3D Environment Mapping System

COMPENG 2DX3 Final Project Winter 2025

Student Name: Kai Hughes
Student Number: 400539889
Date: April 9th, 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 my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by Kai Hughes - 400539889, Hughek26

Contents

1 Device Overview

This system implements modern technology to model closed areas with enhanced precision levels. With two simple buttons, the area scanned will be modeled highly accurately and can be observed from all angles. This 3D model is developed from complex yet configurable software making it easy to use, and expandable for specific requirements.

1.1 Key Features

- **Configurable Bus Architecture**:
  - Programmable system clock: $12\,\mathrm{MHz}$ (PLL-configurable)
  - Maximum CPU frequency: $120\,\mathrm{MHz}$
- **Power Management**:
  - Operating voltage: $5.0\,\mathrm{V}$ DC for stepper motor
  - Operating voltage: $3.3\,\mathrm{V}$ DC for ToF sensor
- **Cost-Optimized Design**:
  - BOM: $<$ \$150 CAD (COTS components)
  - ULN2003: \$6, VL53L1X: \$30, MSP432E401Y + wires: \$100
- **Dual-Channel Serial Communication**:
  - UART: $115\,200\,\mathrm{baud}$ (8N1, hardware flow)
  - I2C: $10\,\mathrm{kbit/s}$ (standard mode)
- **Memory Architecture**:
  - Volatile: $256\,\mathrm{kB}$ SRAM (real-time buffer)
  - Flash: $1024\,\mathrm{kB}$
  - Architecture: ARM Cortex-M4F, 32-bit, Harvard architecture (separate instruction and data paths)
- **Receiver-Side Software**:
  - Software: MATLAB R2021b+
  - Libraries: Serial Communication API, MATLAB Plotting Tools
  - Data format: XYZ file and 3D plot

1.2 Pin Mapping

Microcontroller

Pin	Function
PJ0	ToF Reset
PJ1	Stepper EN
PB2	I2C SCL
PB3	I2C SDA
PH0-3	Motor
UART0	R/T

Stepper Motor

Pin	Function
IN0	Ctrl Signal
IN1	Ctrl Signal
IN2	Ctrl Signal
IN3	Ctrl Signal
V+	5V
V-	GND

ToF Sensor

Pin	Function
VCC	3.3 Power
GND	Ground
SCL	I2C Clock
SDA	I2C Data

LED

Pin	Function
PN1	Scan Status
PF4	UART
PF3	Measurement

2 General Description

This system is controlled by two user switches. As SW2 is pressed, the full 360° scan begins. To end the plotting process, the other user switch, SW1, sends a termination indicator to the computer. The scanning process involves the time of flight sensor (ToF) gathering distance measurements after rotating a specific amount of times. By the end of the full rotation it would have collected 64 measurements. Between measurements, D3 and D4 will be flashing to indicate a successful UART transmission and distance measurement respectively. D1 is a status indicator representing a ready status; when D1 is on, it is ready for user input for another rotation or termination. Then, these measurements are processed relative to its position within the rotation. This data is being communicated with the computer to be plotted in MATLAB to visualize the scan.

2.1 Signal Acquisition and Processing

The ToF used is also an integrated circuit (IC) which includes the signal conditioning and ADC on top of the transducer

- **Phonon Detection**:
  - $940\,\mathrm{nm}$ VCSEL laser emitter
  - SPAD (Single Photon Avalanche Diode) receiver array
- **Analog Preprocessing**:
  - Prepares the analog signal from the SPAD receiver for reliable digital conversion.
  - Improves signal quality by filtering out noise and adjusting levels before ADC.
- **Analog to Digital Conversion**:
  - Converts the conditioned analog signal into a digital value for processing.
  - Provides discrete distance measurements with 12-bit resolution from continuous input.

2.2 Serial Communication Protocol

Table 1: UART Packet Structure (32 bytes)

Field	Description
TOF & Board	I2C Protocol
Board & PC	UART Protocol

The ToF sensor communicates with the microcontroller using SDA/SCL lines. The board acts as the master and initiates communication via a START condition which addresses the slave device (ToF).

