COMPENG 2DX3 – Microprocessor Systems Project

Final Project Report

Kun Xing, xingk8, 400460968

Date of Submission: Wednesday April 9, 2025

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

Device Overview

Features

The 2DX3 Final Project LIDAR system integrates several components to enable 3D spatial mapping and visualization. The system's core features include:

1) Integrated LIDAR System:
    - 360-Degree Distance Measurements: The system captures distance data in the x-y plane and rotates 360° to map the surrounding environment.
    - Orthogonal Displacement Samples: Fixed displacement sampling along the z-axis enables 3D mapping when combined with the 360° measurements.
    - Data Communication via USB: Captured data is transmitted to a PC via UART, with Python scripts facilitating communication between the microcontroller and external device.
2) Texas Instruments MSP432E401Y Microcontroller:
    - Cortex-M4F Processor: The microcontroller is powered by a Cortex-M4F processor with a 96 MHz bus speed, ensuring fast processing capabilities for real-time data collection and control of the system's components [1].
    - Operating Voltage: The microcontroller operates within a voltage range of 2.5V to 5.5V, providing flexible integration into various power configurations and ensuring stability across different voltage levels [1].
    - Memory: The MSP432E401Y microcontroller has a 1024KB flash memory, 256KB SRAM, and 6KB EEPROM [1]. This memory ensures smooth execution of the system without overflow or slow access.
    - Serial Communication: Communication with the VL531LX ToF sensor is managed through the I2C interface at a maximum speed of 400 kHz, allowing for quick data transfer between the sensor and microcontroller. Additionally, data is sent to the host PC through UART communication at a baud rate of 115200 BPs.
3) UNL2003 Stepper Motor Controller:
    - Precision Motor Control: The system uses a 28BYJ-48 stepper motor, which has 512 steps per revolution, providing precise control over the sensor's rotational position for 360° data acquisition.
    - Wide Voltage Range: The ULN2003 stepper motor driver supports an operating voltage range of 5V to 12V, allowing for flexibility in power configurations.
    - LED Feedback: The motor controller includes 4 onboard signal LEDs that provide feedback on motor operations.
4) VL53L1X Time-of-Flight (ToF) Sensor:
    - High Accuracy Distance Measurement: The VL53L1X sensor provides accurate distance measurements with an error margin of ±20 mm, making it suitable for precise measuring applications [1]. This sensor is capable of measuring distances up to 4 meters, making it suitable for indoor and smaller mapping applications [2].
    - Infrared Laser Technology: The sensor uses infrared laser pulses to measure distance by calculating the time it takes for the pulse to return after being reflected from the target. The sensor features an emitter that sends out the laser pulse and a receiver that detects the reflected photons, enabling the measurement.

- Operating Voltage: The sensor operates within a voltage range of 2.6V to 3.5V [2].
- Serial Communication: The sensor communicates with MSP432E401Y microcontroller via the I2C interface, allowing for efficient and fast data exchange. The sensor supports a maximum of 400 kHz I2C interface, which enables high-speed data transfer to the microcontroller.

5) Data Communication:
   - I2C for Sensor communication: The sensor data is communicated from the ToF to the microcontroller over I2C.
   - UART for PC Communication: The microcontroller communicates with the PC using UART, at a baud rate of 115200 bps. Data transmission to the PC allows for further processing, including 3D visualization.
   - Python Support for Data Processing: Data is processed and visualized in real-time using the Python library Open3D, which allows for 3D mapping and modelling of the environment.
6) 3D Visualization:
   - Real-Time Graphical Output: The system uses the Open3D Python library to create real-time visualizations of the mapped space. The 3D data is rendered as a model, providing an intuitive view of the environment.

General Description

The 2DX3 LIDAR system integrates several components to capture, process, and communicate 3D spatial data. Central to the system is the VL53L1X Time-of-Flight (ToF) sensor, which uses infrared light emission to measure distances to objects. The sensor emits pulses of infrared laser light (typically around 942 nm) and then measures the time it takes for the light to reflect off a surface and return to the sensor [2].

