COMP ENG 2DX4

Final Project

YIMING CHEN 400230266
cheny466@mcmaster.ca

Google Drive Links

1. Project Demonstration
   https://drive.google.com/file/d/1834mf2pD6TzD8n1pxulLxLUgXt0ew2O2/view?usp=drivesdk

2. Design Question
   https://drive.google.com/file/d/1gHq8R-_eUGfYCIXAI25eExYdifre2n6w/view?usp=drivesdk

1. Device Overview

1.1 Features

This device was constructed using a TI MSP432E401Y microcontroller combined with a VL53L1X time-of-flight (ToF) module. The MSP432E401Y has the following specifications:

- Operating voltage of 2.5 - 5.5V
- Default bus speed of 120MHz, (adjusted to 12MHz for this device)
- Two 12-bit SAR ADC modules, each up to 2 Mbps.
- 1024KB of flash memory, 256KB of SRAM and 6KB EEPROM
- Eight UARTs, each with a baud rate of up to 7.5 Mbps for regular speed and 15 Mbps for high speed

The VL53L1X module can measure distances up to 400cm away at a frequency of up to 50Hz, with an operating voltage of 2.6 – 3.5 V.

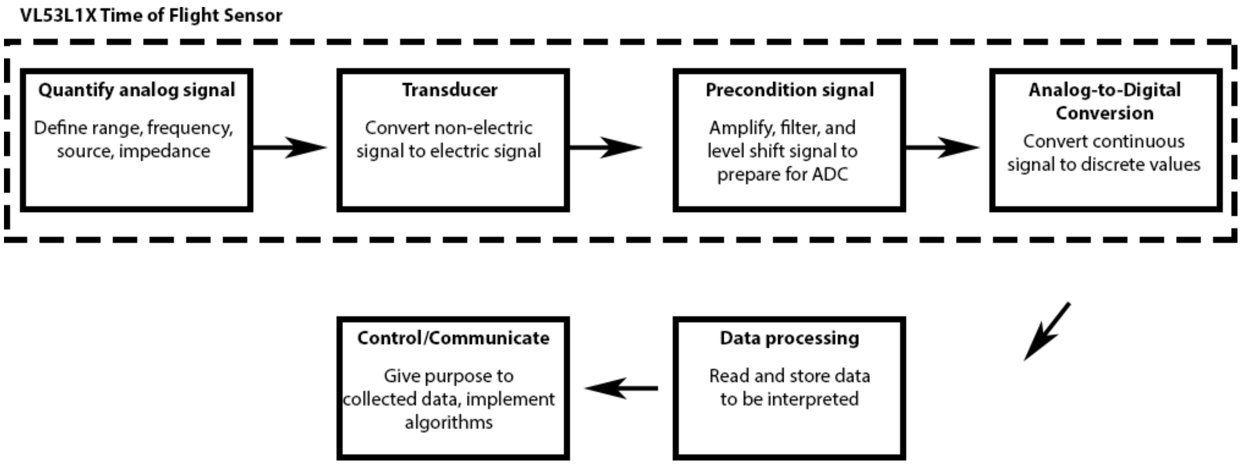
1.2 General Description

This device is an embedded system which integrates a ToF module -- VL53L1X to map enclosed indoor environments. This device is intended to be used as a component of other systems (e.g., robotics navigation, autonomous drone, layout mapping, etc.). Its core components include:

1. Digital I/O: momentary push button to control data acquisition.
2. Digital I/O: LED status of each distance measurement.
3. Transducer: ToF module to measure distance in y-z plane.
4. Data processing: coordinate collection, computation, and storage of distance and displacement.
5. Manually activated collection of new data, 360 degrees at a time, once defined fixed displacement reached.
6. Implementation mode: interrupt design implementation.
7. Control: rotation of stepper motor is controlled to process data collection with ToF module.
8. Communicate/Control: data communicated between ToF module and microcontroller.
9. Communicate/Control: data communicated to PC application, distance and displacement data stored.
10. Communicate: data read from Excel spreadsheet and visualized using Open3D.

The data that would be collected by the sensor will be stored in an Excel sheet formatted the same way as the given dataset.

