Microprocessor Systems Project 2DX3
Spatial Mapping System

Instructors: Dr. Athar/Doyle/Haddara

Areeba Irfan - L07 - irfana20 - 400378045

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by [**Areeba Irfan, irfana20, 400378045**]

General Description

This spatial measurement system creates a 3D visualization of an indoor space by scanning and collecting distance points of nearby objects. The motor will turn 360 degrees and periodically scan and send the data every 22.5 degrees for a total of 16 sample data points every rotation. Once the desired number of scans are completed, the data is sent for 3D visualization through. The system works by acquiring waves of light from the sensor, which is converted through ADC to be read from the microcontroller.

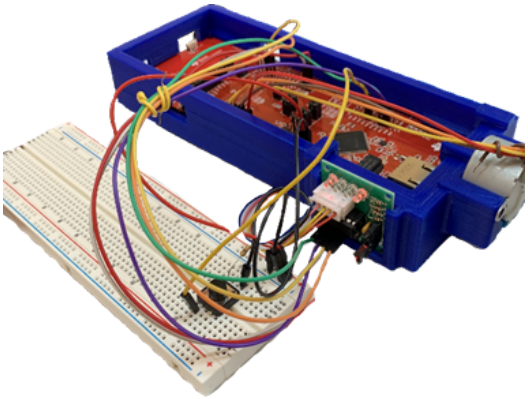

Figure 1: Spatial Mapping System

The system is composed of the Texas Instrument MSP432E401Y, VL531X Time of Flight Sensor, and the ULN2003 Driver connected to a stepper motor, as well as wires, a resistor, two LEDs, and a pushbutton. The VL531X Time of Flight Sensor is mounted onto the stepper motor using a 3D printed box so it is able to scan data as the motor rotates. The code for the microcontroller was programmed in C using Keil and data acquisition/visualization in python.

The overall system uses serial to data communication. The microcontroller and Time-of-Flight sensor communicate using I2C protocol. Once the data is sent from the Time-of-Flight sensor to the microcontroller, the microcontroller and PC communicate through UART protocol. Finally, the data received from the sensor is visualized using Open3D in python.

Key Features

Texas Instrument MSP432E401Y

- Default Bus speed of 120 MHz (16 MHz for this device)
- 2.5 – 5.5 V operating voltage range
- 1024 KB flash memory, 256KB SRAM, 6KB EEPROM
- Programmed in C
- Cost: ~$70.36
- ADC: 20 channel x 12 bit ADC

VL531X ToF Sensor

- I2C Interface
- 2.6 – 5.5 operating voltage range
- Maximum distance measurement of 4000 mm
- 900 nm invisible laser emitter
- 50 Hz ranging frequency
- Cost: ~$11.29
- Long distance mode used

ULN2003 Driver

- Unidirectional
- 512 steps per rotation
- 5 – 12 V operating voltage range
- Cost: ~ $3.99

Communication Protocols

- I2C: VL531X ToF Sensor to Texas Instrument MSP432E401Y - Frequency of 100kHz
- UART: Texas Instrument MSP432E401Y to PC (python) - Baud rate of 115.2 kbps

