COMP ENG 2DX3

Final Project Report

Instructor: Dr. Doyle, Dr. Athar, Dr. Haddara

Date: April 17, 2023

Abdullah Thabit – L04 – thabita – 400170696

Contents

TrueMeasure Indoor Spatial Imaging Device

Device Overview

Features

- Compact, fully self-contained system
  - 240x80x160 mm
- Captures multiple YZ plane measurements at pre-set X displacements to synthesize a 3D spatial image
- Interactive python program provides real-time updates during runtime and plots collected data
  - Open3d library for spatial imaging
  - Pyserial for UART communication
- System integrates a fully configurable microcontroller
  - Texas Insturemnts MSP432E401Y (~$75)
  - ARM Cortex M4F [1]
  - 1024 kB flash memory and 256 Kilobytes ram [1]
  - 32-bit bus [1]
  - Bus Speed: 48 MHz
  - Operating Voltage: ~5V [1]
- Unipolar Stepper motor
  - 28BYJ-48 (~$12)
  - 11.25º steps
  - Operating voltage: 5 V
- Programmable ToF sensor (~$7)
  - Used for Analog-to-digital conversion (ADC)
  - VL53L1X
  - 4 m maximum range [2]
  - Up to 50 Hz frequency [2]
  - Operating voltage: 3.3 V
- External push buttons control stepper motor and ToF sensor
- Onboard microcontroller LEDs provide measurement and status updates
- Return-to-home functionality allows user to re-capture data points at a given X-displacement
- I^2C serial communication facilitates data transfer between microcontroller and ToF sensor (Baud rate:100 kb/s)
- UART communication between micro and Python interfacing program (Baud rate: 115200 bits/s)
- C program is stored in microcontroller memory. Collected data stored in host device storage.

General Description:

The TrueMeasure is an indoor spatial imaging device which uses a cost-effective and accurate time of flight (ToF) sensor (VL53L1X) to collect data points along the YZ plane. The ToF sensor is mounted to a unipolar stepper motor programmed to rotate in 11.25º steps to achieve a 360º rotation. Through incremental displacements along the X-axis, several YZ planes are graphically manipulated to develop a high-fidelity 3D model of a space. Compared to LiDAR imaging, the components are relatively inexpensive, costing less than $100.

The TI MSP432E401Y microcontroller runs at a 48 Mhz bus speed and is configured with three external push buttons to control the stepper motor and data acquisition. The MSP432E401Y uses a 5V source to power the entire system and is equipped with 1024 kilobytes of flash memory and 256 kilobytes of random-access memory (RAM) [1], which store the program and temporary program variables during runtime. Onboard microcontroller LEDs provide status updates during runtime to ensure the program operates as intended. A Python user interface facilitates real-time program updates and develops a 3D plot of the collected data. If data collection is interrupted, a return-to-home function controlled by the third push-button resets the device position and data acquisition.

I2C serial communication transmits data from the ToF sensor to the TI microcontroller at a 100kb/s baud rate – this allows for complete control over the VL531L1X ToF sensor and successful data acquisition. UART serial communication transfers data between the microcontroller and the host device (baud rate: 115200 bits/s). Information transmitted through UART facilitates the Python user interface and allows for the graphical representation of measurements collected by the ToF sensor.

