“Gontrol”

A computerized goniometer for magnetic field optimization of spin resonance of nitrogen vacancies in diamond

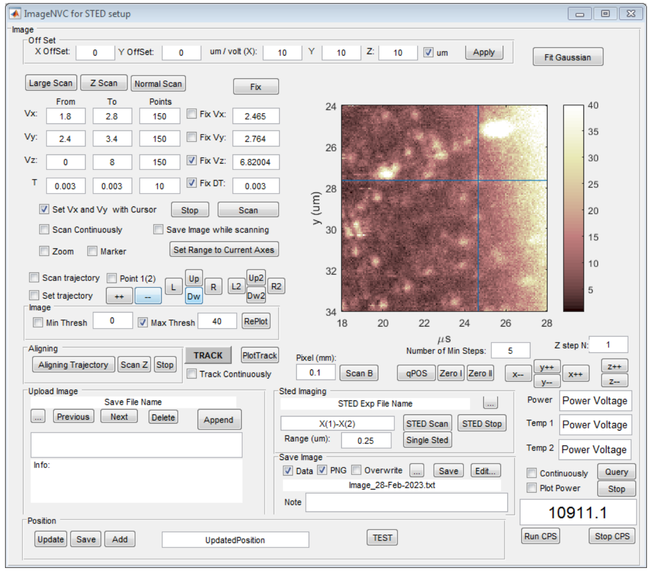
Marshal Dong

05/01/23

Background

Goal:

to improve the Nitrogen Vacancy Center (NV centers) Electron Spin Resonance (ESR) measurement by making magnetic field alignment easier.



ImageNVC for STED setup

Image

Off Set

X OffSet: 0 Y OffSet: 0 um / volt (X): 10 Y 10 Z: 10 ☑ um

	From	To	Points		
Vx:	1.8	2.8	150	□ Fix Vx:	2.465
Vy:	2.4	3.4	150	□ Fix Vy:	2.764
Vz:	0	8	150	☑ Fix Vz:	6.82004
T	0.003	0.003	10	☑ Fix DT:	0.003

☑ Set Vx and Vy with Cursor

□ Scan Continuously □ Save Image while scanning

□ Zoom □ Marker

□ Scan trajectory □ Point 1(2)

□ Set trajectory

Image

□ Min Thresh 0 ☑ Max Thresh 40

Aligning

□ Track Continuously

Pixel (mm): 0.1

Number of Min Steps: 5 Z step N: 1

Upload Image

Save File Name

Info:

Sted Imaging

STED Exp File Name

X(1)-X(2)

Range (um): 0.25

Save Image

☑ Data ☑ PNG □ Overwrite

Image_28-Feb-2023.txt

Note

Power Power Voltage

Temp 1 Power Voltage

Temp 2 Power Voltage

□ Continuously

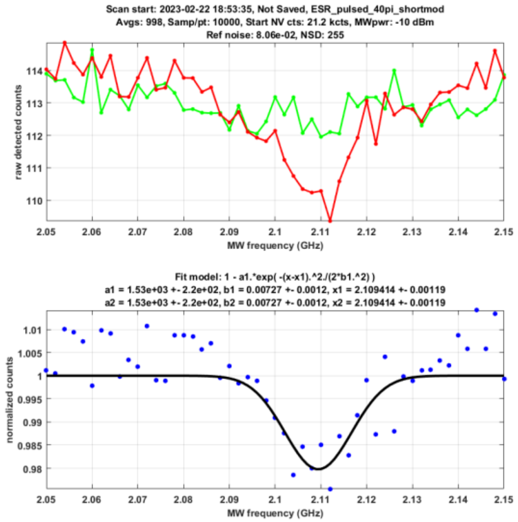
□ Plot Power

10911.1

Position

UpdatedPosition

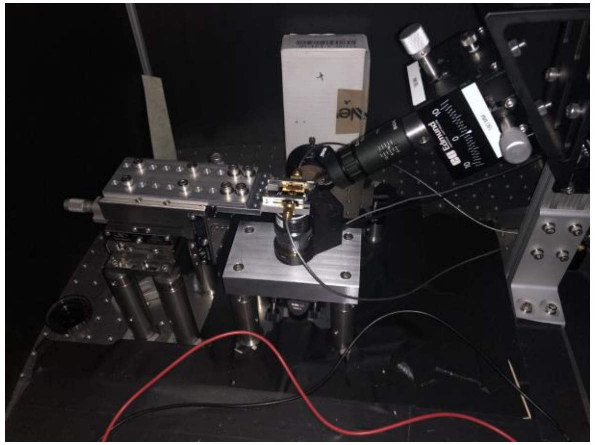
A NV Scan



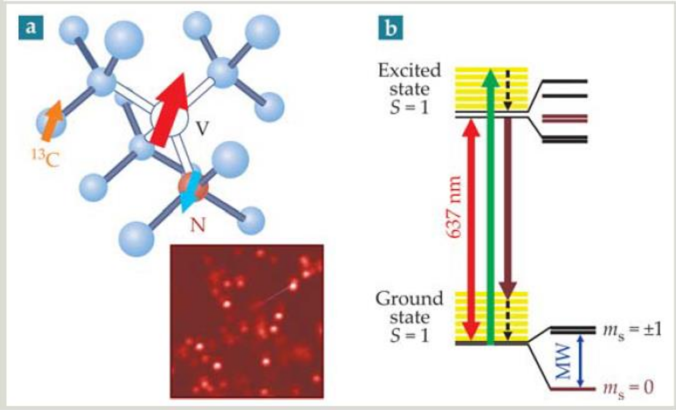
Scan start: 2023-02-22 18:53:35, Not Saved, ESR_pulsed_40pi_shortmod
Avgs: 998, Samp/pt: 10000, Start NV cts: 21.2 kcts, MWpwr: -10 dBm
Ref noise: 8.06e-02, NSD: 255

Fit model: 1 - a1.*exp(-(x-x1).^2./(2*b1.^2))
a1 = 1.53e+03 +- 2.2e+02, b1 = 0.00727 +- 0.0012, x1 = 2.109414 +- 0.00119
a2 = 1.53e+03 +- 2.2e+02, b2 = 0.00727 +- 0.0012, x2 = 2.109414 +- 0.00119