Block Diagram

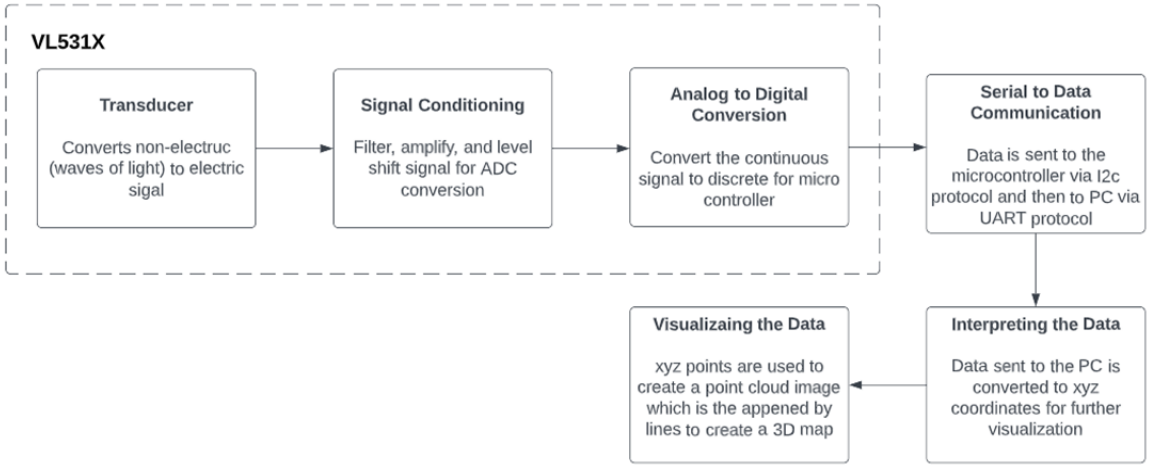

Figure 2: System's Data Flow

Device Characteristics

VL531X ToF Sensor	Texas Instrument MSP432E401Y
VDD	N/A
VIN	3.3V
GND	GND
SDA	PB3
SCL	PB2
XSHUT	N/A
GPIO1	N/A
ULN2003 Driver	**Texas Instrument MSP432E401Y**
VIN	5V
GND	GND
IN1	PH0
IN2	PH1
IN3	PH2
IN4	PH3
User LEDs	**Texas Instrument MSP432E401Y**
Measurement status	PF0
VL531X ToF Sensor status	PF4
Push Button	**Texas Instrument MSP432E401Y**
Start data acquisition/Start rotation	PM0
Device Set-up	
Bus Speed	16 MHz
COM Port	COM3 (Depends on Device)
UART Baud Rate	115.2 kbps

Distance Measurement

The spatial measurement system utilizes the VL531X ToF Sensor to acquire distance measurements. The Time-of-Flight sensor uses LIDAR technology to measure how long it takes for emitted pulses of 940 nm infrared laser light to reach a near object and be reflected to the detector. For this device, the long distance mode was utilized to ensure that all measurements are accurately being scanned, even if they are far away. The time delay between the emitted and received signal is measured to calculate the distance. The distance is measured through the Time-of-Flight principal formula,

$$Measured\ distance\ =\ Photon\ travel\ time\ /\ 2\ *\ Speed\ of\ light$$

The resulting calculation is then manipulated and communicated to the microcontroller in millimeters. For the microcontroller to interact with the VL53L1X, the manufacturer (ST) provides an application programming interface (API). The API provides a software interface to hardware devices which enables the operating system and other computer programs, allowing them to access hardware functions without the knowledge of precise details about the used hardware. The following API functions were used:

VL53L1X_BootState:

- Checks that the sensor has been booted

VL53L1X_SensorInit:

- Called once to initialize the sensor with default configuration

VL53L1X_StartRanging

- Used to make a distance measurement

VL531L1X_CheckForDataReady

- Returns ‘1’ when new ranging data is ready

VL53L1X_GetDistance

- Main get ranging data function
    - Provides the distance measured in mm

VL53L1X_ClearInterrupt

- Clears the interrupt
    - Strongly recommended to enable the next interrupt whe new ranging data is ready

VL53L1X_Stop

- Stops the current ranging operation

Once the Time-of-Flight sensor starts getting data, it is communicated to the microcontroller using I2C protocol. These data points can be seen by using UART communication protocol between the PC and microcontroller. The resolution of the final 3D mapping depends on the number of samples taken on the rotations. For this system, a sample size of 16 measurements was chosen to map the area. This requires a rotation of 32 steps or 22.5 degrees everytime the sensor takes a measurement. 16 sample measurements were chosen to produce a final high resolution image of the scan.

The device starts by prompting the user to enter in the amount of rotations they would like for their scan. This number corresponds to the x coordinate displacement by increasing the x value by 200 mm each rotation. Once the user enters the number of scans, the device will wait for the button to be pressed. A polling-driven solution has been implemented for this device as it constantly scans for an external push button to be pushed to start the scanning. The button has also been implemented to start the data acquisition process. Once the button has been pressed, the sensor will start communicating the distances measured to the microcontroller. The distances being sent to the microcontroller are then sent to the PC for further processing. Within the PC, the measurements are being calculated to be converted to xyz coordinates. Once a rotation has been completed, the sensor will stop and the variable keeping track of the number of rotations, as well as the x displacement will increase. Everytime a rotation has been completed, the stepper motor will rotate counter clockwise, allowing the sensor to return to home. This also allows for any tangled wires to untangle as well. The user will now walk forward to account for the x displacement in their next measurement. The device will then wait for the button to be pressed again and this process will be repeated until the number of rotations has been completed.

