COMP ENG 2DX3
Final Project Report - 3D Spatial Mapping System

lnstructor: Dr. Haddara/Doyle/Athar

Suchir Ladda – L09 – laddas - 400517569

Date: April 6, 2025

Table of Contents

Device Overview

The device is a small, lightweight LIDAR (light detection and ranging) system that is composed of several off the shelf components. The system was designed to map out indoor spaces in 3D producing an accurate depiction of the room. The brains of this system is the MSP-EXP432E401Y microcontroller which works alongside the MOT-28BYJ48 stepper motor, ULN2003 driver board, and the Pololu VL53L1X ToF sensor.

Features

Item Name	Purpose & Link	Price
MSP-EXP432E401Y Microcontroller	https://www.digikey.ca Serves as the brains for this project	$72.54 CAD
MOT-28BYJ48 stepper motor	https://www.amazon.ca/Stepper-28BYJ-48	$2.80 CAD
ULN2003 driver board	https://www.amazon.ca/Stepper-28BYJ-48-ULN2003	$2.07 CAD
Pololu VL53L1X ToF sensor	https://www.pololu.com	$18.95 USD

Table 1:

The performance features of the individual parts used in this design are as follows:

The MSP-EXP432E401Y microcontroller is powered by an ARM Cortex-M4 32-bit CPU that supports floating point operations and runs at a maximum CPU clock frequency of 120 MHz. For this particular design, the bus speed is set to 12 MHz. It offers multiple data transfer interfaces including JTAG, Micro USB, and Ethernet ports. The board is equipped with two user buttons that include built-in debouncing and pull-up/down resistor capabilities, along with four user-programmable LEDs. Memory-wise, it provides 1 MB of flash storage and 256 kB of SRAM. The microcontroller also features high-speed I2C modules, UART communication set to 115200 bps, and GPIO pins that support interrupts, PWM, and ADC functionalities.

The 28BYJ-48 stepper motor is a unipolar 4-phase motor with a 5.625°/64 stride angle. In this design, it is activated in full steps, resulting in a stride angle of 2.8125°/128. The motor operates within a voltage range of 5 to 12 VDC, making it suitable for various low-power applications.

To control the stepper motor, a ULN2003 driver board is used. This driver supports a 5–12 V power supply and includes onboard 4-way signal lights for debugging and monitoring. It is powered by the Texas Instruments ULN2003AN IC, which allows for reliable and efficient driving of the motor coils.

Finally, the Pololu VL53L1X time-of-flight (TOF) distance sensor is utilized for range measurements. It communicates via the I2C protocol at a speed of 100 kbps and can operate within a voltage range of 2.6V to 5.5V. The sensor features a 940 nm invisible Class 1 VCSEL emitter and offers up to a 27° field of view. It supports three distance measurement modes: short-range (up to approximately 130 cm), medium-range (up to around 300 cm), and long-range (up to about 400 cm), providing flexibility depending on the application's needs.

General Description

This project presents a budget-friendly, compact indoor 3D LIDAR solution as an alternative to expensive commercial systems. At its core, the device integrates an MSP-EXP432E401Y microcontroller with a 28BYJ-48 stepper motor, a VL53L1X time-of-flight (ToF) sensor by Pololu, and a ULN2003 driver board to control the motor.

The MSP432E401Y serves as the central processing unit and interfaces with all peripherals. It features a 32-bit ARM Cortex M4F running at 120 MHz, 1 MB of flash memory, and 256 kB of SRAM, complete with floating point support. For this implementation, its internal bus frequency is configured to 12 MHz.

After the reset button is pressed, the system operation begins when the user initiates data collection by pressing the onboard button connected to PJ1, which also activates LED D2 (on pin PN0) to signal readiness. A second press, this time on the PJ0 button, starts the motor’s rotation and briefly flashes LED D3 (on PF4) each time a new measurement is recorded and sent. Both inputs are managed through GPIO interrupt routines.

The motor, a 28BYJ-48 model, is a 4-phase unipolar stepper motor driven through the ULN2003 board. Operating at 5V, the motor has a full step angle of 11.25° per step. The scanning sequence is structured to rotate the motor through a full 360°, pausing every 11.25°—resulting in 32 discrete stops, each corresponding to a measurement point.

The distance sensing is handled by the Pololu VL53L1X ToF sensor, which emits a 940 nm Class 1 laser and uses a SPAD (single photon avalanche diode) array for detection. The sensor operates between 2.6V and 5V and can measure distances up to 4 meters. Activated via PJ0, the sensor captures a reading at each 11.25° interval as mentioned previously. The internal circuitry of the sensor does a lot of heavy lifting as it performs transduction, signal conditioning, and analog-to-digital conversion before transmitting digital data to the microcontroller over I2C. Communication is handled via the standard SDA (data) and SCL (clock) lines.

The sensor returns a package of measurement data that includes the range status, measured distance, signal rate, ambient light level, and the number of active SPADs. After each reading, the microcontroller relays this information via UART to a connected computer.

On the PC side, MATLAB is used to handle the incoming serial data stream. The script extracts parameters such as angle, distance, depth, etc. and transforms the polar coordinates into 3D Cartesian coordinates for y and z. The X coordinates are calculated by manually increasing them with each step to simulate depth. These are then rendered in a 3D scatter plot, connecting all measurement points taken at the same depth level to produce a visual representation of the scanned environment.

