COMPENG 2DX3: Microprocessor Systems Project **- Final Project - Spatial Mapping Using Time of Flight**

Rahim Aziz - L02

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

Department of Electrical and Computer Engineering, McMaster University

Submission Date: April 17th, 2023

Table of Contents

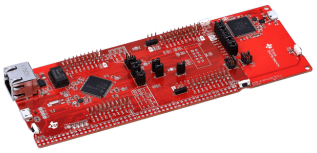
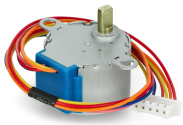
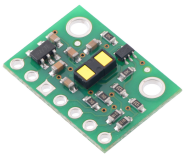
Device Overview

This device is a small-scale, portable, and inexpensive LIDAR (Light Detection and Ranging) system designed for mapping out indoor spaces in 3D. The core of the system is the Texas Instrument MSP-EXP432E401Y microcontroller that works in conjunction with two onboard buttons, the ULN2003 driver board, the 28BYJ-48 stepper motor, and the Pololu VL53L1X time of flight (ToF) sensor.

Overview

- 3D spatial map generated by MATLAB
- ToF sensor communicates live distance measurements with microcontroller via I2C
- Microcontroller communicates live data to PC via UART at 115200 bps baud rate
- Microcontroller powered via 5V micro-usb cable
- Measurement status LED (PF0) and additional status LED (PN0)
- 12 MHz microcontroller bus speed
- 2-onboard push buttons to control stepper motor rotation and to enable ToF data acquisition and transmission
- ADC Sampling Rate: $f_{sample} \geq 2f_{signal}$

Hardware

Hardware	Image	Datasheet	Price
MSP-EXP432E401Y Microcontroller		https://www.ti.com/lit/ug/slau748b/slau748b.pdf	$47.99
28BYJ-48 Stepper Motor		28BYJ-48 – 5V Stepper Motor	$11.73
Pololu VL53L1X TOF		VL53L1X	$18.95 (USD)
ULN2003 Stepper Motor Driver		4 Phase ULN2003 Stepper Motor Driver PCB	$1.35

Software

Software Package	Role	Link
MATLAB	Visualization	MathWorks Account Sign In
Keil MDK (C)	Microcontroller functions (Communication, GPIO, Interrupts, etc..)	Keil MDK

Features

The performance features of the individual parts can be seen listed below

MSP-EXP432E401Y Microcontroller

- ARM Cortex - M4 32 bit CPU (Supports floating point operations)
- Maximum CPU clock frequency 120 MHz
- Bus speed for this design 12 MHz
- JTAG, Micro USB, Ethernet ports to transfer data
- 2 User buttons featuring debouncing & pull up/down resistor capabilities
- 4 user LEDs
- 1 MB of flash, 256kB of SRAM
- I2C modules with high-speed mode support
- UART communication capabilities operating at 115200 bps for this design
- GPIO pins that support interrupts, PWM, and ADC

28BYJ-48 Stepper Motor

- Unipolar 4 - phase stepper motor
- 5.625°/64 stride angle
- Activated in full steps for this design
- For this project a 2.8125°/128 stride angle is used
- 5 - 12 VDC operating voltage

ULN2003 Stepper Motor Driver

- 5-12 V power supply
- Onboard 4-way signal lights
- Powered by Texas Instruments ULN2003AN IC

Pololu VL53L1X TOF

- I2C communication protocol at 100kbps
- Up to 27° FOV available on the distance sensor
- Operating Voltage: 2.6V - 5.5V
- Emitter: 940 nm invisible Class 1 VCSEL
- Three distance mode options; Short: Up to ~130 cm, Medium: Up to ~300 cm, Long: Up to ~ 400 cm