Block Diagram

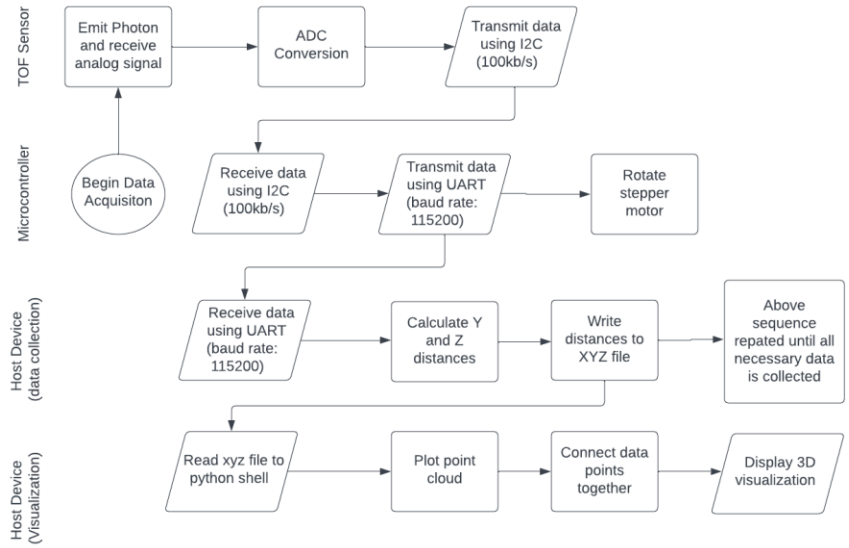

Figure 1 Block Diagram

Device Characteristics Table

Table 1 - Device Characteristics Table

Microcontrolled Bus Speed:	48 MHz
External Push Button Configuration	Active-Low configuration
Power Requirements	5 V (USB-A) to power MSP432E401Y (microcontroller supplies 5 V to stepper motor and 3.3 V to ToF sensor) [1][2]
Status LED	Onboard LED D1 flashes during program initialization. When Program initialization is complete, flashing stops. The status LED will also flash when the program interrupts, programmed to button 1-3, are raised.
Measurement LED	Onboard LED D2 flashes when the ToF sensor has received a measurement. In some instances, multiple measurements are taken to improve accuracy. In these cases, the LED will blink multiple instances.
Return-to-home	The third push button triggers the return-to-home interrupt. Once activated, the stepper motor will rotate until the home position is reached. Once at the home position, the device will be disarmed, and the stepper motor will be toggled off. To continue measuring, both features must be activated (press buttons 1 and 2)
Python libraries needed	math, open3d, numpy, and pyserial
Python customization	To adjust the number of scans or X-displacement per scan, the "xDisp" and "num_scans" variables must be adjusted in the Python code
Maximum range	4 m (in dark, ideal conditions.) [2]
Starting Position	ToF sensor emitter pointing upward at 90^0
Baud Rate	I2C → 100Kbps; UART → 115200 bps

Pin Assignments:

Table 2 - Microcontroller Pin Configurations

Microcontroller Pin:	Function:
PE0	Stepper motor control toggle (input)
PF0	Data acquisition control toggle (input)
PD0	Return-to-home trigger (input)
PH0-3	Stepper motor control (output)
PB2	I^2C SCL (ToF sensor)
PB3	I^2C SDA (ToF sensor)
PD2	Periodic interrupt to verify 48 MHz bus speed

Detailed Description

Distance Measurement

The VL53L1X ToF sensor is used to measure distances accurately. The module uses a 940 nm invisible laser to emit and receive a photon [2]. Since the speed of light is a constant, the time elapsed between the emission and return of the photon is used to determine the distance of an object [3].

$$distance = c \times \frac{time\ for\ photon\ to\ return}{2}$$

The manufacturer provided ToF sensor application programming interface (API) allows for complete customization of the module [2]. To control the module, the VL53L1X sensor is configured in a leader/follower arrangement with the TI microcontroller. I^2C communication protocol at a 100bps baud rate facilitates data transfer between the microcontroller and the ToF sensor. To arrange I^2C, the microcontroller pin PB2 is used as the system clock and PB3 is used as the data line.

During the ToF sensor initialization, several settings were adjusted from default configuration to improve long-range distance measurements:

1. Distance Mode set to “long”
2. Timing budget increased to 500 ms
3. Time waited between measurements increased to 500 ms

By committing these changes, more time was allotted for the sensor to capture measurements, effectively increasing its range closer to the 4 m maximum [2] – this is critical for distance measurement since range is the primary limiting factor of the ToF sensor.

Error detection and distance value averaging are safeguards that ensure that accuracy of measurements captured by the ToF sensor. When data is retrieved from the ToF sensor using the StartRanging() function, the RangeStatus() bit is checked. If the bit is nonzero, the data is invalid [2], so it is dumped and remeasured before the program continues. Additionally, rather than using a single measurement, the device captures three data points of a single measurement and uses the average value as the final distance. This way, the error associated with the measured distance is reduced. The GetDistance() function is used to collect the distance measurement from the microcontroller in mm.