Once the sensor has acquired all of the distance measurements, the data has been sent to the PC through UART communication protocol from the microcontroller. There are two python files that are used for the PC visualization of the system. The first file is where the user prompts the number of scans. Here the data is being communicated from the microcontroller. Within this python file, an xyz file is being created where all the data will be stored on the hard drive. Before being sent to the xyz file, the data is first converted to y and z coordinates. There is no need to convert to x coordinate because that is being manually incremented in the code. The following formulas are used to convert the distance data to y and z coordinates,

```
y = hyp*math.cos(angle)

z = hyp*math.sin(angle)
```

The variable hyp is the acquired data point as it acts as the hypotenuse for the calculations. The angle variable is calculated using the following formula,

```
angle = (motor_steps/512)*2*math.pi
```

This formula calculates the angle based on how many steps the stepper motor has taken. Both the serial and math libraries are needed to run and use this file.

The following shows an example of a calculation using a measured distance of 1050 and angle of 67.5 .

$$y = 1050 * cos(67.5°) = 970.0735$$
$$z = 1050 * sin(67.5°) = 401.8176$$

Visualization

Once all measurements have been scanned and converted to xyz coordinates, the file is closed and saved onto the hard drive. The other python file utilizes the numpy and open3D libraries to visualize this data. The file starts off by opening the created .xyz file and first creates a cloud point visualization when the file is run. The lines are then appended between adjacent points in each x increment to create a final 3D map of the surrounding area scanned.

Application Example

To test the functionality of the system, a hallway was scanned to see how the device would work together to display the 3D mapping. The following hallway was used to test the accuracy of the device.

Figure 3: Hallway to be scanned

It should be noted that the 3D scan should not be a completely straight rectangular prism due to the staircase railing and side of the wall being at different displacements. This is expected to produce some spikes on one side of the wall in the final 3D representation. Since the other wall is straight, a clean line can be expected for that side. The long distance mode is activated so that the sensor will be able to reach these distances accurately while scanning.

Instructions

The following instructions are provided for the application highlighted above to display the functionality and use of the spatial mapping device to the user.

1. Ensure that the VL531X ToF Sensor is successfully mounted onto the stepper motor. This can be done through a variety of ways. For instance, a 3D printed piece or connected lego pieces.

2. Follow the circuit schematic (figure) to successfully connect the Texas Instrument MSP432E401Y, VL531X ToF Sensor, ULN2003 Driver/stepper motor, PC, and the push button.

3. Once everything has been connected, the C code in the Keil IDE should be translated, built, and loaded onto the Texas Instrument MSP432E401Y. Once the microcontroller is reset, the user should see LED 3 on the board flash twice. This will indicate that the VL531X ToF Sensor has been booted successfully and ready to take distance measurements.

4. The python file that will receive the measurements should now be opened and be set to the right configuration. The baud rate should be 115.2 kbps, but the COM port should be adjusted to the user's computer.

5. Run the python file and enter in the number of rotations desired.

6. Press the button to start the first rotation. Once the rotation has been completed, press the button again. Repeat this process until the number of rotations has been achieved.

7. Once all data has been acquired, open the second python file and run it.

8. Now the user will be able to see the point cloud visualization of the acquired data. To view the image at different angles, the mouse can be clicked and dragged to change the view

9. To view the final 3D representation, close the point cloud visualization window and the final 3D map will pop up. The mouse can be used once again to view the image in different angles.

Expected Outcome

The following images of the hallway were produced after following all instructions.

