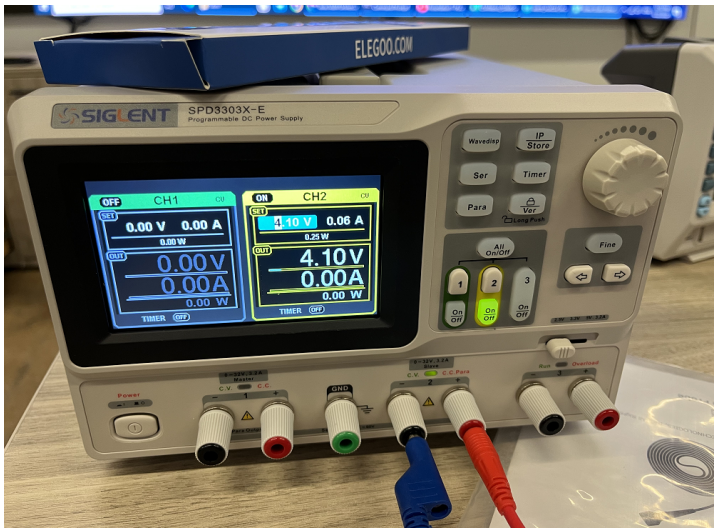
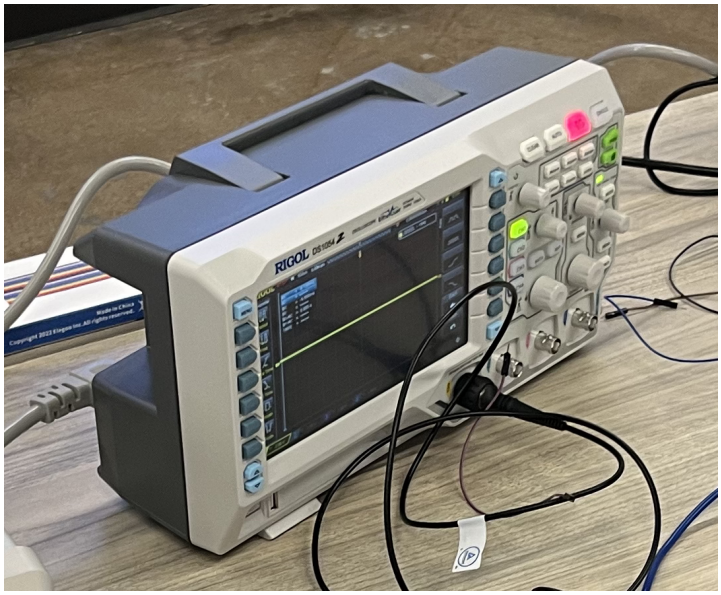
Electro-tactile Interface System Documentation

Apparatus

Programmable DC Power Supply (SIGLENT/ SPD3303X-E)

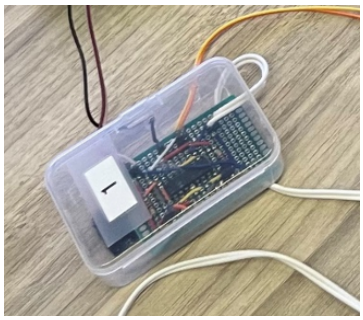


Oscilloscope Calibrator (RIGOL/ Fluke 9500B)



Arduino UNO R3

Stimulator



Justification of the Electro-tactile Feedback System Safety

Please read this before connecting devices.

We believe that our electro-tactile feedback system – which provides sensory augmentation – is a non-significantly risky device because it does not present the potential of serious risk to the health, safety, or welfare of subjects on whom it will be used.

- All of the components in our electro-tactile feedback system will be placed outside subjects, thus eliminating any concerns (e.g., infection, inflammation) presented by implantable devices.
- Our electro-tactile feedback system does not support or sustain human life.
- Its components have not any serious risks to the health, safety, or welfare of subjects.

In below paragraphs, you can find more detailed reasons why our electro-tactile feedback system is a non-significant risky device.

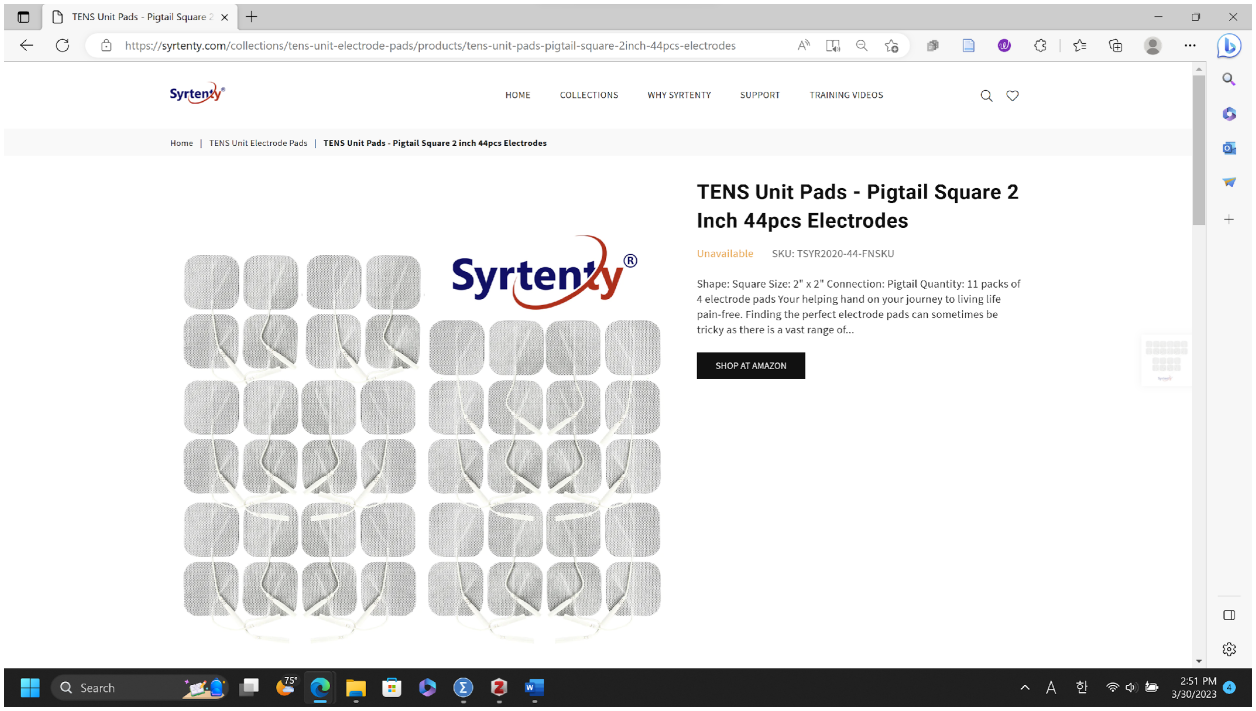
- The bipolar gel electrode of our electro-tactile feedback system is attached on subjects' hand/arm. Thus, it will not restrict any motion or inhibit the ability to walk or stand. And the bipolar gel electrode will not also trigger any skin trouble because its conductive gel pads are obtained from a conductive gel pad of a commercial stimulation pad (model: Syrtenty TENS Unit Pads, web link: https://syrtenty.com/collections/tens-unit-electrode-pads/products/tens-unit-pads-pigtail-square-2inch-44pcs-electrodes) widely used in clinics.

- Our electrical stimulator in our electro-tactile feedback system is isolated from the body of subjects by a thick plastic insulation, and only the gel electrode makes a contact with subjects' skin.

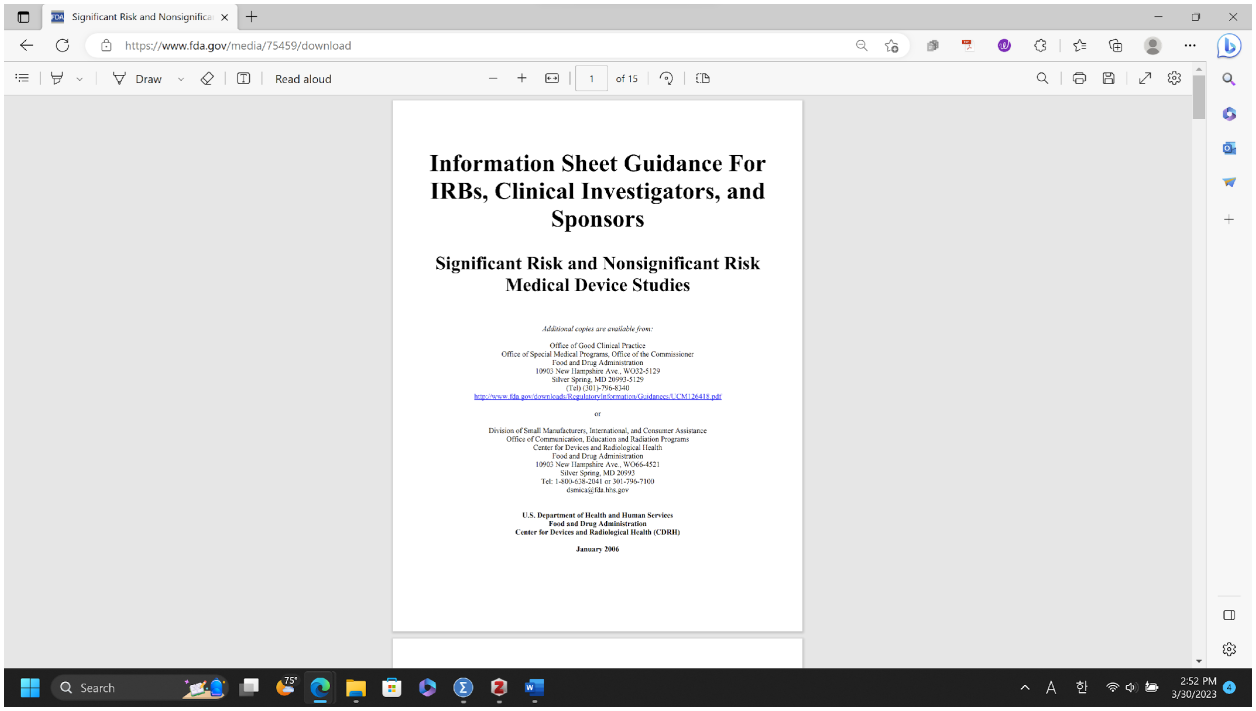
- Our electrical stimulator does not generate any risky pulses because of its limited output current (20 mA). Transcutaneous Electric Nerve Stimulation (TENS) devices, which are usually used in the treatment of pain and are categorized as nonsignificant risk devices (according to the *FDA Information Sheet Guidance for IRBs, Clinical Investigators, and Sponsors: Significant Risk and Nonsignificant Risk Medical Device Studies*, page 9, weblink: https://www.fda.gov/media/75459/download), produce 105 mA pulses (see https://www.electrotherapy.org/tens). However, the current output of our electrical stimulator is less than by using 1 kΩ series resistor (with maximum 20 V applied voltage).