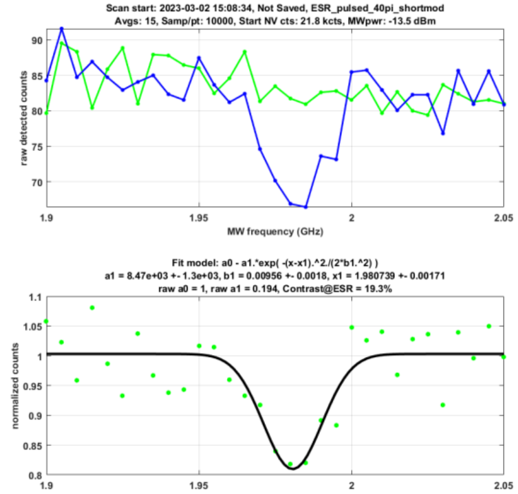
Misaligned Magnet & Poor ESR Contrast



Our Experiment



NV Structure

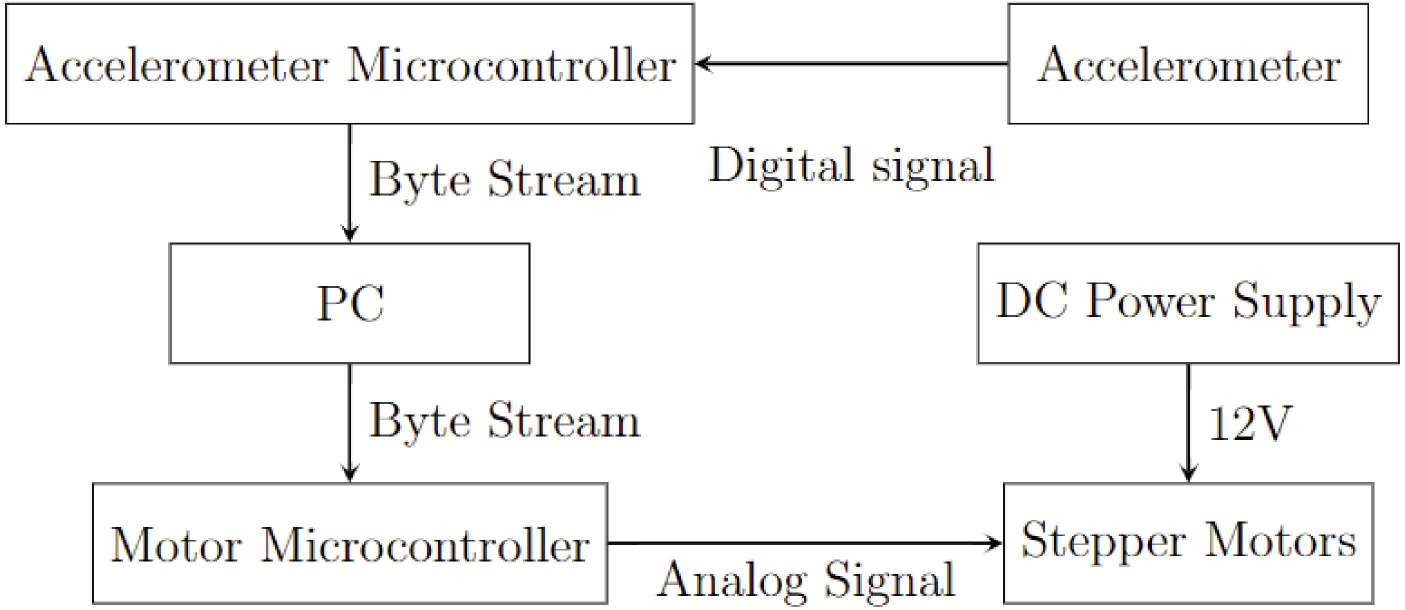


Scan start: 2023-03-02 15:08:34, Not Saved, ESR_pulsed_40pi_shortmod
Avgs: 15, Samp/pt: 10000, Start NV cts: 21.8 kcts, MWpwr: -13.5 dBm

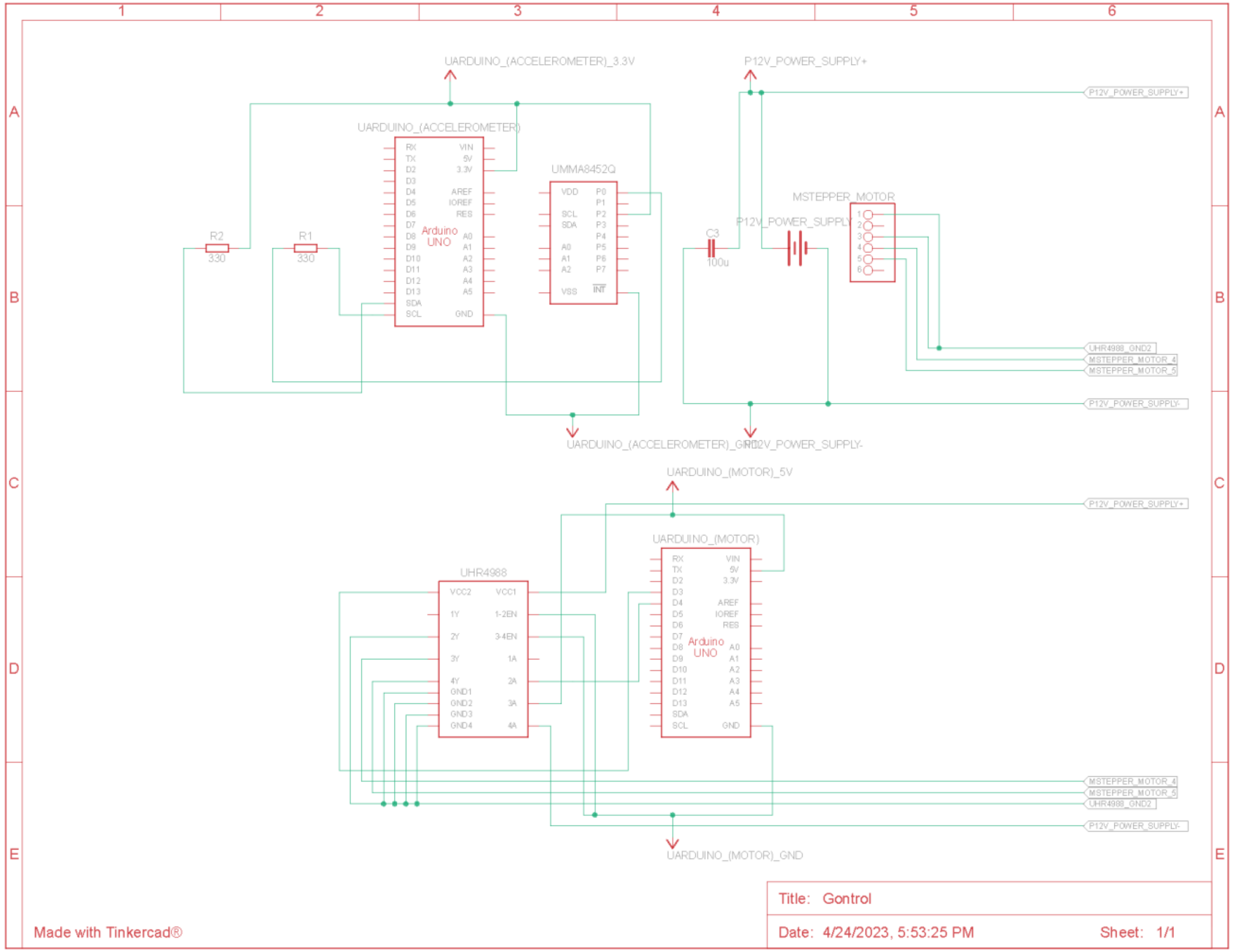
Fit model: a0 - a1.*exp(-(x-x1).^2./(2*b1.^2))
a1 = 8.47e+03 +- 1.3e+03, b1 = 0.00956 +- 0.0018, x1 = 1.980739 +- 0.00171
raw a0 = 1, raw a1 = 0.194, Contrast@ESR = 19.3%