The distance measurement process begins with the emission of the infrared light pulse by the ToF sensor. The emitted light is reflected back by the target, and the time for the light to travel to the target and return is measured. Using the speed of light, the sensor calculates the distance as shown in Figure 1 [2]. The measurement is divided by 2 to account for the fact that the light travels to the target and then back to the sensor, covering the distance twice. This division ensure that the sensor only calculates the distance one-way from the sensor to the target. The raw data is an analog measurement of the light pulse's travel time, which is then converted into a digital distance value using the sensor's analog-to-digital conversion (ADC). This digital value is then communicated to the MSP432E401Y microcontroller via the I2C interface**.**

The I2C communication protocol allows for efficient and fast data transfer from the ToF sensor to the microcontroller, enabling real-time updates of the distance measurements. The system uses a UART serial communication link to send this data to a host PC for visualization. The PC utilizes Python scripts with the Open3D library to process and display the 3D mapped environment in real-time.

This approach ensures that the system accurately captures data with an error margin of ±20 mm and provides reliable communication between the sensor and system. The combination of infrared laser technology, conversion of light pulses to digital values, and I2C communication makes the 2DX4 LIDAR system reliable for spatial mapping and visualization.

$$Distance = ToF \times Speed\ of\ Light \times \frac{1}{2}$$

Figure 1: ToF distance formula

Block Diagram

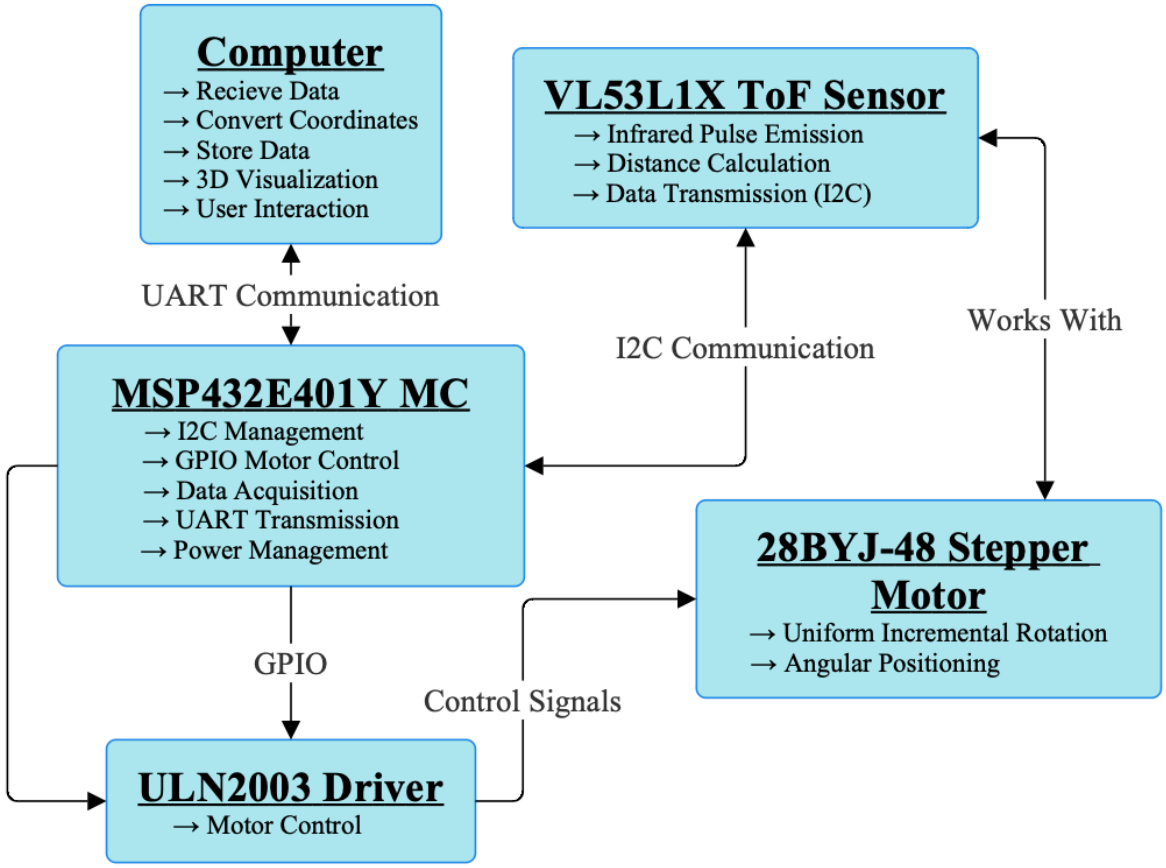

Figure 2: Block Diagram

Device Characteristics Table

MSP432E401Y Microcontroller

Feature	Characteristics
Bus Speed	26 MHz
Serial Port	COM5
Measurement Status GPIO	PF4
UART Tx	PF0
Additional Status	PN0
Baud Rate	115200

UNL2003 Stepper Motor Driver

Feature	Characteristics/Pin
VDD	5V
GND	GND
IN 1	PM0
IN 2	PM1
IN 3	PM2
IN 4	PM3

VL53L1X Time-of-Flight Sensor

Feature	Characteristic/Pin
VDD	–
GND	GND
VIN	3.3V
SCL	PB2
SDA	PB3
GPIO	–
XSHUT	–

Detailed Description

Overview of System Functionality

This LIDAR system utilizes the VL53L1X Time-of-Flight (ToF) sensor to perform precise distance measurements by emitting pulses of infrared light. These pulses travel towards surrounding objects and are reflected back to the sensor's photon-sensitive detector. The fundamental concept behind the system is measuring the time interval between emission and detection of these pulses. By using the speed of light as a reference, the sensor calculates the exact distance to an object, making it highly reliable in many conditions.