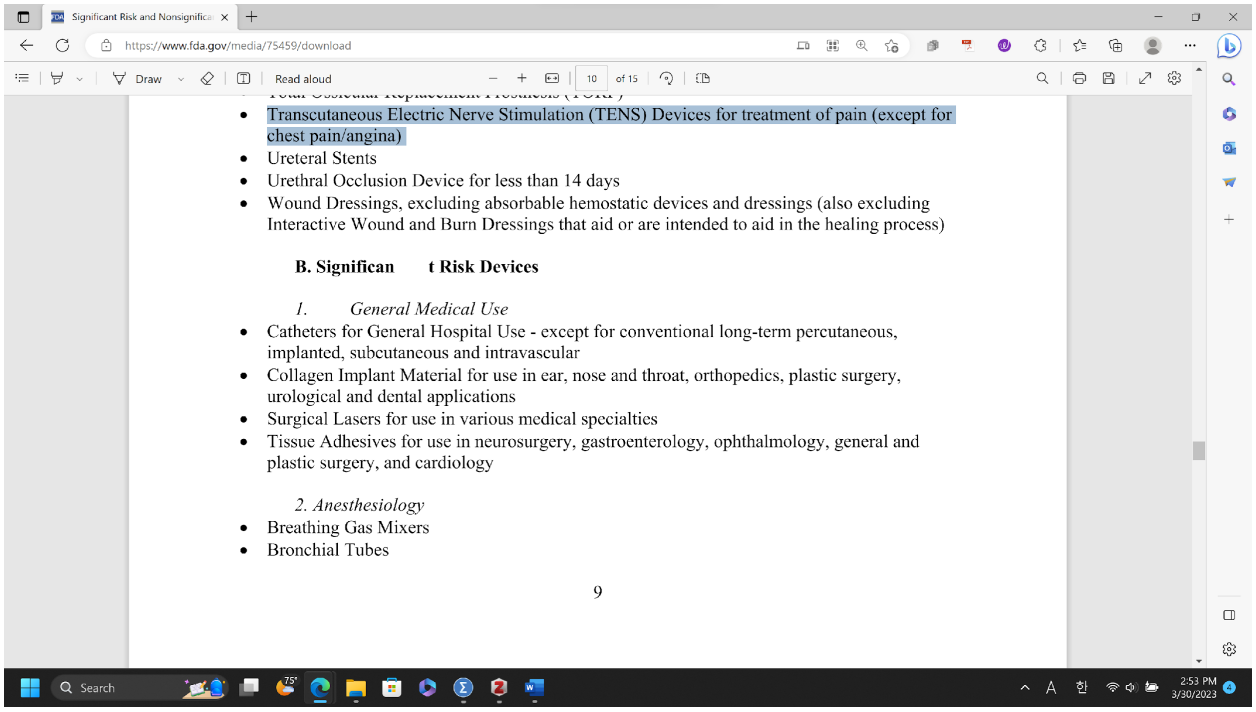
- In addition, our electrical stimulator is protected by a medical-grade surge protector (model: SPS415HGULTRA, web link: https://tripplite.eaton.com/support/sps415hgultra). Therefore, there is no risk of electric short by surge.

Link (03/30/2023):
https://syrtenty.com/collections/tens-unit-electrode-pads/products/tens-unit-pads-pigtail-square-2inch-44pcs-electrodes