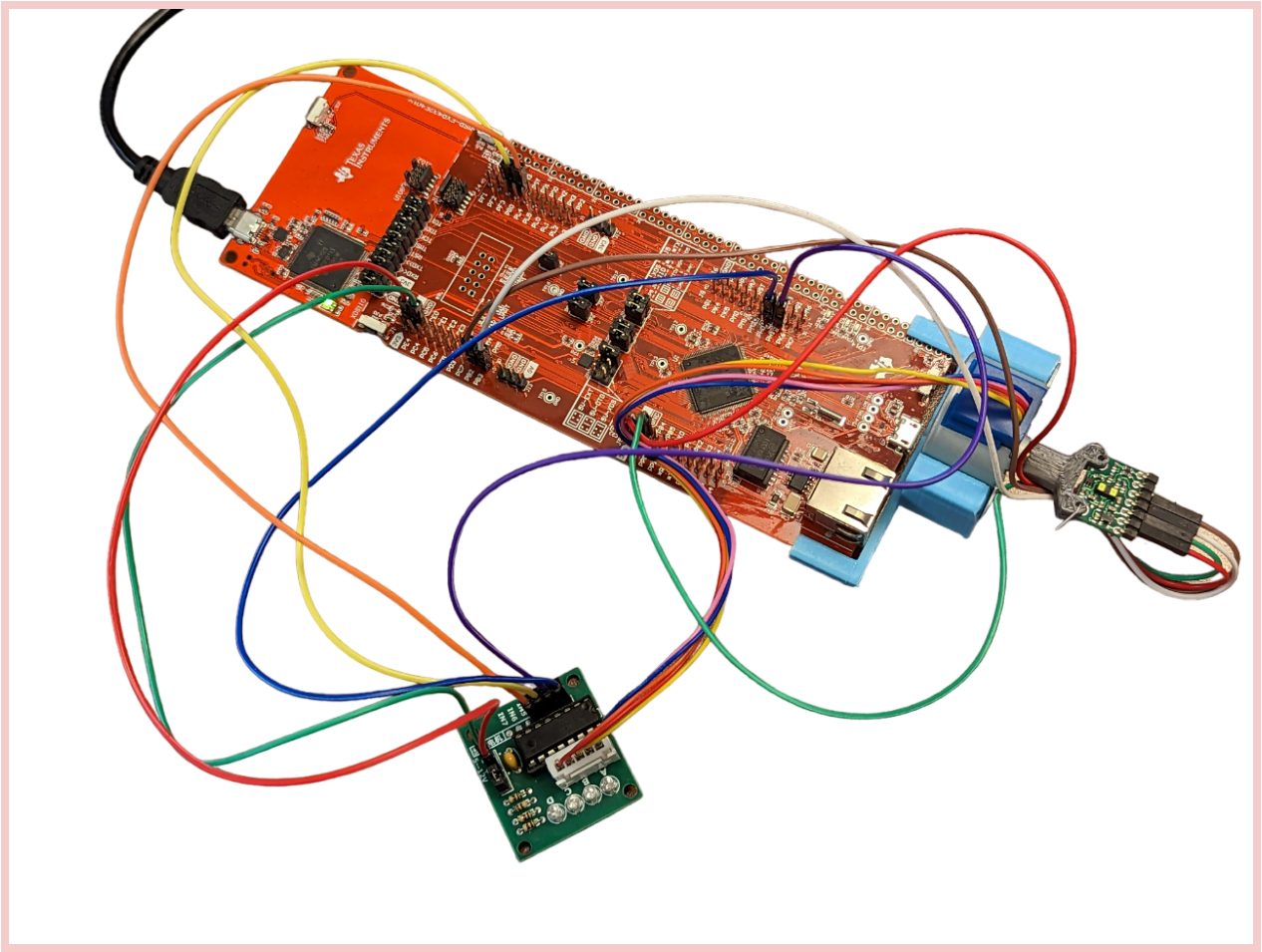


Figure 1: Physical implementation of the LIDAR system

General Description

The primary goal of this device is to create a low-cost and portable alternative to commercial LIDAR (Light Detection and Ranging) systems for indoor purposes.

This device is a 3D LIDAR system composed of the MSP-EXP432E401Y microcontroller, 28BYJ-48 stepper motor, Pololu VL53L1X time of flight sensor, and the ULN2003 stepper motor driver. The MSP432E501Y microcontroller is the heart of the system and interfaces with all the aforementioned peripherals. The MPS432E501Y is equipped with a 120MHz, 32-bit ARM Cortex M4F CPU, 1MB of flash, 256kB of SRAM, and floating point capabilities. For the purposes of this system, the bus speed was set to 12MHz. This system begins by a user pressing the onboard button, PJ0, to enable data acquisition and transmission, followed by pressing the onboard button, PJ1, to begin the rotation of the stepper motor. Both buttons utilize GPIO interrupts to call the desired functions. When PJ0 is pressed, the LED D2 (PN0) is toggled on to indicate that the system is ready to acquire and transmit data. When PJ1 is pressed, it flashes the LED D4 (PF0) every time a measurement is acquired and transmitted.

The 28BYJ-48 is a 4 phase unipolar stepper motor that receives power from the ULN2003 stepper motor driver. The stepper motor ist triggered by the onboard button, PJ1. This stepper motor operates at 5V, and has a stride angle of 5.625°/64 steps. For the purposes of this system, the stepper motor rotates a full rotation (360°) in 2.8125° increments, stopping a total of 128 times for each measurement acquisition.

The Pololu VL53L1X ToF is a laser ranging sensor that utilizes a 940 nm Class 1 laser and a SPAD (single photon avalanche diode) receiving array. This sensor operates between 2.6V - 5V and is capable of detecting measurements up to 4 m. Within this system design, the sensor is enabled by button PJ1 and acquires a distance measurement for every 2.8125° rotation of the stepper motor. The ToF is receiving analog signals, however to manipulate the data the system requires a digital signal. The analog to digital conversion process occurs all aboard the VL53L1X ToF. This process includes transduction, conditioning, and finally analog to digital conversion, resulting in a digital representation of the collected data. Finally, the sensor communicates with the microcontroller via the I2C serial communication protocol. The SDA pin represents the I2C data line and the SCL pin represents the I2C clock line. In this project the ToF transmits the range status, distance, signal rate, ambient rate, and SPAD number back to the microcontroller.

Once each measurement is acquired, it is transmitted via UART to a PC for the visualization process. The parameters transmitted include the range status, distance, stepper motor angle, depth, and SPAD number. The visualization program written in MATLAB receives the above values through one of the computer's communications ports and parses the incoming serial data. MATLAB then converts the polar coordinates (distance, radius, angle) to cartesian coordinates (x,y,z) and then proceeds to plot the data on a 3D scatter plot connecting the appropriate points within the same depth plane.

