# Microprocessor Systems Project: COMPENG 2DX3

# Final Project Report

April 10th, 2024

Hunter Boyd

boydh4

400384654

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 our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

# 1. Device Overview

## 1.1 Features

- Embedded Spatial Measurement System
- MSP432E401Y SimpleLink Microcontroller
    - Operating Voltage: 2.5-5 V
    - Configurable Bus Speed up to 120 MHz (Default 24 MHz)
    - Two 12-bit SAR-Based ADC Modules, Supporting Up to 2 Msps
    - Eight Universal Asynchronous Receivers/Transmitters (UARTs)
    - Ten $I^2C$ Modules
    - 1024KB of flash memory
    - 256 of SRAM
    - Programmed using C or Assembly Language via Keil IDE
    - ~$70
- VL53L1X Time of Flight Sensor:
    - Low-power Microcontroller
    - Operating Voltage: 2.6-3.5 V
    - Size: 4.9 x 2.5 x 1.56 mm
    - Infrared Emitter: 940 nm
    - Receiver FOV: 27º
    - Maximum Distance: 400 cm
    - I2C interface
    - ~ 10$
- 28BYJ-48 Stepper Motor
    - Rated for 5 V DC
    - 2048 steps per revolution
    - ~ $5
- Two Active High Push Buttons Using GPIO Ports PL0 and PL1 of the uC
    - PL0 is used to initialize distance scanning and a full clockwise revolution of the stepper motor
    - PL1 is used once a full scan is finished to rotate the stepper motor counter clockwise to return the ToF to home
- Device scans 3D spaces and displays the spatial layout graphically via Open3D
- Operates at a default 24 MHz bus speed (adjustable up to 120 MHz)
- 12-bit resolution
- Uses UART with a 115200 baud rate for serial communication between uC and PC
- Uses I2C with MSP432E401Y SimpleLink Microcontroller as leader and VL53L1X as follower for communication between uC and VL53L1X

## 1.2 General Description

The Embedded Spatial Measurement System, built around the MSP432E401Y SimpleLink Microcontroller, is a compact and economical solution for precise indoor spatial mapping. Operating at a bus speed of 24 MHz, it can be adjusted up to 120 MHz to accommodate various data processing needs. The system features a VL53L1X Time-of-Flight sensor for distance measurement, mounted on a stepper motor for 360-degree environmental scanning.

Controlled by the provided C and Python code, the system achieves a seamless integration of hardware operations and data visualization. The C code, developed in the Keil IDE, is tailored for real-time sensor data acquisition and motor control, while the Python script, leveraging libraries such as Open3D, facilitates the rendering of spatial measurements into a 3D model on a PC interface.

User interaction is made intuitive through physical push buttons, dictating the start and stop of data collection and motor movement. Onboard LEDs provide visual feedback correlating to measurement progress, enhancing the user experience.

This system is not only an affordable alternative to larger-scale LIDAR systems but also a versatile educational platform, supporting a range of applications from autonomous navigation to architectural mapping. Its design allows for easy customization, making it adaptable to specific project requirements and user-defined coding environments.