The core controller of the system is the Texas Instruments MSP432E401Y microcontroller, which integrates all system operations aside from the 3D visualization. It manages the communication between the sensor and the motor, performs distance data processing, and handles UART communication with the computer.

The system consists of:

1. VL53L1X ToF Sensor: responsible for measuring distance via infrared pulses.
2. MSP432E401Y Microcontroller: controls the sensor and motor, handles communication, and processes data.
3. 28BYJ-48 Stepper Motor: rotates the ToF sensor for full 360° coverage.
4. PC with Python Scripts: processes and visualizes the 3D data generated by the system.

The microcontroller initiates and oversees sensor operation through I2C communication, controls motor rotation through GPIO pins, and communicates collected data to the computer over UART.

Distance Measurement

The LIDAR system operates fundamentally around the VL53L1X Time-of-Flight (ToF) sensor, a precise measurement device for generating spatial data. The sensor's primary function involves emitting infrared laser pulses that travel through the surrounding environment and reflect off objects within its effective range. These returning pulses are captured by the sensor's photon-sensitive detectors. By measuring the elapsed time between pulse emission and detection, the sensor calculates the exact distance in millimeters using the known speed of light as shown in Figure 1 with an example in Figure 3 [2].

Let's say the VL53L1X sensor detects a Time of Flight (ToF) of:

$$ToF = 5.2\,ns = 5.2 \times 10^{-9}\,s$$

Using the speed of light:

$$c = 3 \times 10^{8}\,m/s$$

We apply the formula:

$$Distance = ToF \times c \times \frac{1}{2} = (5.20 \times 10^{-9})(3 \times 10^{8})\left(\frac{1}{2}\right) = 0.78\,m$$

Final Distance = 780 mm

Figure 3: Example calculation

The system is initialized through the Texas Instruments MSP432E401Y microcontroller, which sets up the I2C communication protocol necessary for reliable data exchange with the VL53L1X sensor. After initialization, the sensor remains idle, waiting for user initiation through a polling mechanism rather than interrupt-based control. Specifically, the microcontroller continuously checks the state of a GPIO pin connected to an onboard button (PJ1). This polling-based approach provides simple and direct control over starting and stopping measurement sequences.