Block Diagram

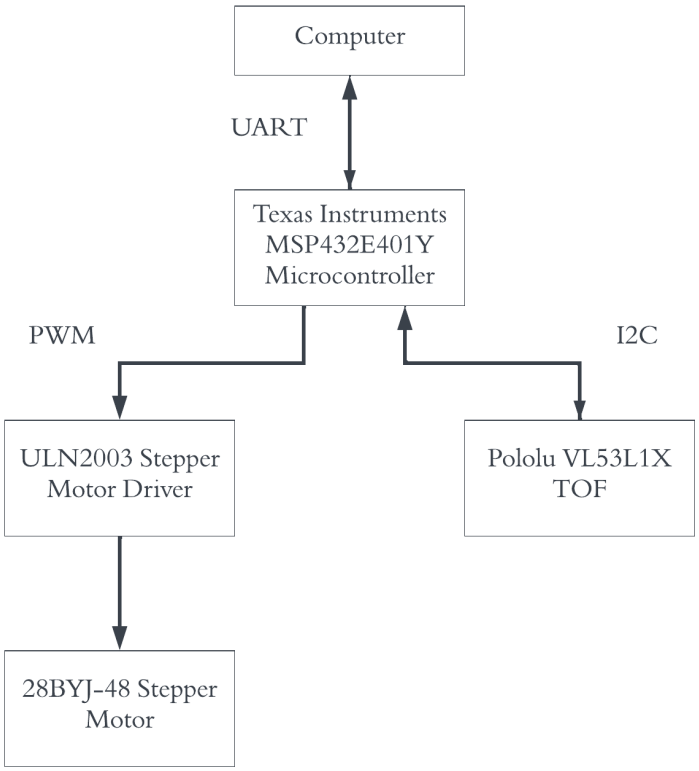


Figure 1: Component block diagram

Data Flow Graph

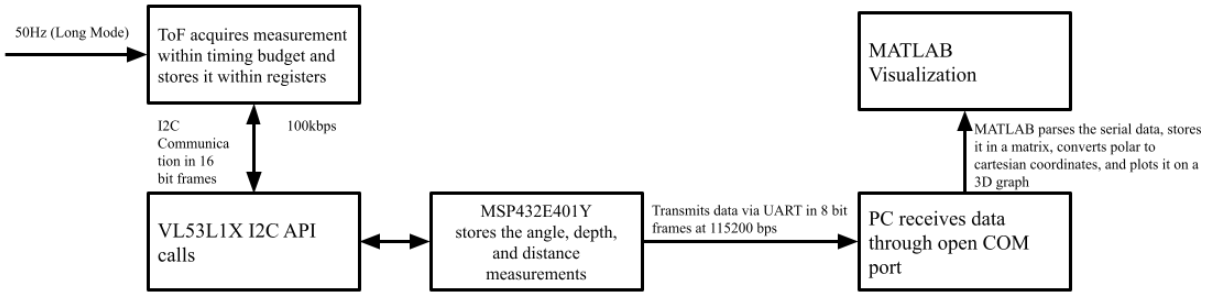


Figure 2: Data Flow Graph

Device Characteristics Table

Table 1: Component power requirements

Component	Power Requirement (Voltage)
MSP-EXP432E401Y Microcontroller	+5V Via Micro USB cable
ULN2003 Stepper Motor Driver	+5V from microcontroller
28BYJ-48 Stepper Motor	+5V from ULN2003 Stepper Motor Driver
Pololu VL53L1X TOF	+3.3V from microcontroller (Can operate from +2.6V to +5V)

Table 2: MSP-EXP432E401Y Pinout descriptions

MSP-EXP432E401Y Pin	Signal Description	Description
GPIO [H0:H3]	PWM	Output for Stepper Motor Driver
GPIO Port J [J0:J1]	GPIO Interrupt	GPIO button that throws interrupts (J0 - For stepper rotation, J1 - for enabling measurements & transmissions)
GPIO B2	SCL	I2C Clock line
GPIO B3	SDA	I2C Data line
MicroUSB	UART	Outputs data to PC through UART protocol
GPIO E0	PWM	Used for bus speed verification using SYSTICK();
GPIO F0	Toggle/Flash	Onboard LED D4
GPIO N0	Toggle/Flash	Onboard LED D2

Table 3: Communication protocol parameters

Parameter	Value
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	100 kbps (ToF supports up to 400 kbps)
I2C address VL53L1X	0x29
UART Frame	1 start bit, 8 data bits, 1 stop bit (No parity)
UART Baud Rate	115200 bps

Table 4: Timing parameters

Parameter	Value
MSP-EXP432E401Y Bus speed	12MHz
Inter Measurement Delay	100ms
Measurement Time Budget	200ms

Detailed Description

Distance Measurement

The apparatus utilized to perform distance measurements is the VL53L1X Time-of-Flight distance sensor. The device is composed of two electronic pads, a 940 nm (Class 1) laser emitter and a SPAD receiver array. The process for acquiring a distance measurement begins by emitting an infrared laser, which bounces off local surfaces and returns to the receiver array. The SPAD (single photon avalanche diode) receiver functions very similarly to a traditional diode. The SPAD is a pn-junction which is subjected to a high reverse voltage that is extremely close to the breakdown voltage. When a photon enters this region, it creates an electron-hole pair. The reverse bias voltage creates an electric field which accelerates the electron-hole pair to the corresponding region of the diode. If enough energy from the photon is imparted, it can create an avalanche effect where the collisions propagate creating more electron-hole pairs and thus create a current.

Utilizing the above principle, the ToF sensor is able to calculate the distance (in millimetres) using the known velocity of the emitted photons and the time delay sensed between the emission and reception of those photons. The equation used to determine the distance is as follows: $\Delta d = c \bullet \frac{\Delta t}{2}$. In this equation c represents the velocity of light/photons (300,000 km/s) emitted by the device. In this equation Δt represents the difference in time between the emission of the photons and the collision of photons with the SPAD. The time difference is divided by 2 because Δt accounts for the time that light takes to travel to the object and back. This data is then transmitted to the microcontroller through the I2C serial communication protocol.

Upon booting up the microcontroller, the Keil C code stored in the flash memory is instantly run. This code begins by initializing the GPIO pins, LED's, interrupts, communication protocols, and the ToF. Upon initialization, the ToF clears local interrupts, sets its distance mode to long, initializes its timing budget and sets the inter-measurement delay. Now the ToF is prepared to collect distance measurements with the specified parameters to suit this project. Now, the microcontroller waits for specific triggers/flag variables through an interrupts method. These interrupts are triggered by 2 onboard push buttons. The onboard button, PJ0, is used to enable data acquisition and transmission by toggling a global flag variable '`scan_enable`'. This button also toggles the LED D2 (PN0) to ensure the user knows the system is primed for collecting and transmitting data. The onboard button, PJ1, is used to begin a single full rotation (360°) of the stepper motor by toggling a global flag variable '`rotate`'.

The above ToF sensor is rotated 360 degrees to capture the entire surrounding space using a mount that connects it to the 28BYJ-48 stepper motor. The ToF samples measurements along the yz-plane. The sensor begins in a position such that the sensor is facing upwards. The stepper motor rotates its axel when the inner coils are energized in the appropriate sequence of full steps. Due to the inner gear ratio of the stepper motor, it has a stride angle of 5.625°/64 steps. Therefore, to rotate a full revolution (360°) it requires the stepper motor to perform a total of 512 steps. For the purposes of this design, each measurement was taken every (2.8125°) to ensure a more precise 3D reconstruction. The above rotations are controlled by the function 'stepperCounter()'. This function begins by polling a distance measurement

if the appropriate flag variable is set. Then it rotates the stepper motor and increments the angle counter. Then, the function checks if the stepper motor has rotated 2.8125°, by applying a modulo 4 to the angle counter. This was calculated by knowing that 512 steps corresponds to a 360° rotation. Therefore, a single step corresponds to 0.703125°. From this information we can calculate the corresponding step count to stop on which turns out to be every 4 steps. ($\frac{0.703125°}{1\ step} = \frac{2.8125°}{x\ steps} \Rightarrow x\ steps\ =\ 2.8125/0.703125\ =\ 4\ steps$). Therefore, every 4 steps/2.8125° the stepper motor stops rotating, and toggles the '`scan_enable`' variable to true. Lastly, the function checks that the angle has reached a full 360° rotation. If this is true, then the stepper motor will reset the angle count, increase the depth(displacement), and prepare for the motor to rotate in the opposing direction upon the user's input. The purpose of rotating in the opposite direction for the next set of measurements is to ensure the ToF wires do not get tangled and interfere with the measured data.