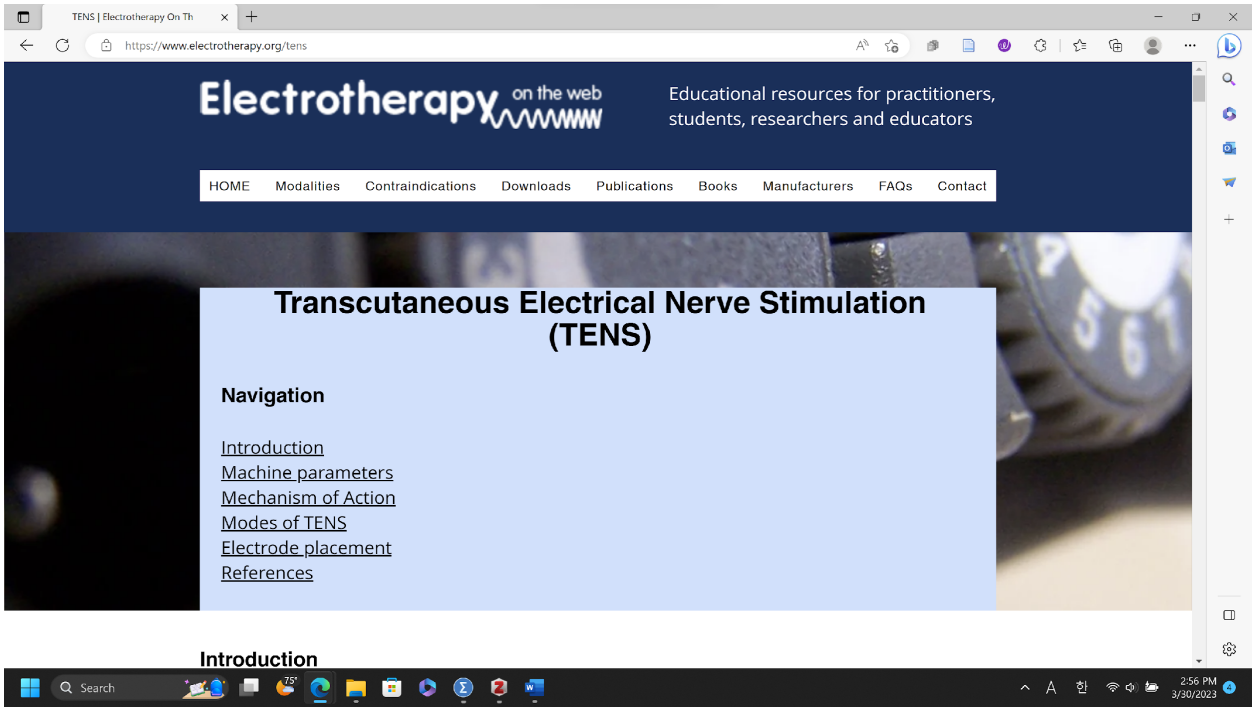


Link (03/30/2023):
https://www.fda.gov/media/75459/download

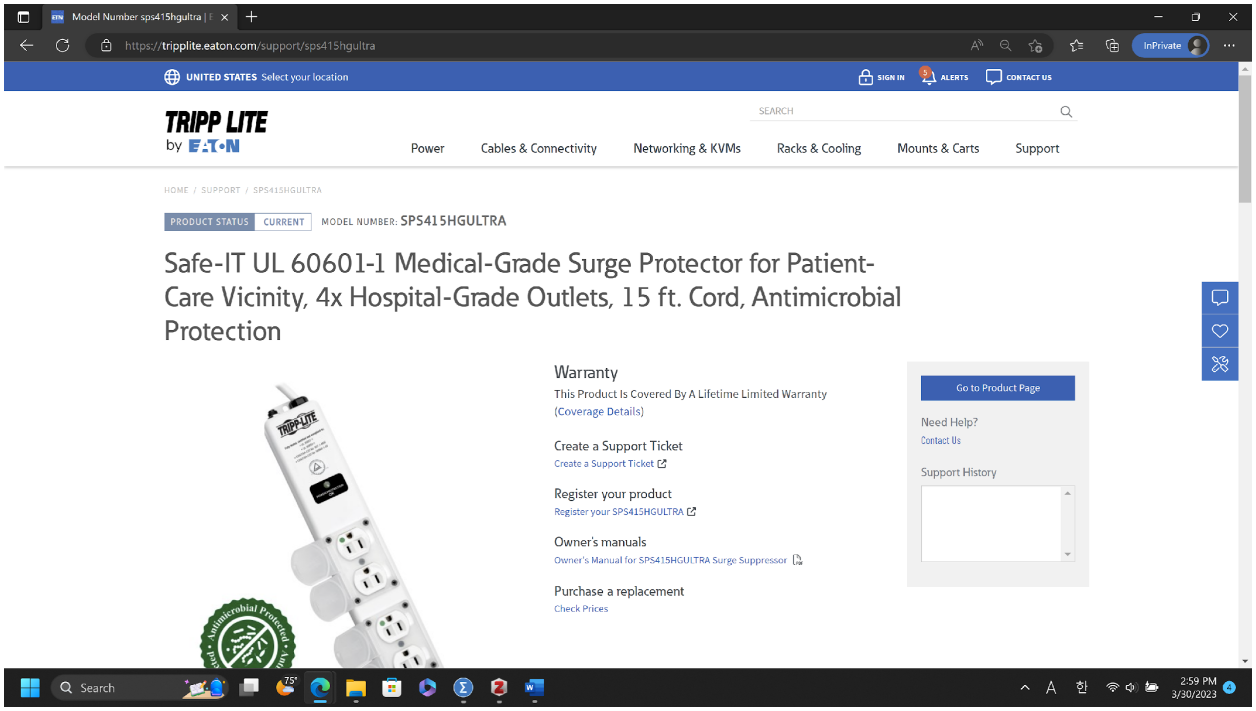




Link (03/30/2023):
https://www.electrotherapy.org/tens



Link (03/30/2023):
https://tripplite.eaton.com/support/sps415hgultra



Electro-tactile feedback System Experiment Preparation

1) Electro-tactile feedback system

The configuration our electro-tactile feedback system. It consists of an electrical stimulator and gel electrode. In the **electrical stimulator**, a self-made printed circuit board (PCB) generates electrical pulses by utilizing pulse width modulated (PWM) signals delivered from a micro controller unit (MCU). The MCU (model: Arduino UNO/Due) is linked to the VR-based construction equipment simulator. In other words, electro-tactile feedback will be provided corresponding to actions in the two platforms. A **custom-made bipolar gel electrode** forwards the electrical pulses to subjects' skin. It has a pair of thin copper pads, with 5 mm spacing, printed on a flexible PCB. The dimension of each pad is 8 mm x 8 mm, and it is connected to the wire by a through-hole soldering. The conductive side of each pad is painted with a silver epoxy (8331D, MG Chemicals, Canada) to make a rough surface, and a conductive gel pad is added to the surface for good attachment to the skin. The conductive gel pad, directly contacted to the skin, is obtained from a conductive gel pad of a commercial stimulation pad (model: Syrtenty TENS Unit Pads, web link: https://syrtenty.com/collections/tens-unit-electrode-pads/products/tens-unit-pads-pigtail-square-2inch-44pcs-electrodes) widely used in clinics.

2) Check list before attaching the electro-tactile feedback system to human

Device checklist

- Please use a medical-graded (having a circuit protector) power strip (very critical).
- Please set voltage of your DC power supply under 19 V to protect the electrical stimulator and your computer.
- Please set current of your DC power supply under 0.06A to protect participants.
- Please check a polarity of your DC power for the electrical stimulator (Orange: +, white: -).
- Please adjust the frequency of the electrical stimulator between 10 Hz and 100 Hz.
- Please always check there is electric shorts between electrical lines and in the electrical stimulator.

Experiment checklist

- Please check exclusion criteria for every participant before every experiment.
  - **Y, N)** Have sensitive electronic implantable medical devices such as a deep brain stimulator or a pacemaker in the upper body
  - **Y, N)** Sensitive or allergic to any kind of adhesive
  - **Y, N)** Have history or overt sign of neurological disease or disorder (e.g., epilepsy)
  - **Y, N)** Have a problem standing or walking
  - **Y, N)** Have a cognitive impairment so that you are unable to follow simple commands
  - **Y, N)** Have a heart disease

Y, N) Unable to understand English

Y, N) Are pregnant (based on a subject's statement)

- Before every experiment, please ensure if the electrical stimulator operates within normal range. (Voltage: 0~19V, current: 0~0.06A) with your oscilloscope.
- Before attaching electrodes to every participant, please ensure if your DC power supply is off.
- After attaching electrodes to every participant, please ensure if the voltage of your DC power supply is under 1 V before activating the channel of your DC power supply, connected to the electrical stimulator.
- Please ensure that the attached electrodes are physically separated and do not contact each other anytime. Also, please ensure that the conductive gel does not make an electrical short between the attached electrodes.
- Before every experiment, please properly adjust the output frequency and output amplitude of the electrical stimulator for every participant.
- Please increase the voltage of your DC power supply by 0.5 V.
- Please always be sure if every participant shows any medical symptoms (e.g., sharp pain on the skin where the electrodes are attached).
- Please turn off your DC power supply before removing electrodes for every participant.

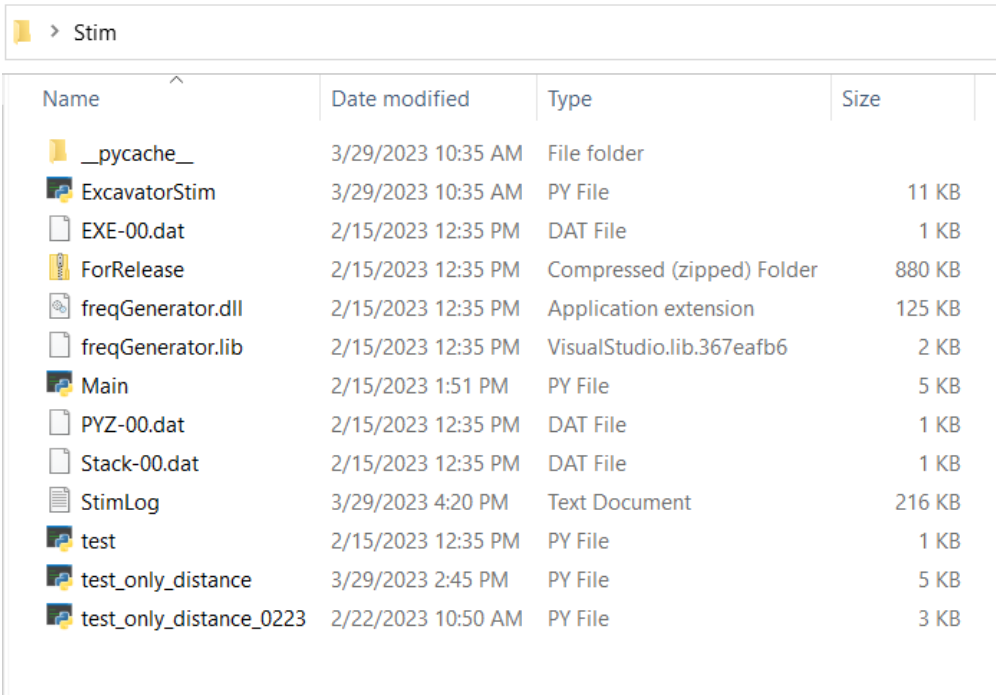
3) Selection of optimal stimulation

To clearly perceive the stimulation in construction equipment VR simulator, a single bipolar gel electrode will be attached to the palmar digital nerve of each subject's hand (Fig. 1) or arm. The distal part of the palmar digital nerve is composed mostly of cutaneous axons and located at the bony area with minimal fats and muscles near to the skin. Therefore, transcutaneous electrical stimulation of the skin over this nerve easily elicits the electro-tactile feedback on the fingertip. Note that the location of the electrode would be changed slightly because subjects' in-born sensitivities on transcutaneous electrical stimulation are different.

4) Understanding of Coding before connecting the apparatus

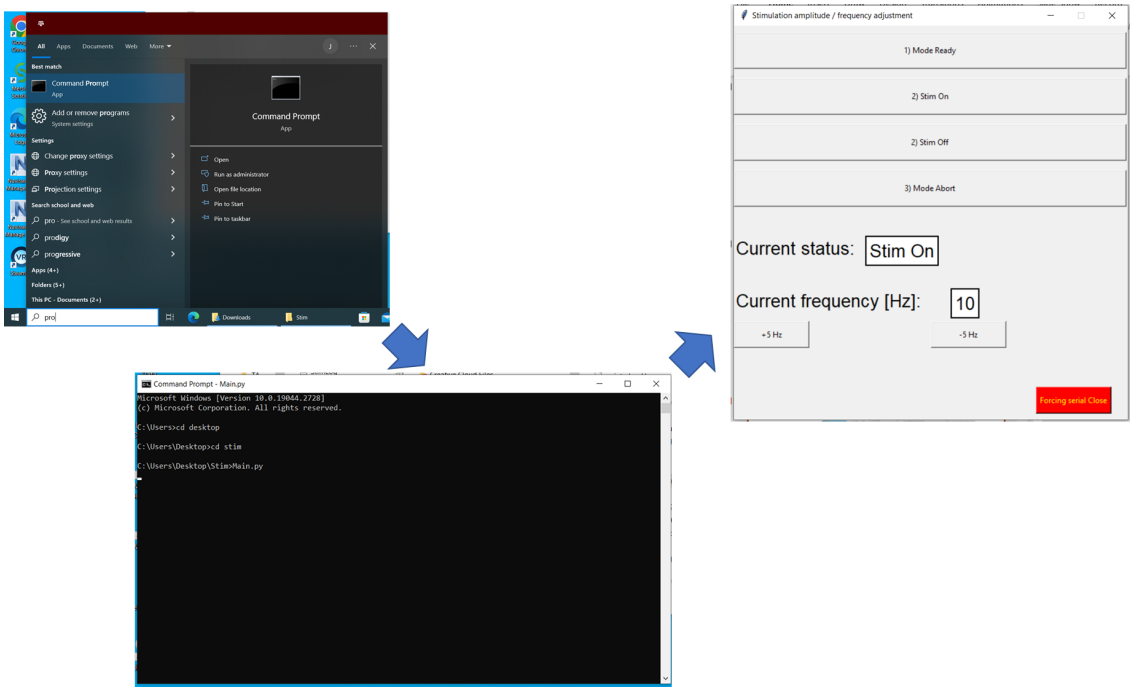
Coding

- All codes are in “Stim” folder (C:\Users\Desktop\Stim)
- In the “Stim” folder there are several python files. Among them below are the important codes for this system.
    - Main.py
    - ExcavatorStim.py
    - test_only_distance.py