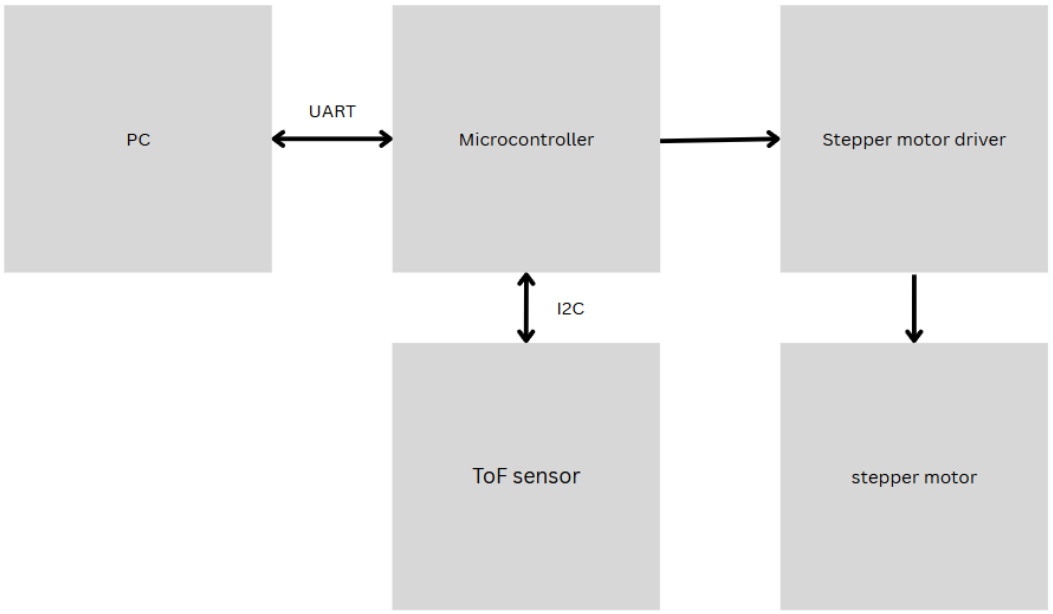


Figure 1: Block diagram of components

Device Characteristics Table

MSP-EXP432E401Y		ULN2003 driver board		VL53L1X ToF	
Bus Speed	12 MHz	V+	5V	Vin	3.3V
Measurement LED	PF4 (LED D3)	V-	GND	GND	GND
Status LED	PN0 (LED D2)	In1	PH0	SCL	PB2
Push Button 1 and 2	PM0,PM1	In2	PH1	SDA	PB3
Serial Port	COM7	In3	PH2		
Baud Rate	115200 BPs	In4	PH3		

Software used		
Software name	**Purpose**	**Link**
Keil	Programming the MSP-EXP432E401Y microcontroller	https://www.keil.com/
MATLAB	3D visualisation of hallway	MATLAB

Detailed Description

Distance Measurement

The VL53L1X time-of-flight (ToF) sensor operates by projecting infrared light from its internal emitter. When this light strikes a nearby surface, it reflects back and is detected by the receiver module. The sensor determines the object's distance by calculating the time interval between the outgoing and returning light pulses. The formula used is:

Distance = (flight time / 2) × speed of light

Once this raw measurement is acquired, the sensor internally handles the conversion process: transforming the optical signal through transduction, applying signal conditioning, and completing analog-to-digital conversion. The final digital output is sent to the microcontroller using the I2C communication protocol.

When the microcontroller powers on, it immediately executes the Keil C program stored in flash memory. This startup routine handles the setup process, which includes configuring GPIO pins, enabling LEDs, setting up interrupts, initializing communication protocols, and preparing the time-of-flight (ToF) sensor.

During initialization, the ToF sensor disables any active local interrupts, switches its distance mode to long-range, defines its timing budget, and establishes the delay between consecutive measurements. These parameters tailor the sensor's behavior for optimal performance within the scope of this system.

Once everything is initialized, the microcontroller enters a standby state where it listens for input through hardware interrupts triggered by two onboard push buttons. One of these, connected to pin PJ1, activates data acquisition and UART transmission by flipping a global flag variable named `scan_enable`. Pressing this button also switches on LED D2 (located at PN0), giving a visual cue that the device is ready to begin sensing and sending data.

The second onboard button, linked to PJ0, is responsible for starting the rotation of the stepper motor. It toggles a separate global flag called `rotation`, which initiates a full 360° sweep of the motor for a complete scan cycle.

After this, functions are called in the main control loop of the program, starting the process of the ToF sensor collecting values. Since a full rotation of 360° requires 512 steps, each step accounts for 0.703125°, and 16 steps equal 11.25°. When this condition is met, the system blinks a measurement LED and sets a flag to instruct the program to retrieve a distance sample. The rotations are monitored by the stepperCounter() function that increments the angle counts every time a flag is set. To check if a rotation of 11.25° was made, a modulo of 16 is taken with the total number of steps giving a value of 0.703125° per step. Using this information we can say that 11.25/0.703125 = 16.

To prevent the ToF sensor's wiring from becoming tangled during operation, the system includes a check to determine if the stepper motor has completed a full 360° rotation. Once this condition is met, the angle counter is reset, the `depth` value is incremented, and the motor is set to rotate in the reverse direction for the next scanning pass—pending user input. This is done by toggling a variable `dir` which in turn energises the coils of the stepper motor in the reverse order of what was done before, doing this makes the motor rotate in the opposite direction.

When the global flag `scan_enable` is set to true, the scanning process becomes active. Within the `stepperCounter()` routine, the program initiates a distance measurement every 11.25° of motor rotation. During each of these intervals, the microcontroller polls the ToF sensor and waits for fresh measurement data. The retrieved data is sent to the MSP432 via I2C and includes RangeStatus, Distance, Step, Depth and SpadNum.

The `step` and `depth` variables are maintained manually within the Keil C code, while the remaining values are fetched directly from the ToF sensor's internal registers through the provided VL53L1X C API. The `step` count acts as a proxy for the motor's angular position, where each step equates to 0.703125°. Thus, multiplying the step count by this value provides the current rotation angle.

