**Comments on the relationship between the acceleration and applied power voltage (linearity, non-linearity, etc). (5 pts)**

There is a non linear relationship, but the more power you send faster it will rotate the motor.

**Python**

# NOTE 1

# If your power supply goes into an error state (i.e., the word

# error is printed on the front of the device), use this command

# power\_supply.write("\*CLS")

# to clear the error so that you can rerun your code. The supply

# typically beeps after an error has occured.

# import various libraries necessary to run your Python code

import time # time related library

import sys,os # system related library

import numpy as np

import pyvisa as visa

import matplotlib as mpl

import matplotlib.pyplot as plt

mpl.style.use('ggplot')

ok\_sdk\_loc = "C:\\Program Files\\Opal Kelly\\FrontPanelUSB\\API\\Python\\x64"

ok\_dll\_loc = "C:\\Program Files\\Opal Kelly\\FrontPanelUSB\\API\\lib\\x64"

sys.path.append(ok\_sdk\_loc) # add the path of the OK library

os.add\_dll\_directory(ok\_dll\_loc)

import ok # OpalKelly library

#%%

# Define FrontPanel device variable, open USB communication and

# load the bit file in the FPGA

dev = ok.okCFrontPanel() # define a device for FrontPanel communication

SerialStatus=dev.OpenBySerial("") # open USB communication with the OK board

# We will NOT load the bit file because it will be loaded using JTAG interface from Vivado

# Check if FrontPanel is initialized correctly and if the bit file is loaded.

# Otherwise terminate the program

print("----------------------------------------------------")

if SerialStatus == 0:

print ("FrontPanel host interface was successfully initialized.")

else:

print ("FrontPanel host interface not detected. The error code number is:" +

str(int(SerialStatus)))

print("Exiting the program.")

sys.exit ()

#%%

# Since we are using a slow clock on the FPGA to compute the results

# we need to wait for the result to be computed

#0 is a write, 1 is a read

#address 6 was made to be PC\_control because we were trying some things.

#this is where we begin writing to the registers to enable both the Gyroscope and the

#Accelerometer

def Use\_Sensor(R\_or\_W, dev\_addr, sub\_addr, read\_count, data, dev):

########### Writes the Contrl Reg ############################

dev.SetWireInValue(0x01, read\_count) #how many time to iterate

dev.SetWireInValue(0x02, R\_or\_W) #Decides read or write

dev.SetWireInValue(0x03, data) # Data to write

dev.SetWireInValue(0x04, sub\_addr) #Sets the Sub Adress

dev.SetWireInValue(0x05, dev\_addr) #Sets the Dev Adress

dev.UpdateWireIns()

dev.SetWireInValue(0x00,1) #Starts FSM

dev.UpdateWireIns()

dev.SetWireInValue(0x00,0) #Turns FSM Off

dev.UpdateWireIns()

dev.UpdateWireOuts()

if(read\_count == 1):

addr = 0x21

elif(read\_count == 2):

addr = 0x22

elif(read\_count == 3):

addr = 0x23

elif(read\_count == 4):

addr = 0x24

elif(read\_count == 5):

addr = 0x25

elif(read\_count == 6):

addr = 0x26

else:

return

if (R\_or\_W == 1):

x = dev.GetWireOutValue(addr)

return x

else:

return

def twobit(m1, m2):

val\_x = (bin(m1 + (m2 << 8)))

if((int(val\_x, 2) >> 15)):

val\_x = int(val\_x, 2) - (1<<16)

else:

val\_x = int(val\_x, 2)

return val\_x

def Moveaccel(direction, time,dev):

dev.SetWireInValue(0x07,0)

dev.SetWireInValue(0x06,direction)

dev.UpdateWireIns()

X\_A = np.zeros(5)

Y\_A = np.zeros(5)

Z\_A = np.zeros(5)

for i in range(5):

dev.SetWireInValue(0x07,1)

dev.UpdateWireIns()

Use\_Sensor(0, 0x32, 0x20, 1, 0x97, dev) #write to Lin Accell CTRL\_REG\_A 8'b01010111

time.sleep(.0007)

X\_L\_A = Use\_Sensor(1, 0x33, 0x28, 1, 0, dev) #Reads Lin Al from X Low

time.sleep(.0007)