Aligned Magnet & Good ESR Contrast

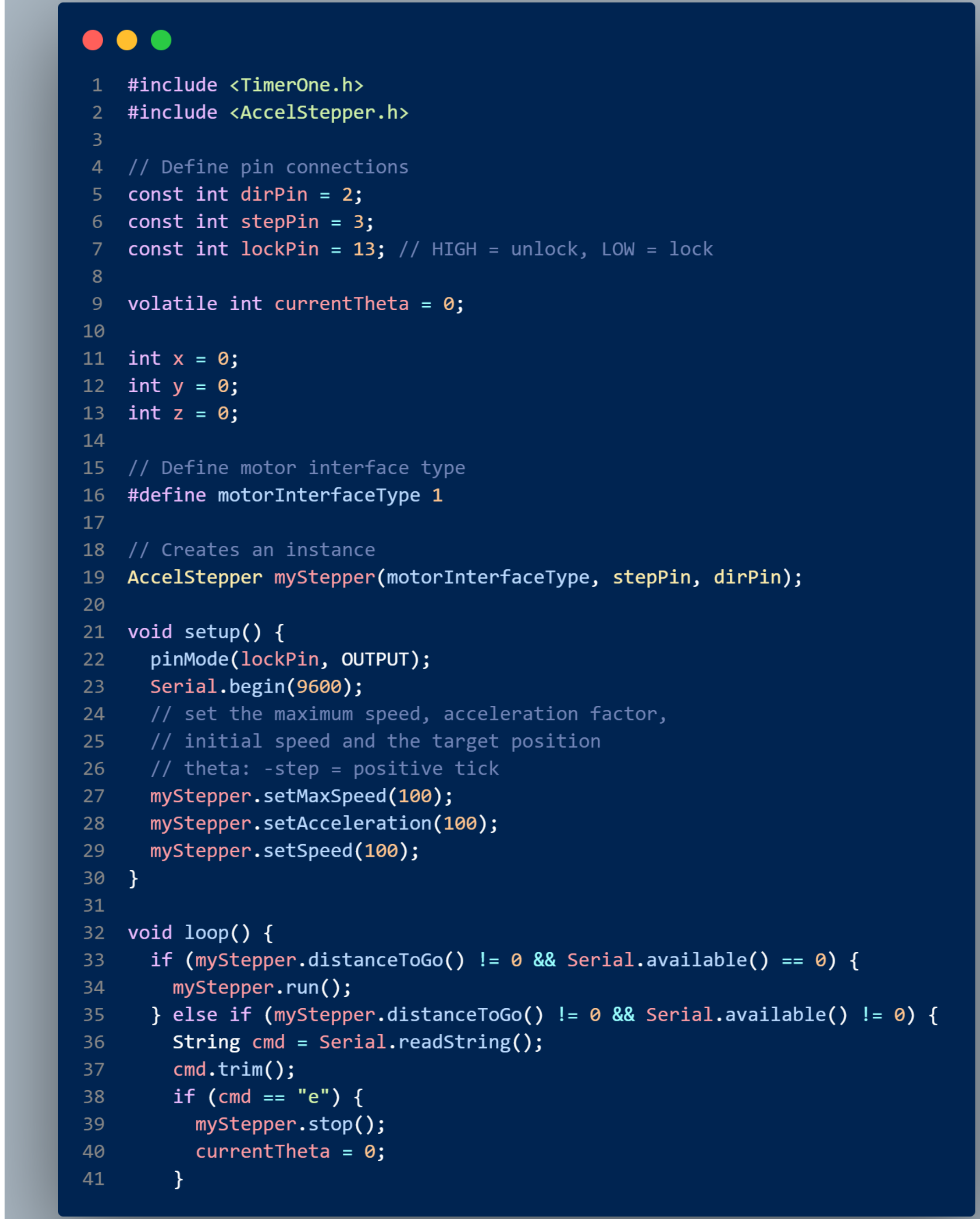
Block Diagram



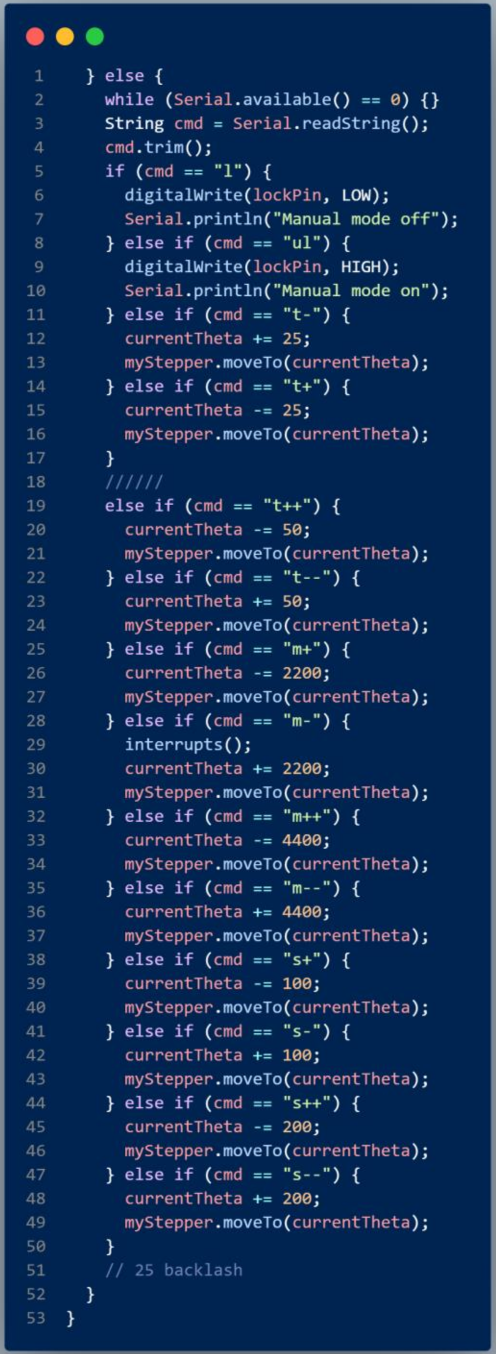
Electronic Schematic



Arduino Code : Stepper Motor

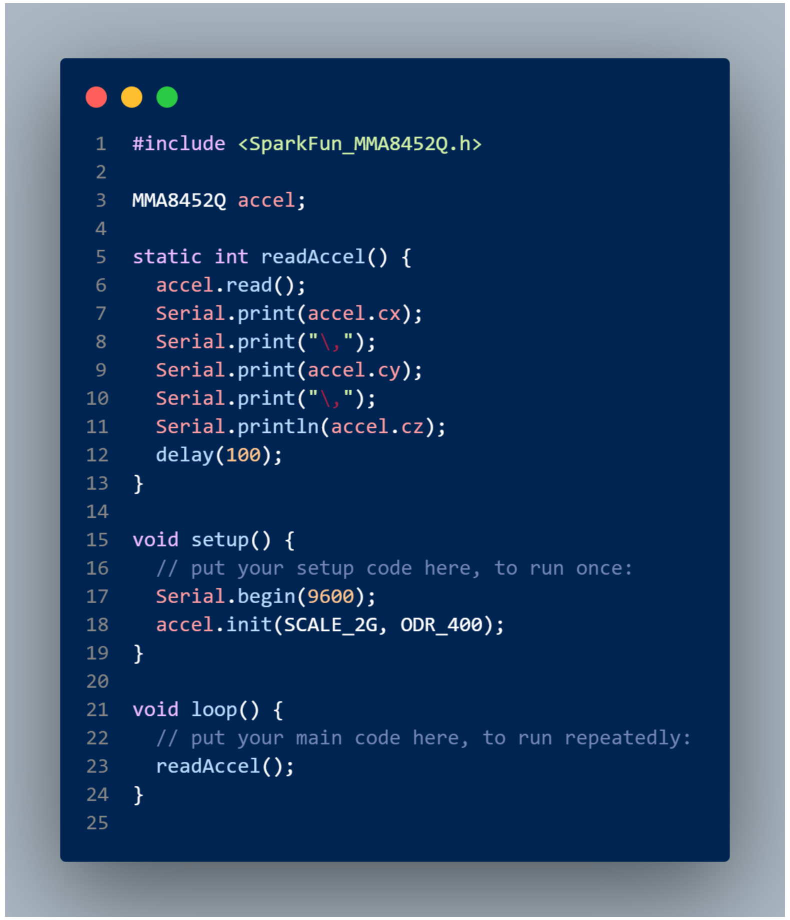


```
#include <TimerOne.h>
#include <AccelStepper.h>

// Define pin connections
const int dirPin = 2;
const int stepPin = 3;
const int lockPin = 13; // HIGH = unlock, LOW = lock

volatile int currentTheta = 0;

int x = 0;
int y = 0;
int z = 0;

// Define motor interface type
#define motorInterfaceType 1

// Creates an instance
AccelStepper myStepper(motorInterfaceType, stepPin, dirPin);

void setup() {
  pinMode(lockPin, OUTPUT);
  Serial.begin(9600);
  // set the maximum speed, acceleration factor,
  // initial speed and the target position
  // theta: -step = positive tick
  myStepper.setMaxSpeed(100);
  myStepper.setAcceleration(100);
  myStepper.setSpeed(100);
}

void loop() {
  if (myStepper.distanceToGo() != 0 && Serial.available() == 0) {
    myStepper.run();
  } else if (myStepper.distanceToGo() != 0 && Serial.available() != 0) {
    String cmd = Serial.readString();
    cmd.trim();
    if (cmd == "e") {
      myStepper.stop();
      currentTheta = 0;
    }
```



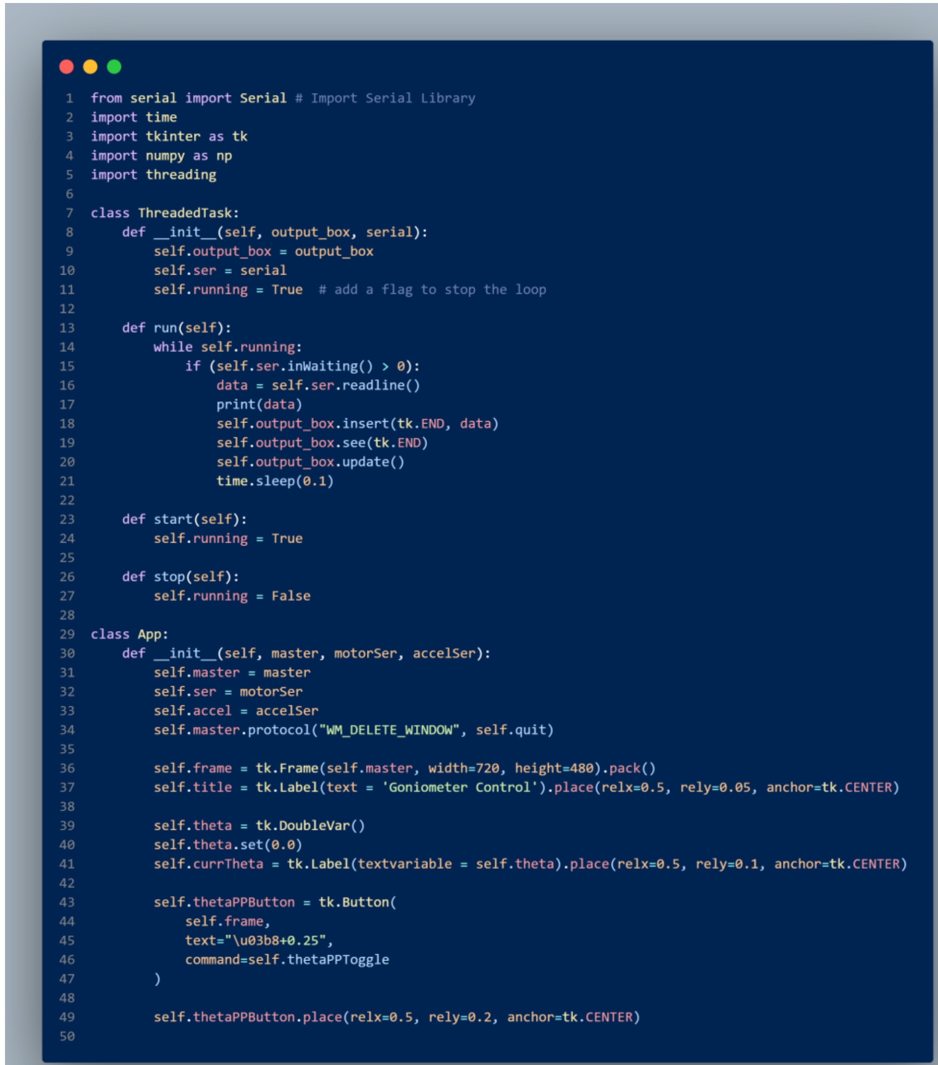
```
  } else {
    while (Serial.available() == 0) {}
    String cmd = Serial.readString();
    cmd.trim();
    if (cmd == "l") {