To emulate forward displacement through a 3D environment, the depth variable increases by one unit (representing 1 meter in this setup) each time the motor completes a full revolution. After each reading, the collected data is sent from the microcontroller to the PC via UART, using a micro USB connection to handle serial communication.

Visualization

After the data is sent from the microcontroller, MATLAB takes over the computational and visualization tasks. It first establishes a serial connection with the computer through COM7, which is opened to begin receiving incoming UART data. As each data packet arrives, MATLAB processes it to prepare for plotting.

The raw input is parsed and organized into a structured matrix formatted as: [check_bit, distance, angle, depth, spadNum].Once the angle and distance are available, they are transformed from polar to Cartesian coordinates using basic trigonometric relationships:

- $y = \text{distance} \times \cos(\text{angle})$
- $x = \text{distance} \times \sin(\text{angle})$

In this project, the x value obtained from the equation is treated as the z coordinate to help simulate the vertical structure of the scanned environment. Meanwhile, the depth value—incremented with every full 360° rotation of the stepper motor—is used as the x coordinate in the final 3D model.

The result is a plot of points on the yz plane extended over the x axis (depth), allowing MATLAB to build a 3D visualization of the scanned area.

Application Example

This section of the report is going to demonstrate a sample scan of a 3D space within the McMaster campus. This example illustrates the Lidar system as well as detailed instructions for setting up. It was scanned from the second floor of the John Hodgins Engineering Building (JHE) according to the last digit of my student number. It has been presumed that both Keil and MATLAB are successfully installed and configured in your preferred device. In addition, it has also been presumed that the circuit is the same as shown in the device overview section.

The use of laser pulses allows lidar technology to find the distance to things, supporting the creation of accurate three-dimensional representations of physical environments. It is a type of advanced remote sensing that plays a large part in many sectors thanks to its accuracy as well as versatility.

In geographic and structural mapping, lidar is used extensively in the creation of highly detailed digital elevation models. Such models are important decision tools used in the design of urban infrastructure, flood hazard assessment, and management of natural resources. The use of lidar also extends into the automotive world, where it has been adopted as a foundation in the development of self-driving cars. By offering real-time environmental awareness, it supports object detection, collision avoidance, and navigation.

Aside from transportation planning and urban planning, lidar scanning has also made significant contributions in fields such as archaeology and forestry. In archaeological studies, it helps uncover hidden or buried features by producing topography with sub-surface information. Likewise, in forest management, lidar helps build precise forest density and structure models, assisting governments in monitoring ecosystems, planning land management, and evaluating hazards such as landslides and fire-risk areas.

Instructions

1) Using the micro USB cable in the black TI LaunchPad box, connect your MSP-EXP432E401Y microcontroller to the USB port in your PC. Ensure that you are plugging in your cable into the port seen in the top left of the image below.

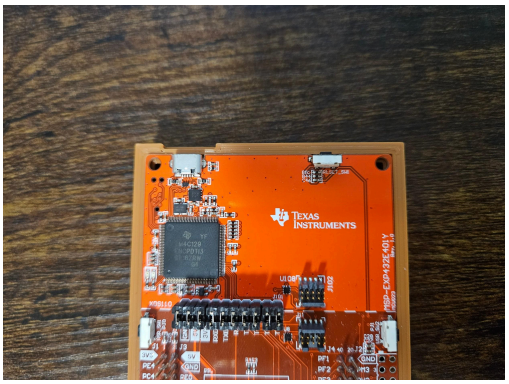


Figure 2: Upper portion of microcontroller

2) Ensure that you know what port is configured to UART in your device. This can be done by running the command: python -m serial.tools.list_ports -v. As seen below, COM7 is using UART in my case
    a) In MATLAB change the value of X in: s = serialport(ports(X), 115200); so that it corresponds to your UART port