## 1.3 Block Diagram

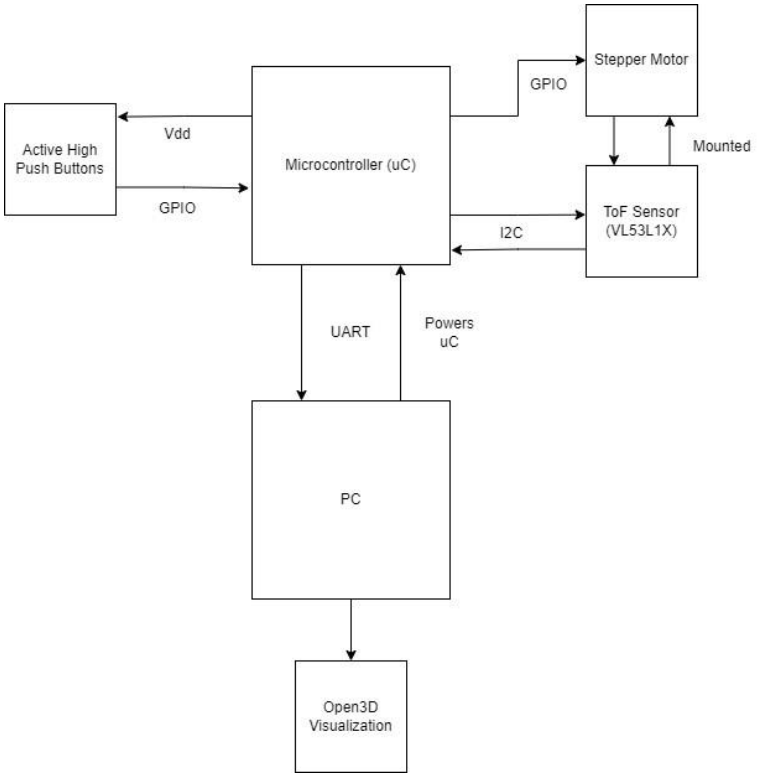


*Figure 1: Block Diagram*

## 2. Device Characteristics

| Feature | Specification |
|---|---|
| **Microcontroller** | MSP432E401Y SimpleLink |
| Operating Voltage | 2.5V – 5V |
| Bus Speed | Default 24 MHz, configurable up to 120 MHz |
| Memory | 1024KB flash, 256KB SRAM |
| ADC Modules | Two 12-bit SAR-based, up to 2 MSPs |
| UART Communication | Baud rate of 115200 for PC interfacing |
| GPIO Input Ports | PL0, PL1 (active high push buttons) |
| GPIO Output Ports | PH0-PH3 , PN0, PN1, PF0, PF4 |
| Serial Port | COM5 (Confirgurable) |
| $I^2C$ Modules | Ten modules for sensor and peripheral comms |
| **Time of Flight Sensor** | VL53L1X |
| Operating Voltage | 2.6V – 3.5V |
| SCL | PB2 on uC |
| SDA | PB3 on uC |
| Maximum Distance | 400 cm |
| **Stepper Motor** | 28BYJ-48 |
| Operating Voltage | 5V |
| Steps/Revolution | 2048 |
| IN1, IN2, IN3, IN4 | PH0, PH1, PH2, PH3 on uC |
| Visualization | Open3D interface for 3D spatial representation |
| Programming Languages | C for microcontroller, Python for PC interface |
| Development Tools | Keil IDE for firmware, Python environment for PC |
| User Interface | Push buttons for control, LEDs for status. PL0 push button initializes stepper motor as well as sensor for spatial measurement. PL1 returns stepper motor to home position. |
| Output Format | 3D spatial data visualized graphically |

*Table 1: Device Characteristics*

## 3. Detailed Description

### 3.1 Distance Measurement

The ToF sensor measures distances using Light Detection and Ranging (LIDAR). It does so by emitting a 940 nm pulse of light. This light then hits an object and returns to the sensor, and the VL53L1X measures the time it takes for the emitted pulse of light to return. The distance can then be calculated based on the known speed of light and the time to detect the light reflection. Per measurement, there are multiple bursts of light and the average time of return is taken for the final distance calculation. This distance is sent to the SDA via $I^2C$ and received by the microcontroller.