      digitalWrite(lockPin, LOW);
      Serial.println("Manual mode off");
    } else if (cmd == "ul") {
      digitalWrite(lockPin, HIGH);
      Serial.println("Manual mode on");
    } else if (cmd == "t-") {
      currentTheta += 25;
      myStepper.moveTo(currentTheta);
    } else if (cmd == "t+") {
      currentTheta -= 25;
      myStepper.moveTo(currentTheta);
    }
    //////
    else if (cmd == "t++") {
      currentTheta -= 50;
      myStepper.moveTo(currentTheta);
    } else if (cmd == "t--") {
      currentTheta += 50;
      myStepper.moveTo(currentTheta);
    } else if (cmd == "m+") {
      currentTheta -= 2200;
      myStepper.moveTo(currentTheta);
    } else if (cmd == "m-") {
      interrupts();
      currentTheta += 2200;
      myStepper.moveTo(currentTheta);
    } else if (cmd == "m++") {
      currentTheta -= 4400;
      myStepper.moveTo(currentTheta);
    } else if (cmd == "m--") {
      currentTheta += 4400;
      myStepper.moveTo(currentTheta);
    } else if (cmd == "s+") {
      currentTheta -= 100;
      myStepper.moveTo(currentTheta);
    } else if (cmd == "s-") {
      currentTheta += 100;
      myStepper.moveTo(currentTheta);
    } else if (cmd == "s++") {
      currentTheta -= 200;
      myStepper.moveTo(currentTheta);
    } else if (cmd == "s--") {
      currentTheta += 200;
      myStepper.moveTo(currentTheta);
    }
    // 25 backlash
  }
}
```