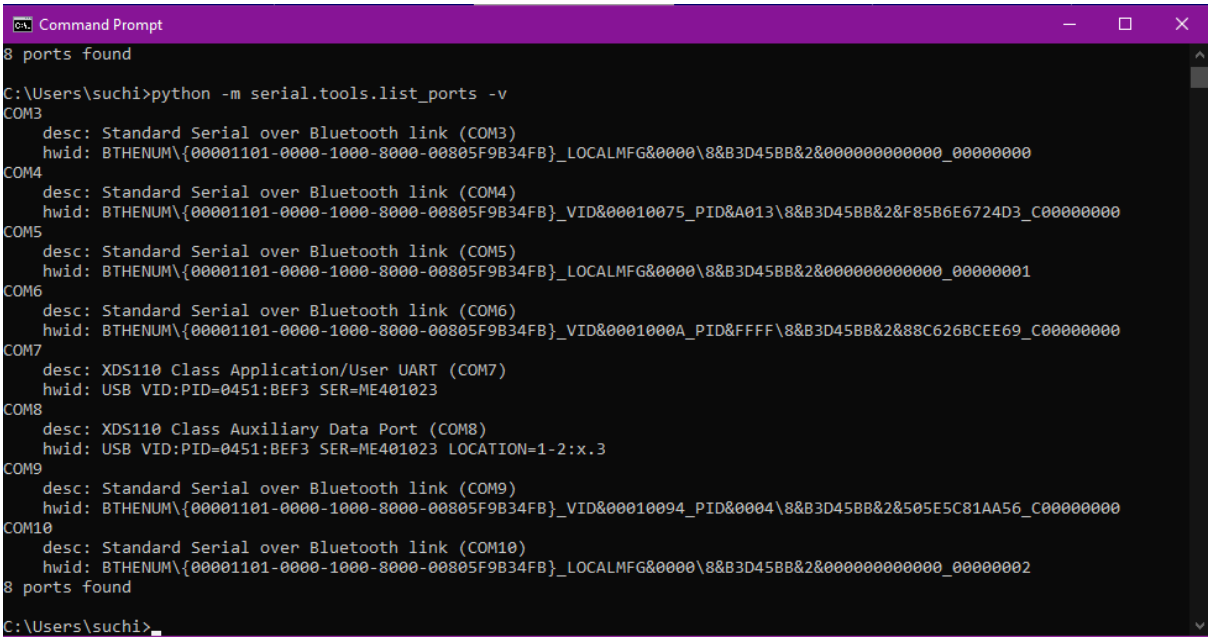


Figure 3: Command prompt displaying all ports

3) Depending on the length and how detailed the desired scan is, the variables depth, and measurements in the matlab code can be changed. Depth correlates to how many scans/steps you want to take in the X direction, the larger the value, the longer your hallway. Measurements changes the detail of the scan, currently it is set to 32 meaning measurements are taken every 11.25° degrees.
4) Ensuring proper connections are made, open the Keil project
5) Click on ‘Translate’ → ‘Build’ → ‘Load’
6) Flash the program onto the microcontroller by pressing the reset button in the right upper corner of figure 2.
7) Upon pressing the reset button, all 4 onboard LEDs will blink indicating everything is running properly.
8) In order to make sure the ToF reading get graphed, go back to MATLAB and press run to start the program
9) Position the board as desired and press PJ1. Onboard LED D2 should light up indicating the ToF sensor is ready to transmit values to the board
10) Press PJ0 to begin motor rotation. The motor will begin spinning in 11.25° increments with LED D3 blinking each time the motor pauses. This blink indicates a measurement was made

11) Once an entire revolution of the motor is complete, the motor will stop on its own. Then proceed to move the setup in the X direction as far desired, once ready, press PJ0 again to begin your next set of measurements.
12) Repeat step 11 as many times as needed until all of the depths have been measured
13) After all measurements are complete, MATLAB will generate a precise 3D plot showing the dimensions of the scanned space.

Expected Output

The output generated by the system can be evaluated by comparing the 3D reconstruction to its real-world counterpart at the designated scanning site. For this project, the assigned scanning location was location J (JHE 2nd floor). Below are visuals that show the physical hallway and the digital model produced by the completed LIDAR device.

As observed in the scan results, the device effectively captured the spatial layout and features of the environment. Figure 5 to 8 clearly shows the geometry of the hallway. It can be seen that it accurately measured the changing height in the ceiling. It also took into account the long bench on the side. This overall shape is faithfully represented in the scan.

Looking at figure 4, you can clearly see the resemblance between the 3D graph and the actual space, showing a strong alignment between the scan data and the physical environment. It should be noted that the hallway had a lot of wide open spaces way beyond the 4m range of the ToF sensor as well as a lot of reflective surfaces resulting in some skewed values.

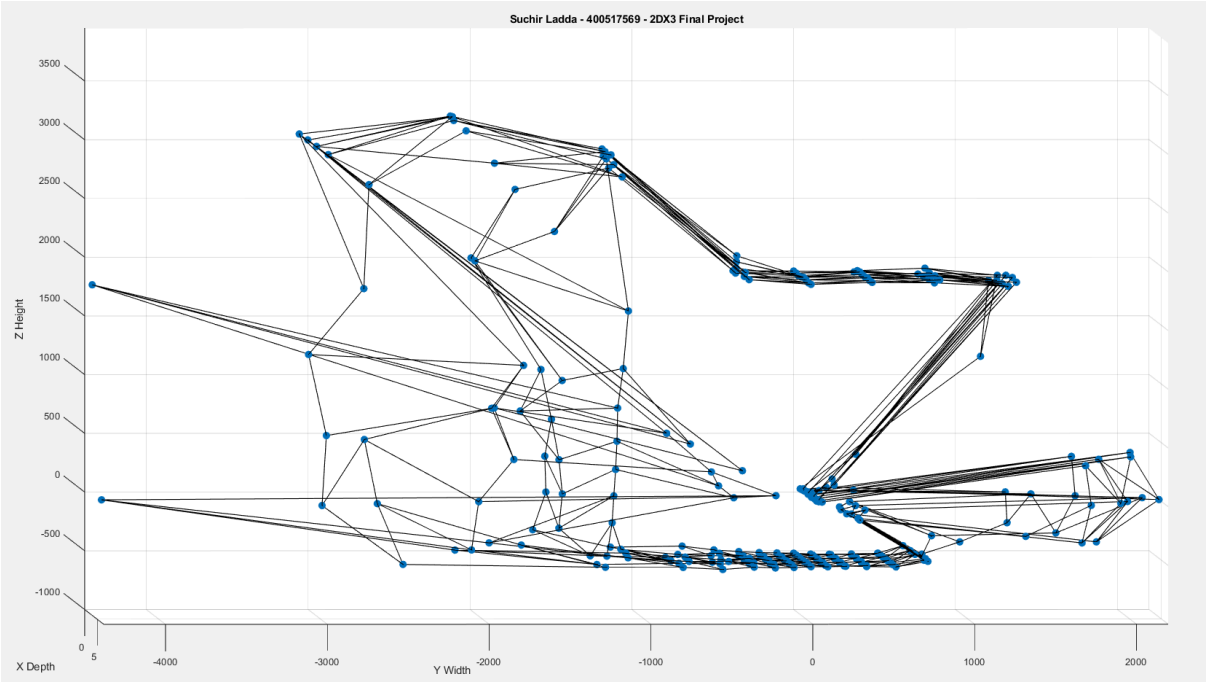


Figure 4: Matlab output of hallway

Note that the result is a plot of points on the yz plane extended over the x axis (depth), allowing MATLAB to build a 3D visualization of the scanned area.