The ToF sensor is mounted to a unipolar stepper motor to capture a 360º scan of the YZ plane. The 28BYJ-48 is a unipolar stepper motor that is configured in full-step mode. Using PH0:3, logic low is transmitted in a sequence which induces clockwise or counterclockwise motion. A step angle of 11.25º is configured to ensure reasonable resolution of a scan. As such, 32 measurements are taken per YZ plane (360/11.25 = 32).

Since measurements from the ToF sensor are represented in mm, trigonometry is used to find the Y and Z components for each measurement. With a starting angle of 90 degrees above the horizontal, the Y and Z components are the following:

$Y = distance\ \times \cos\left(\frac{\pi}{180^0}\ \times angle\right)$ where angle is [-270, 450] degrees

$Z = distance\ \times \sin\left(\frac{\pi}{180^0}\ \times angle\right)$ where angle is [-270, 450] degrees

To begin capturing data points, button 1 and button 2 (Stepper motor control toggle & Data acquisition control toggle) must be enabled (pressed once). Pressing these buttons enables a program status flag which indicates that start of data acquisition. I data capture is disrupted, button 3 (return-to-home trigger) will trigger an interrupt service routine which stops data restarts data acquisition of a YZ plane at a given x-displacement

Sample calculation:
ToF sensor starts at position = 90 deg.

$$Y = distance\ \times \cos\left(\frac{\pi}{180^0}\ \times 90\right) = 0\ mm$$

$$Z = distance\ \times \sin\left(\frac{\pi}{180^0}\ \times 90\right) = distance\ \times 1 = +distance\ mm$$

Visualization

To develop 3D spatial reconstruction, multiple YZ measurement planes are required. Displacing the device using regular increments along the X-axis and connecting the YZ planes is an effective way of developing 3D models. The device allows the user to customize the number of YZ measurements along with the x-axis displacement between YZ planes. To increase the resolution of the 3D model, the number of measurements should be high with small increments along the X-axis (default X-displacement is 40 cm).

Once the measurements are collected using the ToF sensor and written into an xyz file, the open3d python library is used to visualize the data. A point cloud representation may be used to attain an understanding of the data captured by the ToF sensor.

To achieve a 3D model, the data points are assigned vertex values by allocating them to an array. Next, the vertexes for each YZ plane must be connected using straight lines. By connecting the vertexes of a given plane using the “array.append” function, a 2D rendering is formed. After connecting lines through each individual plane, the “array.append” function must be used to connect the sequential YZ planes together to form a 3D model. For instance, considering an x-displacement to 1 m per measurement, the YZ plane at x = 0 must connect with the YZ plane at x = 1 m. Once straight lines are appended, a 3D image is formed. The LineSet() and draw_geometries functions from the open3d library are used to generate the 3D model.

Application Example

Figure 2 - Assigned Hallway (G) in ETB, McMaster University

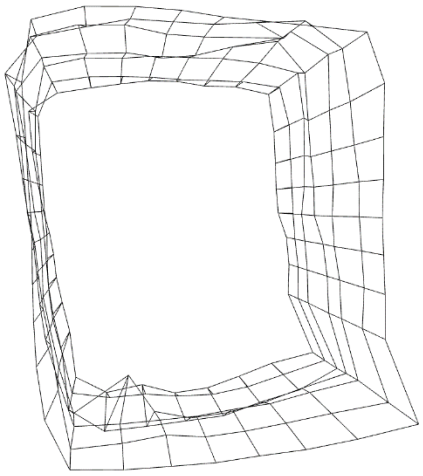

Figure 3- 3D model of ETB hallway in Figure 2

Figures 2 and 3 demonstrate hallway G, an assigned space in the Engineering Technology Building at McMaster University, and a digital reconstruction developed by the device. While some image distortion exists, the higher portion of the ceiling is oriented on the right side of the 3D spatial construction. For this example, 7 YZ plane measurements were taken, with a 70 mm X-displacment. To improve image resolution, the X-displacement per plane should be reduced and the number of YZ planes should be increased.

