COMPENG 2DX3: Microprocessor Systems Project - Final Project

Spatial Mapping Using Time of Flight

Aryan Karnati - karnatia - 400507538 - L09

Instructor: Dr. Yaser M. Haddara, Dr. Shahruhk Athar, Dr. Thomas Doyle

Department of Electrical and Computer Engineering, McMaster University

Submission Date: April 9th, 2025

Table of Contents

Device Overview....4
- Hardware Used....5
- Software Used....6
- Features....6
- General Description....7
- Block Diagram....9
- Data Flow Graph....9

Device Characteristics....9

Detailed Description....11
- Distance Measurement....11
- Visualization....12

Application Example....13
- Setup Instructions....14
- Scanned Location....15
- Reference Images....16
- Expected Output....17

Limitations....18

Circuit Schematic....19

Programming Flow Chart....20

References....21

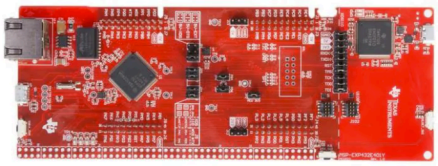
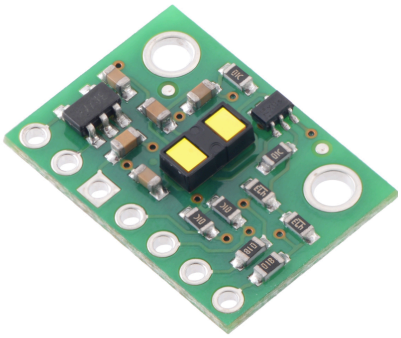
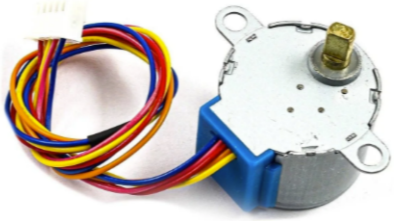
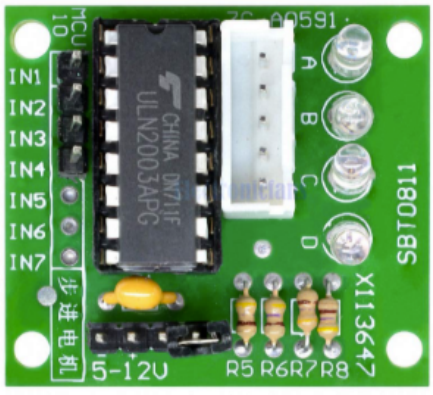
Device Overview

This portable device is a LIDAR (Light Detection and Ranging) system designed for creating 3D maps in indoor environments. Its central component is the Texas Instruments MSP-EXP432E401Y microcontroller, which works in conjunction with one on board push button, a ULN2003 driver board, a 28BYJ-48 stepper motor and a Pololu VL53L1X time-of-flight (ToF) sensor.

Overview

- Microcontroller powered from a 5V micro-USB cable
- 1 onboard push button to control the start and the stop of the system
- Measurement status LED (PF4)
- UART status LED (PF0)
- Additional status LED (PN1)
- ToF sensor communicates distance data with the microcontroller in real time utilizing I2C
- Microcontroller communicates data to the PC utilizing UART at a 115200 bps baud rate in real time
- Spatial map created by MATLAB

Hardware Used

Hardware	Image	Price	Datasheet
MSP-EXP432E401Y Microcontroller		$47.99	https://www.ti.com/lit/ug/slau748b/slau748b.pdf?ts=1743453336719
Pololu VL53L1X TOF		$24.85	https://www.pololu.com/file/0J1506/vl5311x.pdf
28BYJ-48 Stepper Motor		$11.39	https://www.digikey.ca/en/htmldatasheets/production/1839399/0/0/1/28byj-48
ULN2003 Stepper Motor Driver		$1.95	https://www.hadex.cz/spec/m513.pdf

Software Used

Software Tools	Purpose	Link
Keil MDK (C)	Programming Microcontroller	https://developer.arm.com/Tools%20and%20Software/Keil%20MDK
MATLAB	3D Map Visualization	https://www.mathworks.com/products/matlab.html

Features

MSP-EXP432E401Y Microcontroller

- Core: 120 MHz ARM Cortex-M4F (with hardware floating-point)
- Bus Speed for this design: 26 MHz
- JTAG, Micro USB, Ethernet ports to transfer data
- Utilizing 1 push button featuring debouncing & pull up/down resistor capabilities
- 4 LEDs
- Flash Memory: 1 MB
- SRAM: 256 kB
- I2C modules
- UART communication capabilities
- Baud rate for this design: 115200
- GPIO pins that support interrupts, PWM, and ADC

Pololu VL53L1X TOF

- Operating Voltage: 2.6V - 5.5V
- Field of View: 15^{o} - 27^{o}
- Range: Up to 4 meters
- I2C Protocol at 100kbps

28BYJ-48 Stepper Motor

- Unipolar stepper motor
- 4 Phases
- The motor used in full step for this design
- Operating Voltage: 5-12 VDC

ULN2003 Stepper Motor Driver

- Power supply: 5-12V
- Indicators: 4 onboard LEDs to show step signals
- Powered by Texas Instruments ULN2003AN IC

General Description

The primary purpose of this device is to develop an affordable and portable alternative to commercial LIDAR systems, specifically for indoor use. This solution aims to make spatial sensing technology more accessible for indoor mapping, obstacle detection, and robotics applications.

This device is a 3D LIDAR system utilizing an MSP-EXP432E401Y microcontroller, a Pololu VL53L1X time-of-flight sensor, a 28BYJ-48 stepper motor, and a ULN2003 stepper motor driver. The core component of the system is the MSP-EXP432E401Y microcontroller, which has a 120 MHz system clock, 32-bit ARM Cortex M4F CPU, 1MB of flash, 256kB of SRAM, and floating point capabilities. For this specific system, a bus speed of 10 MHz was used. To start the system, the user must push the on board button PJ0. This enables the spinning of the motor, data acquisition, and data transmission. The button utilizes polling logic by consistently checking if the user has pushed the button and the outcome will be based on the input of the button. Once the button has been pressed, the motor will start rotating clockwise, taking distance measurements and sending the data to MATLAB for processing. After one full rotation of 360^{o} the motor will rotate back to the original position without taking any measurements to set itself up for the next scan. When the button is pressed anytime during the scanning cycle, the system will stop and not take any measurements. When PJ0 is pressed, three LEDs will light up, and each one has a different functionality. LED 0 (PN1) will stay on when in the scanning phase and will turn off when in the resetting phase. LED 2 (PF4) will blink every time a distance measurement is taken. LED 3 (PF0) will flash (toggle) once at the beginning of every scanning cycle.