There are several steps to finish the operation. The VL53L1X module collect distance (y-z plane) data acquisition. The signal is captured via the transducer. The captured signal will then be processed and discretized into a digital signal, which is the process of (A2D). Then the digital measurement will be transferred and store in an excel document and finally visualized using Open3D in python. The displacement in the x-axis is captured manually, by moving the device 20cm in each step along the x-axis.


VL53L1X Time of Flight Sensor
Quantify analog signal
Define range, frequency, source, impedance
Transducer
Convert non-electric signal to electric signal
Precondition signal
Amplify, filter, and level shift signal to prepare for ADC
Analog-to-Digital Conversion
Convert continuous signal to discrete values
Data processing
Read and store data to be interpreted
Control/Communicate
Give purpose to collected data, implement algorithms
Figure 1: Diagram of Basic Working Process

2. Device Characteristics

Feature	Detail
Motor Driving Pins	PH0 – PH3
SCL/SDA Pins	PB2/PB3
Bus Speed	12MHz
Communication Port	COM5
Communication Speed	115.2 kbps
Button	PM0

Refer to the table above is the ports needed to set up the device. The user should build the circuit accordingly and then to process the data through Python or something alike.

3. Detailed Description

3.1 Distance Measurement

To collect the distance data we will need to use three components:

a. Transducer: to convert the non-electric signal into electric signal;
b. Pre-conditioning circuit: to function as a filter for the data can be converted into digital form more easily;
c. ADC: to discretize the conditioned analog signal into a digital form.

VL53L1X ToF module is used for all the three requirements. It will detect the distance using the reflection of light for calculation. So the environment for the device to work should be choose at a room with comparatively dark light.

The distance data will be collected after each small rotations of the motor, then send the data to PC via UART. PC will then receive the data through PySerial. In this project, the data will transfer through recivedata.py which will open serial communication port COM5 at 115.2 kbps, 8N1. Then the script will write the data to Excel usting Python library called xlsxwriter. In the excel datasheet, each line represents the stepper motor rotates after a specific angle and then the module collects the data. Each locumn is a separate plane of the idsplacement. The calculation to establish the (x,y,z) plots from the raw measurements will be discussed here. The raw data is measurement on the y-z plane. These coordinates can be converted using trigonometry. The two components of the calculation are the angle and the distance measurement. Angles can be calculated from the following formula:

$$angle = \frac{360}{512} * motor\ position + 270\ degrees$$

For the Keil program, each movement for the motor increases at a pace of 360/512 degrees, and 512 steps for a full rotation.

After the ToF module collects the data for which the raw data is supposed to be a distance data. We will then apply trigonometry method to obtain for the y and z coordinates.

$$y = \frac{distance}{1000} * \sin\,(angle)$$

$$z = \frac{distance}{1000} * \cos\,(angle)$$

And for each squad of the four squads that build a complete coordinate in any plane. The different calculation will be used every four movements of the motor, which is 45 degrees (11.25 for each step). The sign of the corresponding coordinates varies each time when the measurement goes into the new region.

At last, after recording these values into the .xyz file, the open3d program will help to establish the following graphs.

3.2 Displacement Measurement

For the displacement measurement, the displacement is made manually by changing the device along x-axis increments of 200 mm. This displacement will automatically be presented after running the open3d.py program. There is no need for any extra modification to be made.

At each displacement, starting at x = 0 and incrementing by 200mm, a set of 512 distance measurements will be taken. 512 steps is the number of steps for the motor to rotate for a full cycle.

3.3 Visualization

For my computer, the configuration is Dell G5 (i7-9750H CPU @ 2.60GHz) and the graphics card is GTX 1660ti. 3D visualization of data is performed by importing data in the .xyz file and using Open3D library. Then the functions in the library will visualize the data.

The program is run by Python 3.8-64bit. The visualization is done by Open3D which includes functions to read and create the coordinates to place the corresponding points. Once the data is ready to be used by the library, a point cloud will be created using Opend3D function, the point cloud is also used to create a line set which contains all points in the point cloud, through this the presentation by the program will be more obvious and straightforward.

Under the 3-Program.zip folder, two sets of data have been collected for the user to test. However, because the movement along x-axis is done manually, the results may not be that precise. Besides, the problem also took when fixing the sensor with the motor and the wires are really messy, so there may be some small error taken place. The basic shape for the environment is done well I think.

4. Application Example

Before starting to implement the components together, please refer to the safety manual to make yourself safe.