Arduino Code – Accelerometer

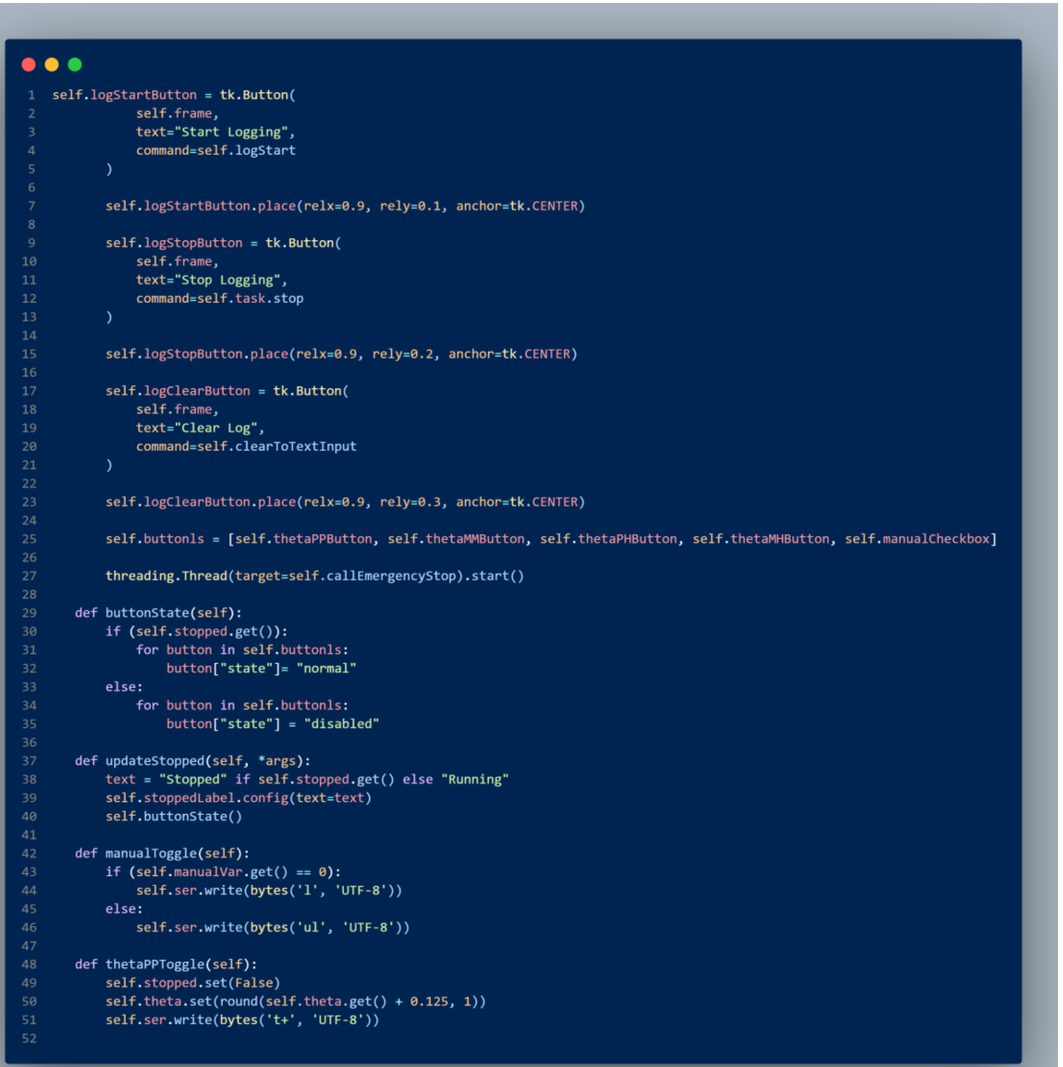


```
#include <SparkFun_MMA8452Q.h>

MMA8452Q accel;

static int readAccel() {
  accel.read();
  Serial.print(accel.cx);
  Serial.print("\,");
  Serial.print(accel.cy);
  Serial.print("\,");
  Serial.println(accel.cz);
  delay(100);
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  accel.init(SCALE_2G, ODR_400);
}

void loop() {
  // put your main code here, to run repeatedly:
  readAccel();
}