The 28BYJ-48 is a 4-phase unipolar stepper motor powered through the ULN2003 driver module. It is activated using the onboard push button PJ0. It operates at 5V, the motor features a stride angle of 5.625° per 64 steps. In this system, the motor completes a full 360° rotation in 11.25° increments, taking a total of 32 measurements for each scanning cycle.

The Pololu VL53L1X Time-of-Flight (ToF) sensor is a laser-based ranging sensor that uses a 940 nm Class 1 laser and a SPAD (Single Photon Avalanche Diode) array to detect reflected signals. The sensor operates from an electrical supply voltage of 2.6V to 5V and can measure range distances of up to 4 meters. In this system, the sensor is activated by the PJ0 button and collects measurements every 11.25° of rotation. The ToF sensor receives analog signals, but to manipulate the data, this system needs digital signals. The analog-to-digital conversion process

occurs on the VL53L1X ToF itself. This process includes transduction and conditioning and then results in a final digital signal. The ToF communicates to the microcontroller utilizing the I2C communication protocol. The sensor sends all the measurement data to the microcontroller which then will be sent to MATLAB for further processing. The SDA and the SCL pins were utilized as the data and clock lines. For this system, the only data being sent from the ToF to the microcontroller was the distance measurement.

After each measurement is captured, the data is transmitted to the PC via UART for visualization. The only transmitted parameter is the distance measurement taken from the ToF. The MATLAB program receives this data through one of the computer's communication ports and parses the incoming serial stream. MATLAB keeps track of the angle for each measurement and the depth of each cycle scan. It then converts the polar coordinates into Cartesian coordinates (x, y, z) and generates a 3D scatter plot. It connects all points that lie within the same depth plane to create a clear spatial representation of the scanned environment.

Once an individual measurement is taken, the resulting data is transmitted to the PC using UART for visualization purposes. The only parameter being transmitted is the distance measurement recorded from the ToF. The MATLAB program is then set up to receive this information from one of the communication ports on the computer and parse the incoming serial stream. MATLAB keeps track of the angle for each measurement and the depth of each cycle scan, converting the polar coordinates into Cartesian coordinates (x, y, z) to create a 3D scatter plot. It connects all points that are located on the same depth plane to create a clear spatial representation of the scanned environment.

Block Diagram

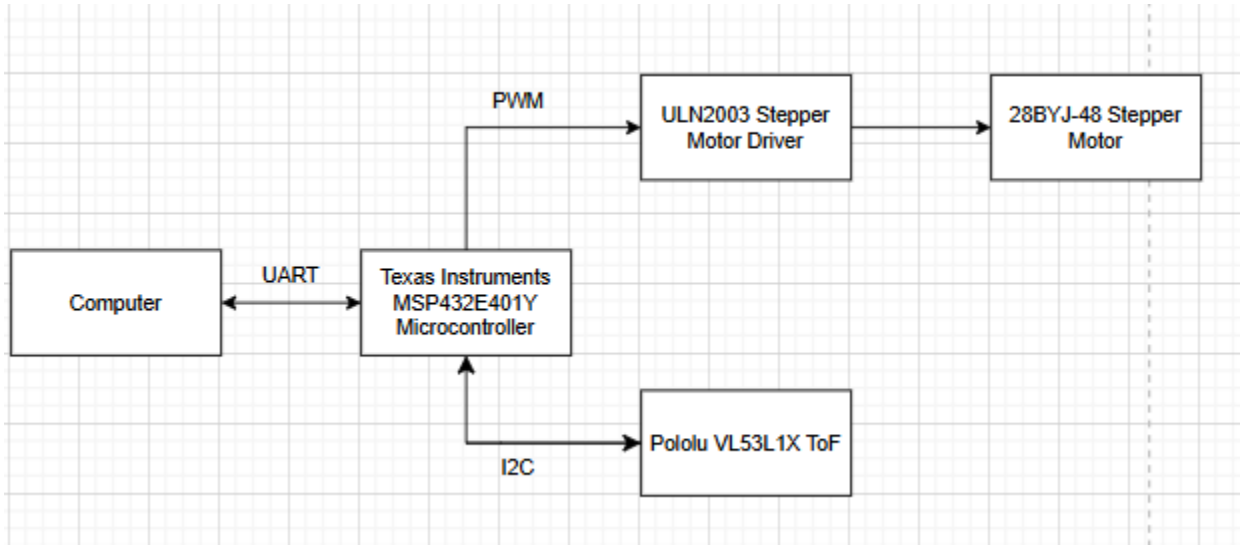


Figure 1: Block Diagram of the Communication of the System

Data Flow Graph

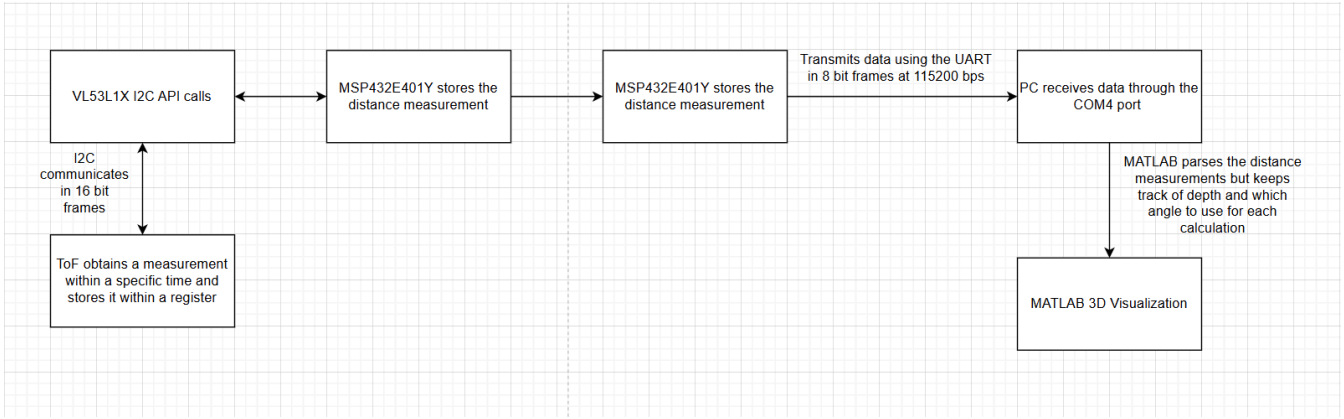


Figure 2: Data Flow Graph of System

Device Characteristics

Component	Power Requirements
MSP-EXP432E401Y Microcontroller	5V from the Micro USB
Pololu VL53L1X TOF	3.3V from the Microcontroller
28BYJ-48 Stepper Motor	5V from the Stepper Motor Driver