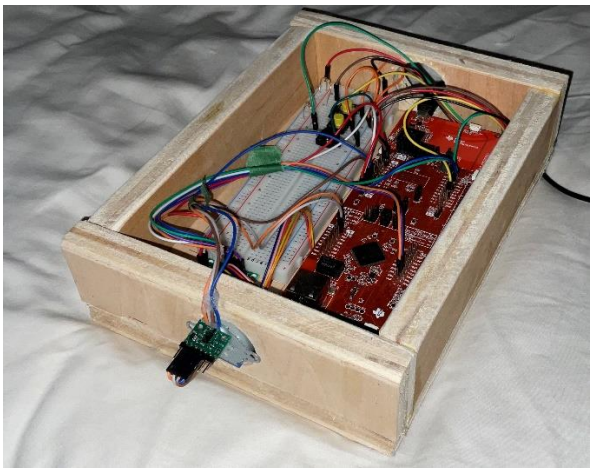

Figure 4 - Device With Correct Sensor Starting Position

To operate the device, the ToF sensor must be held so that the scanner is pointing upward (Figure 4). In this orientation, the Y-plane is defined as the horizontal component of the room and the Z-plane is the vertical component. The location where the device is held is the origin, with the right-side direction being +Y and the upward direction being +Z. Consequently, the leftward direction is -Y and downward direction is -Z. The +X direction is the direction the user displaces themselves for each YZ plane.

To configure the device, the following steps are needed:

1. Connect hardware based on circuit diagram
2. Install a python version supported by open3d on host device (3.8-3.10)
3. Install open3d, pyserial, and numpy libraries
4. Connect microcontroller to host device using USB
5. Identify serial port of device using device manager → Ports
6. Open python file and adjust communication port based on its assignment on the host
7. Adjust the number of measurements and the x-displacement on the python script to desired value (increased number of measurements and smaller x-displacement, high resolution but slow measuring)
    a. Refer to device characteristics table for the variable names for number of measurements and the x-displacement
8. Run python program
9. Restart C program using onboard microcontroller reset button
10. Wait for message on python shell indicating device is ready for measurements
11. Press button 1 and 2 to start measuring
12. Wait until 360 degree scan is complete (stepper motor will stop automatically)
    a. If measuring is interrupted, press button 3 to initiate return-to home functionality. Return to step 10 in this case.
13. Move along the positive x-axis by the pre-set value determined by the Python script
14. Press button 1 to begin measuring next YZ plane
15. Repeat until entire area is covered
16. Inspect point cloud to ensure are as expected (pattern resembles scanned area)
17. Assess 3D rendering to ensure accurate modelling of the space. The features and scale should resemble the physical space if the resolution is high. If the resolution is low, decrease the x-displacement increments to achieve an improved model.

Limitations

1. Range: The ToF sensor has a maximum range of 4m. Distances further than this threshold are not processed correctly
2. Reduced measurement accuracy causes error in 3D renderings. Some measurements may be smaller or larger than the true value because of rounding, causing inaccurate 3D spatial images. Consequently, the final 3D model may look different to the expected result.
    - Floating point values are used to store variables in python. Floating point numbers are 64-bit double precision values [4]. As such, the values are not represented as exact, increasing error for measurements. Several operations using floating point numbers reduces their accuracy.
    - Using trigonometry to calculate values for distances induces a small percent error. Since the trigonometric numbers are stored in floating point data types, an error is associated with the value – this increases measurement uncertainty.