Upon activation, the sensor begins continuous ranging operations, coordinating with incremental rotational movements facilitated by a stepper motor. Each rotational increment of 11.25° triggers the microcontroller to poll the sensor repeatedly until a distance measurement is ready. Once the sensor confirms readiness, the microcontroller retrieves the data, temporarily stores it, and transmits it immediately to a connected computer via UART at a baud rate of 115200 bps. The sensor then clears its readiness flag, resetting itself for the next measurement. A complete measurement cycle involves 32 evenly distributed measurements, ensuring a 360° environmental scan. After completing a rotation in the clockwise direction, the stepper motor reverses, returning the sensor to its initial position, readying the system for subsequent measurement cycles.

Stepper Motor Control for 360° Rotation

Precise sensor positioning is a key factor for accurate LIDAR data collection. The system employs a 28BYJ-48 unipolar stepper motor, driven by the ULN2003 driver, to enable high-resolution rotational control. This motor is suited for incremental movement due to its internal gearing and coil-based stepping mechanism. Each step involves selectively energizing specific internal coils in the motor, generating controlled magnetic fields that incrementally rotate the rotor in precise angular steps. Within the code, this is done by setting GPIO ports to '1' to activate the corresponding coils.

The MSP432E401Y microcontroller precisely sequences the energizing of these coils, advancing the motor rotor in increments of 11.25° per measurement step in the clockwise direction. This incremental rotation guarantees that each spatial measurement taken by the sensor is equidistant in angular displacement. The ULN2003 driver acts as the intermediary between the microcontroller signals and the high-current coils of the motor, ensuring reliable, repeatable movements.

The rotation is systematically controlled, starting from an initial position and completing a full revolution of 360°. Upon completing one full revolution, the microcontroller automatically reverses rotation to reset the system back to the initial angular position. This stepper motor approach ensures high repeatability, consistent angular increments, and comprehensive spatial coverage for every LIDAR scan cycle.

Data Collection and Conversion

Collected raw distance measurements are transmitted from the MSP432E401Y microcontroller to a personal computer through UART serial communication. On the computer side, a Python script leverages the PySerial library to handle incoming UART data efficiently. The received data are initially represented in polar coordinates, characterized by the measured distance and its associated angle.

For visualization purposes, the data must be converted from polar coordinates to cartesian coordinates (x, y, z). Polar coordinates describe spatial positions using radial distances and angles, whereas cartesian coordinates define these positions along perpendicular axes, which are ideal for visual representation. The conversion employs trigonometric equations given below:

- x = distance × cos(angle)
- y = distance × sin(angle)
- z = constant values changing with each rotation cycle

Each angle increment of 11.25° is converted to radians to perform these calculations accurately. The z-coordinate remains constant during each rotation cycle, only changing after each complete measurement set to establish multiple horizontal scanning layers. The calculated cartesian points are systematically stored in a structured `.xyz` file, prepared for visualization purposes.

Displacement Measurement

In the current LIDAR implementation, displacement measurements are manually accounted for within the Python script. Specifically, the vertical displacement (z-axis) between each full rotation of measurements is incremented manually after each scanning cycle. After completing one full rotation (32 steps of 11.25°), the z-axis displacement is incremented by a predefined constant value (250 units/millimeters in this case), which is directly reflected in the `.xyz` coordinate data generated by the Python script. This manual method allows for straightforward adjustments to the vertical scanning increments but introduces the possibility of errors due to manual entry and measurement inaccuracies.

Visualization

Once the data points are processed into cartesian coordinates, they are visualized using the Open3D Python library. Open3D facilitates the representation of spatial data, creating 3D models directly from point clouds. The `.xyz` file generated earlier serves as the direct input, loading into Open3D to create a point cloud representation of the scanned environment.

This visualization process involves generating a 3D point cloud where each data point accurately represents its corresponding real-world position. To enhance the visualization, points within individual rotational slices are connected horizontally, and these slices are linked vertically between layers. This interconnected network of points and lines enhances spatial perception, clearly illustrating the scanned environment's geometry.

Open3D provides interactive tools to navigate the generated 3D models. Users can rotate, zoom, and pan the model interactively, providing a detailed examination from any angle or scale. The interactive model helps to quickly identify any inconsistencies, missing data points, or artifacts introduced during data collection, such as mechanical disturbances or environmental interference.