ULN2003 Stepper Motor Driver	5V from the Microcontroller

MSP-EXP432E401Y Pin	Description	Signal Description
GPIO [H0:H3]	Output for Stepper Motor Driver	PWM
GPIO Port J [J0]	GPIO button J0 uses the polling method to check if detected	GPIO Polling
GPIO B2	I2C Clock line	SCL
GPIO B3	I2C Data line	SDA
GPIO E0	Used for bus speed verification	PWM
GPIO F4 & F0	Onboard LED D3 & D3	Toggle/Flash
GPIO N1	Onboard LED D1	Toggle/Flash
Micro USB	Outputs data to PC through UART protocol	UART

Parameter	Value
UART Frame	1 start bit, 8 data bits, 1 stop bit
UART Baud Rate	115200
I2C Frame	1 start bit, 7 follower address bits, 1 read/write bit, 1 acknowledge bit, 8 data bits, a 1 acknowledge/not acknowledge bit, 1 stop bit
I2C Clock	100kbps
I2C address VL53L1X	0x29

Parameter	Value
MSP-EXP432E401Y Bus Speed	26 MHz
Bus Speed Check Period	2ms

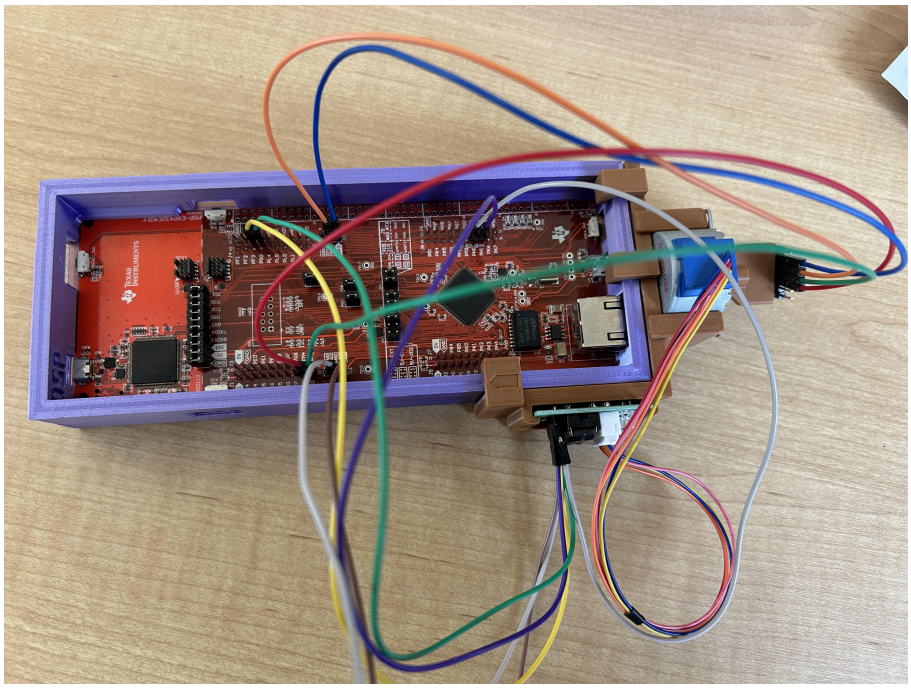


Figure 3: Physical Device

Detailed Description

Distance Measurement

Distance measurements were conducted using the VL53L1X Time-of-Flight (ToF) sensor. This device consists of two main components: a 940 nm Class 1 laser emitter and a SPAD (Single Photon Avalanche Diode) receiver array. For distance measurements, the sensor emits an infrared laser beam that reflects off surfaces and enters the receiver. The SPAD receiver operates much like any other diode, as it has a pn-junction that is biased close to the breakdown voltage. Upon a photon hitting the pn-junction region, the photon is absorbed, creating an electron-hole pair. Due to the high reverse voltage, each created electron-hole pair will be accelerated. If sufficient energy is present, they create an avalanche effect where these electron-hole pairs are rapidly multiplied, creating a measurably detectable signal current.

Using the previously mentioned principle, the ToF sensor can then compute the distance based on the known velocity of the emitted photons and the detected lag between the emission and reception of those photons. For distance calculations, the equation is $\Delta d = c \bullet \Delta t/2$, where c refers to the velocity of light/photons emitted by the device. In the above-stated equation, Δt is the time difference between the emission of the photons and when the photons collide with the SPAD. The difference in time measured is divided by 2, as Δt includes the time it took for light to travel to the object and back to the device. This information is then passed to the microcontroller using the I2C serial communication protocol.

Once the microcontroller is plugged in, the Keil C code stored in the flash memory runs. The code starts by initializing GPIO pins, LEDs, communication protocols and the ToF. Once initialized, the ToF distance mode is set to long and is prepared to collect measurements. The microcontroller waits for specific trigger variables through polling. The polling focuses on the onboard button PJ0. When this button is pressed, the stepper motor begins a single rotation, and data collection and transmission are enabled. This button also turns on LED D1 (PN1) to showcase to the user that the system is collecting data and spinning clockwise.

The ToF sensor rotates 360 degrees to scan the entire surrounding space using a 3D printed mount which connects to the 28BYJ-48 stepper motor. The measurements the ToF captures are along the yz-plane and the depth in the x-plane is kept track of in MATLAB. The sensor begins in an upwards facing position. The inner gear ratio of the stepper motor has a stride angle of 5.625º/64 steps. To rotate a full 360º, it requires 512 steps. For this specific design, every measurement was taken at 11.25°, which results in 32 measurements every 360º. This was chosen to ensure a precise 3D reconstruction of the scanned area. The movement of the motor is controlled by the motor() function. The main code begins by calling the motor() function and incrementing the angle counter. The function checks if the stepper motor has rotated 11.25º by doing $step\_counter \ \% \ 16$. This was calculated using the fact that 512 steps is a 360° revolution. This means that $\frac{360}{512} = 0.703125^{o} \rightarrow \frac{0.703125}{1\ step} = \frac{11.25}{x\ steps} \rightarrow 16\ steps$. This means that every 16 steps, the motor will pause its rotation and take a measurement on the ToF. The code also checks if the step_counter has reached 512 steps showcasing a full revolution has occurred. If this has happened, the motor will reset variables like step_count and angle, and toggle the motor direction. Then the motor will rotate in the opposite direction to reset the motor and ensure the wires do not get tangled.