3. Less accurate resolution due to quantization error:

   - The ToF sensor quantization error is equivalent to the resolution of the device $\frac{VF}{2^n} = \frac{5V}{2^8} = 19.53\ mV$

4. I2C (100 kb/s) and UART (115200 bits/s) serial communications were both implemented for the project
   - Since data is serially transferred over USB, the theoretical maximum data transfer speed is 480 Mbps (USB 2.0) [5]. However, the TI microcontroller cannot transmit data at speeds appropriate for 480 Mbps sampling. Working at these limits will likely cause errors on the receiving end [6]. To work within the threshold of the microcontroller, the standard baud rate is 115200 bps – this may be verified by using Realterm and adjusting the baud rate to assess the maximum allowable value to transmit/receive valid data.
5. The measuring process is relatively slow, so the device is time consuming to use. The primary limiting factors on speed are the ToF sensor and the stepper motor. The ToF sensor has a 50 Hz maximum frequency but is programmed to capture measurements at 500 ms intervals to improve range. Since 3 measurements are captured for every data point, the time of capturing a single measurement is further increased. The stepper motor is limited by a minimum wait time of 2 ms between steps. As such, the speed of the device is reduced. To achieve precise results, a 360-degree plane can take up to 48 s. To determine the wait times for stepper motor and ToF sensor operations, the time elapsed between operations is analyzed. Through trial and error, the minimum time from stepper motor movement and ToF data acquisition for reliable device operation was determined.

Circuit Schematic

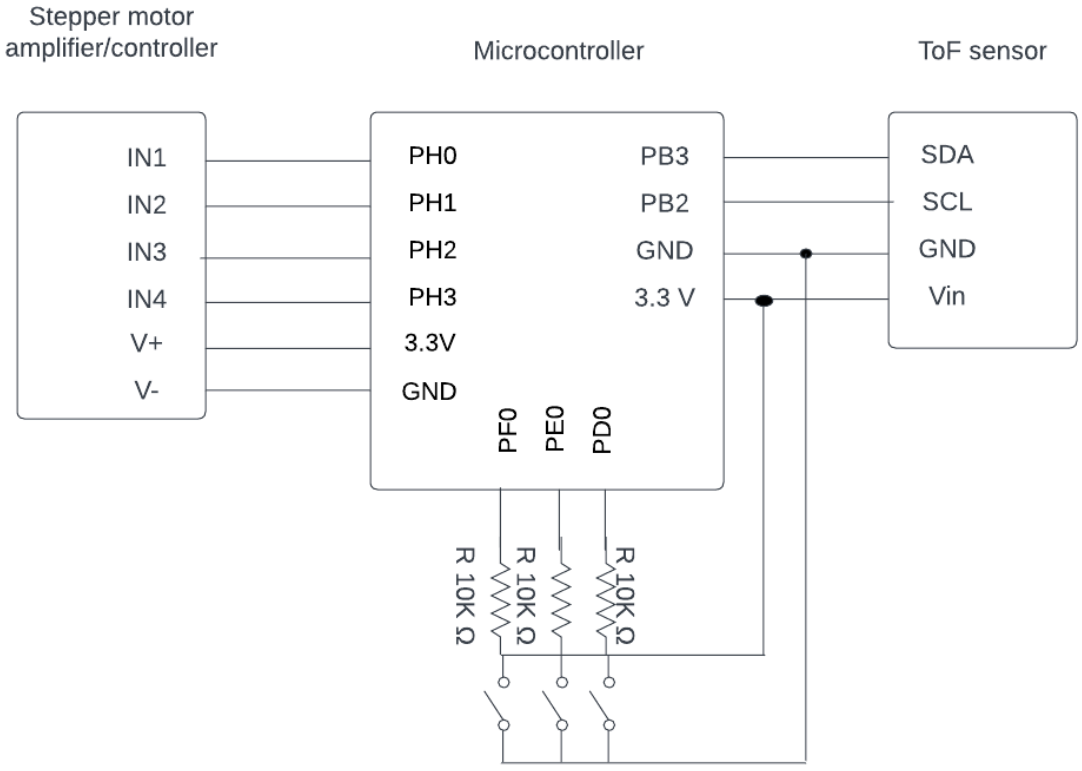

Figure 5- Circuit Schematics of Device (PE0 is button 1; PD0 is button 2; PF1 is button 3)

Programming Logic Flowchart

C program (microcontroller)