X\_H\_A = Use\_Sensor(1, 0x33, 0x29, 2, 0, dev)

time.sleep(.0007)

Y\_L\_A = Use\_Sensor(1, 0x33, 0x2A, 3, 0, dev)

time.sleep(.0007)

Y\_H\_A = Use\_Sensor(1, 0x33, 0x2B, 4, 0, dev)

time.sleep(.0007)

Z\_L\_A = Use\_Sensor(1, 0x33, 0x2C, 5, 0, dev)

time.sleep(.0007)

Z\_H\_A = Use\_Sensor(1, 0x33, 0x2D, 6, 0, dev)

time.sleep(.0007)

X\_A[i] = twobit(X\_L\_A,X\_H\_A)/16384

Y\_A[i] = twobit(Y\_L\_A,Y\_H\_A)/16384

Z\_A[i] = twobit(Z\_L\_A,Z\_H\_A)/16384

#print('Pulse: ' + str(i))

time.sleep(0.5)

print(X\_A,Y\_A,Z\_A)

dev.SetWireInValue(0x07,0)

dev.UpdateWireIns()

return (np.mean(X\_A), np.max(X\_A), np.mean(Y\_A), np.max(Y\_A), np.mean(Z\_A), np.max(Z\_A))

def Movemag(direction, time,dev):

dev.SetWireInValue(0x07,0)

dev.SetWireInValue(0x06,direction)

dev.UpdateWireIns()

time.sleep(1)

X\_A = np.zeros(5)

Y\_A = np.zeros(5)

Z\_A = np.zeros(5)

for i in range(5):

dev.SetWireInValue(0x07,1)

dev.UpdateWireIns()

Use\_Sensor(0, 0x3C, 0x02, 1, 0x00, dev)

time.sleep(0.0007)

X\_H\_M = Use\_Sensor(1, 0x3C, 0x03, 1, 0, dev) #Reads Mag Field X-Y-Z

time.sleep(.00007)

X\_L\_M = Use\_Sensor(1, 0x3C, 0x04, 2, 0, dev)

time.sleep(.00007)

Z\_H\_M = Use\_Sensor(1, 0x3C, 0x05, 3, 0, dev)

time.sleep(.00007)

Z\_L\_M = Use\_Sensor(1, 0x3C, 0x06, 4, 0, dev)

time.sleep(.00007)

Y\_H\_M = Use\_Sensor(1, 0x3C, 0x07, 5, 0, dev)

time.sleep(.00007)

Y\_L\_M = Use\_Sensor(1, 0x3C, 0x08, 6, 0, dev)

X\_A[i] = twobit(X\_L\_M,X\_H\_M)/1000

Y\_A[i] = twobit(Y\_L\_M,Y\_H\_M)/1000

Z\_A[i] = twobit(Z\_L\_M,Z\_H\_M)/1000

#print('Pulse: ' + str(i))

time.sleep(0.5)

dev.SetWireInValue(0x07,0)

dev.UpdateWireIns()

return (np.mean(X\_A), np.max(X\_A), np.mean(Y\_A), np.max(Y\_A), np.mean(Y\_A), np.max(Y\_A))

#%%

# This section of the code cycles through all USB connected devices to the computer.

# The code figures out the USB port number for each instrument.

# The port number for each instrument is stored in a variable named “instrument\_id”

# If the instrument is turned off or if you are trying to connect to the

# keyboard or mouse, you will get a message that you cannot connect on that port.

device\_manager = visa.ResourceManager()

devices = device\_manager.list\_resources()

number\_of\_device = len(devices)

power\_supply\_id = -1;

waveform\_generator\_id = -1;

digital\_multimeter\_id = -1;

oscilloscope\_id = -1;

# assumes only the DC power supply is connected

for i in range (0, number\_of\_device):

# check that it is actually the power supply

try:

device\_temp = device\_manager.open\_resource(devices[i])

print("Instrument connect on USB port number [" + str(i) + "] is " + device\_temp.query("\*IDN?"))