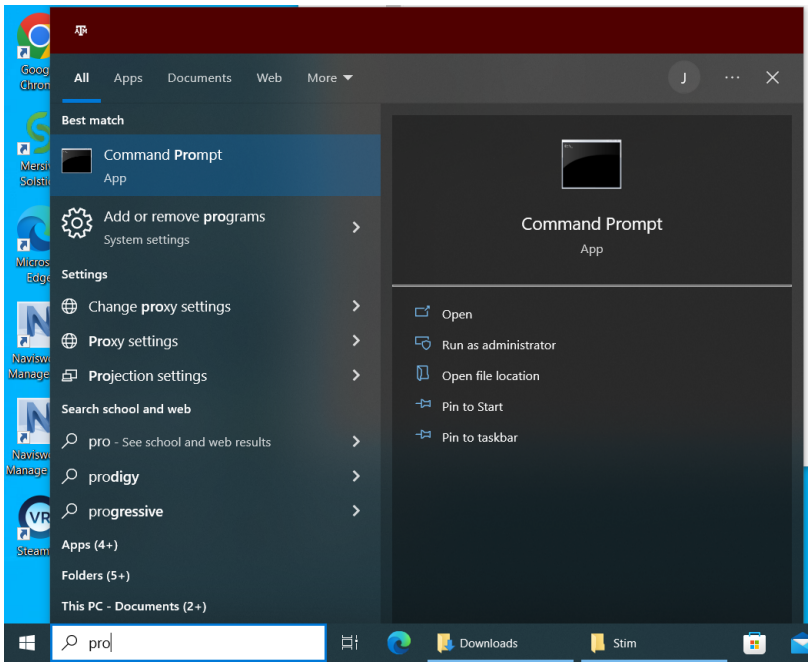


Main.py (C:\Users\Desktop\Stim\Main.py)

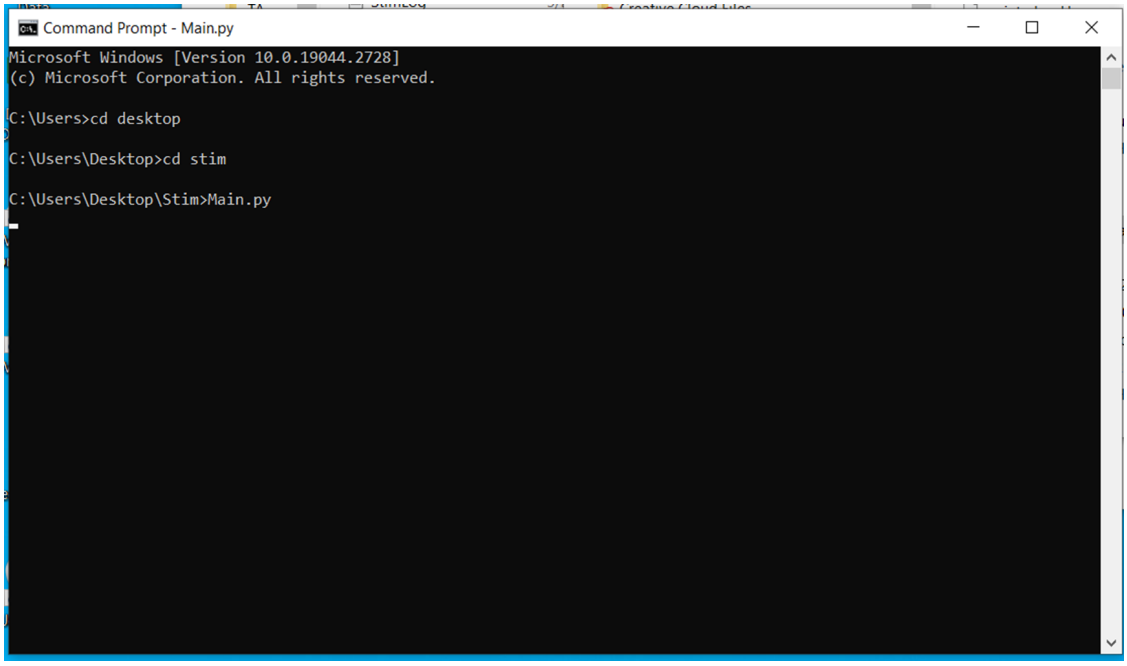
- The purpose of Main.py program is to set up the frequency [Hz] during the experiment.
- How to open the Main.py? (You don't need to change the code. It is a program)



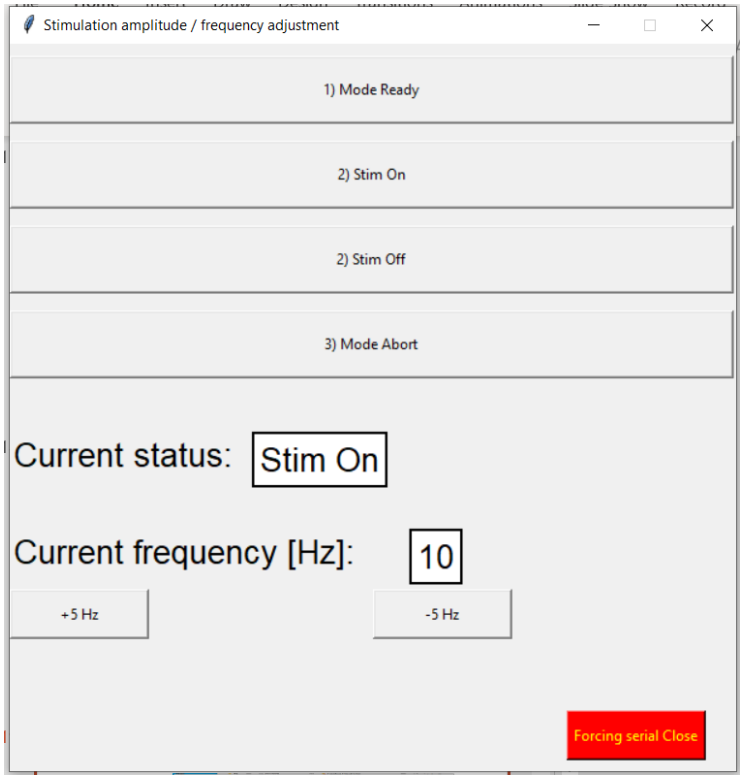
- Open the Command Prompt in Window



- Go to the folder location (C:\Users\Desktop\Stim\Main.py)



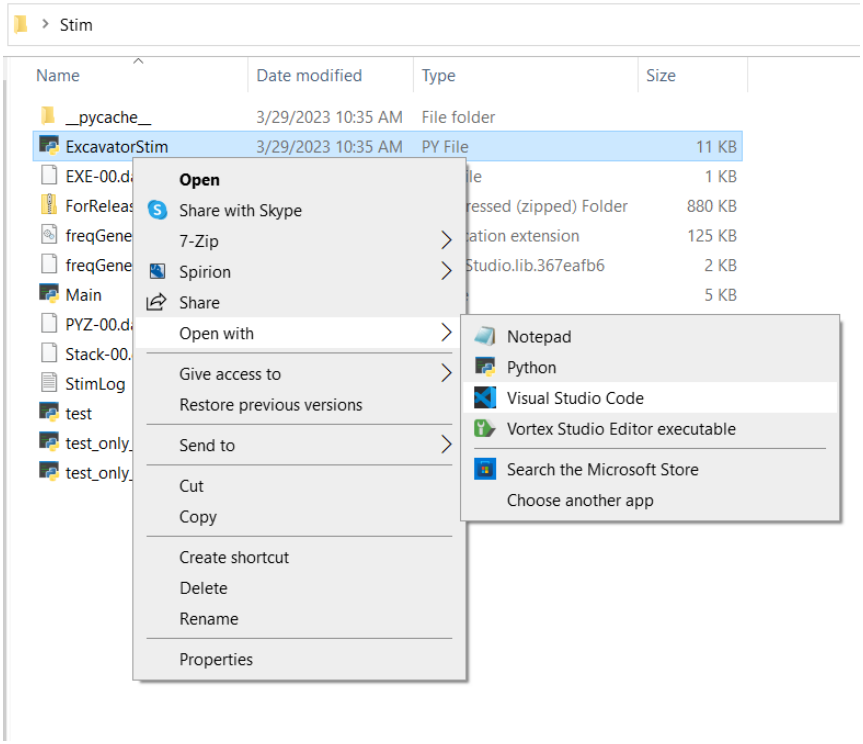
- Open the Main.py software!
- If you open Main.py, it will look like below.
-



-

ExcavatorStim.py (C:\Users\Desktop\Stim\ExcavatorStim.py)

- The purpose of ExcavatorStim.py is to put the appropriate port value (ex. ‘COM8’) which should be the same value as Arduino.
- How to open ExcavatorStim.py?
- Open with Visual Studio Code



- If you open ExcavatorStim.py, it will look like below.

```
# Date: 2022/10/28

# Modules and Parameters ------------------
# Do not edit this part.
import serial
#import ctypes
from time import sleep
from ctypes import *
from datetime import datetime
import time
eventMessage = ""
#-----------------------------------------

# Parameters for operator ----------------------------------------------------------------------------------
port = 'COM8'# Serial communication port number
baudRate = 115200 # Serial communication baudrate
minFreq = 10.0 # For setting minimum stimulation , Unit: Hz, This valus should be over 10.0 Hz
maxFreq = 100.0 # For setting maximum stimulation , Unit: Hz, This valus should be under 100 Hz.
minDis = 10# For setting minimum distance between a bucket and burried pipe. This valus should be positive.
           # But, I recommend you not to change minDis. (10.0 <- optimal value)
maxDis = 100.0 # For setting maximum distance between a bucket and burried pipe. This value should be positive.
#----------------------------------------------------------------------------------------------------------
```