Figure 5: Start of hallway

Figure 6: Showcasing different height in ceiling

Figure 7: Long bench in hallway

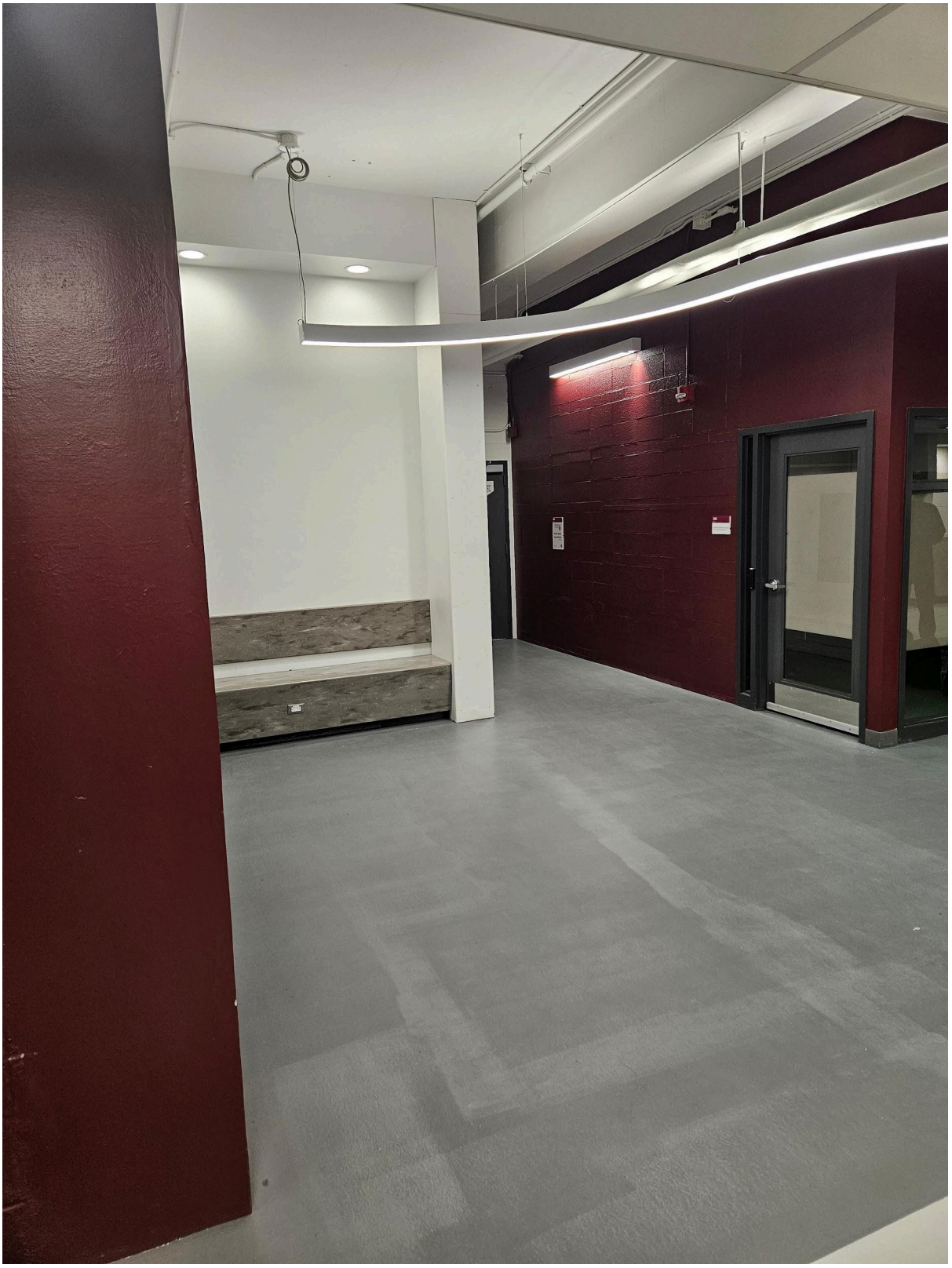


Figure 8: Large opening, outside of ToF range

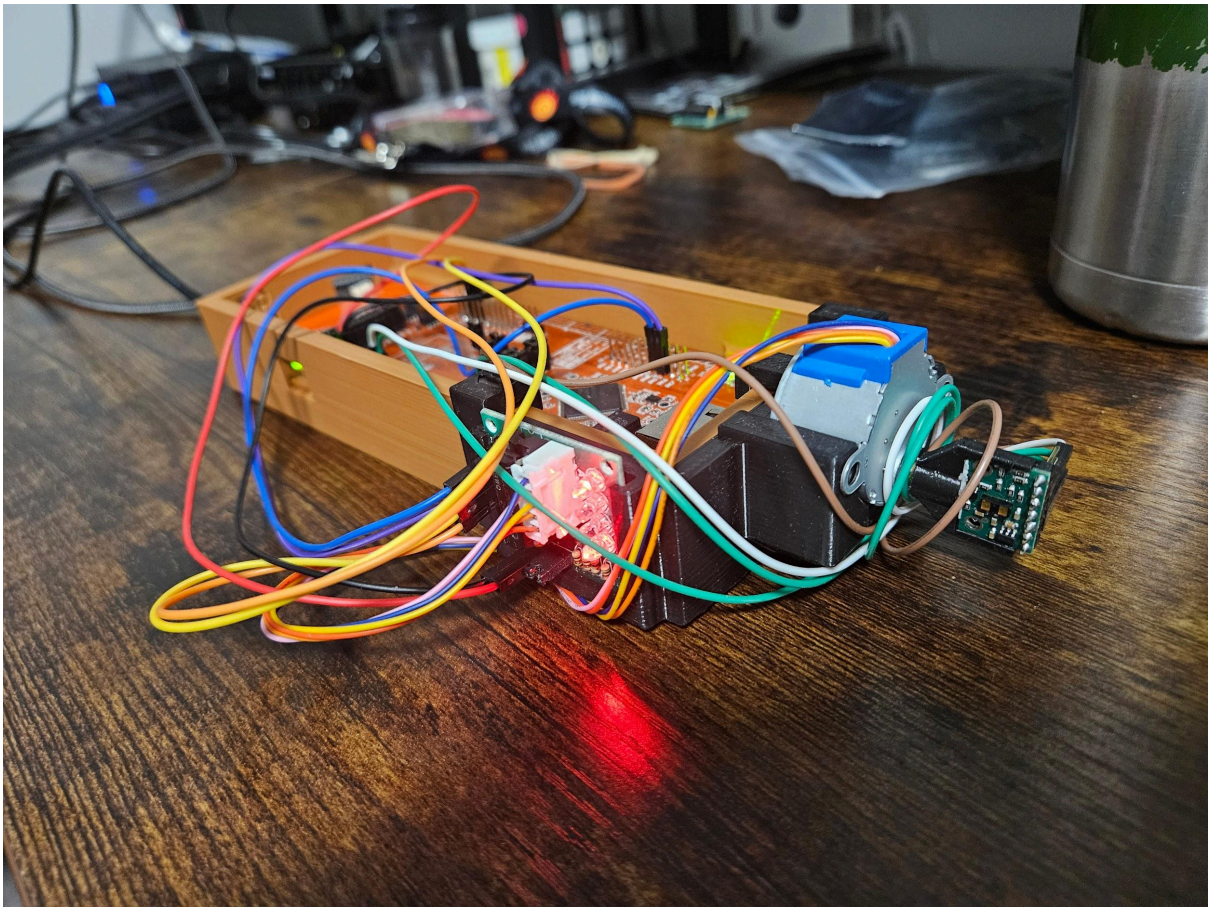


Figure 9: Physical implementation of circuit

Limitations

While this system was designed to be a low-cost and adaptable alternative to high-end LIDAR platforms, several inherent limitations impacted its overall performance and precision.

One major constraint arises from the microcontroller's floating-point capability. The MSP-EXP432E401Y is equipped with a 32-bit ARM Cortex-M4F floating point unit (FPU). Although this allows the device to perform basic floating-point operations such as addition, multiplication, and square roots, it lacks the double-precision and computational efficiency offered by modern desktop CPUs. As a result, the most computation-heavy floating-point operations—particularly the trigonometric functions required to convert polar coordinates (distance and angle) into Cartesian coordinates—were offloaded to the PC. Attempting to run these calculations on the microcontroller would not only reduce precision but also introduce significant delays in processing, especially under repeated or real-time operations. The combination of limited FPU accuracy and the performance cost of onboard trigonometric functions meant that such operations were avoided entirely on the embedded side of the system.

In terms of measurement precision, the VL53L1X time-of-flight sensor used in this project has a resolution of 1 mm, which directly defines its quantization granularity. Using the standard formula for quantization error:

Max Quantization Error = (Maximum Measurable Range) / (2^n bits),
we substitute the known values:

$4000 \text{ mm} / 2^{16} = 0.061035 \text{ mm}.$