if (device\_temp.query("\*IDN?") == 'HEWLETT-PACKARD,E3631A,0,3.2-6.0-2.0\r\n'):

power\_supply\_id = i

if (device\_temp.query("\*IDN?") == 'HEWLETT-PACKARD,E3631A,0,3.0-6.0-2.0\r\n'):

power\_supply\_id = i

if (device\_temp.query("\*IDN?") == 'Agilent Technologies,33511B,MY52301259,3.03-1.19-2.00-52-00\n'):

waveform\_generator\_id = i

if (device\_temp.query("\*IDN?") == 'Agilent Technologies,34461A,MY53207926,A.01.10-02.25-01.10-00.35-01-01\n'):

digital\_multimeter\_id = i

if (device\_temp.query("\*IDN?") == 'Keysight Technologies,34461A,MY53212931,A.02.08-02.37-02.08-00.49-01-01\n'):

digital\_multimeter\_id = i

if (device\_temp.query("\*IDN?") == 'KEYSIGHT TECHNOLOGIES,MSO-X 3024T,MY54440281,07.10.2017042905\n'):

oscilloscope\_id = i

device\_temp.close()

except:

print("Instrument on USB port number [" + str(i) + "] cannot be connected. The instrument might be powered of or you are trying to connect to a mouse or keyboard.\n")

#%%

# Open the USB communication port with the power supply.

# The power supply is connected on USB port number power\_supply\_id.

# If the power supply ss not connected or turned off, the program will exit.

# Otherwise, the power\_supply variable is the handler to the power supply

if (power\_supply\_id == -1):

print("Power supply instrument is not powered on or connected to the PC.")

else:

print("Power supply is connected to the PC.")

power\_supply = device\_manager.open\_resource(devices[power\_supply\_id])

#%%

# The power supply output voltage will be swept from 0 to 1.5V in steps of 0.05V.

# This voltage will be applied on the 6V output ports.

# For each voltage applied on the 6V power supply, we will measure the actual

# voltage and current supplied by the power supply.

# If your circuit operates correctly, the applied and measured voltage will be the same.

# If the power supply reaches its maximum allowed current,

# then the applied voltage will not be the same as the measured voltage.

output\_voltage = np.arange(3, 5, 0.5)

X\_Mean= np.array([]) # create an empty list to hold our values

Y\_Mean= np.array([]) # create an empty list to hold our values

Z\_Mean= np.array([]) # create an empty list to hold our values

X\_Max = np.array([]) # create an empty list to hold our values

Y\_Max = np.array([]) # create an empty list to hold our values

Z\_Max = np.array([]) # create an empty list to hold our values

print(power\_supply.write("OUTPUT ON")) # power supply output is turned on

# loop through the different voltages we will apply to the power supply

# For each voltage applied on the power supply,

# measure the voltage and current supplied by the 6V power supply

for v in output\_voltage:

# apply the desired voltage on teh 6V power supply and limist the output current to 0.5A

power\_supply.write("APPLy P6V, %0.2f, 0.5" % v)

# pause 50ms to let things settle

time.sleep(0.5)

x\_mean,x\_max,y\_mean,y\_max,z\_mean,z\_max = Moveaccel(1, time,dev)

# read the output voltage on the 6V power supply

X\_Mean = np.append(x\_mean)

Y\_Mean = np.append(y\_mean)

Z\_Mean = np.append(z\_mean)

X\_Max = np.append(x\_max)

Y\_Max = np.append(y\_max)

Z\_Max = np.append(z\_max)

# read the output current on the 6V power supply

# power supply output is turned off

print(power\_supply.write("OUTPUT OFF"))

# close the power supply USB handler.

# Otherwise you cannot connect to it in the future

power\_supply.close()

#%% Plot measured data. First convert the data from strings to numbers (ie floats)

X\_Mean\_list=np.zeros(np.size(output\_voltage))

Y\_Mean\_list=np.zeros(np.size(output\_voltage))

Z\_Mean\_list=np.zeros(np.size(output\_voltage))

X\_Max\_list=np.zeros(np.size(output\_voltage))

Y\_Max\_list=np.zeros(np.size(output\_voltage))

Z\_Max\_list=np.zeros(np.size(output\_voltage))

for i in range(len(X\_Mean)):

X\_Mean\_list[i]= float(X\_Mean [i])

Y\_Mean\_list=float(Y\_Mean [i])

Z\_Mean\_list[i]= float(Z\_Mean [i])

X\_Max\_list=float(X\_Max [i])