Once a button press sets the global variable motor_running to true, the scanning process begins in the main while loop. The motor() function is called repeatedly, moving the stepper motor a distance of 0.703125°, or every 16 moves (11.25°), the VL53L1X Time-of-Flight sensor is polled for a new distance reading. This distance is obtained over I2C via function calls from the VL53L1X API and contains the distance measured in mm. Internally, a step counter keeps track of the current angular position of the motor, which is used to compute the current scan angle by

multiplying the number of steps by 11.25°/16. All distances scanned are formatted using sprintf() and sent to the PC over UART via a micro USB cable. All collected data are used to reconstruct the environment scanned in 3D space. The process occurs until 512 microsteps have been taken (complete 360°). The motor will stop moving for 2 seconds and then reverse to scan in the opposite direction. The communication via UART provides the ease of not worrying about the distance data transfer, allowing for real-time updates on a PC during scan attempts.

Visualization

The visualization is done entirely within a MATLAB script, which interprets and plots 3D distance data received through the UART from the microcontroller. The script begins by establishing serial communication on COM4 with a baud rate of 115200. After establishing communication, the program initializes scan parameters, including the total number of scans and the number of steps for each scan; each step represents 11.25° of rotation in a full 360° scan. MATLAB will receive 32 distance measurements over UART for each scan. Each measurement will be read one by one using readline. The distance measurement will then be transformed into a number with str2double and checked to confirm it is a finite numeric value. Once all 32 measurements are received for a scan, the angle associated with each measurement will be calculated by dividing 360° evenly into 32 parts. Each angle will be converted into radians to use for coordinate transformation. The script uses the distance and angle data to calculate Cartesian coordinates, as indicated in the following equations:

$y = \text{distance} \cdot \cos(\theta)$, $z = \text{distance} \cdot \sin(\theta)$, $x = \text{scan index}$

This places each scan on a different X-layer, allowing MATLAB to reconstruct a layered 3D view of the surrounding environment.

When all the scans have been received and converted, MATLAB reshapes the X, Y, and Z coordinate data into matrices by both the current scan number and step index. The final 3D visualization is plotted using MATLAB's plot3 function. Each scan is plotted in a ring of 32 connected points, which represent one full 360° horizontal sweep. The script will then connect each point to the corresponding angular position from the previous scan and the following scan, thus stacking the rings into a 3D mesh.

Application Example

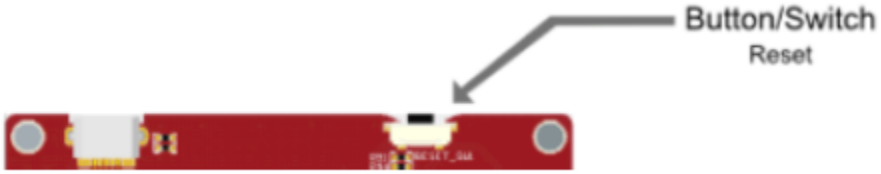
This part of the document contains an example of the 3D scan performed by the custom LIDAR system, as well as steps for setting up the system and gathering measurement data. An example of what this would look like is available for comparison. Following the assignment, the scanned

location was based on everyone's specific student number. The identified location for this specific scan was Location I, the scan was done in McMaster's John Hodgins Engineering Building (JHE). Reference materials to help with comparisons of JHE and what the scan will look like have been provided below. It is assumed that the user has downloaded the required software and has been set up appropriately on the machine. Details for the required software can be found in the Device Overview section. It is further assumed that the hardware is fully assembled. Refer to the Circuit Schematic section for proper hardware setup and wiring information.

Setup Instructions

1. Connect the TI-EXP432E401Y microcontroller to your PC using a micro USB cable. Determine the COM port that the two devices are communicating with by opening up the device manager and check the port section (Windows 11). This port will be defined as XDS110 Class Application/User UART (COM #).
2. Open up the MATLAB script and change the following lines based on the specific COM number and the number of scans that need to be conducted.
    a. `port = "COM4";` - Change the COM number to match the one found in your device manager
    b. `num_scans = 8;` - Change the variable for the amount of times you want to scan a room
3. Assuming the microcontroller is wired correctly and all settings are changed correctly, follow the circuit schematic in the section below to wire up the stepper motor, motor driver board, and the ToF sensor.
4. Open up the Keil project in the Keil IDE
5. Assuming the correct target settings are configured in Keil the process to start flashing the program onto the microcontroller is by pressing 3 buttons.
    a. Translate → Build → Download

6. One the program has been flashed onto the microcontroller, the reset button must be pressed to clear anything left on the board before. The reset button is located next to the micro USB port.



7. Run the MATLAB script, and it should prompt that it is waiting for data from the microcontroller

8. Once ready, press the PJ0 button to start the system. The motor will start spinning and taking distance measurements. Once one full rotation is complete, the motor will reset to its starting position, and you will be able to take a step to take the next scan. Any time you want to pause/stop the system, PJ0 must be pressed again. To fully restart the system the reset button must be pressed.

9. MATLAB should be printing out each step and the distance measurement and will keep track of how many scans are completed and how many are left
10. After all scans are completed a 3D map will be plotted of the scanned environment and the reset button must be pushed to stop the system.