```

Python Tkinter GUI Code

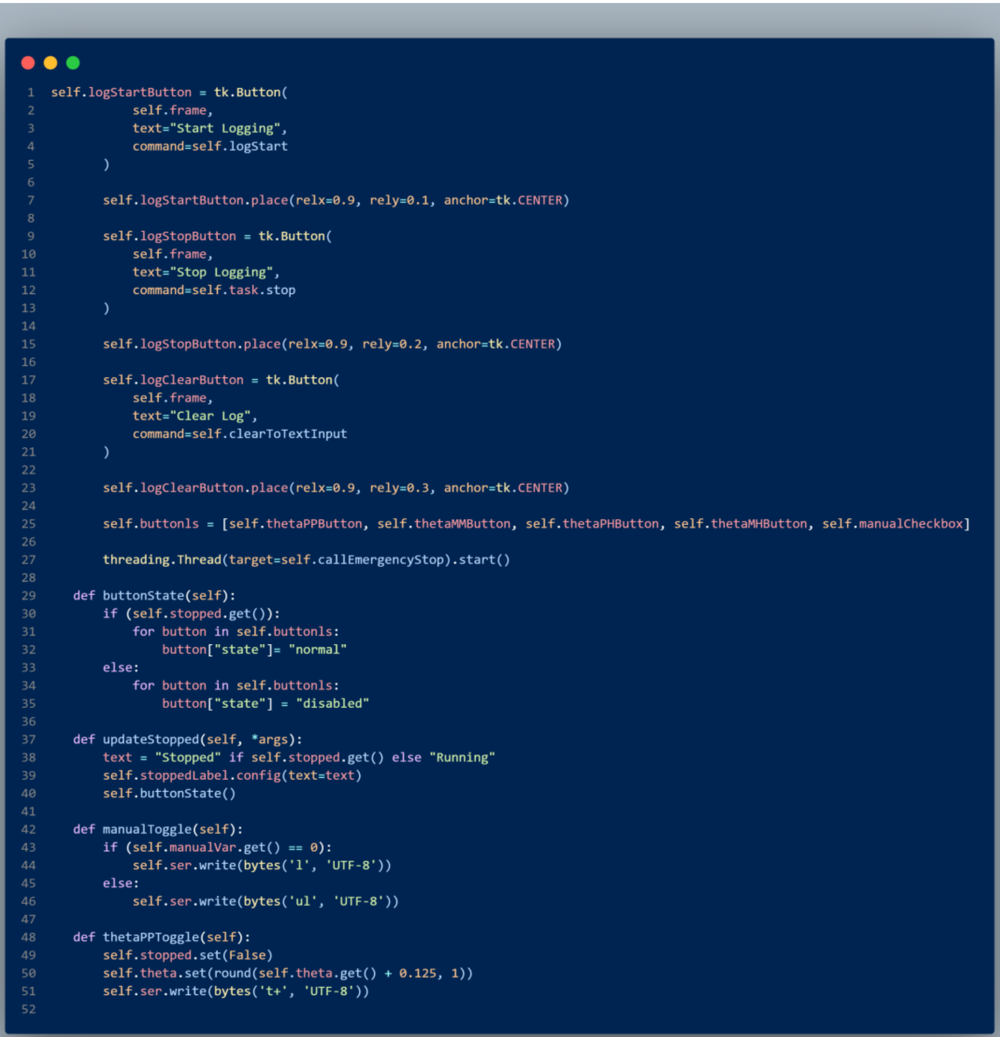


```python
from serial import Serial # Import Serial Library
import time
import tkinter as tk
import numpy as np
import threading

class ThreadedTask:
    def __init__(self, output_box, serial):
        self.output_box = output_box
        self.ser = serial
        self.running = True  # add a flag to stop the loop

    def run(self):
        while self.running:
            if (self.ser.inWaiting() > 0):
                data = self.ser.readline()
                print(data)
                self.output_box.insert(tk.END, data)
                self.output_box.see(tk.END)
                self.output_box.update()
                time.sleep(0.1)

    def start(self):
        self.running = True

    def stop(self):
        self.running = False

class App:
    def __init__(self, master, motorSer, accelSer):
        self.master = master
        self.ser = motorSer
        self.accel = accelSer
        self.master.protocol("WM_DELETE_WINDOW", self.quit)

        self.frame = tk.Frame(self.master, width=720, height=480).pack()
        self.title = tk.Label(text = 'Goniometer Control').place(relx=0.5, rely=0.05, anchor=tk.CENTER)

        self.theta = tk.DoubleVar()
        self.theta.set(0.0)
        self.currTheta = tk.Label(textvariable = self.theta).place(relx=0.5, rely=0.1, anchor=tk.CENTER)

        self.thetaPPButton = tk.Button(
            self.frame,
            text="\u03b8+0.25",
            command=self.thetaPPToggle
        )

        self.thetaPPButton.place(relx=0.5, rely=0.2, anchor=tk.CENTER)

```

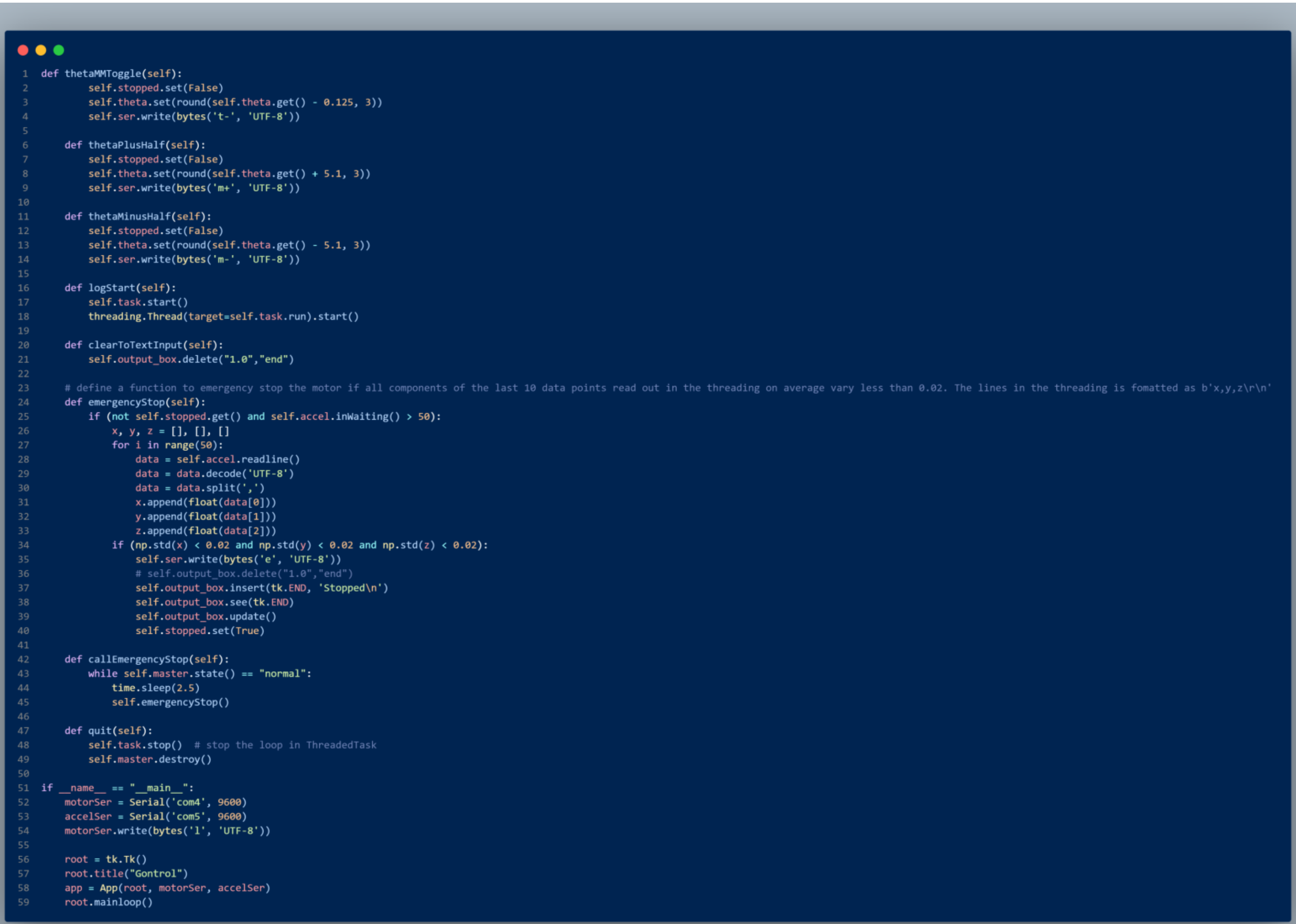


```python
self.logStartButton = tk.Button(
            self.frame,
            text="Start Logging",
            command=self.logStart
        )

        self.logStartButton.place(relx=0.9, rely=0.1, anchor=tk.CENTER)

        self.logStopButton = tk.Button(
            self.frame,
            text="Stop Logging",
            command=self.task.stop
        )

        self.logStopButton.place(relx=0.9, rely=0.2, anchor=tk.CENTER)

        self.logClearButton = tk.Button(
            self.frame,
            text="Clear Log",
            command=self.clearToTextInput
        )

        self.logClearButton.place(relx=0.9, rely=0.3, anchor=tk.CENTER)

        self.buttonls = [self.thetaPPButton, self.thetaMMButton, self.thetaPHButton, self.thetaMHButton, self.manualCheckbox]

        threading.Thread(target=self.callEmergencyStop).start()

    def buttonState(self):
        if (self.stopped.get()):
            for button in self.buttonls:
                button["state"]= "normal"
        else:
            for button in self.buttonls:
                button["state"] = "disabled"

    def updateStopped(self, *args):
        text = "Stopped" if self.stopped.get() else "Running"
        self.stoppedLabel.config(text=text)
        self.buttonState()

    def manualToggle(self):
        if (self.manualVar.get() == 0):
            self.ser.write(bytes('l', 'UTF-8'))
        else:
            self.ser.write(bytes('ul', 'UTF-8'))

    def thetaPPToggle(self):
        self.stopped.set(False)
        self.theta.set(round(self.theta.get() + 0.125, 1))
        self.ser.write(bytes('t+', 'UTF-8'))