The microcontroller then communicates with the PC via UART at 115200 baud (8N1) - this is integrated into the USB connection. Data is transmitted in packets with ToF measurements, angular position, and system status.

$$Distance = \frac{\text{Light Travel Time}}{2} \cdot \text{Speed of Light}$$

To resolve environmental spatial coordinates, these radial measurements are transformed into Cartesian coordinates. With this, the data can be processed in MATLAB to be plotted, completing the visualization.

2.3 Technical Specifications

Table 2: System Technical Specifications

Parameter	Description	Value
Microcontroller	ARM Cortex-M4 processor	12MHz
Time-of-Flight	VL53L1X	30 Hz (max 50Hz)
Data Collection	Angle/Measurement	5.625°
Measurement Range	ToF sensor capability	40mm - 4000mm
Data Rate	UART transmission speed	115200 baud
Software	Plotting	MATLAB
Software	Embedded Development	Keil uVision5

2.4 Block Diagram

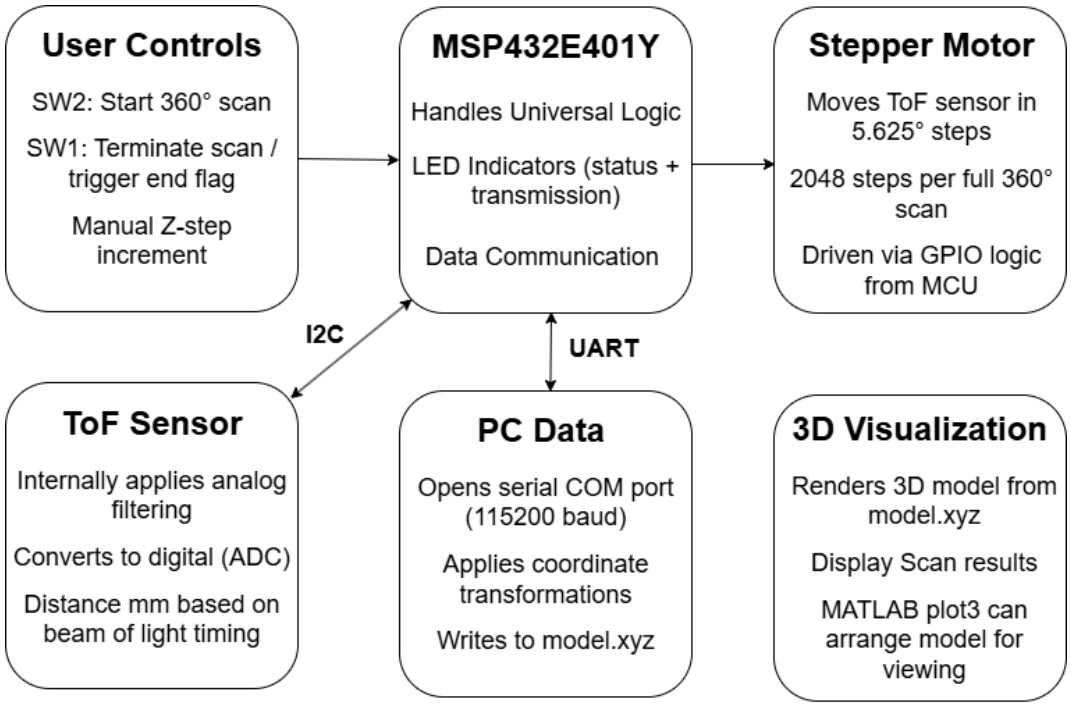


Figure 1: Block diagram of components

3 Detailed Implementation

3.1 Distance Acquisition

The VL53L1X time-of-flight sensor operates by emitting 940 nm infrared pulses through its vertical-cavity surface-emitting laser (VCSEL) diode. Reflected photons are detected by a single photon avalanche diode (SPAD) array receiver. The fundamental distance calculation follows from the time-of-flight equation and coordinates are:

$$d = \frac{c\Delta t_{\text{photon}}}{2} \tag{1}$$

$$x = d\cos\theta \tag{2}$$

$$y = d\sin\theta \tag{3}$$

where:

- d = Measured distance (mm)
- c = Speed of light ($3 \times 10^8\,\mathrm{m/s}$)
- Δt_{photon} = Round-trip time delay (s)
- θ = Stepper rotation angle (0 rad to 2π rad)

The result is a millimeter distance measurement, with a resolution of 1mm and a range of up to 4m under optimal conditions. This sensor also performs operations on the analog measurements for the microcontroller to recieve. That is, transduction, condition and analog-to-digital (ADC) conversions. The VL53L1X then communicates digitally with the microcontroller using I2C.