Once the global '`enable_scan`' variable has been set to true, the scanning function will be called within the '`stepperCounter()`' function every 2.8125°. This function polls the ToF and waits for new incoming data measurements. These measurements are then transmitted over I2C to the microcontroller. These measurements include the RangeStatus, Distance, step, depth, and SPAD number. The variables step, and depth are variables that are manually updated by the Keil program, while the other variables are polled from the ToF's registers using the VL53L1X's integrated C API calls. The step count is used to keep track of the angle of the measurement because it can be converted to an angle measurement by simply multiplying it by $0.703125°/step$. Lastly, the depth variable is incremented by 1 unit (In this design 1 meter) within the Keil code to manually simulate the increasing depth measurement of a 3D space. After each measurement, the ToF then transmits the aforementioned variables to the PC via the UART communication protocol. UART communication between the microcontroller and PC is accomplished through a micro USB connection.

Visualization

The visualization process is all handled by a single MATLAB program. This program begins by initializing some variables related to the total number of depth measurements. For the purposes of this design, since each measurement is every 2.8125°, then the device is going to take 128 measurements for a single full rotation. Next, the program opens the desired communication port to receive data transmitted by the microcontroller. The program reads all the incoming serial data until we have reached the maximum number of points that is fixed by the desired number of depth measurements. Then every incoming serial string is parsed using a function which separates the data at each comma delimiter. The angle variable is calculated by using the step count passed by the microcontroller and multiplying it by the degree per step count unit. The resulting equation is: $angle\ =\ step\ \bullet\ \frac{0.703125°}{step}$. The data is then stored and returned as a 1x3 matrix. The key variables (distance, angle, and depth) are then appended to a larger matrix which will hold all the polar coordinates of every measurement recorded. To plot the data in MATLAB, it requires the coordinates to be in cartesian form. Traditionally, to convert the coordinates to cartesian form from polar we would use some foundational trigonometry identities and the following formulas; $x\ =\ distance\ \times sin(angle)$, $y\ =\ distance\ \times cos(angle)$, $z\ =\ depth$. Luckily, MATLAB has a built-in function for converting polar coordinates to cartesian coordinates.

Once the microcontroller has transmitted all the data for every depth, MATLAB will commence the visualization process independently.

The process begins by converting the polar coordinates to cartesian coordinates and storing it in a temporary matrix. Then every point in this matrix is plotted on a 3D scatter plot. The last part of the visualization process involves connecting all adjacent points in the same depth (displacement) plane (ring) as well as connecting all adjacent depth planes with lines. The first process is done by looping through all the measurements grouped by the same depth (displacement) and creating an offset variable which points to the adjacent point. Then, utilizing MATLAB's 3D plotting functionality, the program can draw a line connecting these adjacent points. The last step involves looping through every point grouped by the same depth and connecting it to the corresponding point in the next depth plane with a line. This process is also accomplished utilizing an offset variable which points to the same point in the next depth. Once this is completed, MATLAB will plot the entire graph and label all the axes.

Application Example

This section of the document will outline an exemplar scan of a 3D space utilizing this LIDAR system. Provided below are the setup instructions for initializing the system and acquiring measurements. Additionally, an expected output created by the system can be used for comparison based on the assigned location for my student number. Based on my student number, the assigned area was location E located inside McMaster's Engineering Technology building (ETB). A map of ETB is provided below along with real life reference photos of the scanned location. The below instructions assume that the necessary software is downloaded and configured for the user's desired system. The required software can be found in the Device Overview section. Additionally, this process assumes that the appropriate steps have been taken to connect the hardware together. For this process, refer to the Circuit Schematic section.

Instructions

1. Connect the TI-EXP432E401Y microcontroller to your PC via a micro USB cable. Determine the COM port that the two devices are communicating with. This port will be defined in Windows 11's device manager and will be labeled as *XDS110 Class Application/User UART (COM #)*.
2. Open the MATLAB script and edit the following parameters based on your use case
    a. `serial_port = serialport(ports(2), 115200, 'Timeout', 120);` - Change the number within `ports(2)` to the number that corresponds with your communication port being used by your computer and the microcontroller. This can be done by running the code once and expanding the variable 'ports' to see what serial ports are active.
    b. `depth_total = 8; % x-axis` - Change this number to correspond with the desired number of depth measurements you would like to take.
    c. `num_measurements = 128;` - Change this number to correspond with the desired number of measurements your ToF will take for a single full rotation. For this application example 128 measurements were taken, meaning a measurement was taken every 2.8125°. Remember, the more measurements you take the more accurate your final reconstruction will be.
3. Assuming your microcontroller is wired correctly as seen in the Circuit Schematic section, you may open the Keil project in the Keil IDE.
4. Assuming the desired target settings are all correctly setup, you can click: *Translate* → *Build* → *Download*

5. Once the project has been flashed onto the board, you may press the reset button located next to the micro USB port. This should run the flashed code. If all the onboard LED's flash, then you can rest assured that the flashing process worked.

6. To enable data acquisition and transmission, press the onboard button PJ0. This should toggle LED D2 on.
7. Go back into the MATLAB script and press the ‘RUN’ button. Nothing should happen because we haven’t begun the stepper motor rotation process
8. To enable a single depth distance measurement (360-degree rotation of the stepper motor) press the onboard button PJ1. You should now see the stepper motor rotating and the onboard LED D4 flashing every time the ToF acquires and transmits a measurement. Additionally, in the MATLAB console you should see measurements being printed in the form of (distance, angle, depth).

9. Once the stepper motor has completed one full rotation, it will stop. You can then proceed to physically move the LIDAR system one depth unit in the desired direction and press the onboard button PJ1 to commence another single depth distance measurement
10. Once the desired number of depths have been measured, MATLAB will automatically generate a 3D scatter plot containing the contours of the scanned area.