Application Note, Instructions, and Expected Output

Figure 4: Scanning hallway

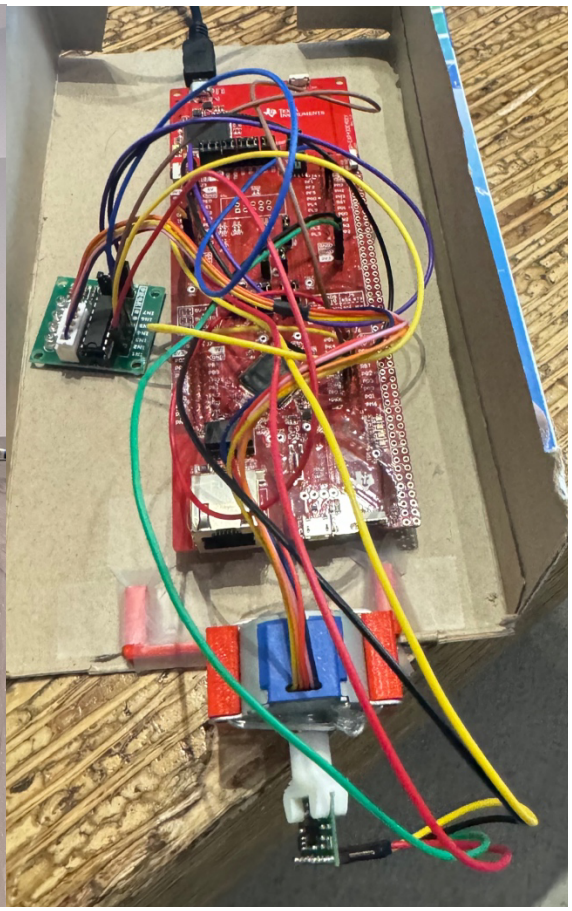

Figure 5: Physical circuit

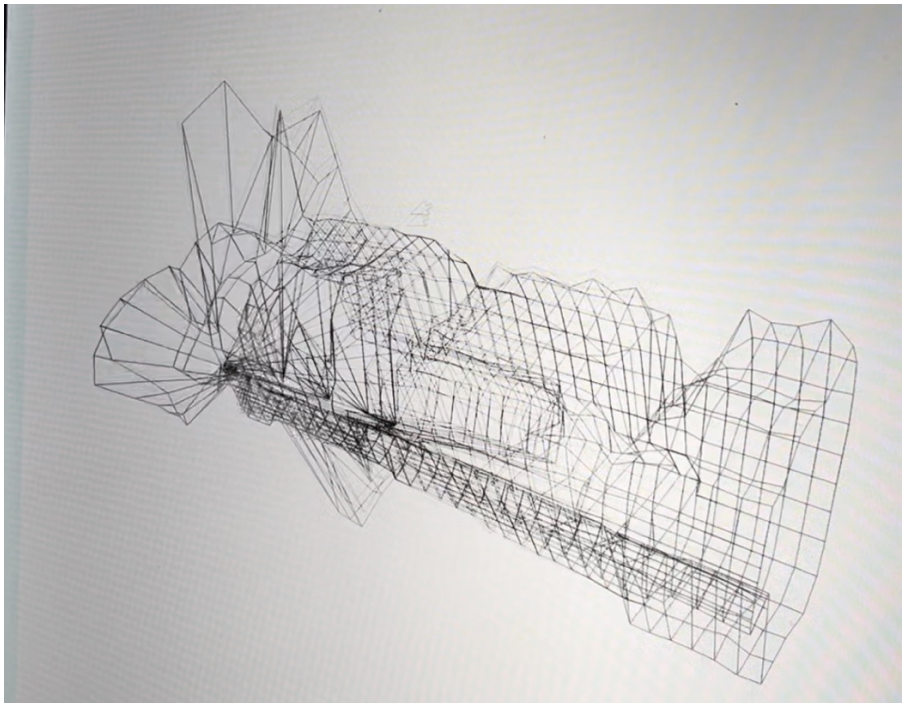

Figure 6: Scanning results

Preparation Checklist: Before beginning, ensure you have the following:

- Uncompressed project files
- A USB cable for board-PC connection

A. Project Initialization

Step 1: Extract Project Contents

- Locate the .zip file provided.
- Extract it to a known path like *C:\Projects\DistanceScanner*.
- Inside, you'll find:
  - Source file and headers for C firmware.
  - Pre-configured Keil project file (*.uvprojx*).
  - Python visualization script.