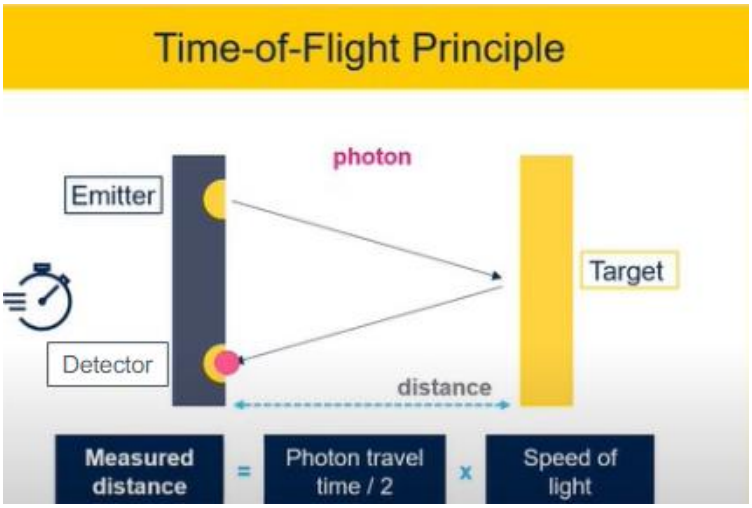


*Figure 2: Distance Calculation [1]*

The microcontroller and the ToF sensor have a leader and follower relationship. There are 4 connections between them. $V_{IN}$ on the VL53L1X is connected to 3.3 V on the uC, GND is connected to GND, SDA is connected to PB3, and SCL is connected to PB2. The microcontroller communicates with the ToF sensor via the VL53L1X Application Programming Interface (API). The sensor is booted using the BootState function, then initialized with the SensorInit function, and measurements are taken with the StartRanging function. This measurement data is grabbed using the GetDistance function and then sent from the microcontroller to PC using UART serial communication.

```
221     status = VL53L1X_GetDistance(dev, &Distance);          //The Measured Distance value
222
223     status = VL53L1X_ClearInterrupt(dev); /* 8 clear interrupt has to be called to enable next interrupt*/
224
225     sprintf(printf_buffer,"%u\r\n", Distance);
226     UART_printf(printf_buffer); //Distance measurement sent to PC via UART
```

*Figure 3: UART Distance Measurement Transmission*

The ToF sensor is initialized in the microcontroller C code. Once the microcontroller and PC programs are synchronized, a button push at PL0 starts the measurement process. To begin, the stepper motor mounted with the ToF sensor will rotate, and an on board LED will turn on to

signal the measurement process is happening. For every 360 degree rotation, the VL53L1X will take 64 uniform measurements. In other words, this embedded system takes a measurement every 5.625 degrees. Each time the motor reaches an increment of 5.625 degrees, a delay of 250 ms takes place to ensure a stable measurement, and an LED is flashed to confirm a measurement has been taken. Then, the measurement is stored in a Distance variable which is transmitted to the PC using UART. When the PC grabs the data with it's Python code, it is checked to make sure it is a valid measurement and represents a number. If it is a valid measurement then it is translated to a xyz coordinate and written to a text file. If the button at PL0 is pressed during the measuring process, the stepper motor will pause and no measurements will be taken. Pressing it again will resume the measurement process. Once the stepper motor has reached a 360 degree rotation, measurements are no longer taken and the stepper motor must be returned to it's home position by pressing the button at PL1. Once it is returned to home, displace the system in your preferred x distance and repeat the measurement process by pressing PL0 again.

## 3.2 Visualization

The Python code is run on the PC and is responsible for the visualization of data. The PySerial library is used for receiving/transmitting data packages serially via UART in a Python environment. In the Python program, the appropriate port is open for serial communication (default COM5). The program asks the user for the uniform x displacement value that the system will be moved by for each measurement. It also asks how many displacements will be taken place. Then, it asks for an input to start the program, which the uC distance measurement program will be waiting for. Once this input is entered, the programs are synched up and distance measurements are ready to be transmitted from the uC to the PC.

The previously user chosen value for the number of displacements taking place are used for a loop condition which runs the specified number of times. For each iteration of this loop, 64 valid measurements are received from the microcontroller. Each of these measurements are converted into integer xyz coordinates using the following calculations:

$$x = displacement * x\_displacement$$

$$y = distance * \sin(5.625 * valid\_measurements)$$

$$z = distance * \cos(5.625 * valid\_measurements)$$

where displacement is the current iteration of the loop ranging from 0 – num_displacements, and x_displacement is the uniform x displacement value the system is moved after each measurement. This works as an accurate coordination system because the angle is incremented by 5.625 degrees for each measurement in the calculations, just as it is on the actual stepper motor. Also, x increments by a set displacement each time as it should be when taking measurements with this system.

After every measurement is taken, they are written to a .xyz file. These coordinates then serve as a scaffold for constructing a virtual model, where the Open3D toolkit renders the points into a navigable and detailed three-dimensional representation. This visualization not only embodies the actual scanned environment but also enables users to interact with the data, facilitating applications that require an immersive spatial analysis.

# 4. Application

## 4.1 User Guide

The following steps are for you to be able to use this embedded system for your application.

1. Connect the MSP432E401Y microcontroller, VL531X ToF sensor, stepper motor, and two push buttons as required by this design in the circuit schematic.