Scanned Location

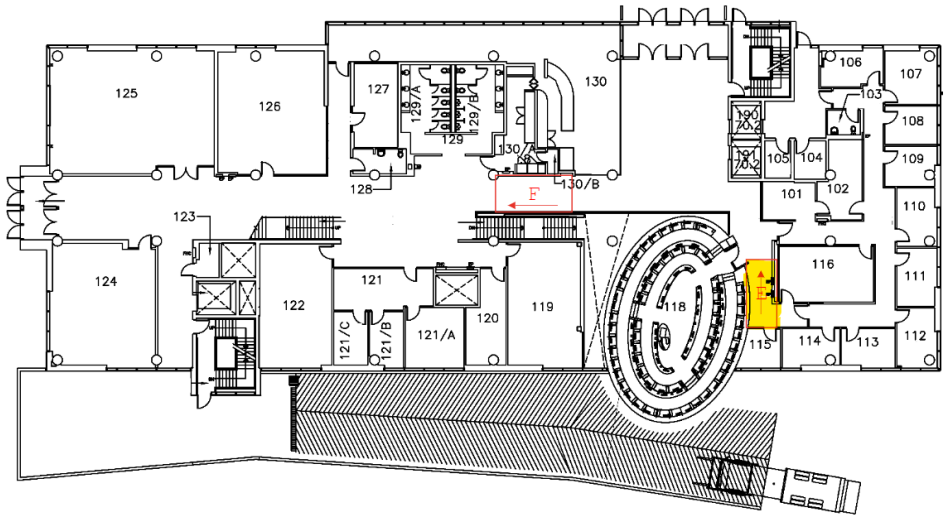


Figure 3: Engineering Technology Building First Floor (ETB)

Reference Images

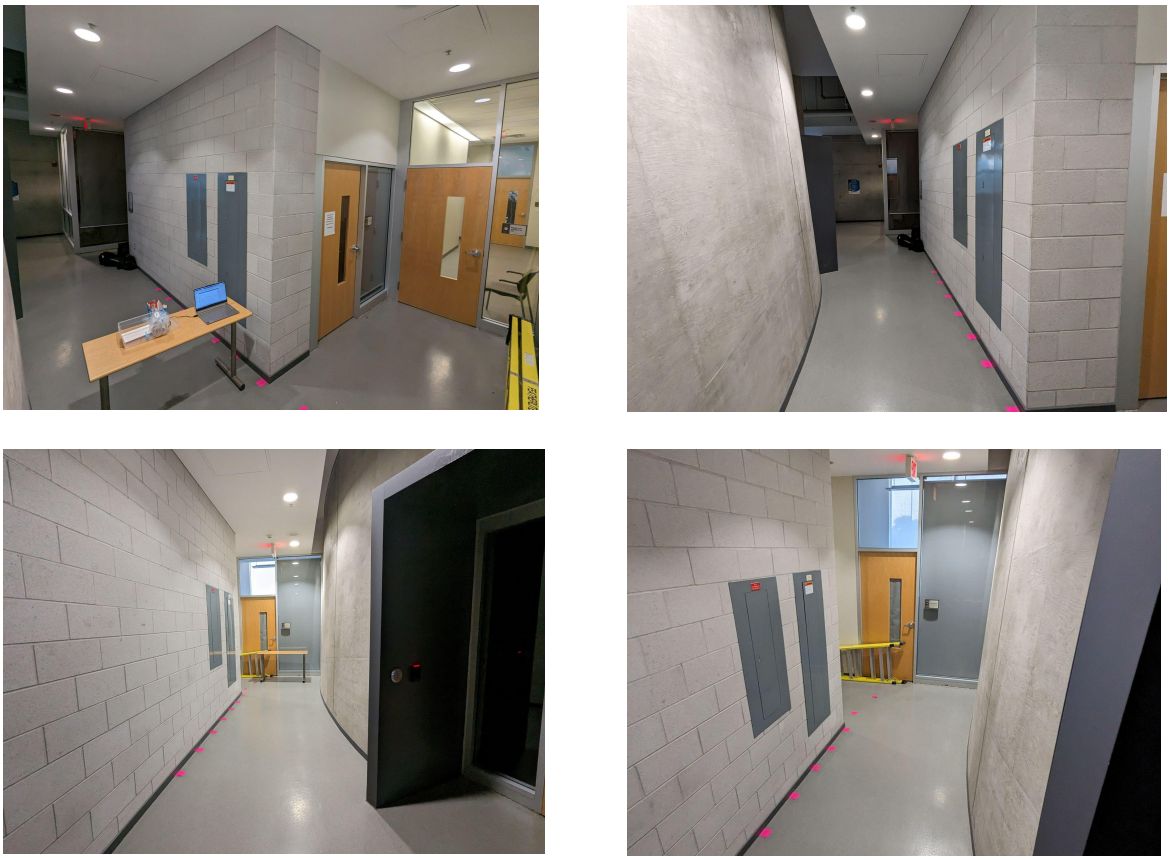


Figure 4: Reference photos of the scanned hallway

Expected Output (MATLAB Visualization)

As seen by the scans below, the system was able to capture the general shape of the hallway as well as various finer details. For example, in figure 5 a) the top view is able to capture the slight curvature on the right of the hallway which appears in real life in the form of the exterior of a lecture hall. Additionally, in 5 c) the isometric view is able to capture an abnormality in the very first depth which so happens to be a ladder which is present in the pictures above.