1. Setup the circuit refer to the circuit schematic. Please make sure for every port and pins, the connection performs well. Attach the VL53L1X module correctly as well as the stepper motor.

2. Download and install a release of Python 3.8 accordingly. Run the setup and ensure Python 3.8 has been added to path. You can verify this step by typing ‘python’ on the command prompt.

3. Please make sure you install the following python library: Open3D, PySerial and numpy. If the package is not installed well, please type *pip install* **** on the command prompt.

After installing all the required software, now we are going to collect the data by the following instructions.

1. Open Keil project, build, translate and load the program on your microcontroller.

2. Press RESET SW1 and press the button to make the stepper motor rotate.

3. Wait the stepper motor rotate for a full cycle. And then move the device 0.2 forward along the x-axis.

4. Repeat step 3 for ten times to collect enough data for the 3D visualization to perform the points cloud.

5. Set COM port to be the appropriate one. For example, my port # is COM5. To determine the COM port being used for UART serial communication, enter the PC’s Device Manager and select ‘Show hidden devices’ under View. Choose the one listed as ‘XDS110 Class Application/User UART’.

6. Once the stepper motor begins to rotate and the python program will start to collect the data and store the data in the .xyz file.

After collecting the data, we will need to open the Open3D.py file to create the visualization results.

1. Open and run the open3d_visualization.py python file and adjust the file name where the data is stored.

2. A new window will appear on the screen automatically after executing the program, which is exactly the graph derived from the data collected.

5. Limitations

1. MSP432E401Y has an IEEE754 32-bit floating point unit (FPU) to support floating points with up to 32 bits of precision. However, it is not that easy to process the data. Instead, we will use Python to assist this process.

2. The Nyquist Rate should be at least twice as the frequency of the bus speed, which is 80 MHz, thus the Nyquist Rate should be 160 MHz. However, in this project I have already been assigned with the bus speed 120 MHz.

3. The wires that connecting the microcontroller and the motor will give a opposite force when the motor is rotating. And VL53L1X is fixed on the motor so this opposite force will pose reverse effects on the movement of the motor. This will decrease the accuracy of the data collected by sensor because each rotation should have been set as a specific angle and the opposite force will change this angle slightly. And through the whole process, these slight inaccuracies will finally make up of a comparatively large inaccuracy.