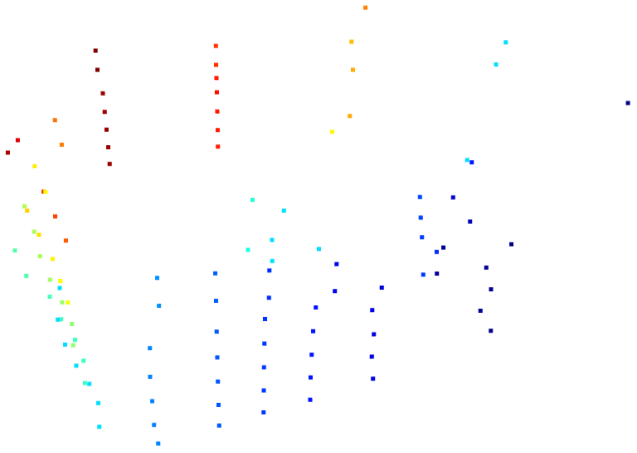

Figure 4: Point Cloud Scan of Hallway

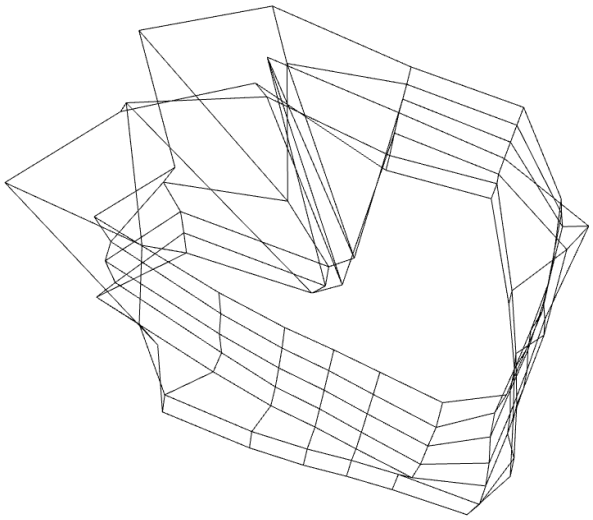

Figure 5: 3D Representation of Hallway

As expected, it can be seen that the final scan has some spiking on one of the walls. This is because the sensor is getting points of both the railing and the other wall which is at a much further displacement than the railing. This can be avoided by scanning a clean hallway with no discrepancies. Within the scan, y and z represent vertical scans, while x is defined as the displacement.

Limitations

One limitation of this device is the microcontroller floating point capability. The Texas Instrument MSP432E401Y uses the Cortex-M4F 32 bit processor. The Cortex-M4F has a floating-point unit (FPU) which can cause some limitations when calculating the xyz coordinates using the trigonometric functions. The limitation is due to the fact that the numbers being processed are only accurate to the 32 bits available. When the trigonometric functions, sine and cos are used, the numbers are capped based on the FPU, which is 32 bits. The combination of these limitations can lead to multiple issues. For instance, reduced system performance. The system might respond slower to the calculations, leading to an increased processing time. Another issue is inaccurate results. The lack of accuracy in the calculations can lead to inaccurate data points.

Another limitation that roots from the device is the quantization error. The quantization error depends on a number of factors including, the number of bits in the converter, noise, and nonlinearities. The quantization error read by the ToF module can be calculated.

$$Max\ distance\ reading\ of\ VL531X\ ToF\ Sensor\ =\ 4000\ mm$$
$$ADC\ Bits\ =\ 16\ bits$$

$$Max\ Quantization\ Error\ =\ Max\ distance\ reading/2^{ADC\ Bits}$$
$$Max\ Quantization\ Error\ =\ 4000\ mm\ /\ 2^{16}$$
$$Max\ Quantization\ Error\ =\ 0.0610\ nm$$

Based on the properties of the port in device manager, it can be seen that the maximum standard serial communication rate that can be implemented with the PC is 128 kbps. The baud rate that was implemented for this project was 115.2 kbps. This was confirmed by setting up the device to ReaTerm and verifying that the data was being sent correctly when configured to 115.2 kbps.

To communicate between the Texas Instrument MSP432E401Y and the VL531X ToF Sensor, I2C protocol was used. The frequency for the communication protocol is 100 kHz.

After reviewing the whole system. It can be determined that the VL531X ToF Sensor and stepper motor are the main element limiting the devices. One limitation of the ToF sensor is that the max distance it can receive is 4000 mm. This means that if the nearby object is further than 4000 mm, the ToF sensor will not be able to accurately determine the distance. This was tested by placing objects further than 4000 mm and seeing the distance measured. The ToF timing

budget can be set from 20 ms to 1000ms. Since this device uses the long distance mode, 140 ms is the timing budget, which indicates that the ranging time for each scan is 140 ms. When scanning, the stepper motor also required time to rotate between each scanning position. When combining the time for the stepper motor to move as well as the timing budget, it is significantly more than the time needed for data transmission times. This means that the ToF sensor and stepper motor are negatively affecting the overall speed of the device. This limitation was tested by going through different speeds of the system and seeing the time required for each data transmission.