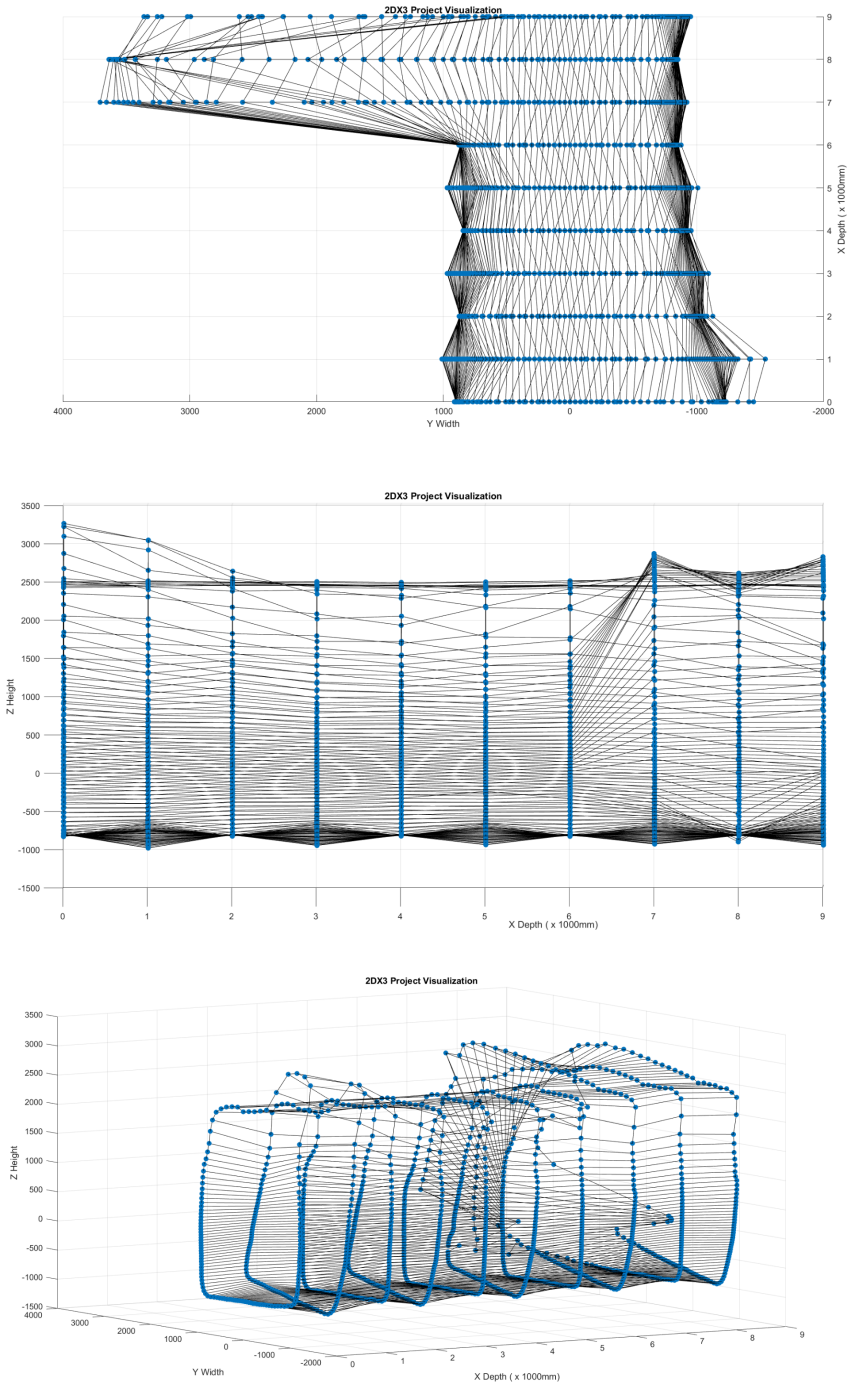


Figure 5 a) Top; b) Side; c) Isometric

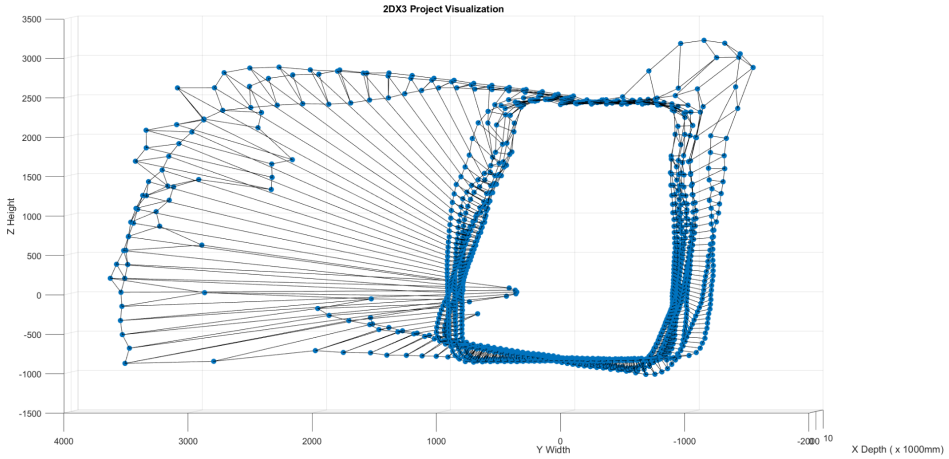


Figure 5 d) Front view

Remember that the measurements taken by the ToF are in the vertical (y-z) plane while the manually incremented depth (displacement) measurement is represented by the x-axis.

Limitations

This system is designed to be a cheap, flexible, and relatively powerful alternative to industrial standard LIDAR systems. Therefore, due to the nature of this design, there are various hardware limitations present in the design.

- Due to the nature of the mounting system and wiring, the stepper motor had to complete full rotations in opposing directions every other step to ensure the ToF wires would not get tangled
- The stepper motor is rated with a +/- 5% non accumulative error for any given step
- After extended periods of usage, the stepper motor was prone to overheating
- The minimum angle per step that the stepper motor could provide was $0.703125°$, therefore the accuracy of the reconstructed scanned space is limited by the aforementioned minimum step angle.
- The stepper motor is a slow component and requires a minimum time delay of 2ms phase activations.
- The VL53L1X is limited to a maximum distance measurement of 4 m at 30Hz
- The presence of varying textured surfaces or non-reflective surfaces can significantly impact the measurement readings of the VL53L1X
- The VL53L1X performance in different lighting conditions can vary a significant amount
- The VL53L1X can utilize a timing budget of 20-1000ms. “Increasing the timing budget increases the maximum distance the device can range and improves the repeatability error.” [1] Additionally, due to this timing budget, the stepper motor is forced to stop its rotation for the equivalent amount of time to ensure that the ToF receives an accurate measurement.
- The maximum quantization error of this system can be described with the following formula $Max\ Quantization\ Error\ = \frac{Maximum\ Reading}{2^{number\ of\ ADC\ bits}}$. From the data sheet it is known that the maximum reading that the ToF sensor can read is 4 m (4000 mm), and the data is stored in a 16 bit format [1]. From the above formula, we can deduce that the maximum quantization error is $Max\ Quantization\ Error\ = \frac{4m}{2^{16}} = \frac{4000mm}{2^{16}} = 0.061035$
- The VL53L1X can only communicate using I2C serial communication protocol, which is limited to a maximum of 400 kHz (Fast Mode). This design utilized I2C at a data transfer rate of 100kbps.
- Therefore, the largest bottlenecks to the system in terms of speed are the stepper motor rotations and the ToF timing budget requirements.
- The maximum UART standard serial communication rate to implement with the pc was 128000 bps. This was verified by opening the Windows 11 ‘Device Manager’ and selecting the drop-down menu for ‘Ports (COM & LPT)’. Then by expanding the ‘Properties’ and then ‘Port Settings’ menu for the ‘XDS110 Class Application/User UART COM5’ object. Within this menu, a selection menu for the ‘Bits per second’ menu appears and lists a maximum serial communication rate of 128000 bps.
- The MSP-EXP432E401Y microcontroller supports a maximum data transmission rate over UART of up to 115.2 kbps half-duplex.[2]

