Microprocessor Systems 2DX3
Final Project Report

James Cameron – camerj22 - 400450866

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 **[James Cameron, Camerj22, 400450866]**

Due April 9, 2025

Table of Contents

Table of Figures

Device Overview

Features

- Embedded Spatial Measurement System comprised of the following components:
- MSP432E401Y SimpleLink Microcontroller
  - Core: 120-MHz Arm Cortex -M4F Processor Core with Floating-Point Unit (FPU)
  - Bus speed up to 120MHz, defaulted to 14MHz
  - 1024KB of Flash Memory
  - 256KB of SRAM
  - Operating Voltage: 3.5-5V
  - Approximately $75
- 28BYJ-48 Stepper Motor and ULN2003 Driver Board
  - 5V unipolar stepper motor with 4 phases
  - Internal 64:1 gear reduction, resulting in 2048 steps per full rotation
  - Provides 34.3 mN*m torque at 15RPM
  - Requires external 5V power supply
  - Cost: $6
- VL53L1X Time of Flight Sensor (ToF Sensor)
  - Fully integrated miniature sensor module (4.9x2.5x1.56mm)
  - 940 nm invisible laser emitter
  - SPAD (single photon avalanche diode) receiving array with integrated lens
  - Up to 400 cm distance measurement, and up to 50Hz ranging frequency
  - I2C interface up to 400 kHz
  - Cost: approximately $20
- On Board Control Button
  - MSP432E401Y Microcontroller button used for general-purpose input
  - Active low configuration
- UART Communication used to communicate between PC and Microcontroller
  - Python Language used to receive data on PC
- I2C used to communicate data between VL53L1X sensor and Microcontroller
  - Baud rate: 115200

General Description

The Spatial Measurement System provides an economical and efficient solution for precise scanning of indoor locations. Leveraging a combination of distance sensing, stepper motor control, and real-time data processing, the system enables users to perform automated scans and generate spatial models of indoor rooms or hallways. The system is powered by the MSP432E401Y SimpleLink Microcontroller, which connects the various components of the system and is responsible for performing all the logic required to allow the scanning to occur [1].

The scanning mechanism is facilitated by a VL53L1X Time-of-Flight (ToF) distance sensor mounted on a 28BYJ-48 stepper motor**,** allowing 360-degree spatial data acquisition with fine angular resolution. User control is managed through a physical push button (PJ1) that initiates and automatically terminates scanning sessions. Once terminated, the Open 3D software constructs the 3D plot of the scanned data, which is transmitted from the microcontroller to the PC via UART. When the sensor scans the distance data, it is transmitted to the microcontroller via I2C. Measurement and system states are indicated via on-board LEDs**,** assigned to display status such as active measurement, data transmission, and rotation reset.

The system is implemented using the Keil IDE, where C code was used to handle the polling logic for the scanning arithmetic. Once the data is sent over UART, a Python script on the PC receives the raw distance and converts it into Cartesian coordinates for plotting. The system can provide an affordable and accurate graphical representation of the scanned environment, making it an optimal choice for educational and learning purposes.