However, since the sensor itself is designed to output data with 1 mm resolution, the actual quantization error is effectively 1 mm, as this is the smallest measurable increment the sensor will report.

Regarding serial communication, the maximum standard UART rate that could be reliably established with the PC was 128000 bps. This value was verified by navigating to the Windows Device Manager, expanding the "Ports (COM & LPT)" section, selecting the UART port associated with the MSP432 (in this case, COM7), and reviewing the maximum supported bit rate in the port settings under the "Bits per second" drop-down menu. Despite the max speed being 128000 bps, I used 115200 bps in my project.

For communication between the microcontroller and the VL53L1X module, the I2C protocol was utilized. This implementation used a standard rate of 100 kHz, which was sufficient for the data throughput required per measurement cycle.

When analyzing the entire system, the two primary performance bottlenecks were the stepper motor and the ToF sensor's timing budget. The stepper motor used—rated with a 0.703125° minimum step size, required a minimum delay of 2 ms per phase activation, which limited the maximum rotation speed. Furthermore, the VL53L1X sensor's measurement cycle depends on a timing budget, which can range from 20 ms to 1000 ms. While increasing the timing budget improves accuracy and range consistency, it forces the stepper motor to halt for the duration of each sensor reading, further extending total scan time. To assess this, multiple timing configurations were tested and benchmarked by measuring the full scan duration under varying timing budgets and stepper motor delay settings. It became clear through these tests that improvements in speed would require either higher-speed motors or a reduction in sensor timing—each with tradeoffs in accuracy and reliability.