3.2 Data Management and Logic

Initialization of the ToF sensor occurs through the `ToF_Sensor_Boot()` function, which configures the I2C communication protocol up to 100kbps. The default implementation, however, is configured at 10 kbps. User interaction is mediated through two external push buttons: PM1 initiates stepper motor rotation through a full $360°$ arc (64 steps at $5.625°$/measurement), while PM0 toggles the `scanEnable` flag that controls UART communication.

The angular resolution of $5.625°$ per measurement interval is achieved through modular arithmetic on the step counter:

$$\theta = \left(\frac{\text{Sample}_i}{64}\right) \times 2\pi \tag{4}$$

where the divisor 64 derives from the amount of samples desired. The relationship $2048\,\text{steps}/360° = 5.689\,\text{steps/degree}$ explains that every 64^{th} step or every invokes the ToF API functions `GetRangeStatus()` and `GetDistance()` to retrieve current ranging data.

3.2.1 Data Acquisition Protocol

- **Begin Communication**: MATLAB transmits start command ('s') after user confirmation

- **Stream Capture**: Continuous readline operation until termination sequence ”999” received (SW1)
- **File I/O**: Raw XYZ coordinates logged to `Model.xyz` with $100\,\mathrm{MB}$ preallocation

3.2.2 Coordinate Transformation

The ASCII data undergoes conversion to Cartesian coordinates through:

$$\mathbf{Data} = \begin{bmatrix} x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix} = \texttt{fscanf(file,'\%f \%f \%f',[3, Inf])} \tag{5}$$

with Δz representing synthetic depth increments per full rotational sweep.

Distance and Displacement Measurement

In this project, **displacement** refers to the linear movement of the Time-of-Flight (ToF) sensor along the z-axis, which is orthogonal to the x-y plane traced out by the sensor's rotation. It represents the sensor's forward motion from its initial position, enabling three-dimensional scanning when combined with rotational measurements. This system manually defines the Z displacement and should be adjusted according to the change in forward displacement - the default is defined on line 254 of the C code.

In the code, this is captured using an interrupt-driven variable, `FallingEdges`, which is incremented each time the user presses the button connected to port SW2. This value determines how many displacement steps have been made.

The z-value is then calculated as:

$$z = \text{FallingEdges} \times 100; \tag{6}$$

This assigns a displacement of $100\,\mathrm{mm}$ per button press, and thus assumes the mechanical system moves by that distance. This fixed-step displacement provides consistent layering for 3D reconstruction but will not account for different stepped displacements.

For each rotational scan, the sensor collects multiple distance readings. These are converted to 3D coordinates where θ is the angle of rotation, z is the manually set displacement, and d is the measured distance from the sensor, as defined in (3.1).

Each (x, y, z) point is transmitted via UART for further processing and 3D visualization. A sample UART output off the 34^{th} sample on the second scan, and distance measured of 1422mm is seen as:

```
Processing Values:
Sample: 34
Angle (rad): 3.337944
Distance (mm): 1422
Z-displacement (mm): 200

UART Output:
-1417.0 -207.1 200
```

These points are then plotted in MATLAB. By incrementing z after each sweep, the system builds a layered volumetric scan of the scanned area.

3.3 3D Visualization

3.3.1 Serial Communication Initialization

The MATLAB interface establishes serial communication through `COM4` using an 8N1 UART configuration at $115\,200\,\mathrm{baud}$. A $135\,\mathrm{s}$ timeout ensures robust error handling and allows for longer delays between measurement intervals. The protocol implements a handshake mechanism where:

$$\text{MCU} \xrightarrow{\text{'s'}} \text{MATLAB} \xrightarrow{\text{ACK}} \text{Data Stream} \tag{7}$$

initiating data transfer upon receiving the start character 's'. Buffered data persists through hardware resets via `flush(s)` command execution. The visualization of the 3D point cloud was conducted in **MATLAB**, chosen for its integrated libraries and ease of data handling. The UART data streamed from the microcontroller was logged into arrays of x, y, and z coordinates. Each sweep of the Time-of-Flight (ToF) sensor corresponds to a full 360-degree rotation at a fixed z-level, producing a circular cross-section of the scanned object.