Y\_Max\_list=float(Y\_Max [i])

Z\_Max\_list=float(Z\_Max [i])

# plot results (applied voltage vs measured supplied current)

plt.figure()

plt.plot(output\_voltage,X\_Mean\_list )

plt.title("Applied Volts vs. Measured Supplied Current for Diode")

plt.xlabel("Applied Volts [V]")

plt.ylabel("Measured Current [A]")

plt.draw()

# plot results (measured voltage vs measured supplied current)

**Python checkpoint 1**

**# -\*- coding: utf-8 -\*-**

**#%%**

**# import various libraries necessary to run your Python code**

**import time # time related library**

**import sys,os # system related library**

**import numpy as np**

**import pyvisa as visa**

**import matplotlib as mpl**

**import matplotlib.pyplot as plt**

**mpl.style.use('ggplot')**

**ok\_sdk\_loc = "C:\\Program Files\\Opal Kelly\\FrontPanelUSB\\API\\Python\\x64"**

**ok\_dll\_loc = "C:\\Program Files\\Opal Kelly\\FrontPanelUSB\\API\\lib\\x64"**

**sys.path.append(ok\_sdk\_loc) # add the path of the OK library**

**os.add\_dll\_directory(ok\_dll\_loc)**

**import ok # OpalKelly library**

**#%%**

**# Define FrontPanel device variable, open USB communication and**

**# load the bit file in the FPGA**

**dev = ok.okCFrontPanel() # define a device for FrontPanel communication**

**SerialStatus=dev.OpenBySerial("") # open USB communication with the OK board**

**# We will NOT load the bit file because it will be loaded using JTAG interface from Vivado**

**# Check if FrontPanel is initialized correctly and if the bit file is loaded.**

**# Otherwise terminate the program**

**print("----------------------------------------------------")**

**if SerialStatus == 0:**

**print ("FrontPanel host interface was successfully initialized.")**

**else:**

**print ("FrontPanel host interface not detected. The error code number is:" +**

**str(int(SerialStatus)))**

**print("Exiting the program.")**

**sys.exit ()**

**#%%**

**#%%**

**# Since we are using a slow clock on the FPGA to compute the results**

**# we need to wait for the result to be computed**

**#0 is a write, 1 is a read**

**#address 6 was made to be PC\_control because we were trying some things.**

**#this is where we begin writing to the registers to enable both the Gyroscope and the**

**#Accelerometer**

**def Use\_Sensor(R\_or\_W, dev\_addr, sub\_addr, read\_count, data, dev):**

**########### Writes the Contrl Reg ############################**

**dev.SetWireInValue(0x01, read\_count) #how many time to iterate**

**dev.SetWireInValue(0x02, R\_or\_W) #Decides read or write**

**dev.SetWireInValue(0x03, data) # Data to write**

**dev.SetWireInValue(0x04, sub\_addr) #Sets the Sub Adress**

**dev.SetWireInValue(0x05, dev\_addr) #Sets the Dev Adress**

**dev.UpdateWireIns()**

**dev.SetWireInValue(0x00,1) #Starts FSM**

**dev.UpdateWireIns()**

**dev.SetWireInValue(0x00,0) #Turns FSM Off**

**dev.UpdateWireIns()**

**dev.UpdateWireOuts()**

**if(read\_count == 1):**

**addr = 0x21**

**elif(read\_count == 2):**