```

Python Tkinter GUI Code Cont.

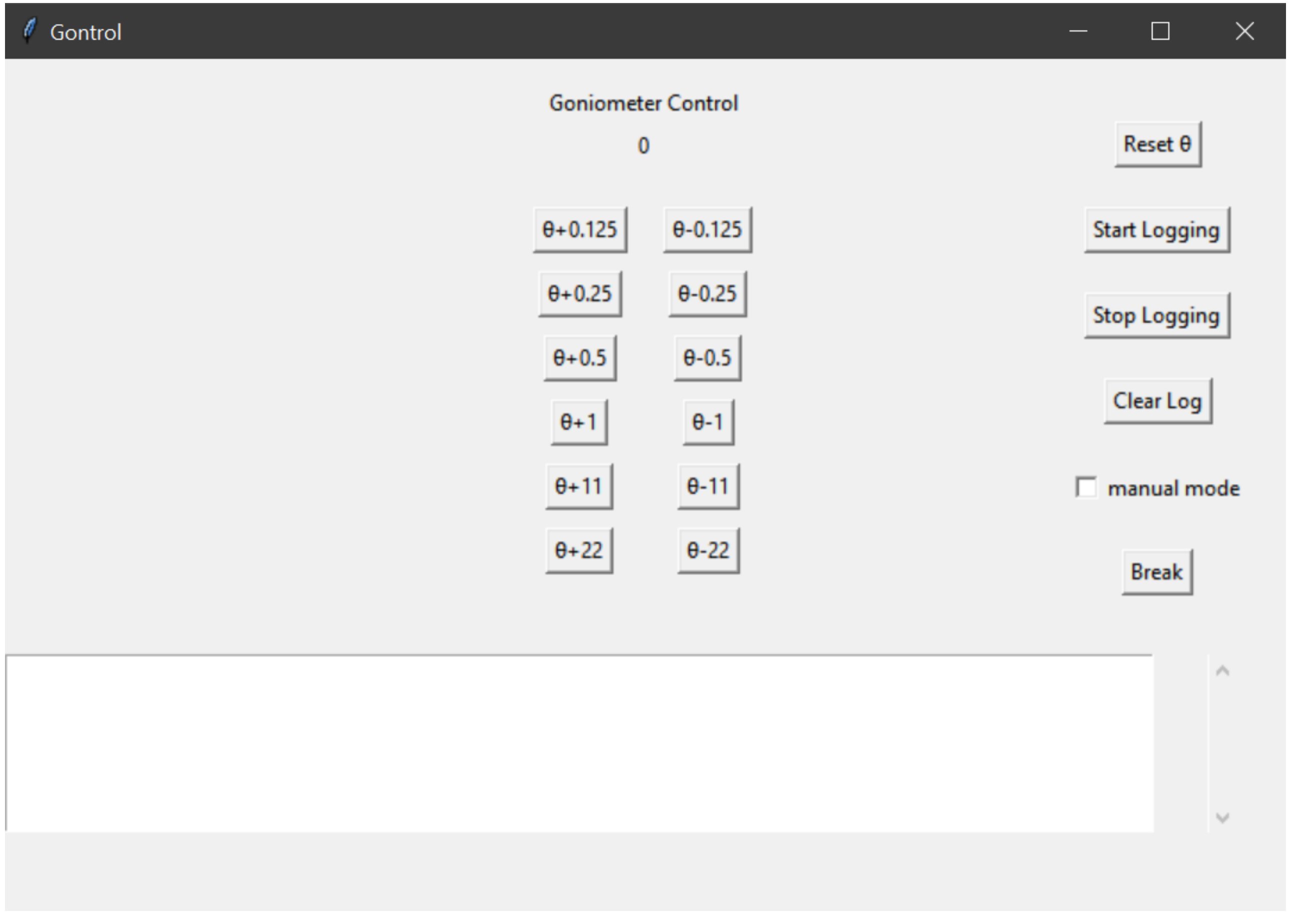


```python
self.logStartButton = tk.Button(
            self.frame,
            text="Start Logging",
            command=self.logStart
        )

        self.logStartButton.place(relx=0.9, rely=0.1, anchor=tk.CENTER)

        self.logStopButton = tk.Button(
            self.frame,
            text="Stop Logging",
            command=self.task.stop
        )

        self.logStopButton.place(relx=0.9, rely=0.2, anchor=tk.CENTER)

        self.logClearButton = tk.Button(
            self.frame,
            text="Clear Log",
            command=self.clearToTextInput
        )

        self.logClearButton.place(relx=0.9, rely=0.3, anchor=tk.CENTER)

        self.buttonls = [self.thetaPPButton, self.thetaMMButton, self.thetaPHButton, self.thetaMHButton, self.manualCheckbox]

        threading.Thread(target=self.callEmergencyStop).start()

    def buttonState(self):
        if (self.stopped.get()):
            for button in self.buttonls:
                button["state"]= "normal"
        else:
            for button in self.buttonls:
                button["state"] = "disabled"

    def updateStopped(self, *args):
        text = "Stopped" if self.stopped.get() else "Running"
        self.stoppedLabel.config(text=text)
        self.buttonState()

    def manualToggle(self):
        if (self.manualVar.get() == 0):
            self.ser.write(bytes('l', 'UTF-8'))
        else:
            self.ser.write(bytes('ul', 'UTF-8'))

    def thetaPPToggle(self):
        self.stopped.set(False)
        self.theta.set(round(self.theta.get() + 0.125, 1))
        self.ser.write(bytes('t+', 'UTF-8'))