Block Diagram

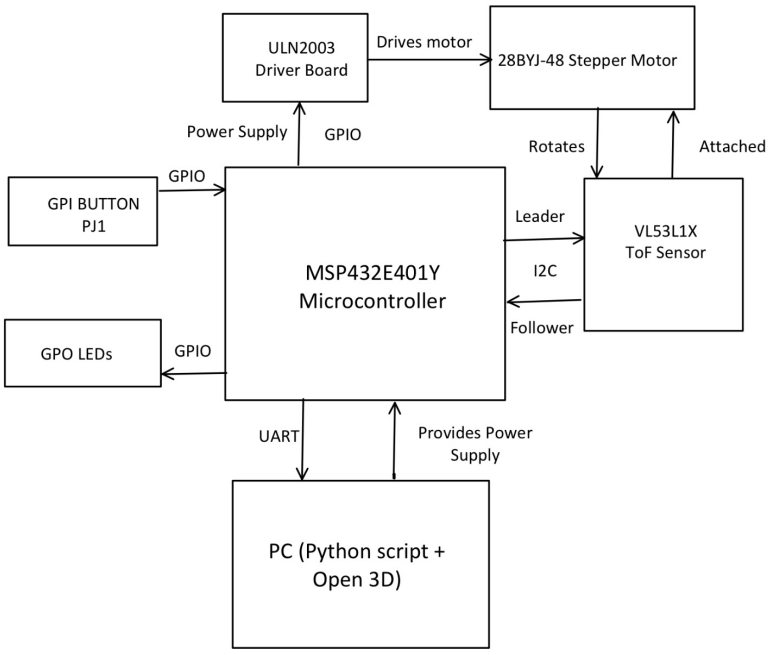

Figure 1: System diagram

Device Characteristics

Component	Specification
Microcontroller	MSP432E401Y
Bus Speed	14 MHz, configurable up to 120 MHz
Operating Voltage	2.5 – 5V
UART baud rate	115200 Bd
I2C	Sensor addressed via 0x29
GPI Pins	PJ1, PG0
GPO Pins	PN0, PF0, PF4, PM1-4,
Alternate Function Pins	PB2, PB3
Memory	1024KB of flash memory
Analog	Two 12-bit SAR-based ADC Modules
Start/Stop button	PJ1, on board button
Measurement Status LED	On board LED D3, PF4
Data byte transmission LED	On board LED D4, PF0
Counterclockwise reset LED	On board LED D2, PN0
Stepper Motor	28BYJ-48
Control Pins	PM1, PM2, PM3, PM4
Voltage	5V
Rotation steps	2048
Rotation Angle	11.25*
Time of Flight Sensor	VL53L1X
size	4.9x2.5x1.56 mm
voltage	2.6-3.5V
Serial Data Pin	PB3
Serial Clock Pin	PB2
Measurement Mode	2 (long), maximum distance 400 cm
Ranging Frequency	50 Hz
Tools	Keil for firmware, VS Code for Python code
Programming Languages	C for Micro. Code, Python for PC code
Graphical Plotting	Open 3D

Detailed Description

Distance Measurement

The VL53L1X is a time-of-flight (ToF) laser-ranging sensor capable of measuring distances up to 400cm. The sensor has three modes, short, medium, and long. Each mode extends the range the sensor can detect. The sensor is set in long mode to allow for all walls in the scanned location to be detected.

The VL53L1X utilizes time-of-flight technology, which involves emitting a burst of infrared light and measuring the time it takes for the reflected signal to return from the target surface. The equation used by the sensor to determine the distance of the detected wall is found below.

$$Measured\ Distance = \frac{Photon\ Travel\ Time}{2} \times Speed\ of\ Light$$

The formula can be explained as follows. When the sensor sends a laser using, the invisible emitter, toward an object, it will bounce off the object and be retrieved by the receiving array. Since the laser must be sent and received, the time it takes to return to the sensor is divided by two. We multiply this by the speed of light in order to get the distance measurement using the following unit cancelling.

The system rotates 11.25 degrees, the sensor acquires the distance, and this process is repeated 31 times to complete a full 360-degree rotation. Once this data is collected, the distance data is sent to the microcontroller via I2C.

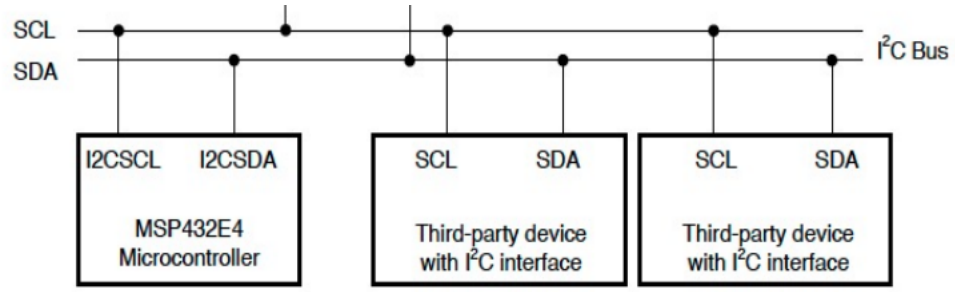

Figure 2: I2C system

The I2C communication protocol is synchronous, meaning it requires a clock to be shared between the leader and follower. In this case, the leader is the microcontroller and the following being the VL53L1X sensor. There are two lines, the first being the SCL (Serial Clock), which is required to ensure data is transmitted at a shared rate between the leader and follower. The other line is the SDA (Serial DATA), used to transfer data between the two devices on the I2C Bus [2].