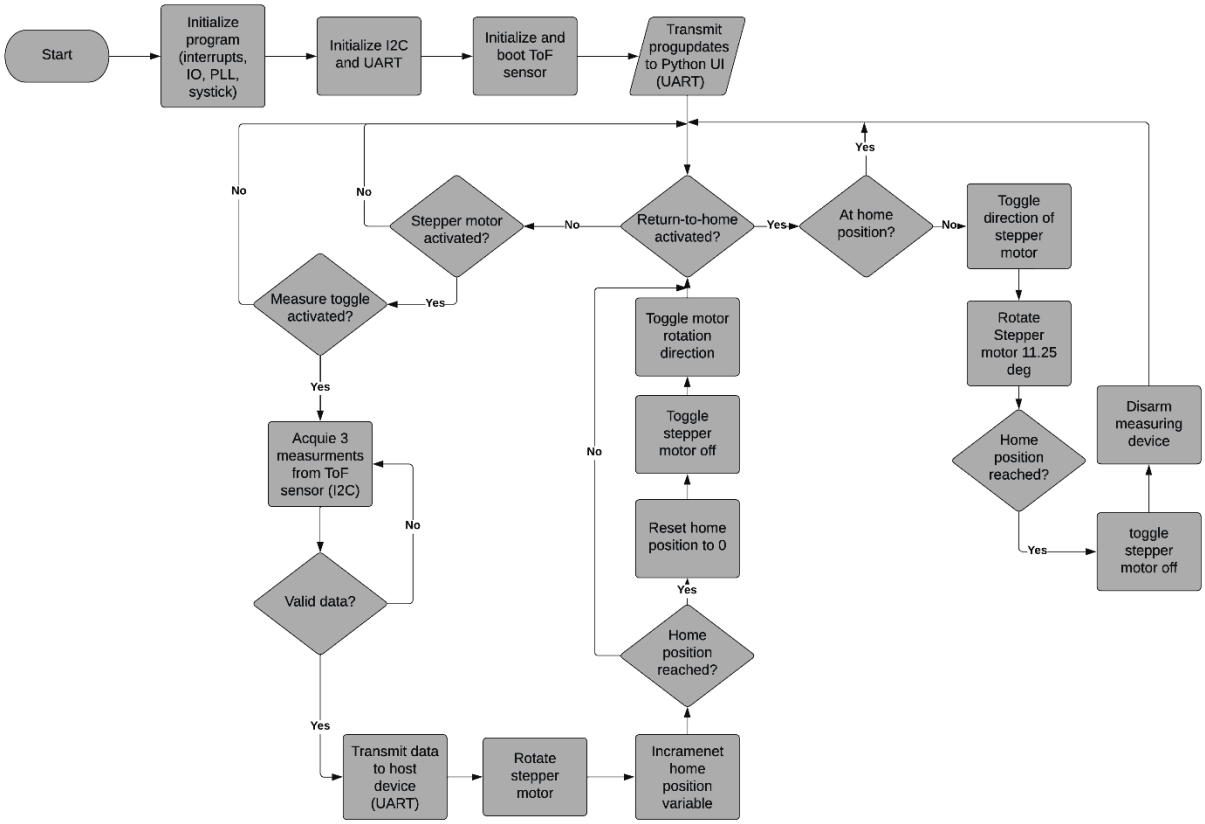

Figure 6- Flow Diagram of C program

Python Program (visualization)