**addr = 0x22**

**elif(read\_count == 3):**

**addr = 0x23**

**elif(read\_count == 4):**

**addr = 0x24**

**elif(read\_count == 5):**

**addr = 0x25**

**elif(read\_count == 6):**

**addr = 0x26**

**else:**

**return**

**if (R\_or\_W == 1):**

**x = dev.GetWireOutValue(addr)**

**return x**

**else:**

**return**

**def twobit(m1, m2):**

**val\_x = (bin(m1 + (m2 << 8)))**

**if((int(val\_x, 2) >> 15)):**

**val\_x = int(val\_x, 2) - (1<<16)**

**else:**

**val\_x = int(val\_x, 2)**

**return val\_x**

**#6 is direction and 7 is enable**

**'''**

**dev.SetWireInValue(0x06,1)**

**dev.SetWireInValue(0x07,1)**

**dev.UpdateWireIns()**

**time.sleep(1)**

**dev.SetWireInValue(0x06,0)**

**dev.UpdateWireIns()**

**'''**

**def Moveaccel(direction, time,dev):**

**dev.SetWireInValue(0x07,0)**

**dev.SetWireInValue(0x06,direction)**

**dev.UpdateWireIns()**

**for i in range(5):**

**dev.SetWireInValue(0x07,1)**

**dev.UpdateWireIns()**

**Use\_Sensor(0, 0x32, 0x20, 1, 0x97, dev) #write to Lin Accell CTRL\_REG\_A 8'b01010111**

**time.sleep(.0007)**

**X\_L\_A = Use\_Sensor(1, 0x33, 0x28, 1, 0, dev) #Reads Lin Al from X Low**

**time.sleep(.0007)**

**X\_H\_A = Use\_Sensor(1, 0x33, 0x29, 2, 0, dev)**

**time.sleep(.0007)**

**Y\_L\_A = Use\_Sensor(1, 0x33, 0x2A, 3, 0, dev)**

**time.sleep(.0007)**

**Y\_H\_A = Use\_Sensor(1, 0x33, 0x2B, 4, 0, dev)**

**time.sleep(.0007)**

**Z\_L\_A = Use\_Sensor(1, 0x33, 0x2C, 5, 0, dev)**

**time.sleep(.0007)**

**Z\_H\_A = Use\_Sensor(1, 0x33, 0x2D, 6, 0, dev)**

**time.sleep(.0007)**

**X\_A = twobit(X\_L\_A,X\_H\_A)/16384**

**Y\_A = twobit(Y\_L\_A,Y\_H\_A)/16384**

**Z\_A = twobit(Z\_L\_A,Z\_H\_A)/16384**

**#print('Pulse: ' + str(i))**

**time.sleep(0.5)**

**print(X\_A,Y\_A,Z\_A)**

**dev.SetWireInValue(0x07,0)**

**dev.UpdateWireIns()**

**return**

**def Movemag(direction, time,dev):**

**dev.SetWireInValue(0x07,0)**

**dev.SetWireInValue(0x06,direction)**

**dev.UpdateWireIns()**

**time.sleep(1)**

**for i in range(5):**

**dev.SetWireInValue(0x07,1)**

**dev.UpdateWireIns()**

**Use\_Sensor(0, 0x3C, 0x02, 1, 0x00, dev)**

**time.sleep(0.0007)**

**X\_H\_M = Use\_Sensor(1, 0x3C, 0x03, 1, 0, dev) #Reads Mag Field X-Y-Z**

**time.sleep(.00007)**

**X\_L\_M = Use\_Sensor(1, 0x3C, 0x04, 2, 0, dev)**

**time.sleep(.00007)**

**Z\_H\_M = Use\_Sensor(1, 0x3C, 0x05, 3, 0, dev)**

**time.sleep(.00007)**

**Z\_L\_M = Use\_Sensor(1, 0x3C, 0x06, 4, 0, dev)**

**time.sleep(.00007)**

**Y\_H\_M = Use\_Sensor(1, 0x3C, 0x07, 5, 0, dev)**

**time.sleep(.00007)**

**Y\_L\_M = Use\_Sensor(1, 0x3C, 0x08, 6, 0, dev)**

**X\_A = twobit(X\_L\_M,X\_H\_M)/1000**

**Y\_A = twobit(Y\_L\_M,Y\_H\_M)/1000**

**Z\_A = twobit(Z\_L\_M,Z\_H\_M)/1000**

**print(X\_A, Y\_A, Z\_A)**

**#print('Pulse: ' + str(i))**

**time.sleep(0.5)**

**dev.SetWireInValue(0x07,0)**

**dev.UpdateWireIns()**

**return**

**pulses = 10**

**print('Acceleration Data')**

**for j in range(pulses):**

**Moveaccel(1,time,dev)**

**Moveaccel(0,time,dev)**