2. Mount the stepper motor to the provided setup, and place the ToF sensor in the provided stepper motor attachment.

3. Connect the USB to your PC and your microcontroller.

4. Install Python 3.6 to 3.10 (No later or Open3D won't work)

5. Install PySerial.

6. Install Open3D.

7. Open your preferred IDE to run Python (ex. Visual Studio).

    i. Open "2dx_project.py"

    ii. Change "COM5" to the correct port found in device manager.

8. Install Keil MDK uVision 5.

9. Open Keil.

    i. Open "2dx_studio_8c.uvprojx".

    ii. Click on Translate, Build, then Load the program onto your microcontroller.

10. Reset the microcontroller.

11. Run "2dx_project.py".

    i. Setup the system in your preferred scanning environment.

    ii. To synchronize the microcontroller Keil program and the Python, press enter when prompted in the terminal.

    iii. Enter the distance you will move the system in between each 360 degree scan when prompted.

    iv. Enter the amount of 360 degree scans you will do when prompted.

12. Press button 0 to begin scanning.

    i. The motor will rotate clockwise 360 degrees and take a measurement every 5.625 degrees.

    ii. Once it has reached 360 degrees, it will stop taking measurements.

13. Press button 1 to untangle the wires once the motor has scanned 360 degrees.
    i. The motor will rotate counter clockwise 360 degrees back to home.
14. Repeat steps 12-13 until you have performed the amount of scans specified earlier.
15. Once the process is complete, the area's 3D model will appear in a new Window.