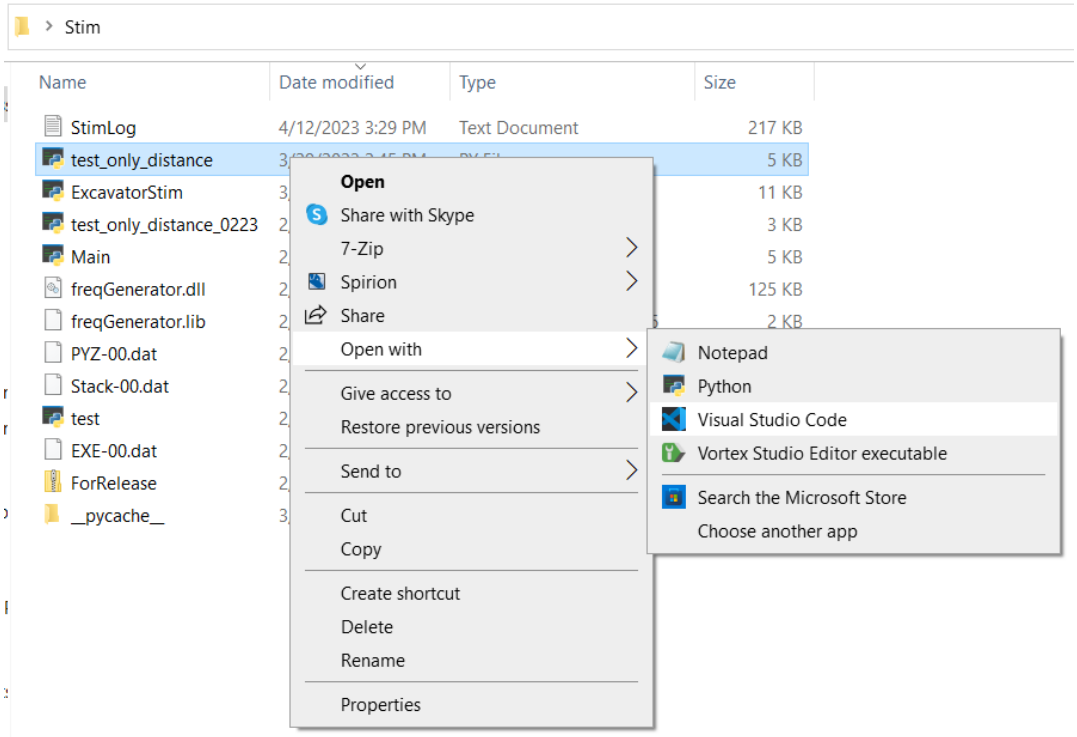
- Find “#Parameters for operator” and “port”

```
# Parameters for operator-----------------------------------------------------
port = 'COM8'# Serial communication port number
```

- The value of the “port” is “COM8”, this should be the same as the Arduino port

test_only_distance.py (C:\Users\Desktop\Stim\test_only_distance.py)

- The purpose of test_only_distance.py is
  1) connecting distance value from VR Platform (ex. Unity, Unreal) to electro-tactile stimulator
  2) to set up the maximum frequency [Hz] during the experiment.
- How to open test_only_distance.py?
- Open with Visual Studio Code



- If you open test_only_distance.py, it will look like below.
-

```python
# !pip install pyserial
# !pip install serial

import Vortex
import time
from datetime import datetime
# import serial
# import pyserial
# import ctypes
from ExcavatorStim import StimGenerator, parameterSetting
# mode = 10 (for API mode), mode = 20 (for frequency and amplitude adjustment mode)
mode = 10
serialConnectionTrigger = 0 # To control serial connection -> 0: Before connected, 1: After connected
pastTime = 0.0 # For time gap
currentTime = 0.0
```

-

- For the 1st purpose of test_only_distance.py is
  1) connecting distance value from VR Platform (ex. Unity, Unreal).
- By using Ctrl+F, search for “global disval”. You can find the result below.
-

```
if (disval<=0.3):
    parameterSetting(minFreqinLink, maxFreqinLink, minDisinLink, maxDisinLink, 1)
else:
    parameterSetting(minFreqinLink, maxFreqinLink, minDisinLink, maxDisinLink, 0)

data_count = 0

def pre_step(extension):
    global disval, data_count
    if disval is None or data_count == 6:
        disvalL = extension.inputs.DistanceL.value
        disvalR = extension.inputs.DistanceR.value
        disval = min(disvalL, disvalR)

        data_count = 1
    else:
        data_count += 1
```

-
- `disvalL` is the linked real-time distance value between the excavator bucket and the **L**eft obstacle in the VR Platform (ex. Unity, Unreal).
- `disvalR` is the linked real-time distance value between the excavator bucket and the **R**ight obstacle in the VR Platform (ex. Unity, Unreal).
- `disval` is the minimum value between `disvalL` and `disvalR`. In this system, `disval` is criteria for giving electro-tactile feedback.
- You can link your VR Platform’s distance value in `disval`. And electro-tactile feedback will be given based on your VR Platform’s distance value.

- How to set the criteria?
- In this coding, if `disval` is less than or equal to 0.3, there will be electro-tactile feedback. Electro-tactile feedback is inversely mapped. This means when `disval` becomes 0, the participant will feel the maximum frequency (ex. 45Hz). When `disval` becomes the criteria distance (in this case 0.3), the participant will feel the minimum frequency (ex. 10Hz). The participant will feel more/less frequency when `disval` is near/far to 0.
- Setting the distance criteria is important.
- By using Ctrl+F, search for “0.3”. You can find the 2 results below. By changing “0.3”, you can change the distance criteria.

```
if (disval<=0.3):
    parameterSetting(minFreqinLink, maxFreqinLink, minDisinLink, maxDisinLink, 1)
else:
    parameterSetting(minFreqinLink, maxFreqinLink, minDisinLink, maxDisinLink, 0)
```

-

```
global serialConnectionTrigger
    global pastTime
    if abs(disval)<=0.3 and serialConnectionTrigger == 0: # Connect serial connection when dis>10m
        StimGenerator(0, mode, 1, float(maxDisinLink+1))
        for _ in range(5):
            StimGenerator(1, mode, 2)
```

```
            #print("Activated Standy")
        serialConnectionTrigger = 1
```

- For the 2nd purpose of test_only_distance.py is
  2) to set up the maximum frequency [Hz] during the experiment.
-

```
# !pip install pyserial
# !pip install serial

import Vortex
import time
from datetime import datetime
# import serial
# import pyserial
# import ctypes
from ExcavatorStim import StimGenerator, parameterSetting
# mode = 10 (for API mode), mode = 20 (for frequency and amplitude adjustment mode)
mode = 10
serialConnectionTrigger = 0 # To control serial connection -> 0: Before connected, 1: After connected
pastTime = 0.0 # For time gap
currentTime = 0.0

minFreqinLink = 10.0 # For setting minimum stimulation , Unit: Hz, This valus should be over 10.0 Hz
maxFreqinLink = 45.0 # For setting maximum stimulation , Unit: Hz, This valus should be under 100 Hz.
minDisinLink = 0.0 # For setting minimum distance between a bucket and burried pipe. This valus should be positive.
                   # But, I recommend you not to change minDis. (10.0 <- optimal value)
maxDisinLink = 0.22 # For setting maximum distance between a bucket and burried pipe. This value should be positiv
#minDisinLink = (minDisinLink/0.1)*10
#maxDisinLink = (maxDisinLink/0.1)*10
```

-
- By using Ctrl+F, search for “maxFreqinLink”. You can find the result below.

```
maxFreqinLink = 45.0 # For setting maximum stimulation , Unit: Hz, This
valus should be under 100 Hz.
```

- In this case the value of “maxFreqinLink” is 45. The unit is Hz. So the maximum frequency is set as 45Hz, now.
- This value (ex. 45) should be the same as the maximum frequency measured with Main.py.
- The process of determining the maximum frequency for each participant will be discussed later.

5) How to connect the apparatus

Arduino

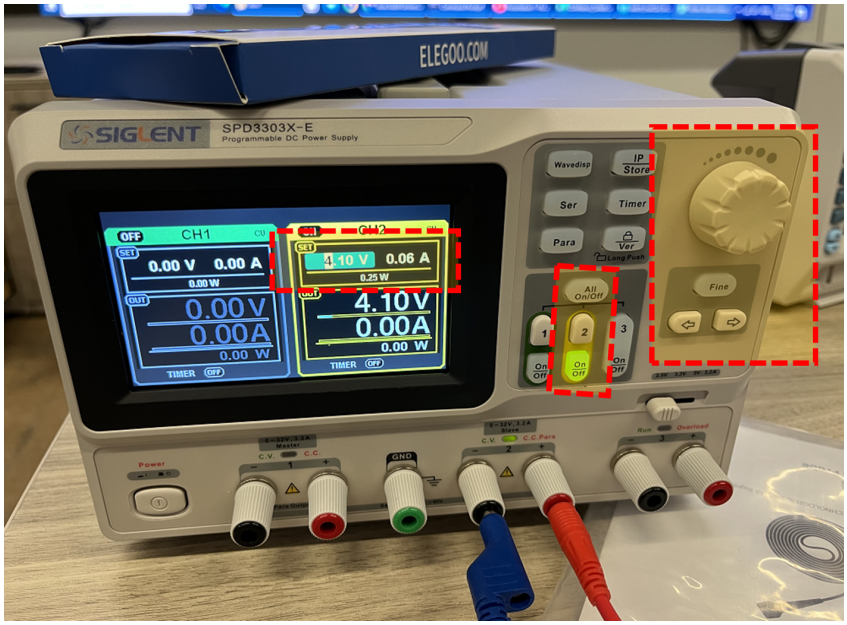
- Connect to Desktop/Laptop with USB cable
- Connect to Stimulator like figure below
- Check the Arduino port

- Open ExcavatorStim.py with Visual Studio Code
- Find “#Parameters for operator” and “port”

```
# Parameters for operator-----------------------------------------------------
port = 'COM8'# Serial communication port number
```