Step 2: Install Keil IDE

- Visit Keil's ARM page (https://www.keil.arm.com) and down uVision5.
- During install, select support for MSP432E401Y via the Pack Installer.
- Allow all dependencies and drivers to install properly.

Step 3: Launch and Load the Project.

- Double-click the *.uvprojx* file from your extracted folder.
- Ensure compilation is error-free.
- Under "*Options for Target*", set the debugger to CMSIS-DAP.
- Connect your microcontroller and click "*Download*" to flash the firmware.

B. Python Environment Configuration

Step 4: Install Python

- Go to https://www.python.org and install any version of Python 3.10.
- **Note: Tick the checkbox that says "*Add Python to PATH*" during installation.**

Step 5: Install Libraries

- Open the Command Prompt and type: "*pip install pyserial numpy open3d*".

C. Hardware and Script Communication Setup

Step 6: Connect Hardware

- Plug the microcontroller into the computer using a micro-USB cable.

- Open "*Device Manager*" then expand "*Ports (COM & LPT)"* to find your board's COM number.

Step 7: Edit Script Settings

- Open the Python script in your preferred editor.
- Modify the line: "*s = serial.Serial(port'COM5', baudrate = 115200, timeout = 10"*.
  - Replace "*COM5"* with your board's detected port.

D. Executing the Scan & Visualization

Step 8: Beginning Scanning

- Run the script.
- Once prompted, press Enter.
- Then press the onboard button PJ1 to initiate scanning.
- Indicators during scan:
  - LED3 will flash when data is sent.
  - LED4 will blink with each distance measurement.
  - After one full revolution, the motor reverses to home position and LED2 will blink.

Step 9: Live Scan Monitoring

- Watch the terminal output to see readings like "*Distance: 728, Index: 18*".

Step 10: Adjusting Scan Layers

- In the script, look for "*numScans = 2*".
- Increase *numScans* to raise the number of vertical layers (slices). Each scan corresponds to a vertical z-step (default: 250 units).

E. Finishing Up

Step 11: Auto-Homing

- The motor automatically performs a reverse rotation after the final layer.
- Once homed, the system returns to idle and awaits the next button press.

Step 12: View 3D Map

- A 3D point cloud viewer will open via Open3D.
- File *graph.xyz* will be saved locally.
- You can load this into a 3D tool like CloudCompare for further inspection.

Limitations

1. What limitations does the microcontroller have in floating-point math and trigonometric calculations?

The MSP432E401Y microcontroller includes a single-precision (32-bit) floating-point unit (FPU), which handles basic floating-point math. However, it lacks efficiency for heavier computations such as trigonometric functions. Calculating sine or cosine values takes ~40 µs per call, significantly slower than a desktop CPU (~1 µs). Since the 3D mapping relies on converting polar data to cartesian coordinates using sin() and cos() functions, these delays accumulate and impact performance. In addition, rounding errors from limited precision can cause visible distortions in the 3D scan. To overcome this, the computational load is shifted to a PC, which uses faster double-precision arithmetic for accurate and smooth rendering.

2. How did you determine the quantization error of the VL53L1X sensor?

The VL53L1X Time-of-Flight sensor reports distance readings with a resolution of 1 millimeter [2]. This means it can only provide integer values in millimeters. Any actual distance falling between whole numbers will be rounded, introducing a maximum possible quantization error of ±1 mm. Since no sub-millimeter accuracy is possible, this rounding error defines the sensor's measurement limit. Therefore, the calculated maximum quantization error is 1 mm, directly determined by the sensor's resolution specification.