```



```python
def thetaMMToggle(self):
        self.stopped.set(False)
        self.theta.set(round(self.theta.get() - 0.125, 3))
        self.ser.write(bytes('t-', 'UTF-8'))

    def thetaPlusHalf(self):
        self.stopped.set(False)
        self.theta.set(round(self.theta.get() + 5.1, 3))
        self.ser.write(bytes('m+', 'UTF-8'))

    def thetaMinusHalf(self):
        self.stopped.set(False)
        self.theta.set(round(self.theta.get() - 5.1, 3))
        self.ser.write(bytes('m-', 'UTF-8'))

    def logStart(self):
        self.task.start()
        threading.Thread(target=self.task.run).start()

    def clearToTextInput(self):
        self.output_box.delete("1.0","end")

    # define a function to emergency stop the motor if all components of the last 10 data points read out in the threading on average vary less than 0.02. The lines in the threading is fomatted as b'x,y,z\r\n'
    def emergencyStop(self):
        if (not self.stopped.get() and self.accel.inWaiting() > 50):
            x, y, z = [], [], []
            for i in range(50):
                data = self.accel.readline()
                data = data.decode('UTF-8')
                data = data.split(',')
                x.append(float(data[0]))
                y.append(float(data[1]))
                z.append(float(data[2]))
            if (np.std(x) < 0.02 and np.std(y) < 0.02 and np.std(z) < 0.02):
                self.ser.write(bytes('e', 'UTF-8'))
                # self.output_box.delete("1.0","end")
                self.output_box.insert(tk.END, 'Stopped\n')
                self.output_box.see(tk.END)
                self.output_box.update()
                self.stopped.set(True)

    def callEmergencyStop(self):
        while self.master.state() == "normal":
            time.sleep(2.5)
            self.emergencyStop()

    def quit(self):
        self.task.stop()  # stop the loop in ThreadedTask
        self.master.destroy()

if __name__ == "__main__":
    motorSer = Serial('com4', 9600)
    accelSer = Serial('com5', 9600)
    motorSer.write(bytes('l', 'UTF-8'))

    root = tk.Tk()
    root.title("Gontrol")
    app = App(root, motorSer, accelSer)
    root.mainloop()
```

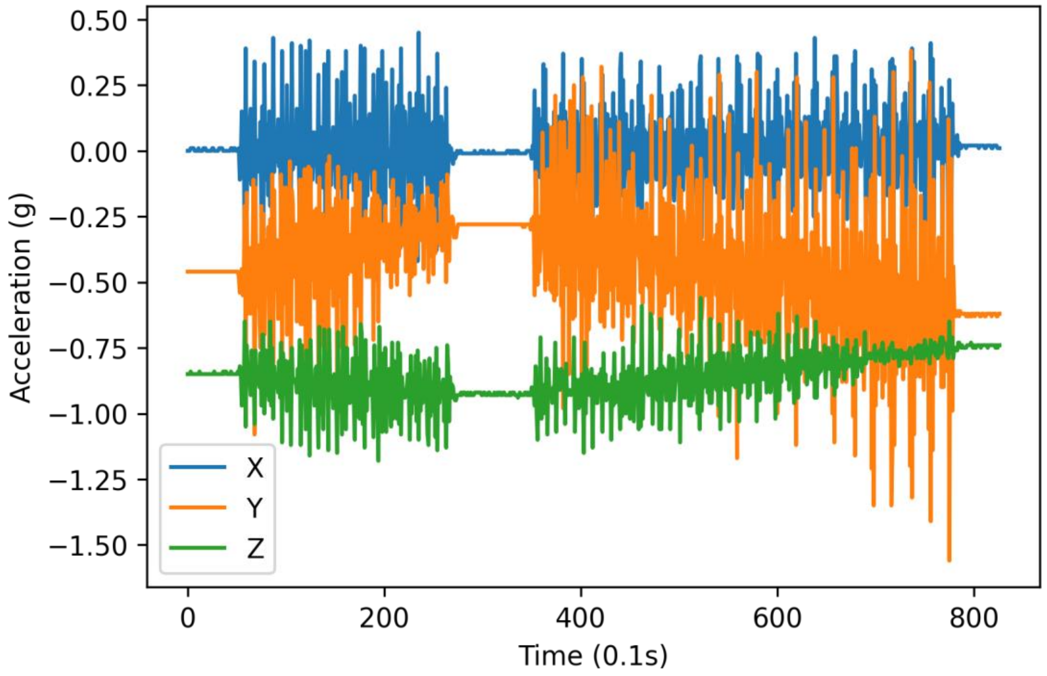
GUI


Gontrol
Goniometer Control
0
θ+0.125
θ-0.125
θ+0.25
θ-0.25
θ+0.5
θ-0.5
θ+1
θ-1
θ+11
θ-11
θ+22
θ-22
Reset θ
Start Logging
Stop Logging
Clear Log
□ manual mode
Break

Works Cited

- Alldatasheet.com. "*HR4988 Datasheet(PDF) - Jiaxing Heroic Technology Co.,Ltd..*" ALLDATASHEET.COM - Electronic Parts Datasheet Search, https://www.alldatasheet.com/datasheet-pdf/pdf/1139938/HEROIC/HR4988.html.
- Childress et al., Physics Today 67(10), 38 (2014); doi: 10.1063/PT.3.2549
- McCauley, Mike. "AccelStepper Library for Arduino." Accelstepper: Accelstepper Library for Arduino, https://www.airspayce.com/mikem/arduino/AccelStepper/index.html.
- "*MMA8452Q 3-Axis, 12-Bit/8-Bit Digital Accelerometer – NXP*". https://www.nxp.com/docs/en/data-sheet/MMA8452Q.pdf.
- Python. "*Tkinter - Python Interface to TCL/TK.*" Python Documentation, Python Software Foundation, https://docs.python.org/3/library/tkinter.html#module-tkinter.

Acceleration Readout



Backup slide to explain the accelerometer configuration process, i.e., how to tell whether the motor is moving from accel. Readings.