Circuit Schematic

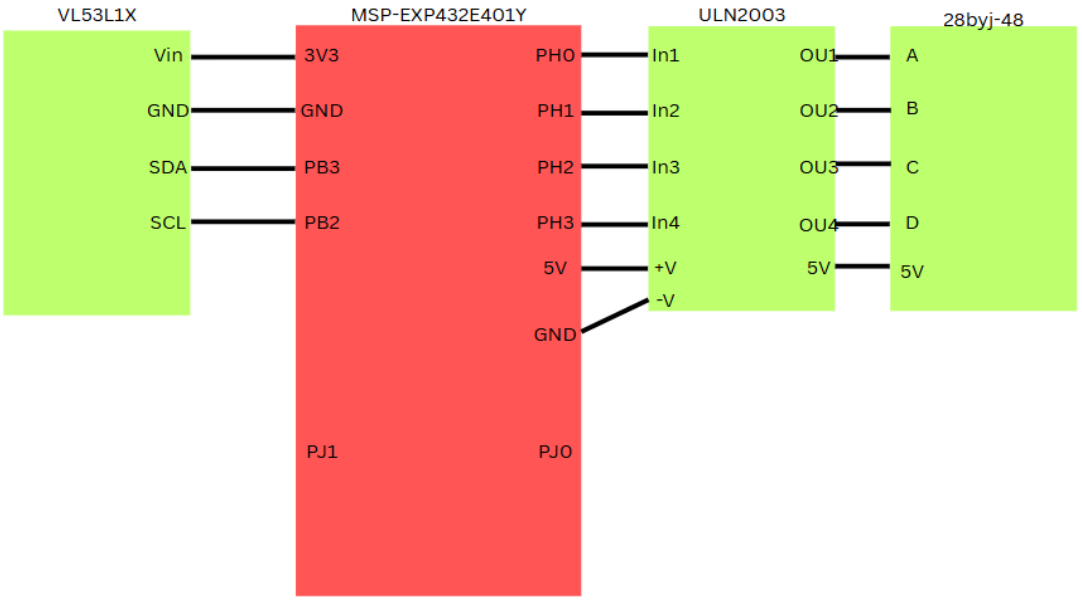


Figure 10: Circuit schematic

Programming Logic Flowcharts

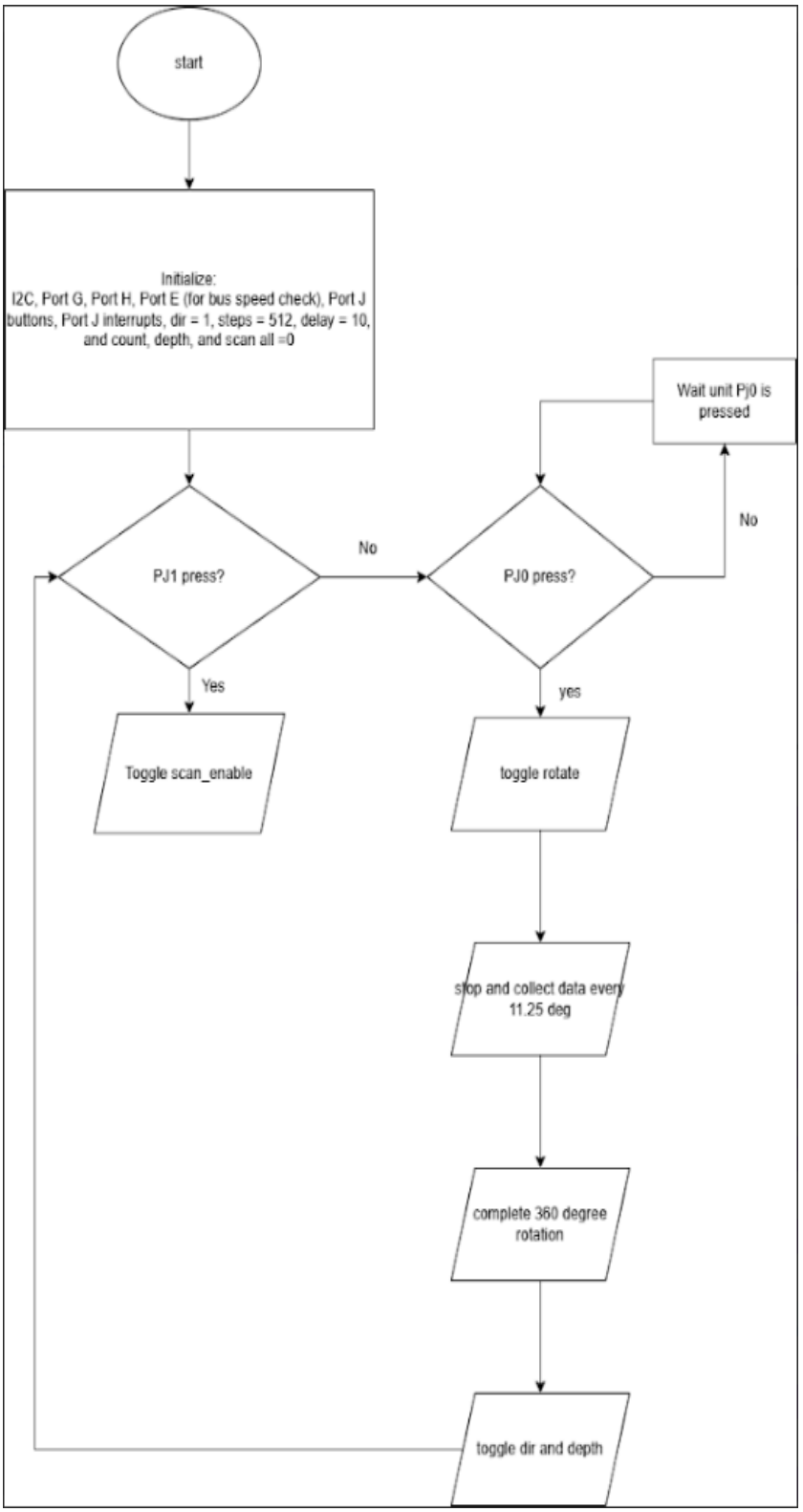
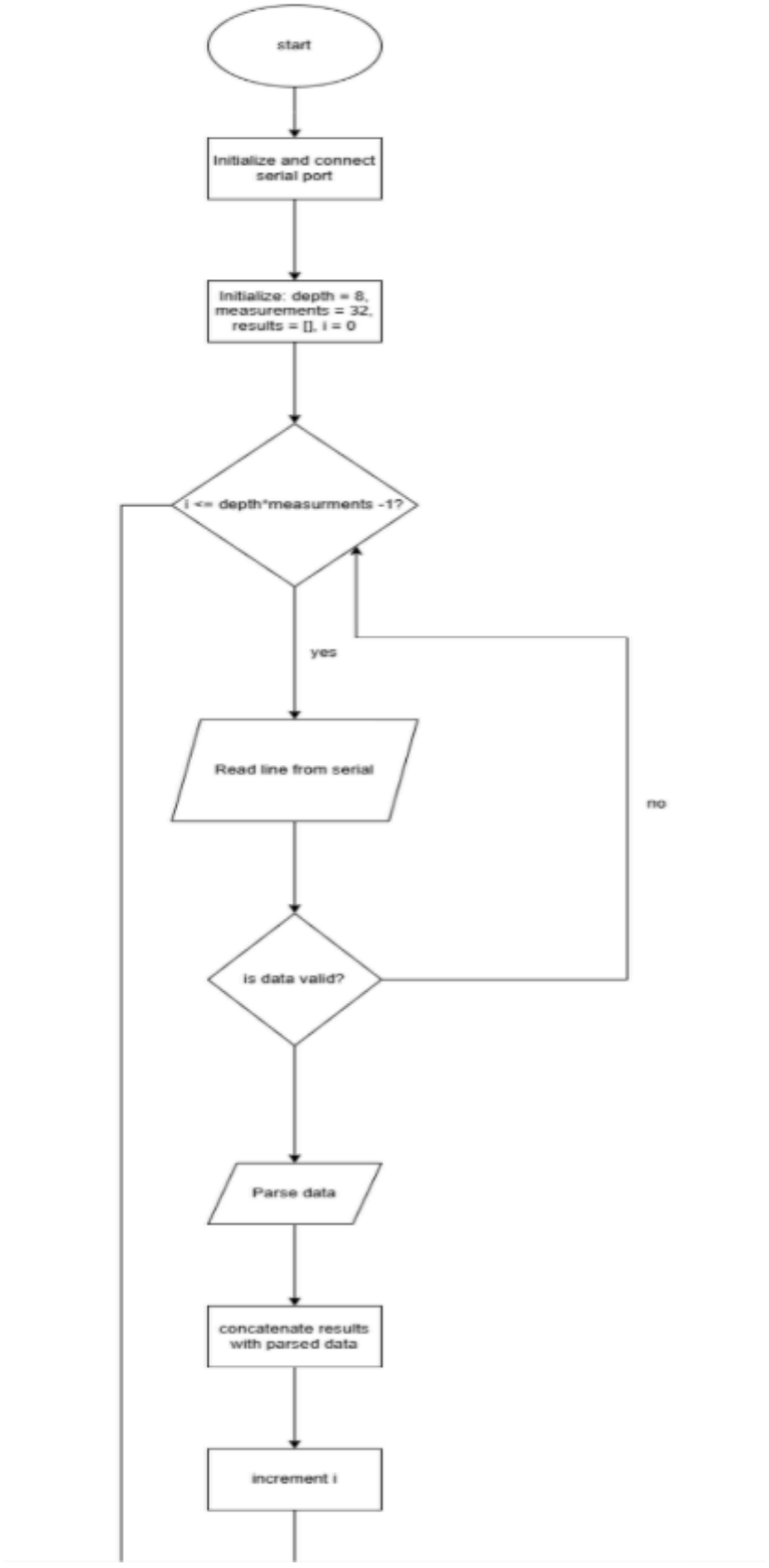