Scanned Location

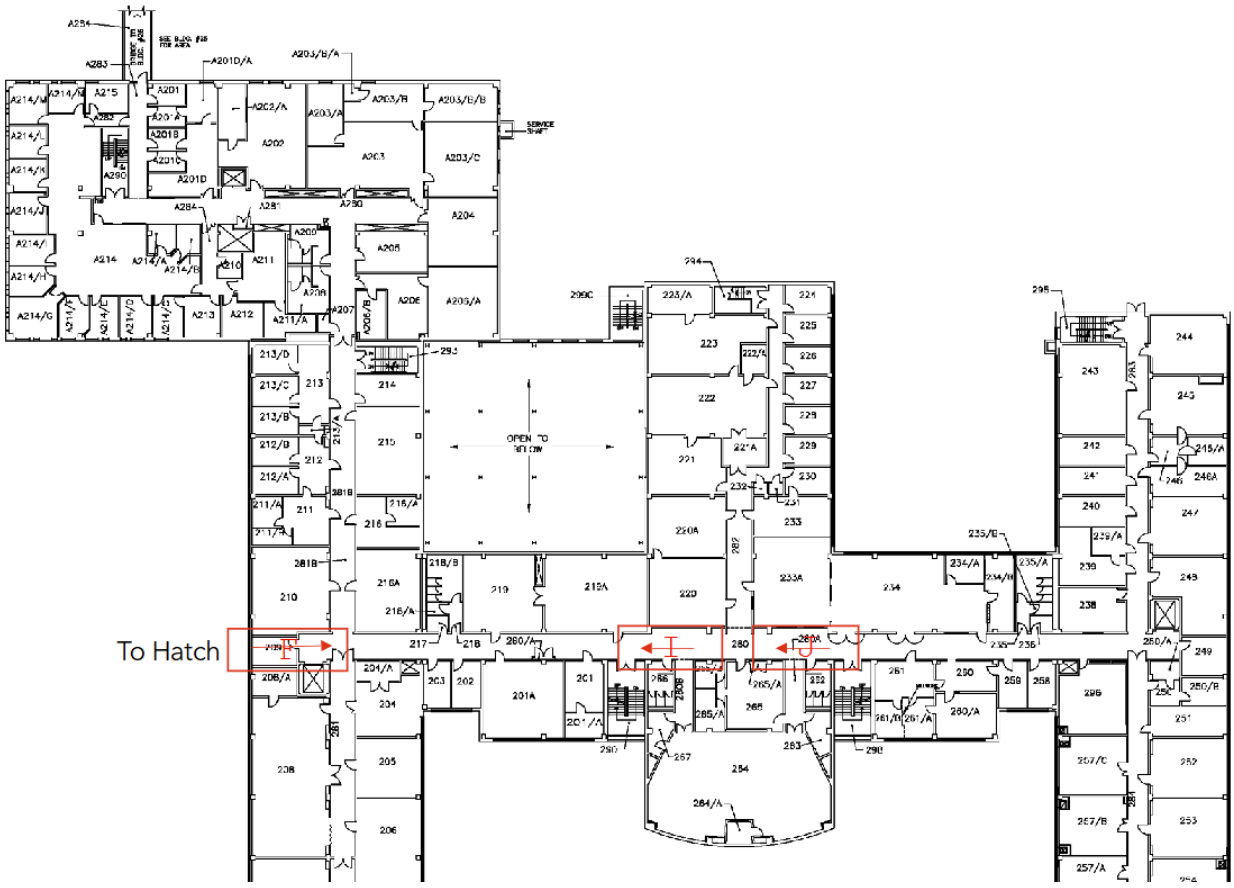


Figure 4: Floor Layout of Building of Scanning Location (Location I)