- The MSP-EXP432E401Y microcontroller contains a IEEE754 compliant 32-bit floating point unit. The Cortex-M4F FPU fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations [2]. Despite this floating point precision, it is still more limited than the floating point precision offered by the FPU on all onboard PC/Laptop CPU's. Therefore, all floating point operations were completed on the PC after the data was transmitted. The most intensive floating point calculations required are the trigonometric functions used to convert polar coordinates to cartesian coordinates. The 32-bit FPU onboard the microcontroller, while technically capable of trigonometric calculations, is significantly less precise and would perform significantly slower than a 64-bit FPU.

Circuit Schematic

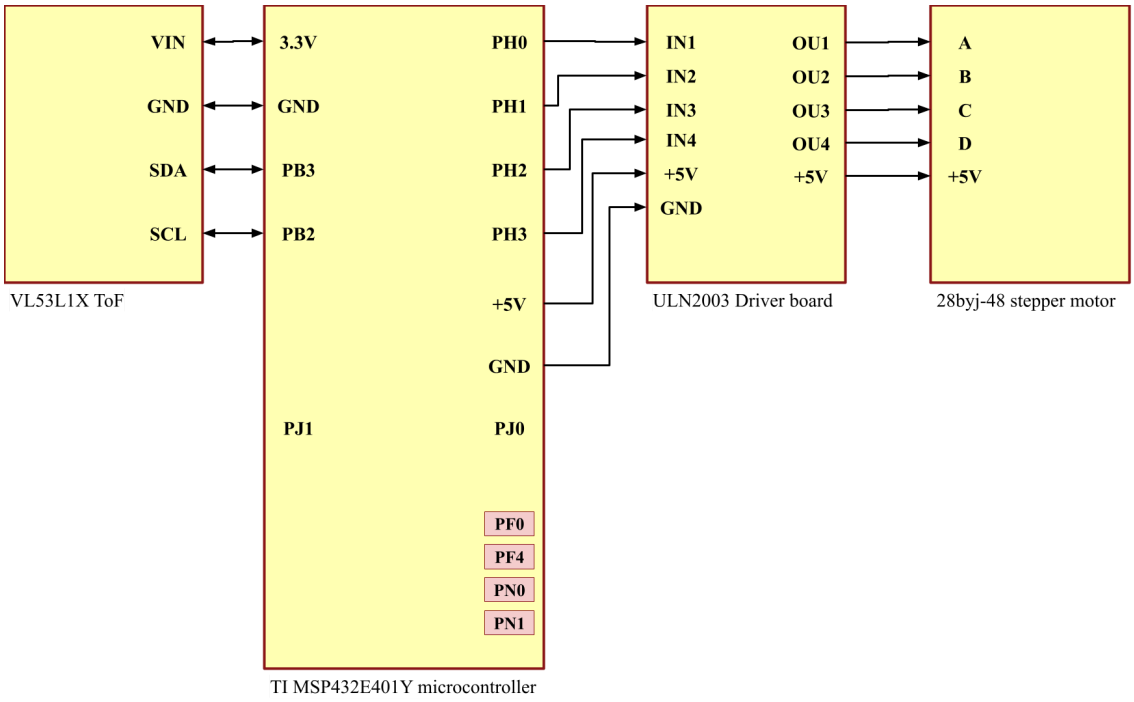


Figure 6: Circuit schematic of the LIDAR system

Programming Logic Flowchart(s)