Figure 11: Keil code flowchart


start
Initialize and connect serial port
Initialize: depth = 8, measurements = 32, results = [], i = 0
i <= depth*measurments -1?
yes
Read line from serial
is data valid?
no
Parse data
concatenate results with parsed data
increment i

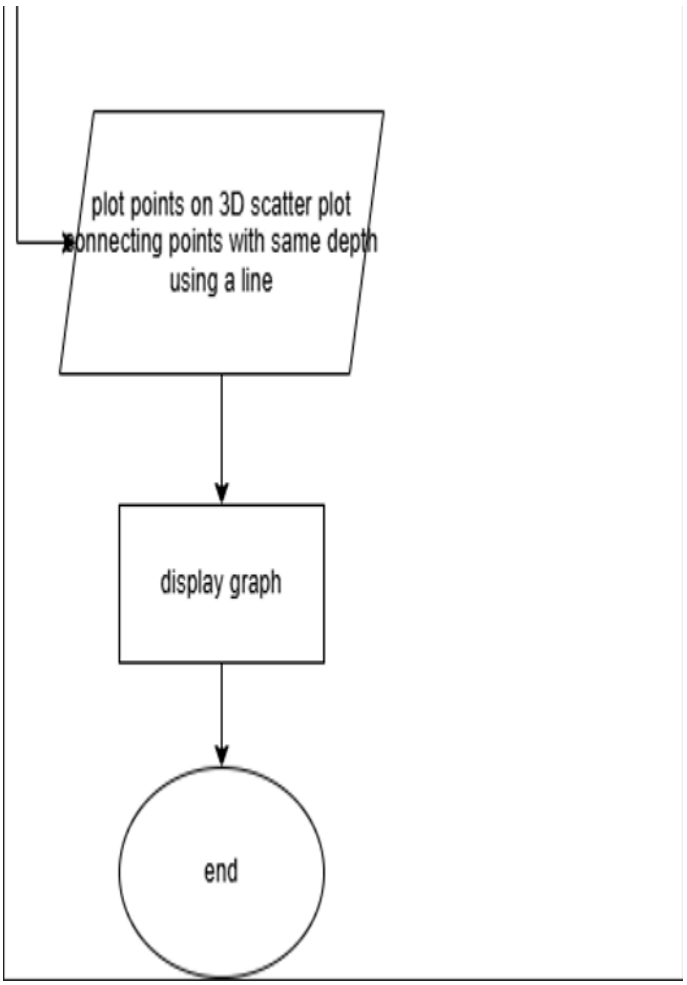


Figure 12: Matlab code flowchart