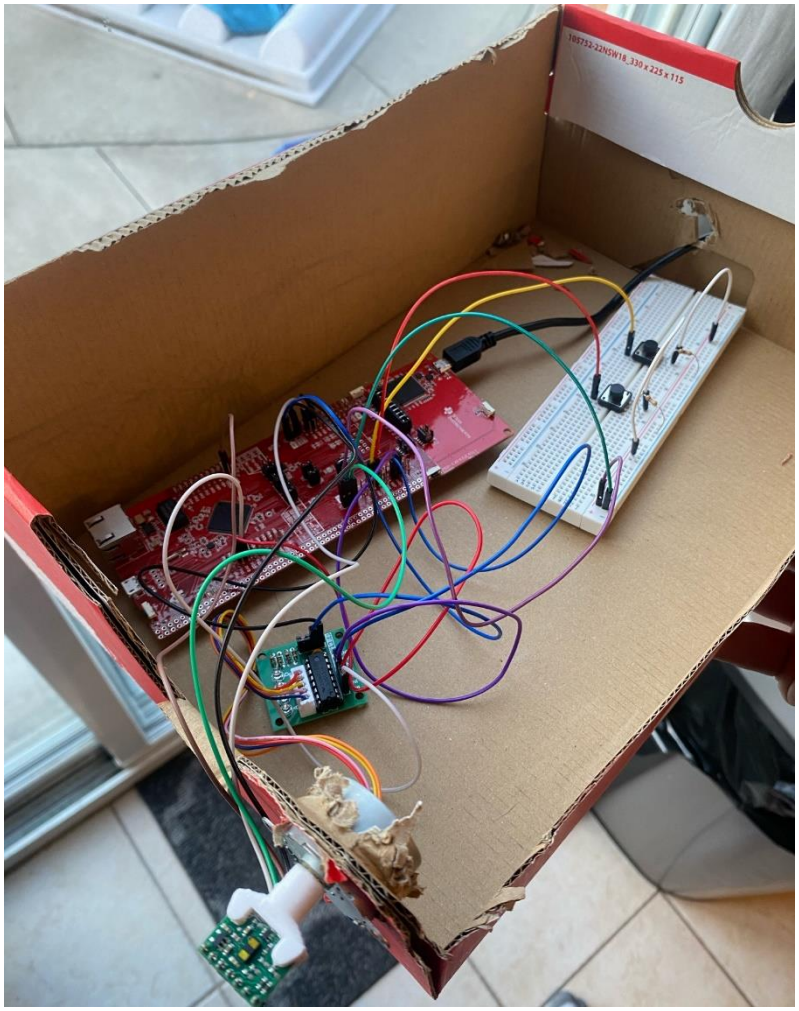

*Figure 4: Design Prototype*

## 4.2 Expected Output


*Figure 5: Location E, Assigned Campus Scanning Location*

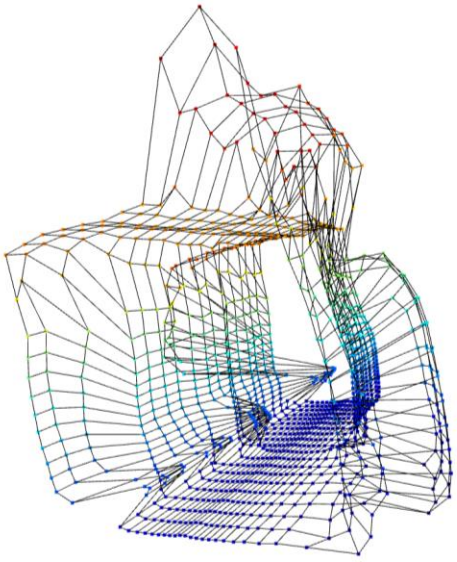

*Figure 6: Location E Open 3D Visualization*

## 5. Limitations

The computational precision of the device is restricted to single precision floating-point operations, which may introduce minor inaccuracies, particularly as the Python-based processing environment employs double precision calculations. During the conversion of measurement data to trigonometric calculations necessary for spatial mapping, the conversion from degrees to radians potentially contribute to slight computational discrepancies.

Serial communication output is another critical aspect, where the device-to-computer rate is capped at a 115200 baudrate, as this is the maximum baudrate parameter acceptable for the PySerial module.

Quantization error, an inherent characteristic of digital systems, manifests in our device with a maximum value of 0.000854. This figure is derived from the full-scale voltage ($V_{FS}$) of 3.5 and a bit length (m) of 12

The speed of the stepper motor is another limiting factor. The minimum delay between steps is 2 ms, which the speed of the motor depends on. Also, there is a trade-off between the amount of measurements taken and motor speed. The more measurements being taken per revolution, the more delay there is in each revolution which results in a slower turn of the motor.

Lastly, the operational range of the ToF sensor, spanning from 40mm to 4000mm, delineates the spatial extent of scannable features. This range sets the boundaries for the measurable environment, limiting the system's application to scenarios that fall within this spatial threshold.