4. The maximum quantization equals the resolution and the resolution can be calculated as:

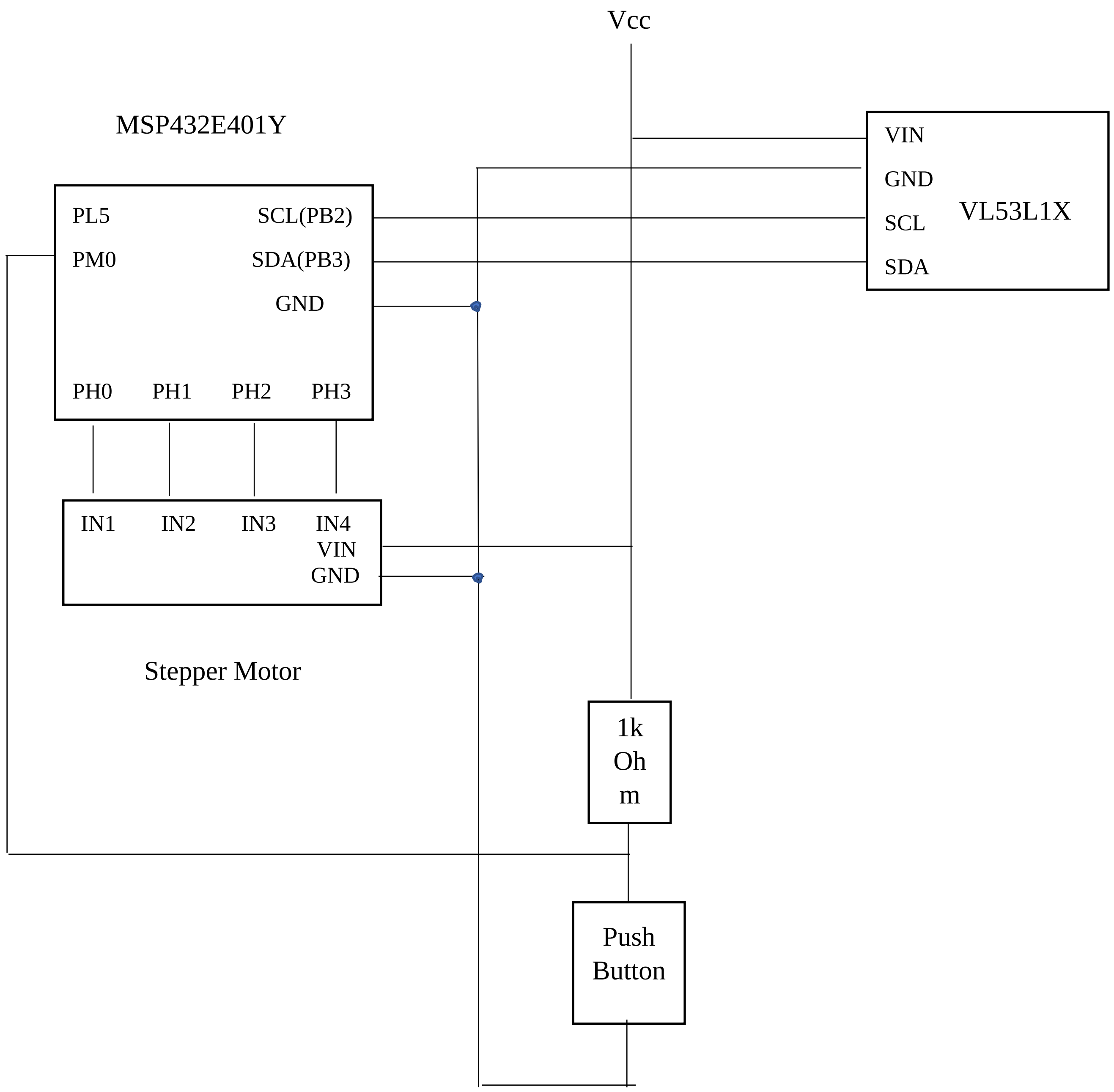
   $$\frac{V_{FS}}{2^m}$$

   m = number of bits and V_{FS} = full-scale voltage (5V in the project).
   The VL53L1X module has an 8-bit ADC, the MSP432E401Y has a 12-bit ADC and the IMU sensor has a 16-bit ADC, and these values give us the result:

   $$ToF\ res. = \frac{5}{2^8} = 19.53\ mV$$
   $$MCU\ res. = \frac{5}{2^{12}} = 1.22\ mV$$
   $$IMU\ res. = \frac{5}{2^{16}} = 76.30\ \mu V$$