For the communication to work, the sensor must be initialized correctly and the specifications must be assigned to ensure correct configuration.

```
// Reset and initialize ToF sensor
VL53L1X_XSHUT();
status = VL53L1X_SensorInit(dev);

if (status == 0) {
    status = VL53L1X_SetDistanceMode(dev, 2); //2 = long 4m max
    status = VL53L1X_SetTimingBudgetInMs(dev, 100);  //time per reading 100ms
    status = VL53L1X_SetInterMeasurementInMs(dev, 100); //time bw measurement
    status = VL53L1X_StartRanging(dev); //making a measurement
```

Figure 3: Sensor initialization code

Four parameters must be set; these include the mode, which is set to 2 for a long distance (maximum of 400cm). Next, the time per reading is set to 100ms. The time between measurements is also set to 100ms. Finally, we acknowledge that we are going to be making a measurement. The VL53L1X API is used to allow the micro. to communicate with the sensor.

```
    status = VL53L1X_GetDistance(dev, &Distance); //returns distance in mm
    status = VL53L1X_ClearInterrupt(dev); //clears for next event

    FlashLED4(3);
    sprintf(printf_buffer, "%u\r\n", Distance);
    UART_printf(printf_buffer); //sending data
```

Figure 4: Distance send code

Once correctly configured, the VL53L1X sensor initiates a distance measurement using the GetDistance function, which stores the result in a variable representing the distance in millimeters. This value is then transmitted to a connected PC via UART (Universal Asynchronous Receiver-Transmitter), enabling real-time scanning and data collection. The process begins upon button press of PJ1 from the user and operates in a loop as the sensor module is rotated in discrete 11.25° increments, resulting in 32 total measurements for a complete 360° scan. At each increment, a new distance reading is captured and sent. Once a full rotation is completed, the user can physically advance the system forward to scan the next section of space, building a stepwise scan of the environment. Once satisfied, the user may press button PJ1 again to stop the scanning procedure, and the 3D render of the room will be generated.

Visualization

The Python script running on the PC is responsible for receiving distance data from the microcontroller, converting it to 3D coordinates, and visualizing the result. For every 11.25° increment, the microcontroller sends a distance value via UART. Once received, the script checks if the message is a valid distance measurement by verifying that it can be parsed as a floating-point number. If the data is valid, it is passed into the convert_polar_to_xyz() function, which converts the polar coordinates (distance and angle) to a 3D coordinate, where x corresponds to the forward step, and y and z form the circular scan.

```python
def convert_polar_to_xyz(dist, angle_deg, offset_x):
    x = offset_x
    y = dist * math.cos(math.radians(angle_deg))
    z = dist * math.sin(math.radians(angle_deg))
    return x, y, z
```

Figure 5: Polar to cartesian conversion function

This process repeats for all 32 measurements in a full 360° scan. After 32 valid points have been collected, the scan index is reset to zero and the "x" position is incremented by 500 mm. This value corresponds to a physical step forward of 50 cm, representing an average stride. Over multiple steps, this results in the accumulation of 3D points, forming a slice-by-slice reconstruction of the environment.

Once all data is collected, a 3D point cloud is generated using Open3D. The generate_point_cloud() function takes the list of coordinates and forms a cloud that can be displayed. Additionally, the build_wireframe_geometry() function connects points across layers to generate a structured wireframe representation, enhancing the spatial model. Finally, the two geometries, the point cloud and the wireframe, are rendered in the same scene using display_scene(), allowing the user to view a full 3D representation of the scanned area.

Application

User Guide

Below is a step by step guide to setting up the complete Spatial Measurment System:

1. Open Keil software, and open the provided Keil project titled “2dx_studio_8c”.
2. Open up your IDE of choice (ex. VS Code) and open the Python file titled “project.py”.
3. Change the COM4 port to the connected port for the micro. shown in the device manager.
4. Install & setup Python 3.6-3.10 (your choice).
5. Install PySerial and Open 3D.
6. Connect the MSP432E401Y microcontroller to the PC.
7. Using wires, connect the following pins to components.
    1. On the ULN2003 Driver Board, connect the + pin to any 5V, and the – pin to any GND on the microcontroller. Connect IN1 pin to PM0, IN2 pin to PM1, IN3 pin to PM2, and IN4 pin to PM3.
    2. Ensure the 28BYJ-48 Stepper Motor is connected to the driver.
    3. On the VL531X ToF sensor, connect the XSHUT pin to PG0. Connect the VIN pin to any 3.3V on the micro. Connect the GND pin to any GND pin on the micro. Connect the SCL pin to PB2, and connect the SDA pin to PB3.
8. Slide the larger 3D printed piece onto the end of the microcontroller board. Place the motor inside of this piece.
9. Mount the smaller 3D printed piece onto the end of the motor. Insert the sensor. Ensure the wires are not tangled around the motor.
10. Return to Keil, and translate, build, and load the C code onto the microcontroller.
11. Press the reset button on the microcontroller (next to the usb port), 3 LEDs should flash if done correctly.
12. Return to the Python file, and run it. The following message should pop up if all done correctly, “Press enter to begin scan…”. Press enter.
13. Press button PJ1 on the microcontroller to begin scanning.
    1. The sensor will now rotate 32 times, by 11.25 degrees. Once completed, it will turning 360 degrees counter clockwise automatically.

2. X, Y, Z values should be shown within the terminal. If not, backtrack and repeat all steps. X represents displacement, with Y and Z representing the vertical distance slices.
3. The system will continue to scan indefinitely, once the motor begins rotating counterclockwise, take a 50cm step forward and wait until next rotation is complete.
4. Repeat step 4 until you are satisfied with the number of scans performed.

14. When you are satisfied with your number of scans, press and hold the PJ1 button to stop scanning.
15. Open 3D will open, and the 3D render of the room/hallway will be displayed. If it is not, you most likely did not download Open3D correctly, or the sensor is not connected correctly.

If set up correctly, the device should look the same as the device below:

