 5		
Type of rod	**Aluminium**	
Initial length of rod	**0.7 meters**	

Initial Temperature (Ti)	Final Temperature (Tf)	Change in Length (Δl) (in mm)
76	71	0.069
71	66	0.069
66	61	0.071
60	55	0.075
55	50	0.075
50	45	0.069
44	39	0.076
Average Coefficient of linear thermal expansion	2.038*10-5 m/(moC)	
Experiment 6		
Type of rod	Aluminium	
Initial length of rod	0.7 meters	
Initial Temperature (Ti)	Final Temperature (Tf)	Change in Length (Δl) (in mm)
76	71	0.079
71	66	0.078
66	61	0.077
60	55	0.077
55	50	0.075
50	45	0.07
44	39	0.071
Average Coefficient of linear thermal expansion	2.151*10-5 m/(moC)	
Experiment 7		
Type of rod	Aluminium	
Initial length of rod	0.7 meters	
Initial Temperature (Ti)	Final Temperature (Tf)	Change in Length (Δl) (in mm)
76	71	0.081
71	66	0.08
66	61	0.07
60	55	0.073
55	50	0.07

50	**45**	**0.072**
44	**39**	**0.065**
Average Coefficient of linear thermal expansion	**2.086*10-5 m/(moC)**	
Experiment 8		
Type of rod	**Aluminium**	
Initial length of rod	**0.7 meters**	
Initial Temperature (Ti)	**Final Temperature (Tf)**	**Change in Length (Δl) (in mm)**
76	**71**	**0.065**
71	**66**	**0.068**
66	**61**	**0.079**
60	**55**	**0.068**
55	**50**	**0.077**
50	**45**	**0.078**
44	**39**	**0.067**
Average Coefficient of linear thermal expansion	**2.049*10-5 m/(moC)**	
Experiment 9		
Type of rod	**Aluminium**	
Initial length of rod	**0.7 meters**	
Initial Temperature (Ti)	**Final Temperature (Tf)**	**Change in Length (Δl) (in mm)**
76	**71**	**0.074**
71	**66**	**0.072**
66	**61**	**0.07**
60	**55**	**0.077**
55	**50**	**0.077**
50	**45**	**0.068**
44	**39**	**0.073**
Average Coefficient of linear thermal expansion	**2.086*10-5 m/(moC)**	
Experiment 10		
Type of rod	**Aluminium**	
Initial length of rod	**0.7 meters**	

Initial Temperature (Ti)	Final Temperature (Tf)	Change in Length (Δl) (in mm)
76	71	0.076
71	66	0.066
66	61	0.07
60	55	0.078
55	50	0.073
50	45	0.073
44	39	0.073
Average Coefficient of linear thermal expansion	2.077*10-5 m/(moC)	

IV. Experiment Module

Coefficient of Linear Thermal Expansion

Objectives

- To specify the coefficient of linear thermal expansion of metals

Theoretical Introduction

When a piece of an object is being heated or cooled it naturally expands towards all sides. Linear thermal expansion is the expansion of an object that is being heated or cooled towards a single axis. The change in the length towards a specific axis follows the following equation:

$$\Delta L=\alpha\times\Delta T\times L0$$

Where ΔT is the change in temperature, Δl is the change in length of the material, lo is the initial length of the material and α is the coefficient of linear expansion. The coefficient of linear expansion increases the increase in length of a specific material per 1-degree Celsius.

To determine the coefficient of linear expansion the equation can be written as:

$$a=l\text{-}lo/(lo\times(T\text{-}To))$$

Equipment Set Used

- Linear thermal expansion apparatus
- Copper, Aluminium, and Steel rods

- Dial Gauge

Procedure

1. Set up the linear thermal expansion apparatus as the images suggests:
2. Measure the initial length of each metal rods
3. Unfasten the dial gauge part and place the rod into the apparatus
4. Place the dial gauge part to touch the rod and check its distance
5. Let the rod heats up with the apparatus
6. When the temperature reaches the desired temperature open the hatch of the apparatus to use the ambient temperature as a cooling mechanism
7. As the rod cools down take notes on the difference of the length and the change of the temperature
8. Repeat until the data taken is sufficient
9. Repeat step 3 for each other metal rods
10. Turn off the apparatus when it's done.

Experiment Data

Type of rod	**Aluminium**	
Initial length of rod	**0.7 meters**	
Initial Temperature (Ti)	**Final Temperature (Tf)**	**Change in Length (Δl) (in mm)**
Average Coefficient of linear thermal expansion	**m/(moC)**	

REFERENCES

References:

[1] Fullerton, D., Schulte, T. and King, K., 2017. Thermal Expansion. [online] Aplusphysics.com. Available at: <https://aplusphysics.com/courses/honors/thermo/expansion.html> [Accessed 5 February 2021].

[2] Engineeringtoolbox.com. 2003. Coefficients of Linear Thermal Expansion. [online] Available at: <https://www.engineeringtoolbox.com/linear-expansion-coefficients-d_95.html> [Accessed 5 February 2021].

[3] Drake, G., 2021. thermodynamics. [online] Encyclopedia Britannica. Available at: <https://www.britannica.com/science/thermodynamics> [Accessed 6 June 2021].

[4] n.d. Thermal expansion of solid bodies. [ebook] Hürth: LD DIDACTIC GmbH, p.4. Available at: <https://www.ld-didactic.de/documents/en-US/EXP/P/P2/P2112_e.pdf?_ga=2.237731976.946945001.1602818715-1070857626.1600976601> [Accessed 10 March 2021].

[5] n.d. Thermal Expansion Apparatus. [ebook] Roseville: Pasco, p.12. Available at: <https://d2n0lz049icia2.cloudfront.net/product_document/Thermal-Expansion-Apparatus-Manual-TD-8856.pdf> [Accessed 10 March 2021].

[6] Bhapat, B., 2017. Experimental Physics II. [ebook] Pashan: IISER PUNE, p.1. Available at: <http://www.iiserpune.ac.in/~bhasbapat/phy221_files/phy221_2017_manual.pdf> [Accessed 10 March 2021].

[7] 2011. Coefficient of Linear Thermal Expansion. [ebook] Glassboro: Rowan University, p.7. Available at: <http://users.rowan.edu/~klassen/dpa/current/IntroLabs/LinearExpansion.pdf> [Accessed 10 March 2021].

CURRICULUM VITAE

Alfian Wildany Putra Harsono

Address: Taman Semanan Indah Blok D10 no.26, Cengkareng, Jakarta Barat
Phone: +62 82114149988
Email: alfianwp.harsono@gmail.com

WORK EXPERIENCE

09/2016 - 11/2016
Programming, PT. Royal Plastindo Megantara

03/2018 – 07/2018
Electrical Engineer, Baust & Co. GmbH-Materialflusssysteme

EDUCATION

2012 - 2015
High School Graduate, Tunas Muda International school

2015 -
University Student, Swiss German University

ADDITIONAL SKILLS

Microsoft Office package: Microsoft Word, Excel, Power Point
Design: Solidworks, Proteus
Programming: C++

REFERENCES

References available on request