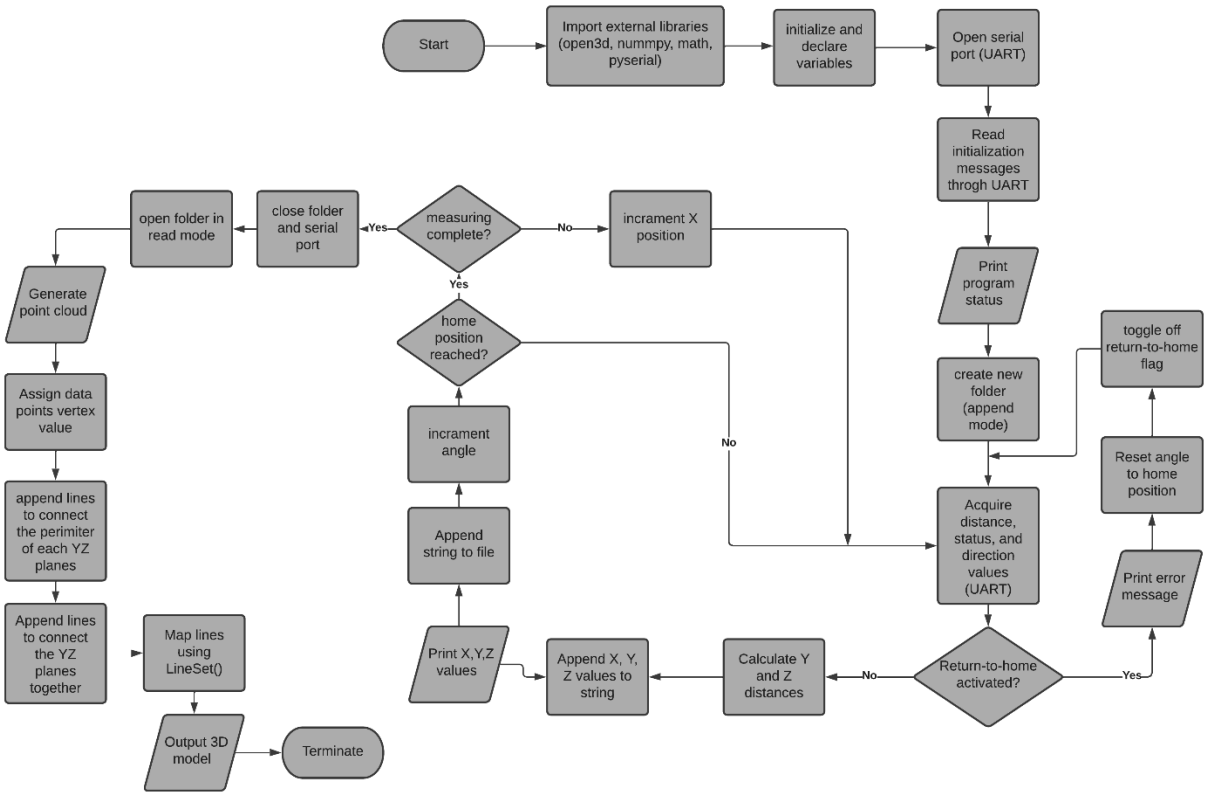

Figure 7- Flow Diagram of Python Code

References

[1] Texas Instruments, Dallas, Texas, United States. *MSP432E4 SimpleLink Microcontrollers Technical Reference Manual***,** (2018). Accessed: Apr. 15, 2023. [Online]. Available: https://www.ti.com/lit/ug/slau723a/slau723a.pdf?ts=1681793770183&ref_url=https%253A%252F%252Fwww.google.com%252F

[2] STMicroelectonics, Plan-les-Ouates, Geneva, Switzerland. *VL53L1X Datasheet,* (2022). Accessed: Apr. 15, 2023. [Online]. Available: https://www.ti.com/lit/ug/slau723a/slau723a.pdf?ts=1681793770183&ref_url=https%253A%252F%252Fwww.google.com%252F

[3] Y. Haddara. (2023). Week 8: Time of Flight Sensor Interfacing and Application [PowerPoint]. Available: https://avenue.cllmcmaster.ca/d2l/le/content/512368/viewContent/4101895/View

[4] Pyhton Software Foundations, Wilmington, Delaware, United States. *The Pyhton Language Reference* (2023). Accessed: Apr. 16, 2023. [Online]. Available: https://docs.python.org/3/reference/

[5] D. K. Sahu and R. Vaishya, "A Survey of USB 3.0 with data Transmission Techniques," IJERT, vol. 02, no. 09, pp. 3175–3178, Sept. 2013. Accessed: Apr. 16, 2023, doi: 10.17577/IJERTV2IS90844. [Online]. Available: https://www.ijert.org/a-survey-of-usb-3-0-with-data-transmission-techniques

[6] J. Lindblom. "Serial Communication." Sparkfun. learn.sparkfun.com (retrieved April 16, 2023).