3. What was the highest serial communication rate you implemented with the PC, and how did you confirm it?

The project uses a UART baud rate of 115200 bits per second, which is a common and stable setting supported by most PCs and USB serial adapters. Verification was done using an oscilloscope connected to the TX pin (PA0) on the microcontroller. The waveform revealed a bit width of ~8.68 µs, translating to the expected 115200 bps, and matches the settings used in both the MCU and the Python visualization script.

4. What protocol and speed were used for communication with the ToF sensor?

Data communication between the microcontroller and the VL53L1X Time-of-Flight sensor is handled via I²C protocol, using the SDA and SCL lines for data and clock signals, respectively. The I²C clock is set to approximately 100 kbps, which aligns with the standard-mode operation of the VL53L1X as per its datasheet [2].

5. What component limited the overall system speed, and how was this tested?

The stepper motor is the main limiting factor in the system due to its mechanical movement constraints. Although the VL53L1X sensor supports fast sampling (up to 50 Hz), the motor requires careful timing to maintain accurate step control [2]. To test its limits, we reduced the step delay incrementally. Once the delay became too short, the motor failed to complete full revolutions or skipped steps. Meanwhile, the sensor continued delivering accurate readings. This

test confirmed that the stepper motor's physical response time is the limiting factor in overall system speed.

(6) Show the steps and calculations to configure your assigned system Bus Speed from Table 1.

$\frac{480\ MHz}{(PSYSDIV+1)} = \frac{480}{(17+1)} \approx 26\ MHz\ (rounding\ down)$ [1]