**print('Magnetic Data')**

**for j in range(pulses):**

**Movemag(1,time,dev)**

**Movemag(0,time,dev)**

**############### Getting Linear Accelleration ###############**

**'''**

**while (1):**

**Use\_Sensor(0, 0x32, 0x20, 1, 0x97, dev) #write to Lin Accell CTRL\_REG\_A 8'b01010111**

**time.sleep(0.5)**

**X\_L\_A = Use\_Sensor(1, 0x33, 0x28, 1, 0, dev) #Reads Lin Al from X Low**

**time.sleep(.02)**

**X\_H\_A = Use\_Sensor(1, 0x33, 0x29, 2, 0, dev)**

**time.sleep(.02)**

**Y\_L\_A = Use\_Sensor(1, 0x33, 0x2A, 3, 0, dev)**

**time.sleep(.02)**

**Y\_H\_A = Use\_Sensor(1, 0x33, 0x2B, 4, 0, dev)**

**time.sleep(.02)**

**Z\_L\_A = Use\_Sensor(1, 0x33, 0x2C, 5, 0, dev)**

**time.sleep(.02)**

**Z\_H\_A = Use\_Sensor(1, 0x33, 0x2D, 6, 0, dev)**

**time.sleep(.02)**

**X\_A = twobit(X\_L\_A,X\_H\_A)/16384**

**Y\_A = twobit(Y\_L\_A,Y\_H\_A)/16384**

**Z\_A = twobit(Z\_L\_A,Z\_H\_A)/16384**

**print(X\_A,Y\_A,Z\_A)**

**time.sleep(.5)**

**############### Getting Magnetic Field ###################**

**while(1):**

**Use\_Sensor(0, 0x3C, 0x02, 1, 0x00, dev)**

**time.sleep(0.5)**

**X\_H\_M = Use\_Sensor(1, 0x3C, 0x03, 1, 0, dev) #Reads Mag Field X-Y-Z**

**time.sleep(.02)**

**X\_L\_M = Use\_Sensor(1, 0x3C, 0x04, 2, 0, dev)**

**time.sleep(.02)**

**Z\_H\_M = Use\_Sensor(1, 0x3C, 0x05, 3, 0, dev)**

**time.sleep(.02)**

**Z\_L\_M = Use\_Sensor(1, 0x3C, 0x06, 4, 0, dev)**

**time.sleep(.02)**

**Y\_H\_M = Use\_Sensor(1, 0x3C, 0x07, 5, 0, dev)**

**time.sleep(.02)**

**Y\_L\_M = Use\_Sensor(1, 0x3C, 0x08, 6, 0, dev)**

**X\_A = twobit(X\_L\_M,X\_H\_M)/1000**

**Y\_A = twobit(Y\_L\_M,Y\_H\_M)/1000**

**Z\_A = twobit(Z\_L\_M,Z\_H\_M)/1000**

**print(X\_A, Y\_A, Z\_A)**

**'''**

**dev**

**dev**

**dev.Close**

**#%%**

**Verilog toplevel —--------------------------------------------------------------------------------------------------------**

**`timescale 1ns / 1ps**

**//////////////////////////////////////////////////////////////////////////////////**

**// Company:**

**// Engineer:**

**//**

**// Create Date: 10/24/2023 03:01:53 PM**

**// Design Name:**

**// Module Name: lab7\_top**

**// Project Name:**

**// Target Devices:**

**// Tool Versions:**

**// Description:**

**//**

**// Dependencies:**

**//**

**// Revision:**

**// Revision 0.01 - File Created**

**// Additional Comments:**

**//**

**//////////////////////////////////////////////////////////////////////////////////**

**module lab7\_top(input sys\_clkn,**

**input sys\_clkp,**

**output wire PMOD\_A1,**

**output wire PMOD\_A2,**

**output I2C\_SCL\_1,**

**inout I2C\_SDA\_1,**

**input wire [4:0] okUH,**

**output wire [2:0] okHU,**

**inout wire [31:0] okUHU,**

**inout wire okAA);**

**//instantiate I2C stuff**

**I2C\_mod I2C\_TR(.sys\_clkn(sys\_clkn),.sys\_clkp(sys\_clkp),.okUH(okUH),.okHU(okHU),.okUHU(okUHU),.okAA(okAA),.DIR1(PMOD\_A2),.pulses(PMOD\_A1),.I2C\_SCL\_1(I2C\_SCL\_1),.I2C\_SDA\_1(I2C\_SDA\_1),.clk(fast\_clk));**

**endmodule**