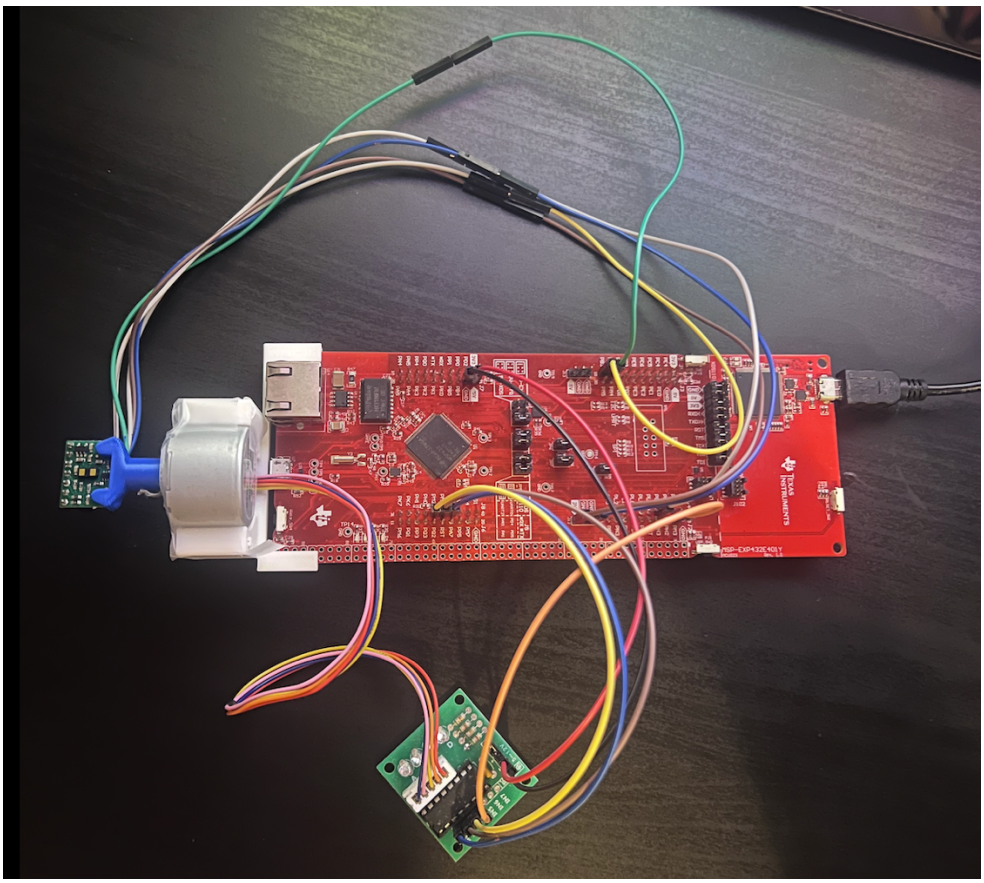

Figure 6: Physical Circuit

Expected Output

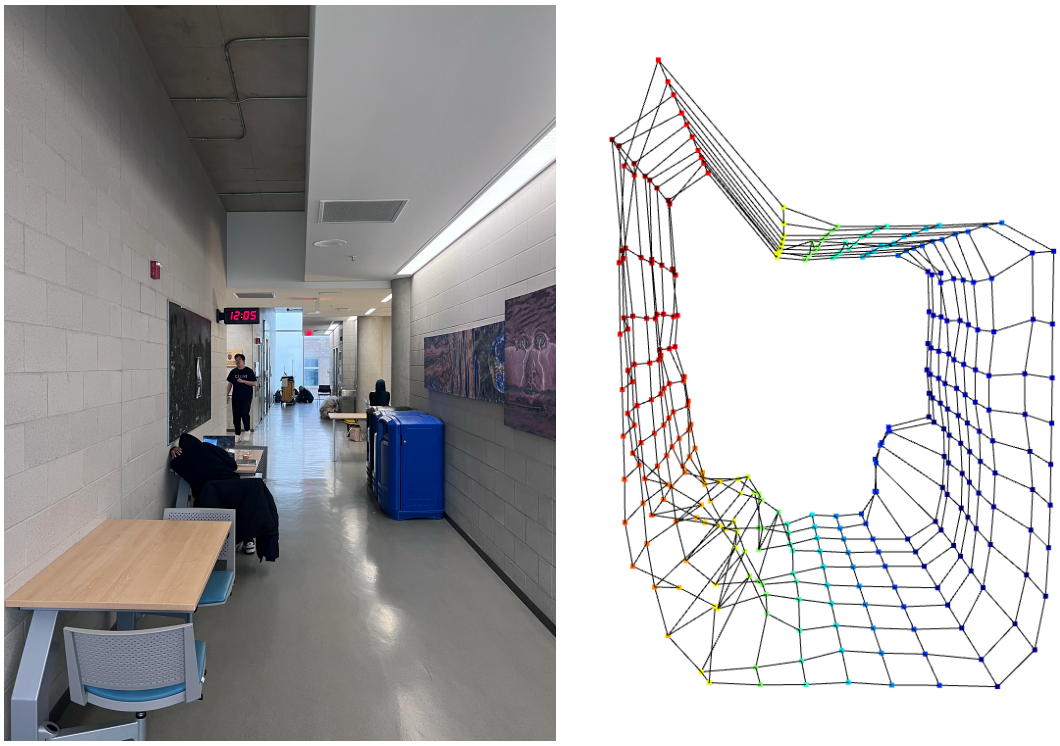

Figure 7: Real location vs. Scanned plot

Limitations

The microcontroller's floating-point unit (FPU) helps with performance, but it still has precision limits that affect calculations, especially when using trigonometric functions like sine and cosine. These functions are needed to convert polar coordinates to Cartesian, but they introduce small rounding errors. When combined with the limitations of floating-point math, these errors can build up over time, slightly distorting distance calculations and reducing accuracy in the final 3D point cloud.

The VL53L1X Time-of-Flight (ToF) sensor used in this project has a resolution of 1 mm, which means the smallest measurable change in distance is 1 mm. Therefore, the maximum quantization error for the module is +-1 mm, since the sensor rounds measurements to the nearest millimeter and cannot detect changes smaller than that.

Serial communication is used between the PC and microcontroller to communicate distance data. A baud rate of 115200 is used. This is the maximum baud rate that can be used in this application due to the software being used such as PySerial. As a result, there is a cap on how fast data can be sent, which can slightly bottleneck real-time performance, especially if more data points or higher-resolution scans are added in the future.

The communication method used between the microcontroller and the sensor is I2C. I2C requires a shared clock amongst devices, therefor the clock speed was set to 100 kHz. This means data is transferred at a rate of 100,000 bits per second. This is fast however it is still a limiting factor as this would not support data coming in from the sensor at speeds higher than 100,000 bps.