Reference Images

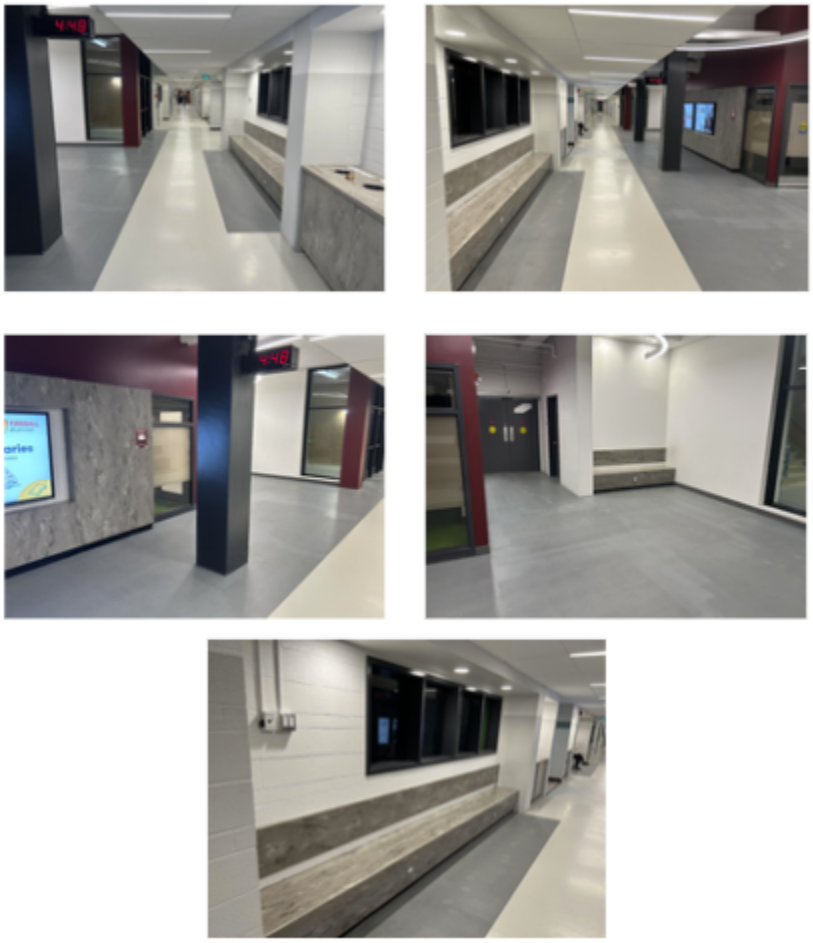


Figure 5: Scanned Location

Expected Output

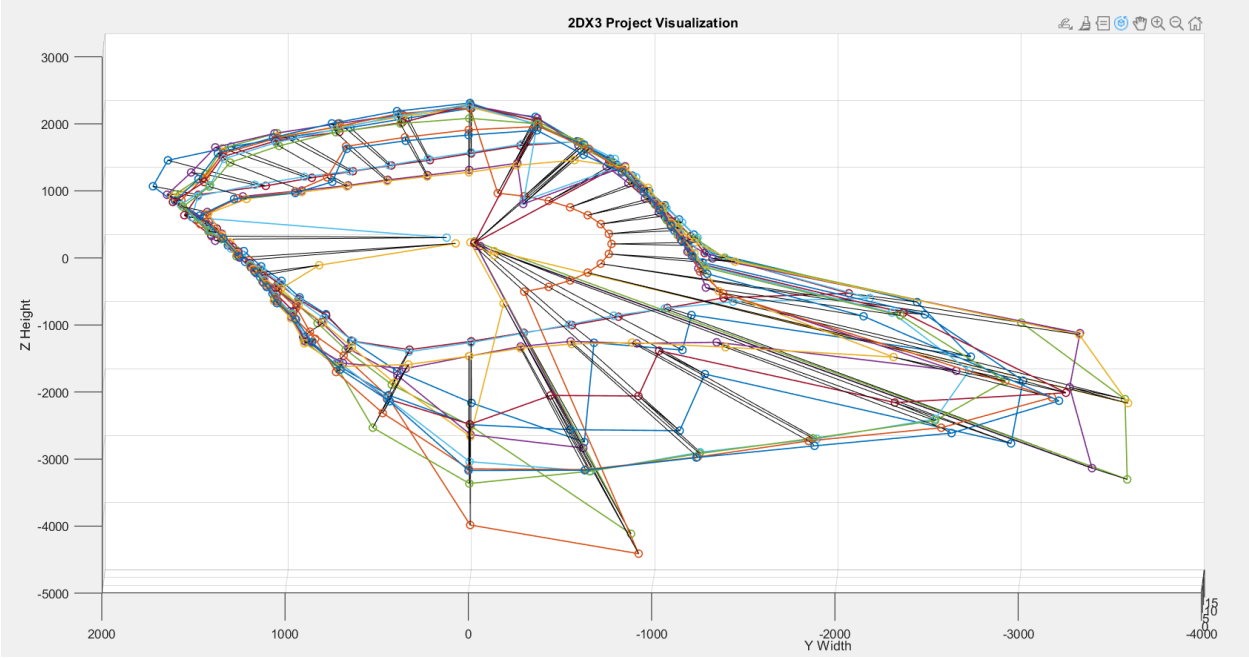


Figure 6: 3D Scan View 1

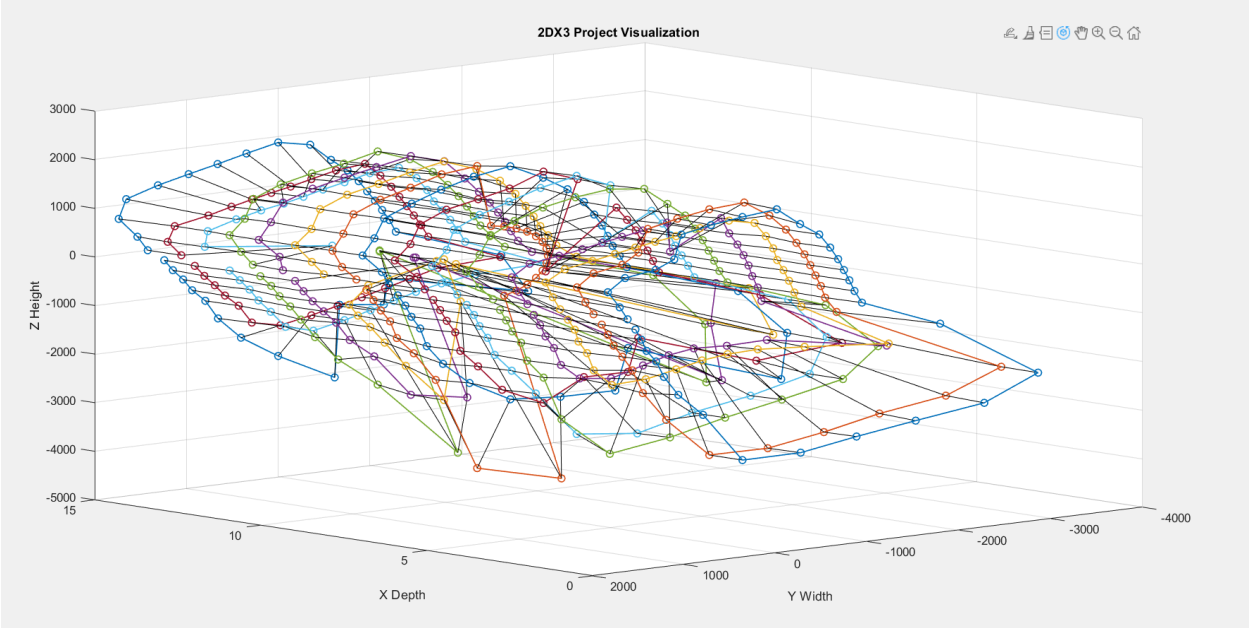


Figure 7: 3D Scan View 2

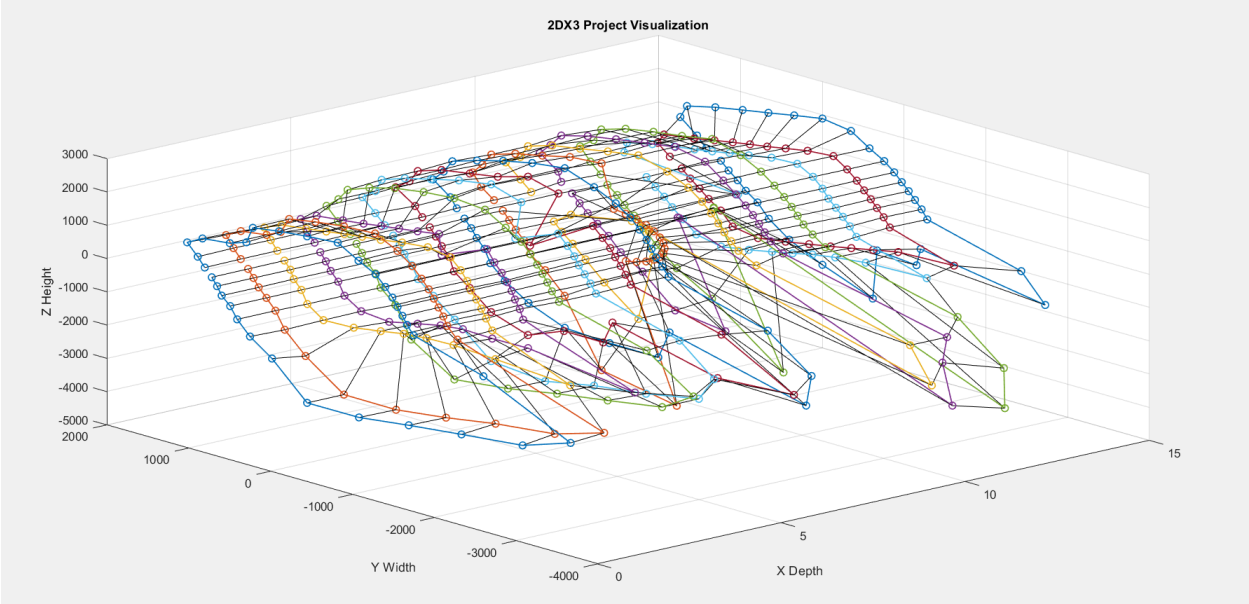


Figure 8: 3D Scan View 3

Limitations

This system is a low-cost and flexible alternative to industrial LIDAR, but it has some hardware limitations.

Sensor (VL53L1X) Limitations

- The VL53L1X has a maximum measurable range of 4 meters at 30 Hz
- The sensor's performance is also impacted by lighting conditions, with noticeable variations between different environments
- Quantization error, based on the 16-bit output resolution and 4000 mm range, can be calculated as:

$$Max\ Quantization\ Error\ = \frac{4000\ mm}{2^{16}}\ =\ 0.061$$

- Since the ToF sensor must stop scanning while capturing data, the timing budget directly affects the stepper motor's movement speed

Stepper Motor Limitations

- A minimum delay of 2 ms is required between motor steps, limiting overall scanning speed
- Due to mounting constraints, the motor must rotate 360° clockwise and then counterclockwise to prevent wire tangling

- Extended operation can cause the stepper motor to overheat, potentially affecting reliability

Communication Constraints

- UART was used to transmit distance measurements to the PC. The configured baud rate was 128000 bps, verified via Windows Device Manager
- The maximum supported UART rate for the MSP-EXP432E401Y microcontroller is 115.2 kbps in half-duplex mode
- I2C speed creates a bottleneck in data retrieval, especially when combined with the sensor's timing delays and the stepper motor's motion.