## 6. Circuit Schematic

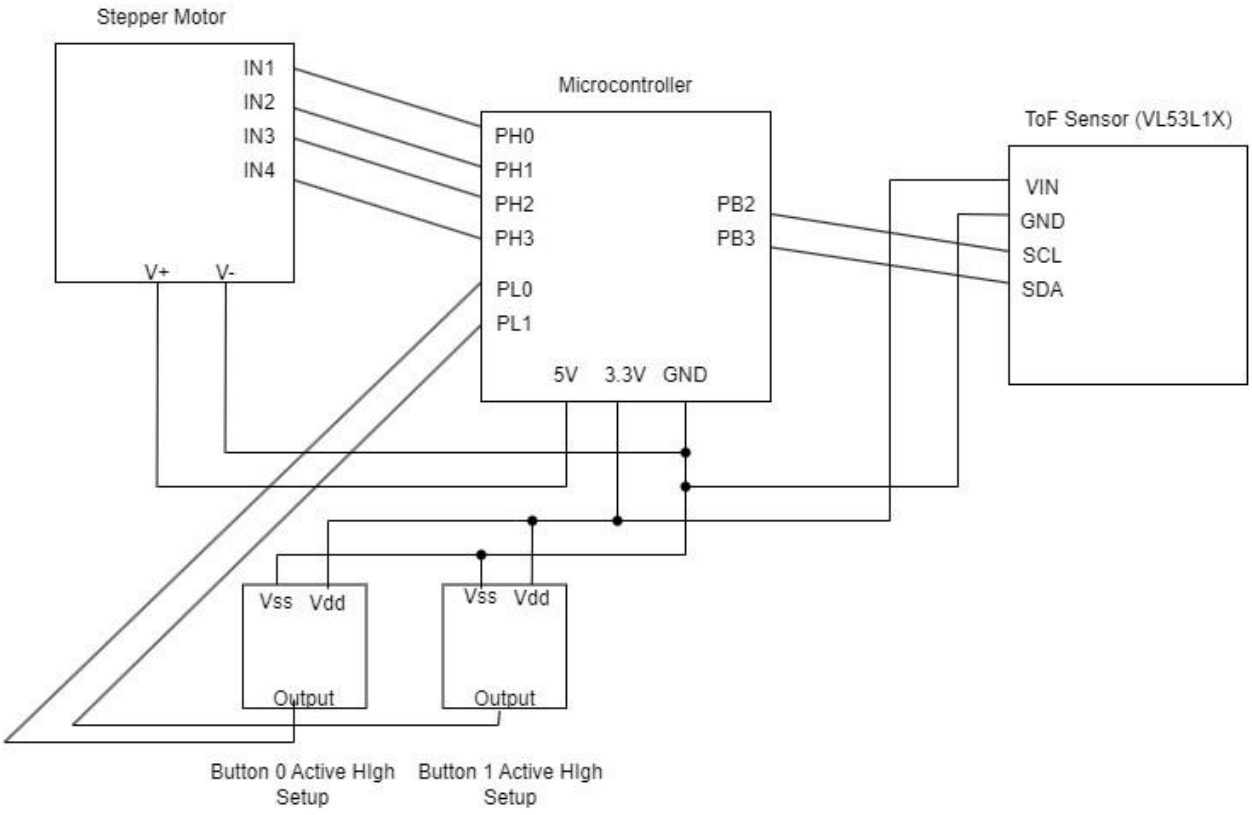


*Figure 7: Design Circuit Schematic*

## 7. Programming Logic Flowcharts

### 7.1 Microcontroller C Distance Measurement Code

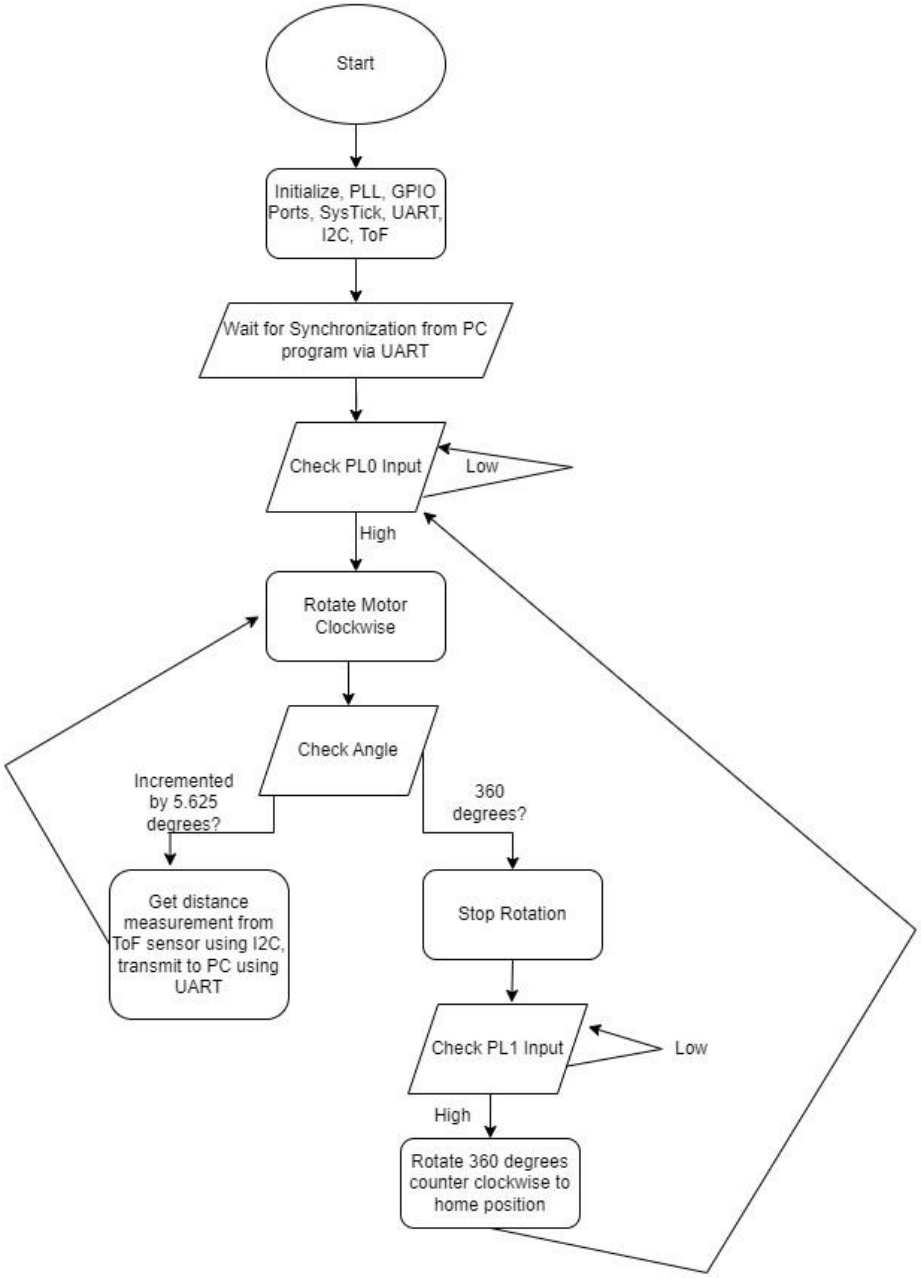


*Figure 8: Microcontroller Flowchart*

## 7.2 PC Python Visualization Code

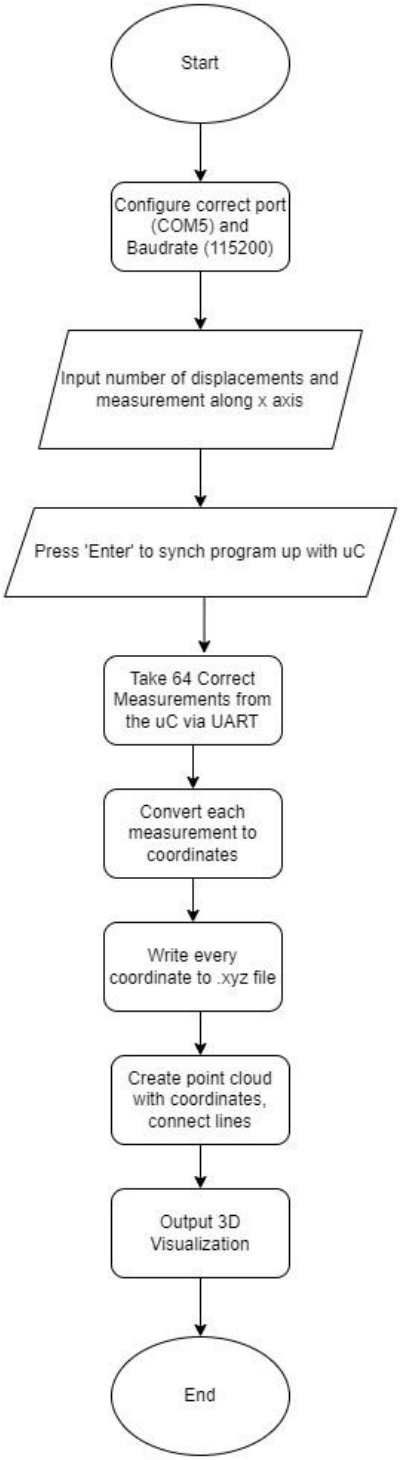


*Figure 9: PC Flowchart*

## 8. References

[1] “VL53L1X Datasheet,” STMicroelectronics, https://www.st.com/resource/en/datasheet/vl53l1x.pdf (accessed Apr. 14, 2024).

[2] “MSP432E4 SimpleLink Microcontrollers - Technical Reference Manual,” Avenue to Learn - McMaster University, https://avenue.cllmcmaster.ca/d2l/le/content/557632/viewContent/4481118/View (accessed Apr. 14, 2024).

[3] T. Doyle, Y. Haddara, and S. Athar, “Studio 8 - Week 8 - ToF Interface and Application,” Avenue to Learn - McMaster University, https://avenue.cllmcmaster.ca/d2l/le/content/557632/viewContent/4481175/View (accessed Apr. 14, 2024).