Circuit Schematic

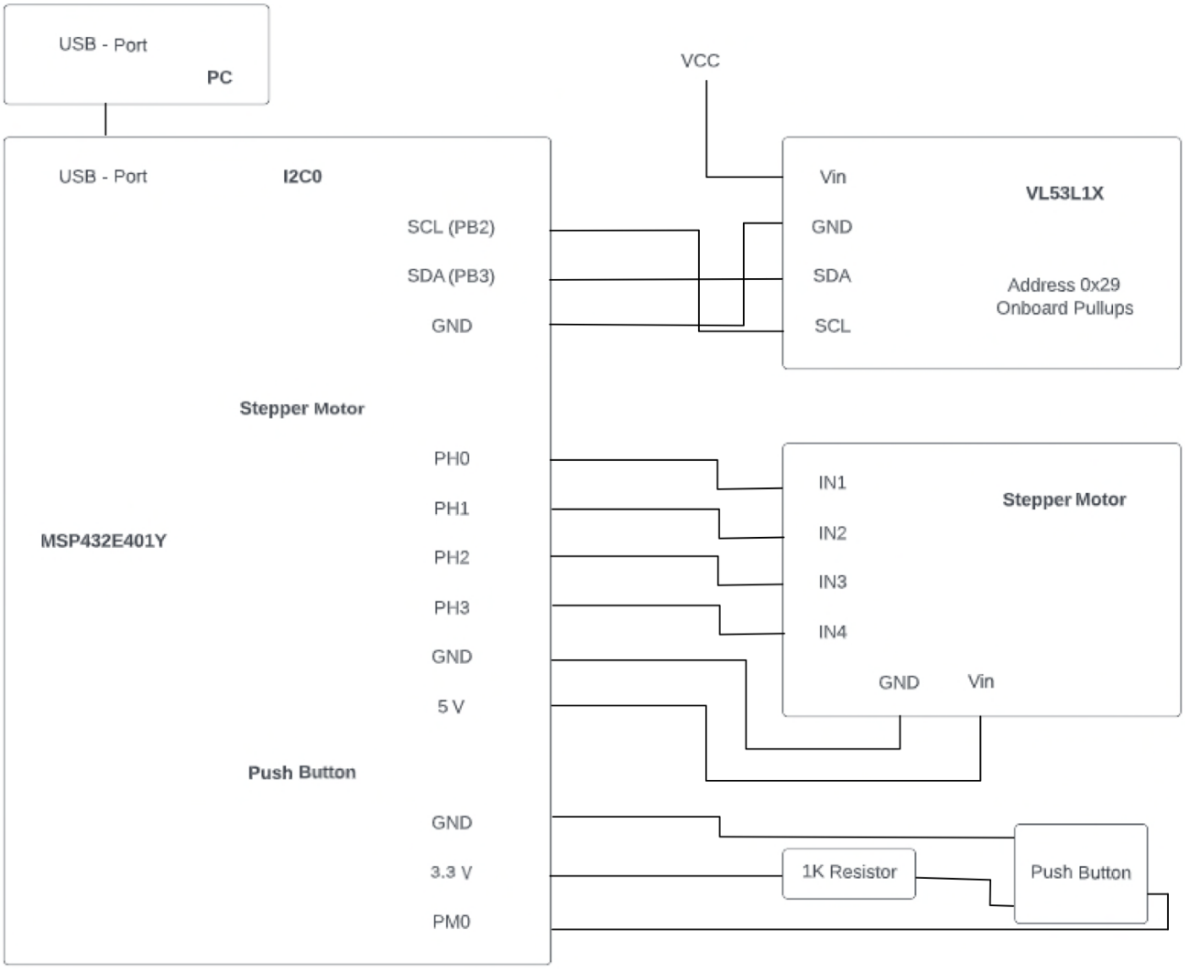

Figure 6: Circuit Schematic of Device

Programming Logic Flowcharts