PSYSDIV = 17

Circuit Schematic

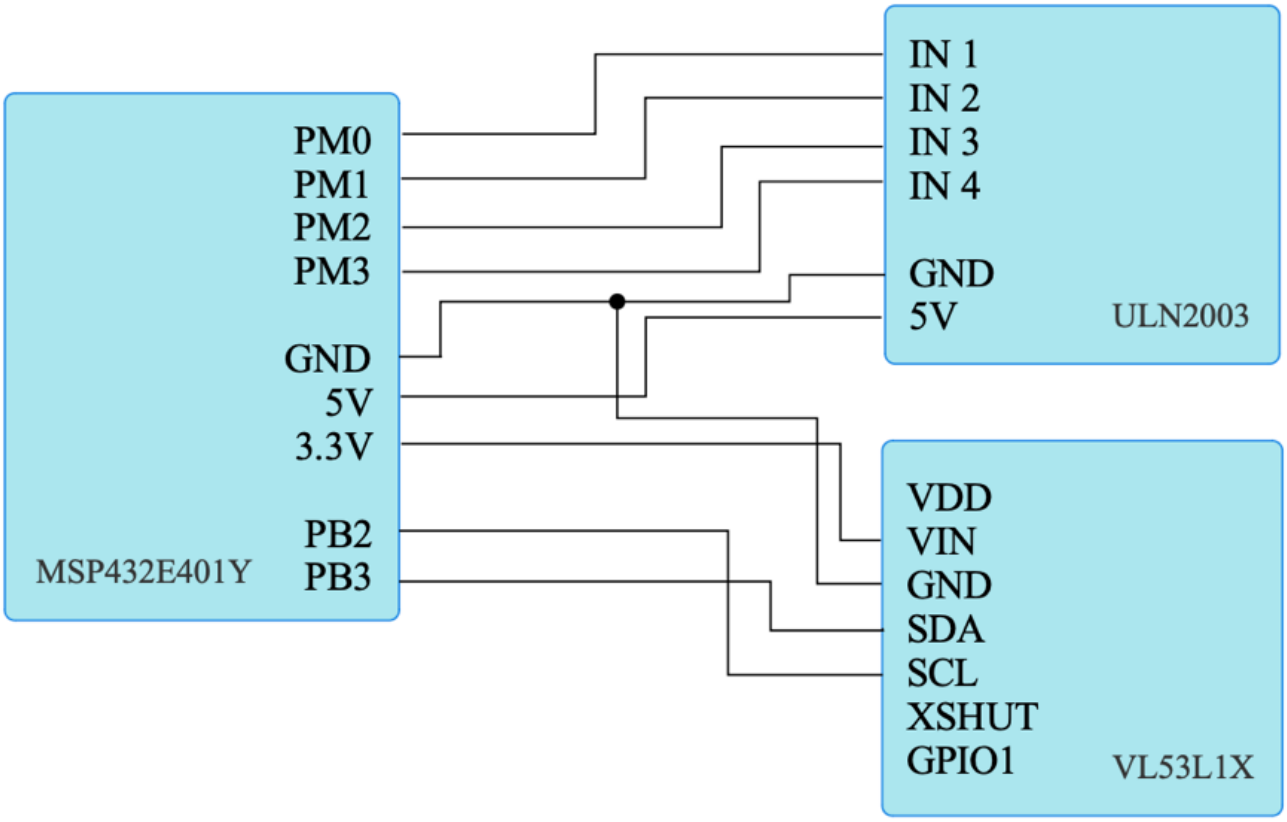

Figure 7: Circuit schematic

Programming Logic Flowchart

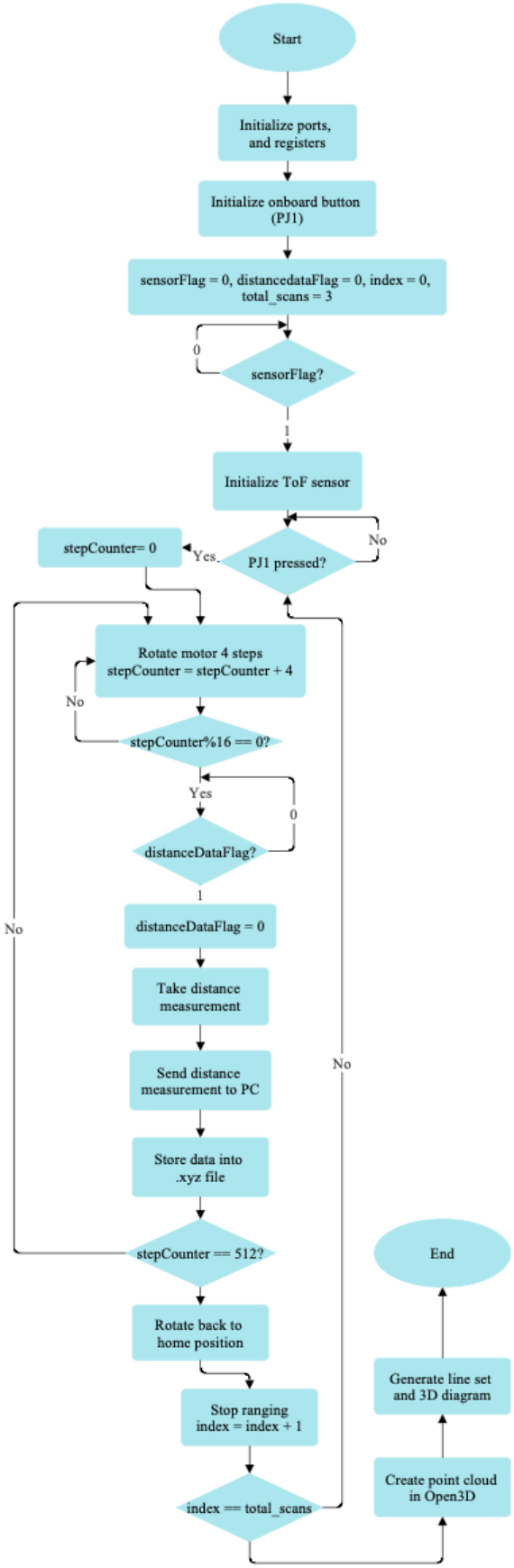

Figure 8: Programming logic flowchart

References

[1] Texas Instruments, "*MSP432E401Y SimpleLink™ Microcontrollers Datasheet*", Accessed: May 8, 2025. [Online]. Available: https://www.ti.com/lit/ds/symlink/msp432e401y.pdf

[2] STMicroelectronics, "*VL53L1X Time-of-Flight Ranging Sensor*", Accessed: May 8, 2025. [Online]. Available: https://www.st.com/en/imaging-and-photonics-solutions/vl53l1x.html