- The value of the “port” is “COM8”, this should be the same as the Arduino port

DC Power Supply

- Don’t connect with any device
- Turn on the DC Power Supply
- Please take a look at
- Make sure and set Voltage ~5V (below 5V), current: ~0.06A (below 0.06A) like below figure
  (in this case CH2 is used)
- 

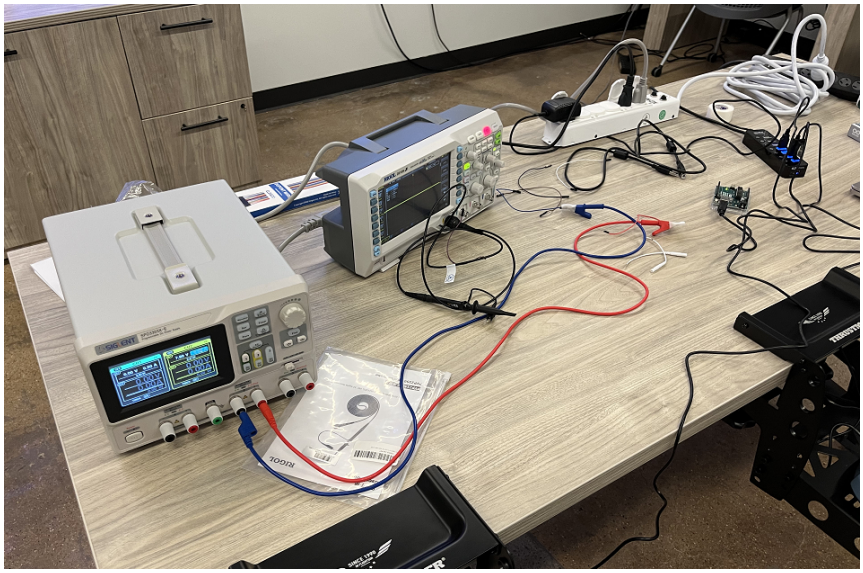
- Connect one side of Red wire (+) and Blue wire (-) to Power Supply like the figure above (in this case CH2 is used)
- Turn off the DC Power Supply

Stimulator

- Should be connect to Arduino already
- Connect the Red wire (+) and Blue wire (-) which are attached to DC Power Supply to Stimulator. (+) (-) of Stimulator is written with tags.
- Turn on the DC Power Supply
- Make sure when turning on the DC Supply, set Voltage ~5V (below 5V), current: ~0.06A (below 0.06A)

Oscilloscope Calibrator (RIGOL/ Fluke 9500B)

- Before attaching electrode directly to a person, it should be tested with an oscilloscope
- If you attach it directly to a person, he/she can accidentally get an electric shock
- Connect Passive Probes
- Make sure there is enough distance between Passive Probes
- Connect Oscilloscope Passive Probes and Stimulator
- Figure

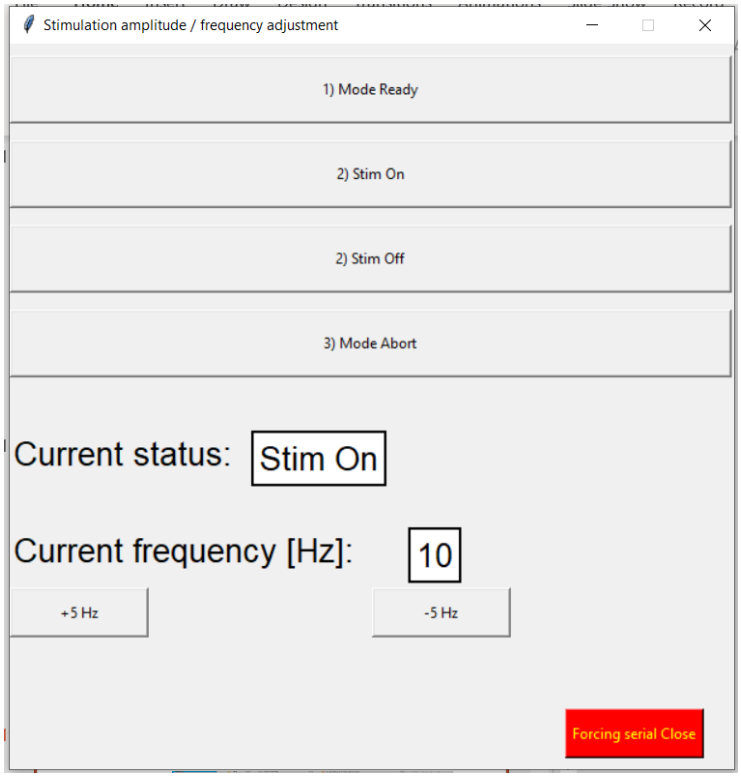


6) Check the electro-tactile system with Oscilloscope and Main.py

- Before attaching electrode directly to a person, it should be tested with an oscilloscope
- If you attach it directly to a person, he/she can accidentally get an electric shock
- Check every apparatus is connected correctly
- Run Main.py

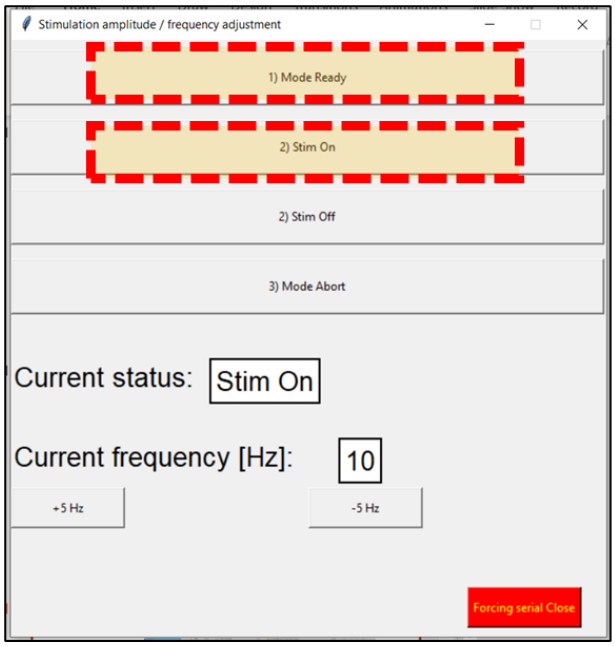
Main.py

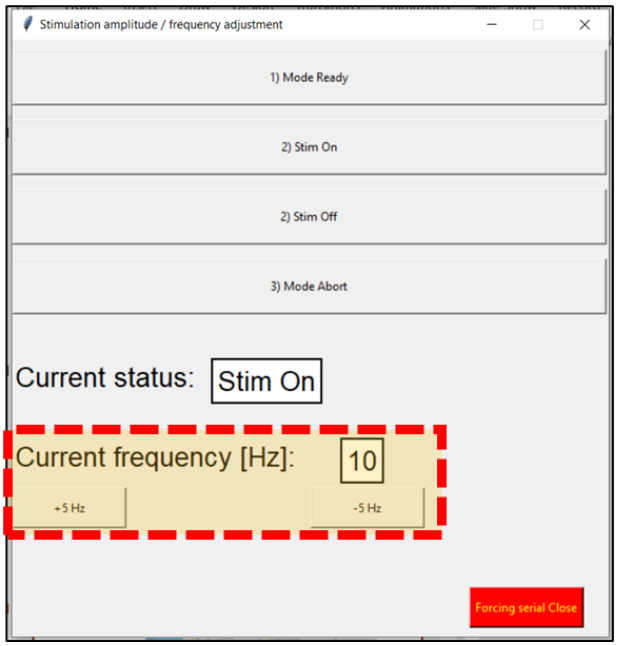
- Open the Main.py software!
- If you open Main.py, it will look like below.
-



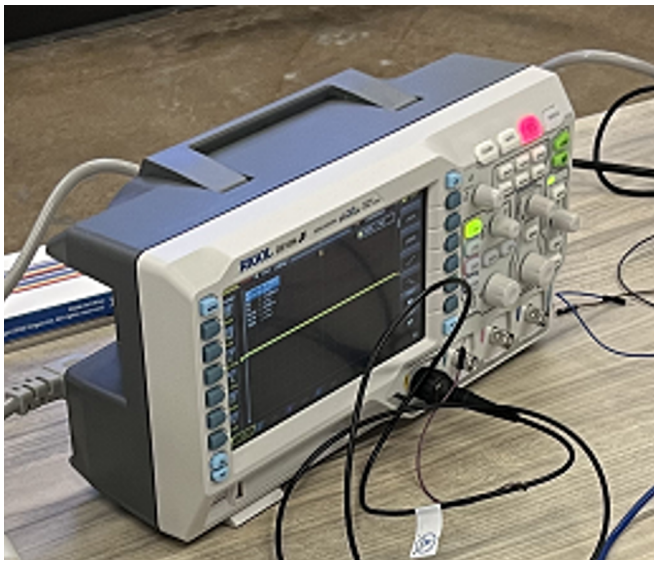
-

- Click “1) Mode Ready”

- 



- Click “2) Stim On”
- Change frequency with “+5Hz” button or “-5Hz” button.
- Check whether the period of the wave changes through the oscilloscope.
- The higher the Hz, the shorter the period will be shown.