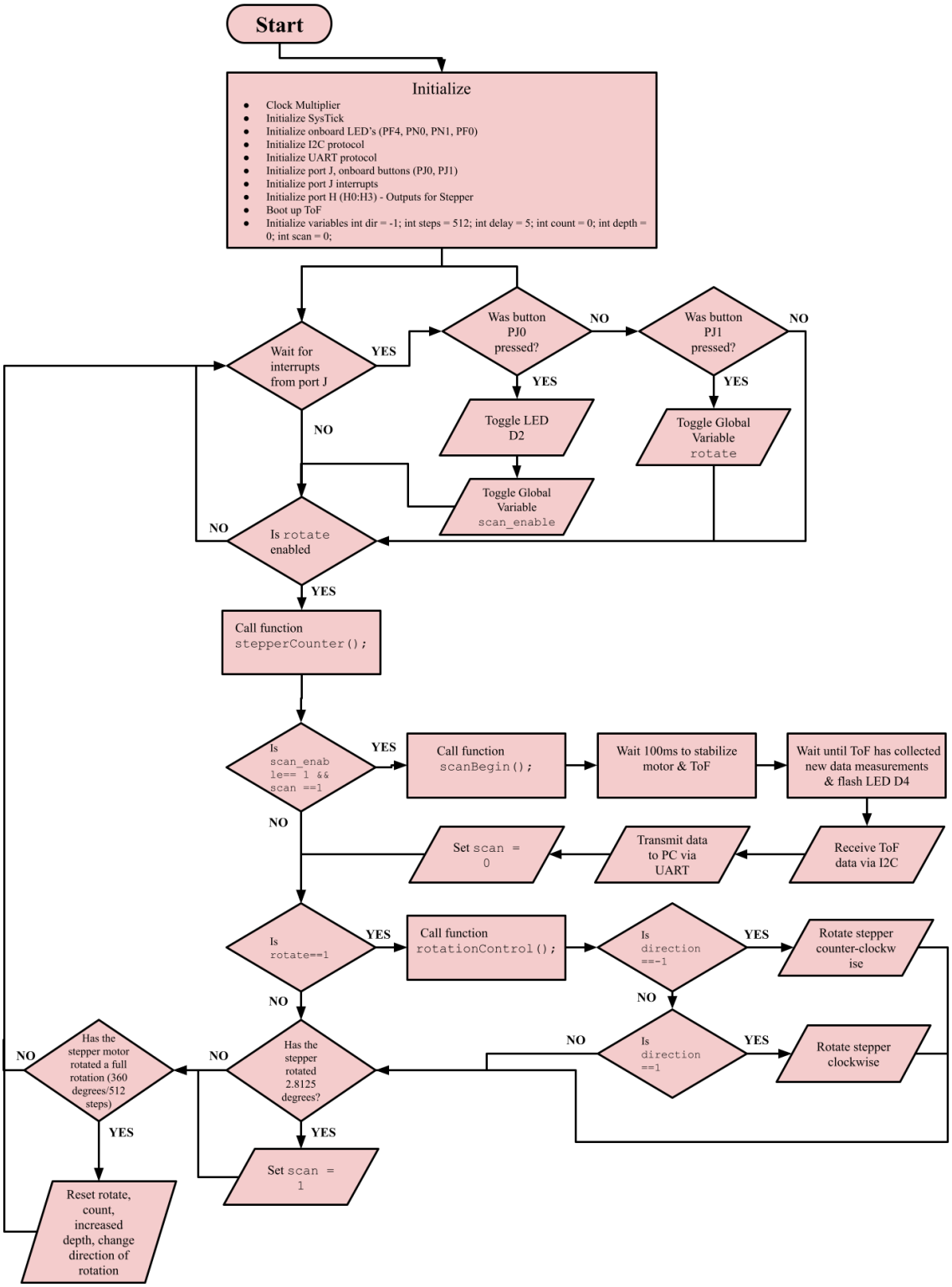


Figure 7: Keil Code flowchart

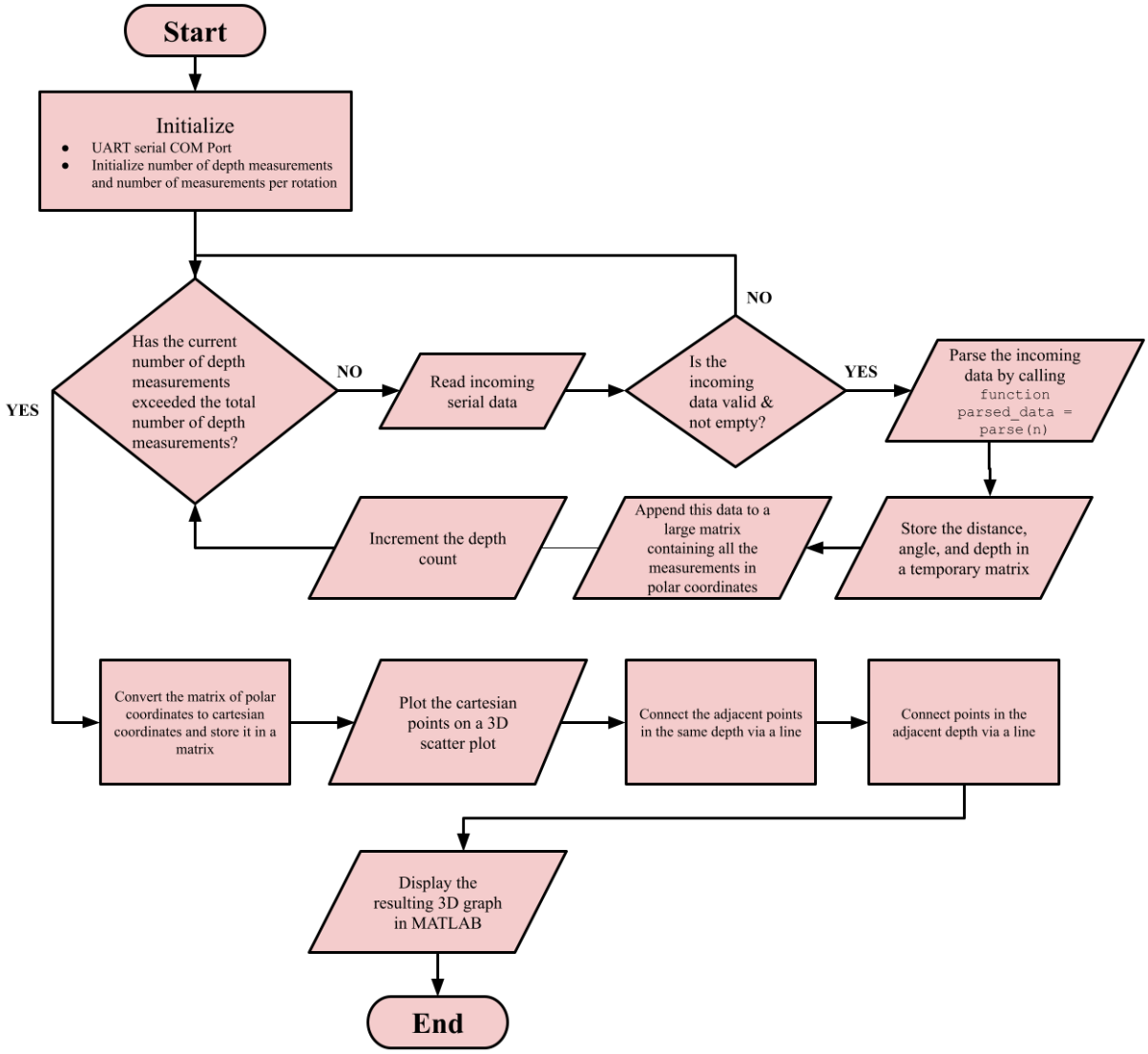


Figure 8: MATLAB visualization flowchart

References

[1] “This is information on a product in full production. VL53L1X A new generation, long distance ranging Time-of-Flight sensor based on ST’s FlightSenseTM technology Datasheet -production data Features,” 2018. Accessed: Apr. 16, 2023. [Online] Available: VL53L1X

[2] “MSP432E4 SimpleLinkTM Microcontrollers Technical Reference Manual,” 2017. Accessed: Apr. 16, 2023. [Online]. Available: MSP432E4 SimpleLink™ Microcontrollers - Technical Reference Manual