Circuit Schematic

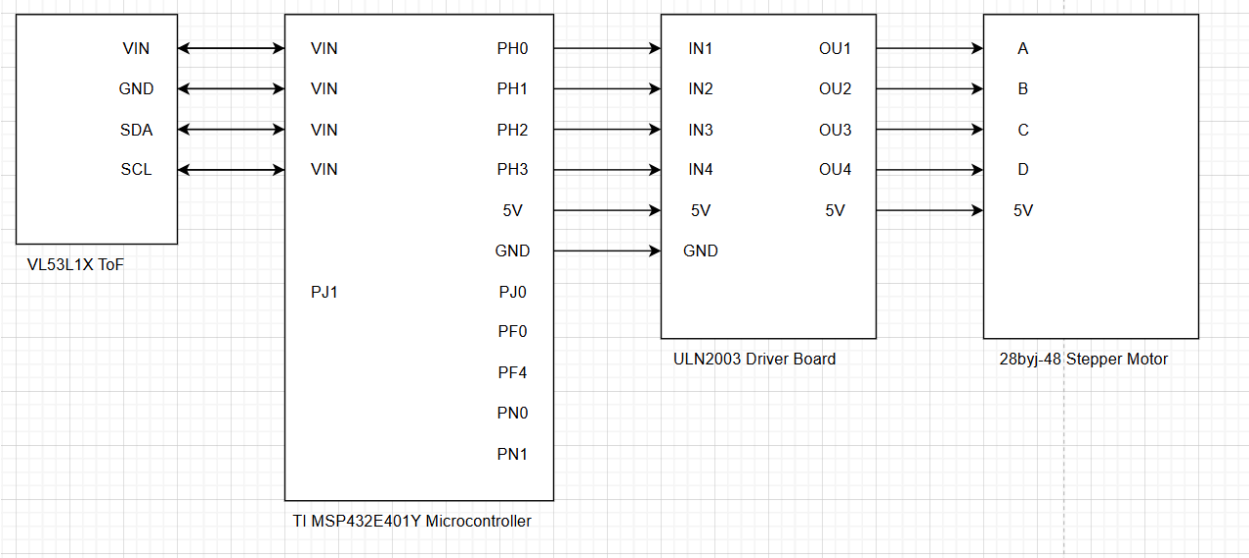


Figure 9: Circuit Schmetaic of Connections

Programming Flow Chart

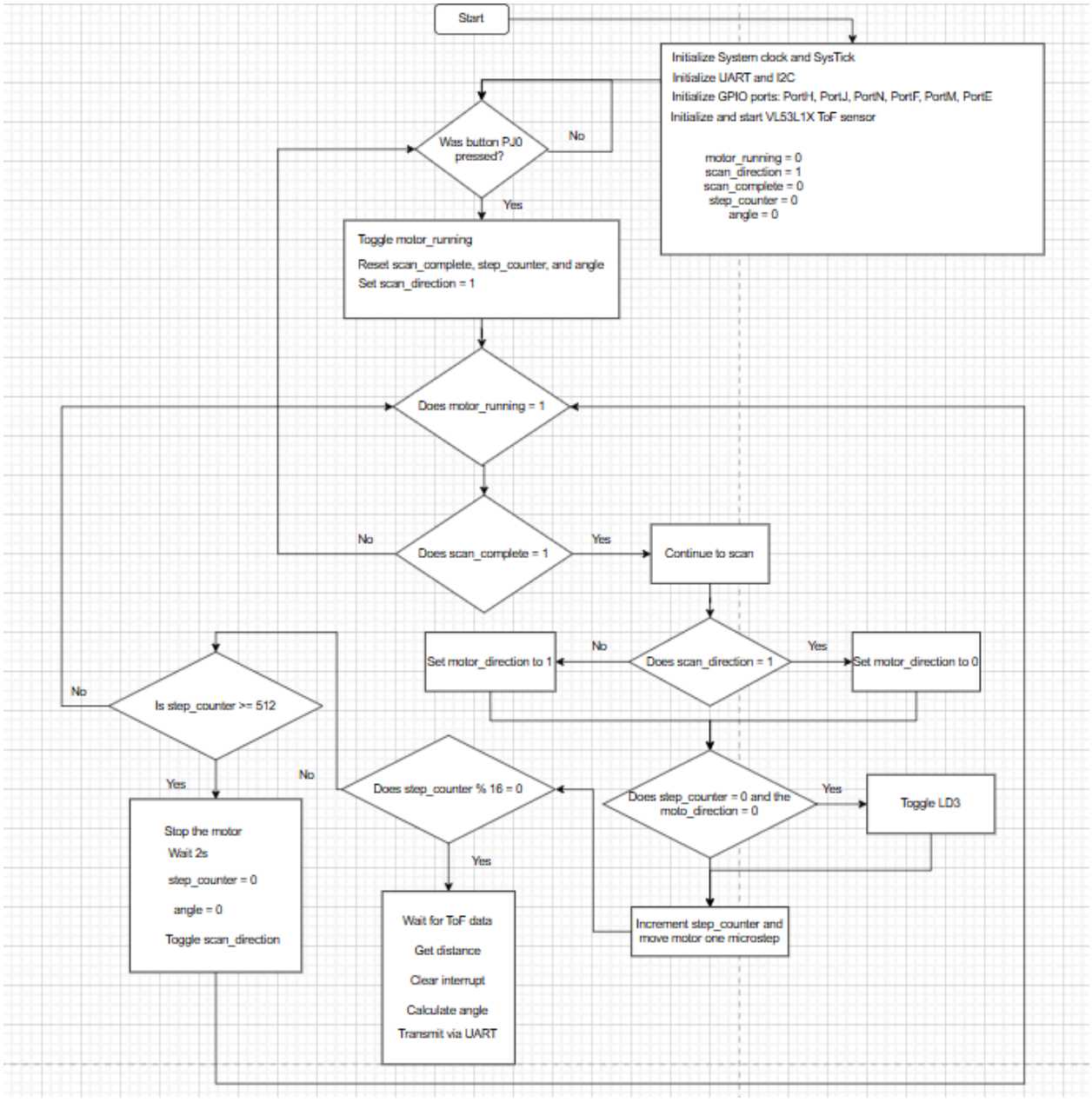


Figure 10: Keil Code Flowchart

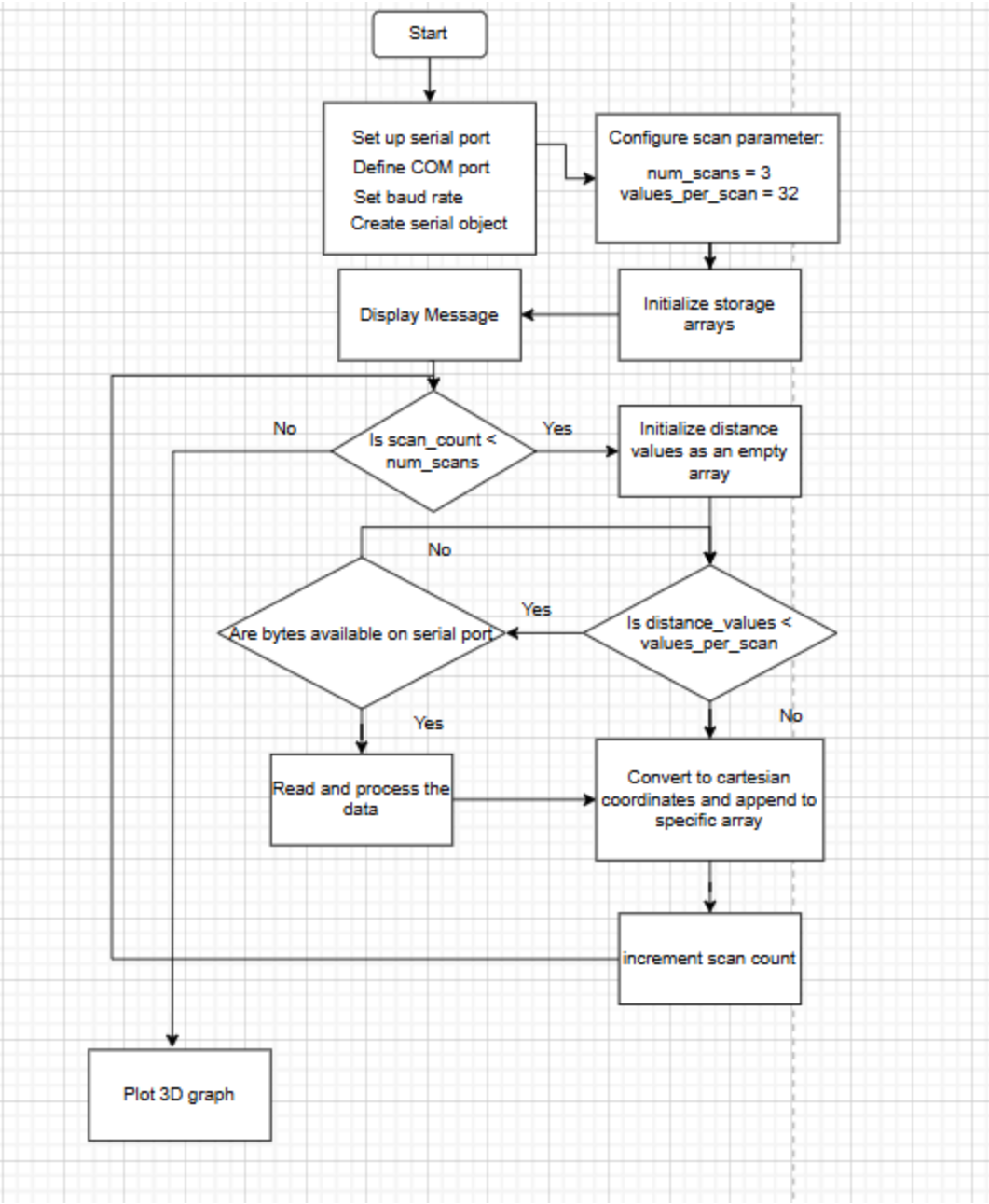


Figure 11: MATLAB Script Flowchart

References

[1] “VL53L1X,” Pololu, https://www.pololu.com/file/0J1506/vl53l1x.pdf [accessed Apr. 3, 2025]

[2] “MSP432E4 SimpleLinkTM Microcontrollers Technical Reference Manual,” Texas Instruments, https://www.ti.com/lit/ug/slau723a/slau723a.pdf [accessed Apr. 3, 2025]