5. The displacement measurement is made manually, so it will pose some small errors on the figures presented.

6. Circuit Schematic



7. Programming Logic Flowchart

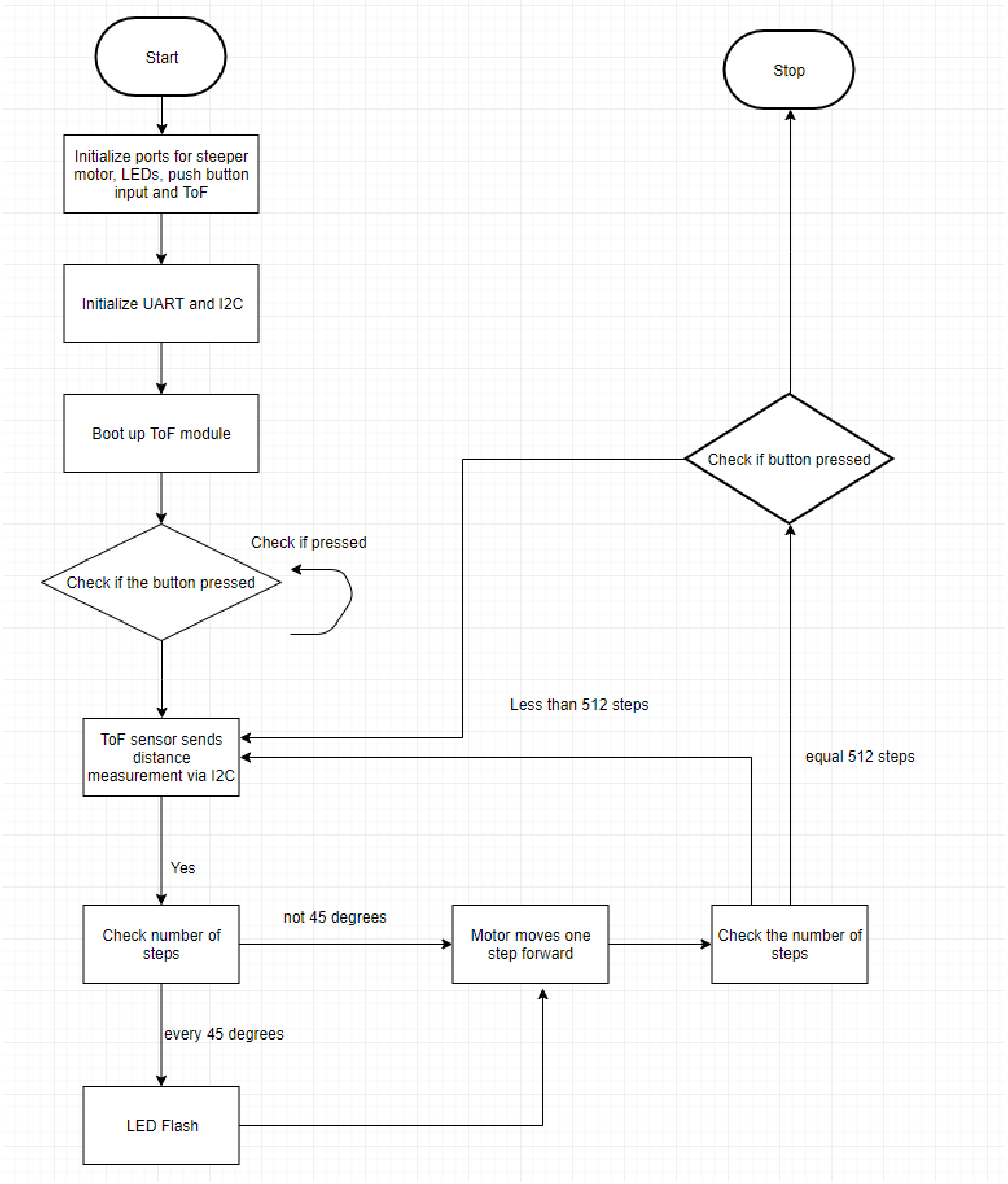

Start
Initialize ports for steeper motor, LEDs, push button input and ToF
Initialize UART and I2C
Boot up ToF module
Check if the button pressed
Check if pressed
ToF sensor sends distance measurement via I2C
Yes
Check number of steps
not 45 degrees
every 45 degrees
LED Flash
Motor moves one step forward
Check the number of steps
Less than 512 steps
equal 512 steps
Check if button pressed
Stop