- If it works well, it means the Stimulator is working.
- Disconnect Oscilloscope and Stimulator like figure below.
-

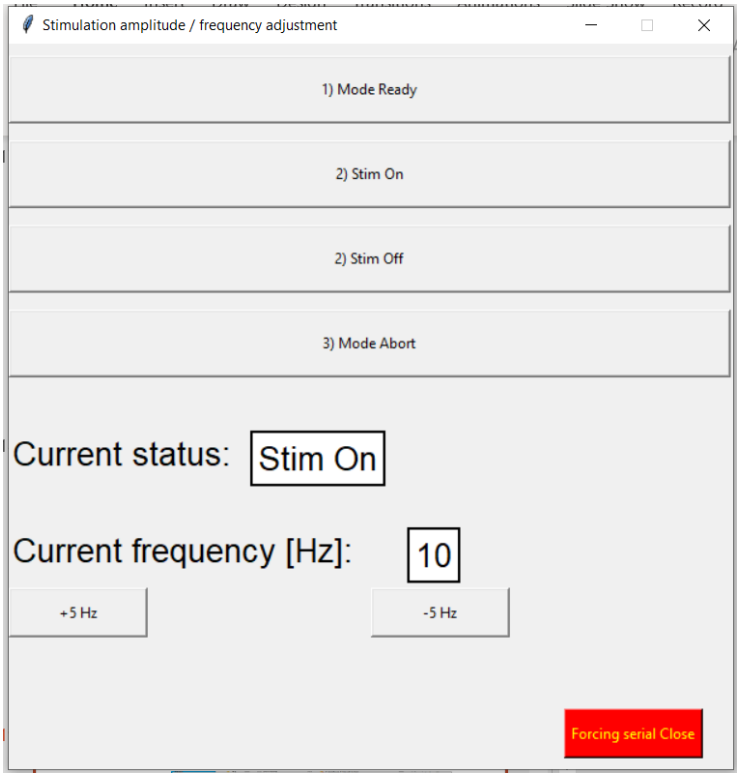
7) Set up the amplitude and frequency for each participant

- Before attaching the electrode directly to a person, it should be tested with an oscilloscope
- After checking with the oscilloscope, disconnect Oscilloscope and Stimulator.

- Connect the electrode and the Stimulator

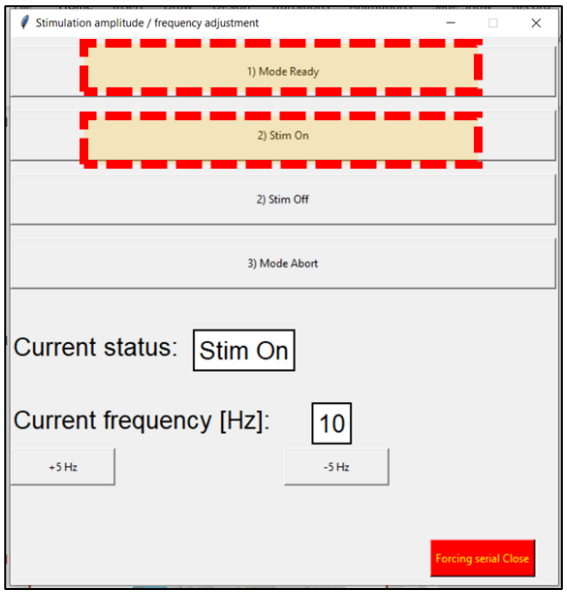
- Attach the electrode to the participant's skin

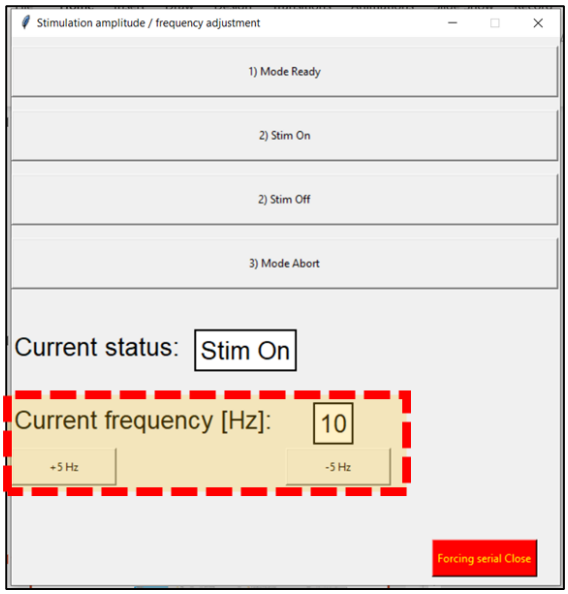
- Open the Main.py software!
- If you open Main.py, it will look like below.
-



-

- Click "1) Mode Ready"

- 

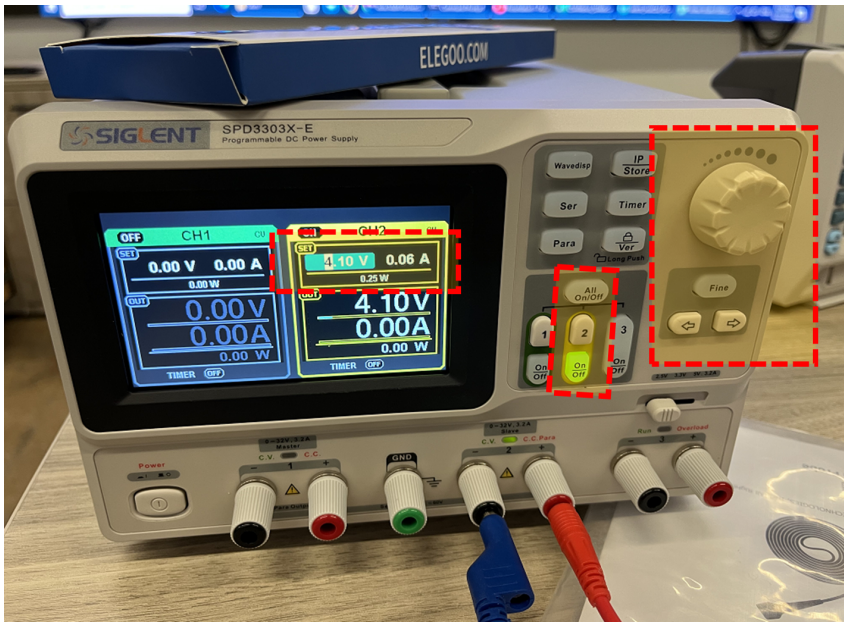


- Click “2) Stim On”
- Make Current frequency [Hz] = 10 Hz

- ***Set up amplitude***

- The stimulation ***amplitude*** will be set as minimum (1 V) first and will be continuously increased with the step of 0.5V until subjects report any sensation. The threshold for perception (T_p) will be decided as the stimulation amplitude when subjects start to feel some sensation.

- Increase Voltage by 0.5V. with DC Power Supply



-

- After establishing T_p, the stimulation amplitude will be increased with the step of 0.5V again until subjects report uncomfortable sensation such as an ache, extreme hot/cold sensation, or sharp/dull pain. The threshold for uncomfortable sensation (T_u) is decided as the stimulation amplitude when subjects start to feel any uncomfortable sensation.

- Mean value (T_m) between T_p and T_u will be used as the default stimulation amplitude during the whole experiment because this value can elicit perceptive electro-tactile sensation without any uncomfortable or unbearable sensation.

- ***Setup frequency***

- First, the lowest frequency (f_o) of stimulation will be set as 10 Hz when its amplitude is fixed to the value identified in the above step.

- Open the Main.py software!
- Change frequency with “+5Hz” button or “-5Hz” button.

- Then, the stimulation frequency will be increased with the step of 5 Hz until when subjects feel any buzzing sensation. The maximum frequency (f_m) will be decided as the frequency when subjects start to feel much more buzzing sensation than pulsing sensation.

- Subjects will be provided stimulation ranging from f_o and f_m.