The VL53L1X has a maximum distance range of 4 meters in long-distance mode and can be affected by surface reflectivity, ambient light, and angle of incidence. Glossy, transparent, or black objects may result in inaccurate readings or inaccuracies. Furthermore, the scanning rate (with a period of 100ms) limits the scanning speed and makes the process relatively slow. An additional delay is required after each rotation to ensure the sensor settles before reading. This compounds the scanning time, particularly when scanning larger environments. As a result, users may need to wait extended periods to complete full scans, making it inefficient for dynamic or time-sensitive applications. In order to minimize this time loss, the delay factor of the stepper motor was minimized to the point where it would prevent any rotation, and determined to be 10ms.

Circuit Schematic

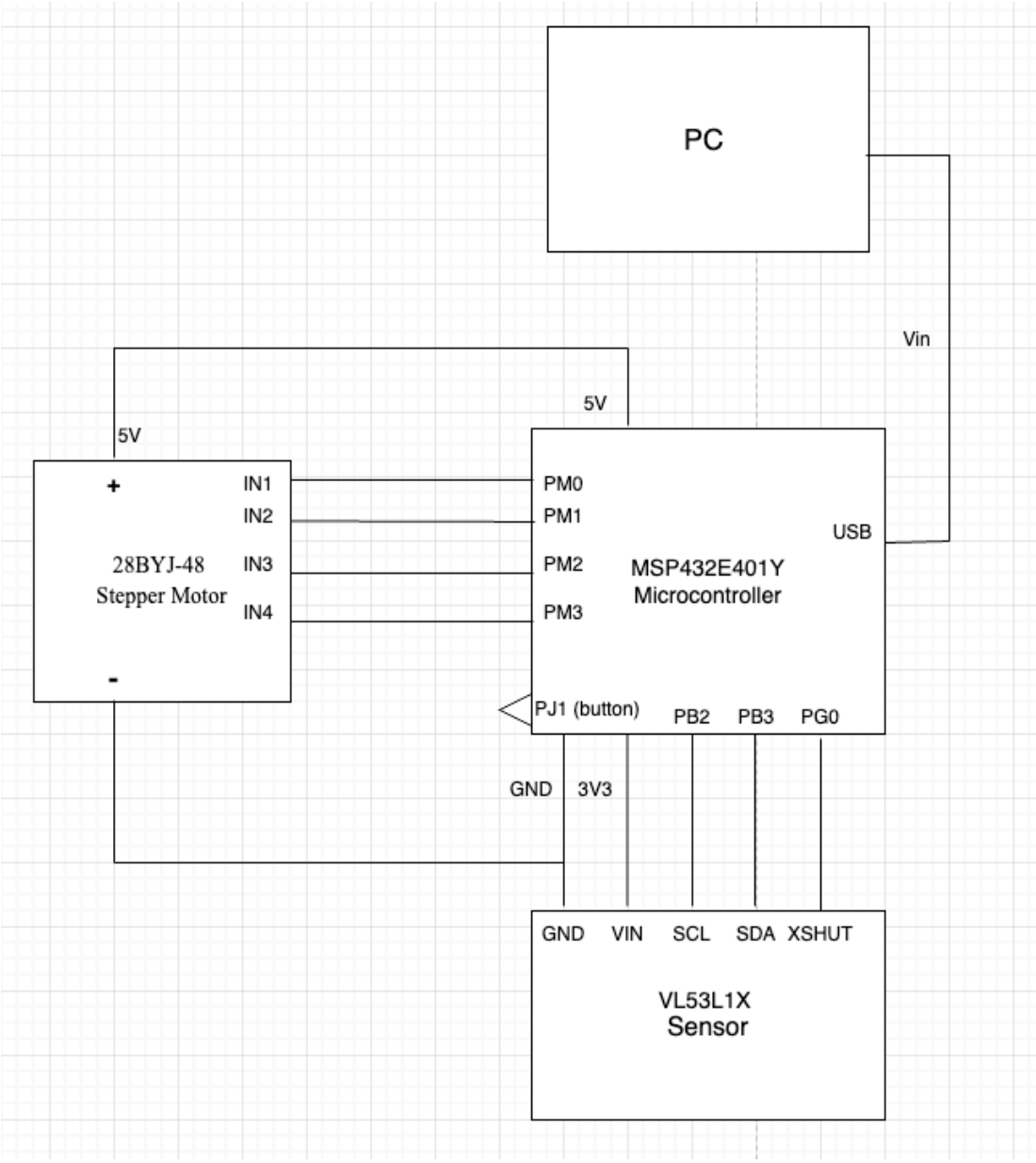

Figure 8: Circuit Schematic & Pin Mapping

Programming Logic Flowcharts

Microcontroller logic flowchart

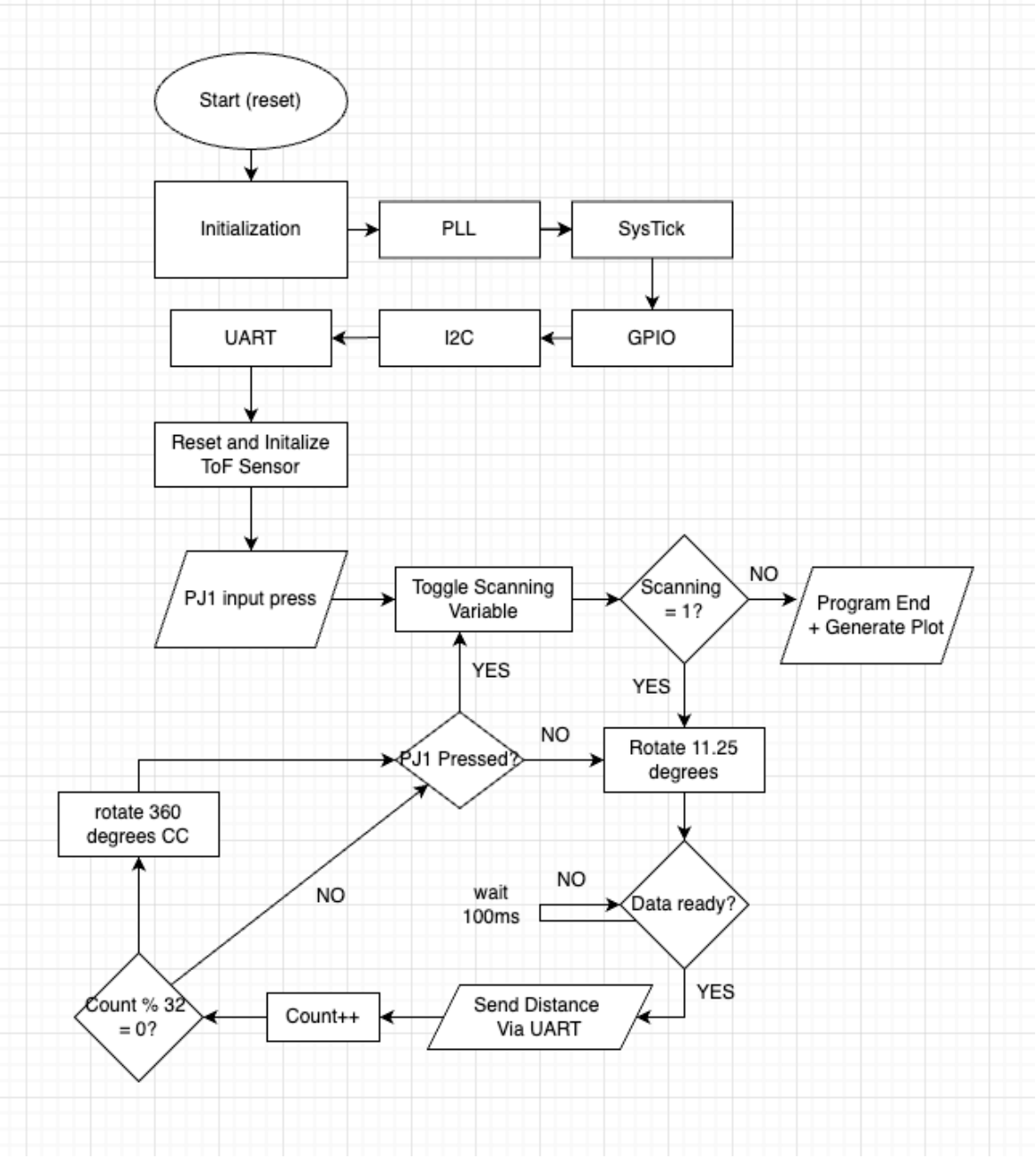

Figure 9: Main Program Logic

Python Code

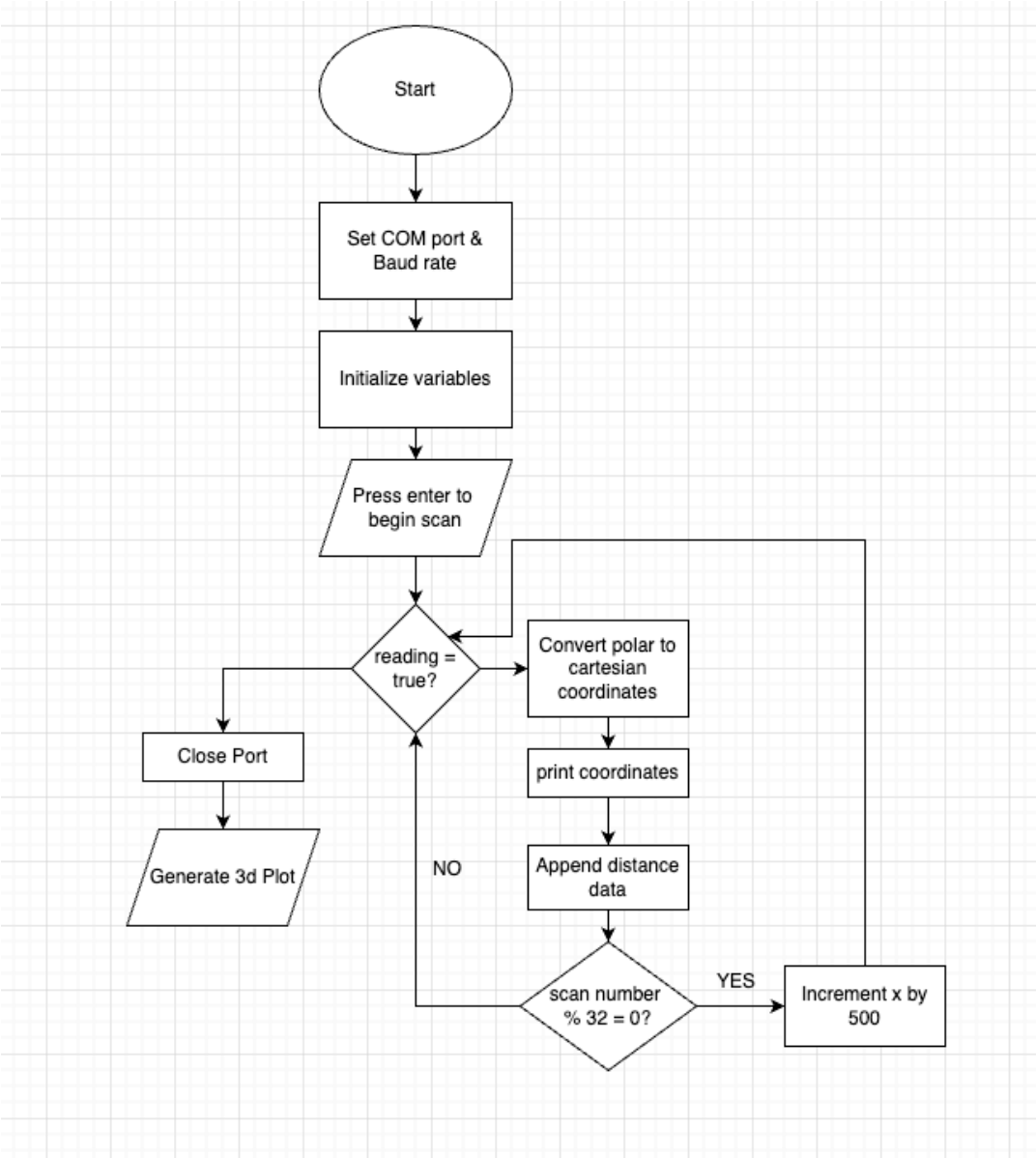

Figure 10: Python code

References

[1] “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).

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