Figure 2 MicroController Flowchart

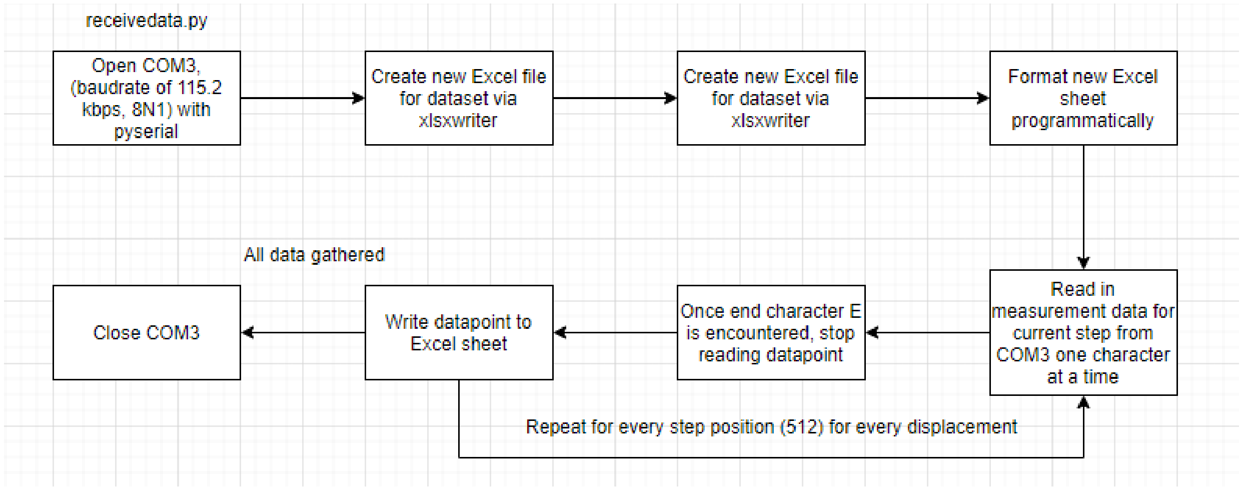

receivedata.py
Open COM3, (baudrate of 115.2 kbps, 8N1) with pyserial
Create new Excel file for dataset via xlsxwriter
Create new Excel file for dataset via xlsxwriter
Format new Excel sheet programmatically
Read in measurement data for current step from COM3 one character at a time
Once end character E is encountered, stop reading datapoint
Write datapoint to Excel sheet
All data gathered
Close COM3
Repeat for every step position (512) for every displacement

Figure 3 Python Flowchart

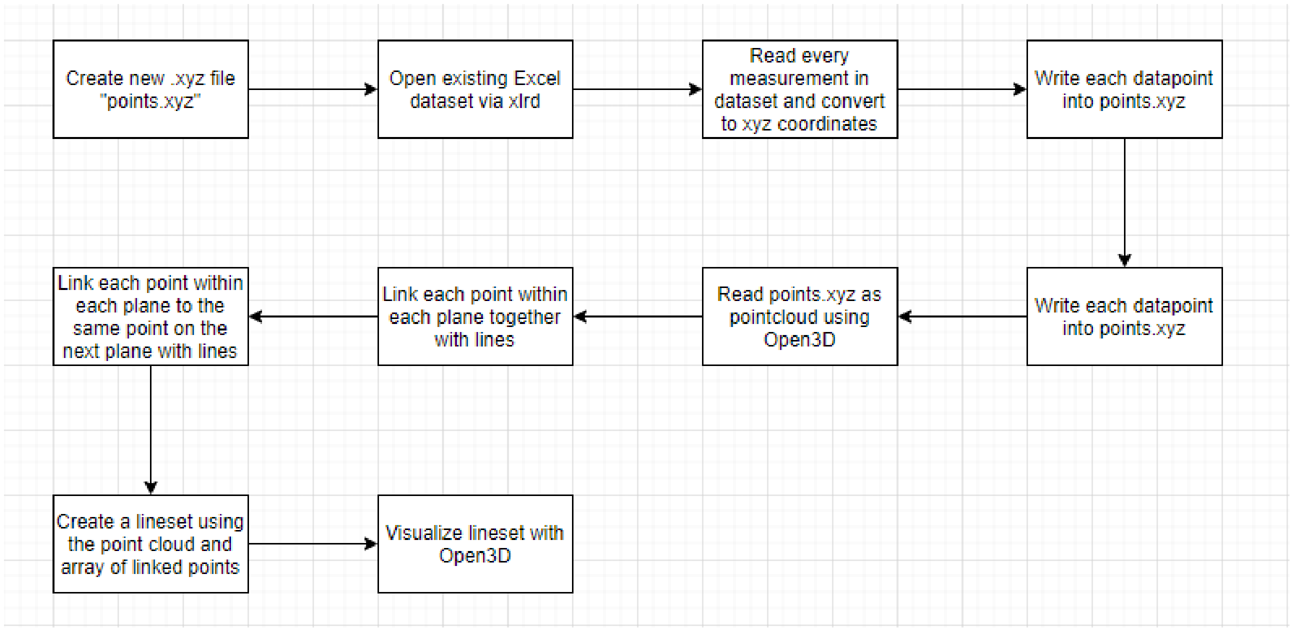

Create new .xyz file "points.xyz"
Open existing Excel dataset via xlrd
Read every measurement in dataset and convert to xyz coordinates
Write each datapoint into points.xyz
Write each datapoint into points.xyz
Read points.xyz as pointcloud using Open3D
Link each point within each plane together with lines
Link each point within each plane to the same point on the next plane with lines
Create a lineset using the point cloud and array of linked points
Visualize lineset with Open3D

Figure 4 Visualization Flowchart