Appendix

ExcavatorStim.py (C:\Users\Desktop\Stim\ExcavatorStim.py)

```
# Date: 2023/03/28

# Modules and Parameters -----------------
# Do not edit this part.
import serial
#import ctypes
from time import sleep
from ctypes import *
from datetime import datetime
import time
eventMessage = ""
#----------------------------------------

# Parameters for
operator-------------------------------------------------------------------------------------
port = 'COM8'# Serial communication port number
baudRate = 115200 # Serial communication baudrate
minFreq = 10.0 # For setting minimum stimulation , Unit: Hz, This valus should be over 10.0 Hz
maxFreq = 100.0 # For setting maximum stimulation , Unit: Hz, This valus should be under 100 Hz.
minDis = 10# For setting minimum distance between a bucket and burried pipe. This valus should be
positive.
            # But, I recommend you not to change minDis. (10.0 <- optimal value)
maxDis = 100.0 # For setting maximum distance between a bucket and burried pipe. This value should be
positive.
#-------------------------------------------------------------------------------------------------
--------

# Parameters for
delay----------------------------------------------------------------------------------
global delayFlag
delayFlag = 0

global timeStart
timeStart = 0.0

global timeCurrent
timeCurrent = 0.0

global delayDesired
delayDesired = 0.05
#-------------------------------------------------------------------------------------------------
--------

def parameterSetting(minFrequency, maxFrequency,minDistance, maxDistance, limitValue):
    global minFreq
    if limitValue==1:
        minFreq = minFrequency
    elif limitValue ==0:
        minFreq =0

    global maxFreq
    maxFreq = maxFrequency
    global minDis
    minDis = minDistance
    global maxDis
    maxDis = maxDistance

"""
def delayfunc(delayTime):
    start = time.time()
    while ((time.time()-start)<=delayTime):
        continue

def FreqGenerator(mode, dis):
    stimFreqVal = 0.0
```

```
    if mode == 10:
        slopeA = (maxFreq-minFreq)/(minDis-maxDis)
        constantB = 0.5 * ((maxFreq+minFreq)-slopeA*(minDis+maxDis))
        if (dis>=minDis and dis<=maxDis):
            #stimFreqVal = slopeA*dis+constantB
            stimFreqVal  = slopeA*dis+constantB
        elif (dis>maxDis):
            stimFreqVal = 0.0
        elif (dis<minDis):
            stimFreqVal = 0.0

    elif mode == 20:
        if (dis>100):
            stimFreqVal = 100
        elif (dis <0):
            stimFreqVal = 0
        else:
            stimFreqVal = dis
    print(stimFreqVal)
    return int(stimFreqVal)

def StimGenerator(comStatus, mode, sync, dis = minDis-1, tempInput = 0):
    #StimGenerator(comStatus, mode, sync, dis = minDis-1, tempInput = 0):
    global delayFlag
    global timeStart
    global timeCurrent
    global delayDesired
    #print("minFreq: ",type(minFreq))
    #print("maxFreq: ",type(maxFreq))
    #print("minDis: ", type(minDis))
    #print("maxDis: ", type(maxDis))
    #Caution****-----------------------------------------------------------------------------------------
    # Please do not edit scripts below. If you revise any codes below,
    #it would lead to electrical shock to subjects, and its your responsibility.
    # ----------------------------------------------------------------------------------------------

    # Function Parameters
-------------------------------------------------------------------------------------------
    # comStatus -> 0: SerialCom open, 1: SerialCom ready, 2: SerialCom active, 3: SerialCom close
    # mode -> To switch between frequency-modulated electro-tactile feedback mode (mode = 10)
    # and amplitude/frequency adjustmnet mode (mode = 20)
    # Sync -> time synchronization --> 0: SerialCom open, 1: SerialCom ready, 2: SerialCom active, 3:
SerialCom close

#-------------------------------------------------------------------------------------------------------
---------

    #Variables of Developer--------------------------------
    timeOut = 0
    returnFreq = 0
    now = datetime.now() # Time Information
    #-----------------------------------------------------

    if comStatus == 0:
        with open("StimLog.txt", "a") as f:
            output = "\n [Log: " + str(now) +"] \n"
            f.write(output)
        try:
            global serialComm
            serialComm = serial.Serial(port, baudRate)
            if serialComm:
                print("Serial comm successfully open")
                with open("StimLog.txt", "a") as f:
                    eventMessage = "Serial comm open mode successfully activated. \n"
                    output = str(now) + " -> " + eventMessage
                    f.write(output)
        except:
```

```
            print("Check your port. Default port is COM5.")
            with open("StimLog.txt", "a") as f:
                eventMessage = "Serial comm open failed. \n"
                output = str(now) + " -> " + eventMessage
                f.write(output)
            return
        serialComm.timeout = timeOut
        delayfunc(0.05)
        ##sleep(0.5) # Wait for completing serial open
        try:
            #libc = ctypes.CDLL('./freqGenerator.dll') # It is to load DLL file.
            with open("StimLog.txt", "a") as f:
                eventMessage = "Serial Comm open mode. Freq calculator successfully loaded. \n"
                output = str(now) + " -> " + eventMessage
                f.write(output)
        except:
            print("Your ""freqGenerator"" has problem. Please check it.")
            with open("StimLog.txt", "a") as f:
                eventMessage = "Serial Comm open mode. Freq calculator not successfully loaded. \n"
                output = str(now) + " -> " + eventMessage
                f.write(output)
            serialComm.close()
            return

        freq = FreqGenerator(mode, dis)
        #print(freq)
        #freq = FreqGenerator(dis, libc, mode, sync)

    elif comStatus == 1:
        freq = str(0)
        serialComm.write(str(int(freq)).encode('utf-8'))
        ##sleep(0.05)
        delayfunc(0.05)
        with open("StimLog.txt", "a") as f:
            eventMessage = "Standby mode activated. "
            output = str(now) + " -> " + eventMessage + "Freq: " + freq + "[Hz]" + "\n"
            f.write(output)
        return

    elif comStatus == 2:
        try:
            #libc = ctypes.CDLL('./freqGenerator.dll') # It is to load DLL file.
            with open("StimLog.txt", "a") as f:
                eventMessage = "Stimulation mode. Freq calculator successfully loaded. \n"
                output = str(now) + " -> " + eventMessage
                f.write(output)
        except:
            print("Your ""freqGenerator"" has problem. Please check it.")
            with open("StimLog.txt", "a") as f:
                eventMessage = "Stimulation mode. Freq calculator not successfully loaded. \n"
                output = str(now) + " -> " + eventMessage
                f.write(output)
            serialComm.close()
            return
        if mode == 10:
            if (type(dis) != float) or (type(minDis) != float) or (type(maxDis) != float) or
(type(minFreq) != float) or (type(maxFreq) != float):
                print("Please check data types of your inputs. serial Comm is closed.")
                serialComm.close()
                return
            if dis<0 or minDis<0 or maxDis<=0 or minFreq<0 or maxFreq<=0 or minDis>=maxDis or
minFreq>=maxFreq:
                print("Please check one of the input values are below zero.Serial comm is closed.")
                serialComm.close()
                return
            #freq = FreqGenerator(dis, libc, mode, sync)
            freq = FreqGenerator(mode, dis)
```

```
            #print(freq)
            with open("StimLog.txt", "a") as f:
                eventMessage = "Stimulation mode. "
                output = str(now) + " -> " + eventMessage+"Freq: " + str(int(freq)) + "[Hz]" + "\n"
                f.write(output)
            #print("test_freq: "+str(freq))
        elif mode == 20:
            freq = FreqGenerator(mode, dis)
            #print(freq)
            #freq = FreqGenerator(dis, libc, mode, sync)
            with open("StimLog.txt", "a") as f:
                eventMessage = "Stimulation mode. "
                output = str(now) + " -> " + eventMessage+"Freq: " + str(int(freq))+ "[Hz]"  + "\n"
                f.write(output)

        # For Arduino Serial comm-----------------------------------------------------------------------
        if delayFlag == 0:
            timeStart = time.time()
            delayFlag = 1
        elif delayFlag == 1:
           if (time.time()-timeStart) >= delayDesired:
            serialComm.write(str(int(freq)).encode('utf-8'))
            delayFlag = 0
            """

        #serialComm.write(str(int(freq)).encode('utf-8'))
        #delayfunc(0.05)

        #-----------------------------------------------------------------------------------------------

    elif comStatus == 3:
        ##sleep(0.5)
        delayfunc(0.05)
        freq = str(0)
        serialComm.write(str(int(freq)).encode('utf-8'))
        delayfunc(0.05)
        ##sleep(1)
        serialComm.close()
        print("Serial comm successfully closed")
        with open("StimLog.txt", "a") as f:
            eventMessage = "Serial comm closing mode successfully activated. \n"
            output = str(now) + " -> " + eventMessage
            f.write(output)
```

test_only_distance.py (C:\Users\Desktop\Stim\test_only_distance.py)

```
# !pip install pyserial
# !pip install serial

import Vortex
import time
from datetime import datetime
# import serial
# import pyserial
# import ctypes
from ExcavatorStim import StimGenerator, parameterSetting
# mode = 10 (for API mode), mode = 20 (for frequency and amplitude adjustment mode)
mode = 10
serialConnectionTrigger = 0 # To control serial connection -> 0: Before connected, 1: After connected
pastTime = 0.0 # For time gap
currentTime = 0.0

minFreqinLink = 10.0 # For setting minimum stimulation , Unit: Hz, This valus should be over 10.0 Hz
maxFreqinLink = 45.0 # For setting maximum stimulation , Unit: Hz, This valus should be under 100 Hz.
minDisinLink = 0.0# For setting minimum distance between a bucket and burried pipe. This valus should be
positive.
             # But, I recommend you not to change minDis. (10.0 <- optimal value)
maxDisinLink = 0.22 # For setting maximum distance between a bucket and burried pipe. This value should be
positive.
#minDisinLink = (minDisinLink/0.1)*10
#maxDisinLink = (maxDisinLink/0.1)*10

# 생성 주기가 60Hz인 데이터를 저장하는 변수
disval = 0

def delayfunc(delayTime):
    start = time.time()
    while ((time.time()-start)<=delayTime):
        continue

def on_simulation_start(extension):
    # Turn on the engine by default
    # extension.outputs.engine.value = True
    # # Set the throttle to 1.0 at the start, 1 represents the pedal pressed all the way in, we want to go
as fast as possible
    # extension.outputs.throttle.value = 1.0
    # # Switch to first gear right at the start
    # extension.outputs.gear.value = 1
    # if(abs(extension.inputs.Distance.value)) == 1:
    #     print("nice")

    print("Initialized")
if (disval<=0.3):
    parameterSetting(minFreqinLink, maxFreqinLink, minDisinLink, maxDisinLink, 1)
else:
    parameterSetting(minFreqinLink, maxFreqinLink, minDisinLink, maxDisinLink, 0)



# 데이터를 카운트하는 변수
data_count = 0

def pre_step(extension):
    global disval, data_count
    # extension
    if disval is None or data_count == 6:
        # 새로운 데이터 생성
        disvalL = extension.inputs.DistanceL.value
        disvalR = extension.inputs.DistanceR.value
        disval = min(disvalL, disvalR)
```

```
        # 데이터 카운트 초기화
        data_count = 1
    else:
        # 이전 데이터를 사용
        data_count += 1
    #print(disval)

    global serialConnectionTrigger
    global pastTime
    if abs(disval)<=0.3 and serialConnectionTrigger == 0: # Connect serial connection when dis>10m
        StimGenerator(0, mode, 1, float(maxDisinLink+1))
        for _ in range(5):
            StimGenerator(1, mode, 2)
            #print("Activated Standy")
        serialConnectionTrigger = 1
        #print("TestA")
    elif abs(disval) <= maxDisinLink and abs(disval)>=minDisinLink and serialConnectionTrigger == 1:
        StimGenerator(2, mode, 3, float(disval))

    elif abs(disval) > maxDisinLink and abs(disval)<=10 and serialConnectionTrigger == 1:
        StimGenerator(2, mode, 3, float(1))


    elif abs(disval) > 10 and serialConnectionTrigger == 1: # disconnect serial connection when dis>10m
        StimGenerator(2, mode, 3, float(maxDisinLink+1))
        StimGenerator(3, mode, 4)
        for _ in range(10):
            print("Abort Standy")
        serialConnectionTrigger = 3
```