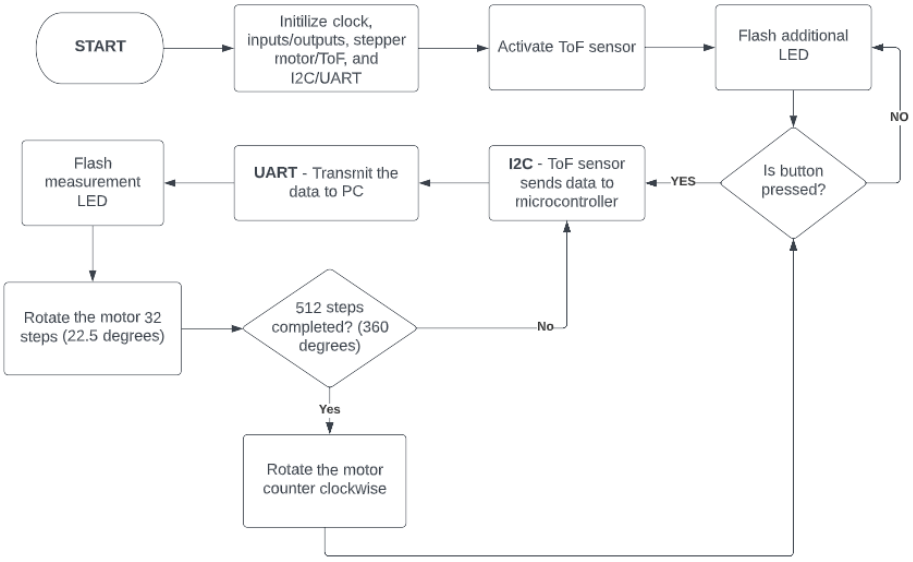

Figure 7: Flowchart of C program in Keil IDE

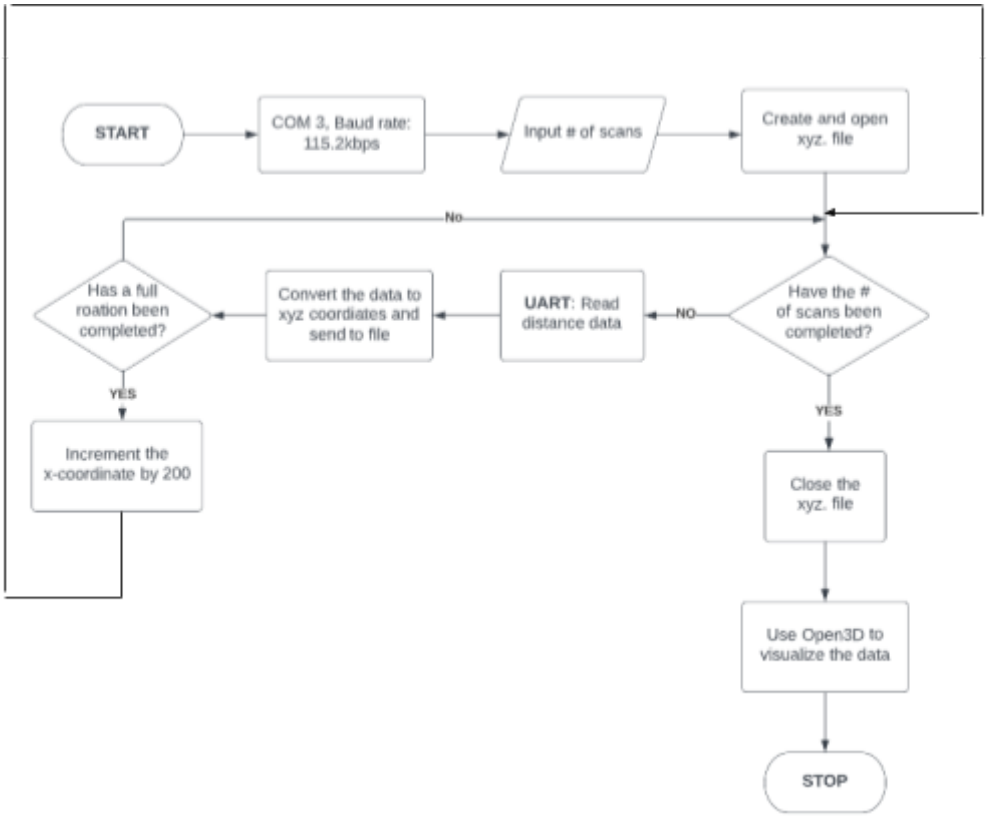

Figure 7: Flowchart of Python Code