3.3.2 Software outline

The steps for producing the 3D point cloud are as follows:

1. **Data Acquisition:** UART data is collected in real-time using a serial terminal from the MATLAB Serial API.
2. **Data Manipulation:** The incoming data, formatted as space-separated `x y z` strings, is parsed line-by-line into numeric arrays. This was done by using `readline(s);` and because of the written format from C, MATLAB can write this to an XYZ file format easily with `fprintf(fileW, "%s\n", x);`
3. **Noise Filtering:** Erroneous or clipped values (e.g., values exceeding ToF max range of 4000) are filtered using a thresholding mechanism.

```
if (fabs(x) > 4000)
    x = prev_x;
if (fabs(y) > 4000)
    y = prev_y;

prev_x = x;
prev_y = y;
```

3.3.3 Plotting

- **Z Segmentation:** After every 64 points (one full rotation), a NaN value is inserted into the arrays to break the polyline and create visual separation between z layers. Seen in 1.

- **Plotting:** The `plot3` function is used to render the data in 3D with customized styling
- **Marker Customization:** Each point is visualized as a 4pt black-filled circle with a 10pt wide cyan connector line for strong visual contrast.
- **Axis Labeling:** The x, y, and z axes are oriented to match the physical scanning setup.

Listing 1: Core plotting mechanism MATLAB

```
file = fopen('Model.xyz', 'r'); % Open the file for reading
data = fscanf(file, '% f  % f  %f', [3, Inf]); % data into a 3xN matrix
fclose(file); % Close the file
% Transpose the data to get columns as vectors
data = data';

x = data(:,1); % First column as x
y = data(:,2); % Second column as y
z = data(:,3); % Third column as z
figure;  % Plotting
hold on;
% Insert NaN values every 64 points to break connections
num_points = size(data, 1);
for i = 1:64:num_points
    end_idx = min(i+63, num_points);
    plot3(x(i:end_idx), y(i:end_idx), z(i:end_idx), 'b-', 'Marker',
    'o', 'MarkerFaceColor', 'black', 'MarkerSize', 4, 'LineWidth', 10);
end
```

This configuration allows for clean, layered scans that reveal the spatial structure of the object being measured. Also, by writing and exporting the data into a xyz rather than plotting the points directly, it allows for the separation of the plotting code. With this, code can be written with only listing (1) to plot other xyz models.

4 Application

4.1 Application Example: 3D Environmental Mapping

The system rotates incrementally, capturing distance data at discrete angular intervals to construct a 3D model of the surrounding environment. This application is ideal for indoor SLAM and robotics navigation. This data can help the robot understand where it's surrounding are and communicate it to other devices. In this system the Z value was manually inputted but with a configuration with vehicle-like systems, the z value can be calculated as well. This would further improve the accuracy. Other potential applications for this device include mapping indoor spaces for building layouts or monitoring small areas for changes over time. Many companies are interested in space optimization for storage units - this system can assist with the initial stages and can be integrated with analysis tools to automatically define optimized organization layouts.

4.2 System Setup Instructions

Components:

- MSP432E401Y LaunchPad
- VL53L1X ToF Sensor
- ULN2003 Stepper Motor (with driver)
- UART-USB connection to PC
- MATLAB (for data visualization)
- Keil uVision IDE

4.3 Instructions and Setup Procedure

This setup process assumes the user has all required software installed and configured correctly. This includes:

- Keil uVision IDE (for compiling and flashing the MSP432E401Y)
- MATLAB (for serial communication and data visualization)
- Correct board configuration for the TI MSP432E401Y

Steps to setup and run the system:

1. **Connect the MSP432E401Y board to the computer via USB.** Identify the COM port by:
    - Pressing the Windows key and searching for "Device Manager"
    - Expanding `Ports (COM & LPT)` and noting the number for "XDS110 Class Application/User UART (COM#)"

2. **Configure MATLAB script for serial communication.**
    - Open the MATLAB script and navigate to line 7:

```
s = serialport("COMX", 115200, "Timeout", 135);
```

    - Replace `COMX` with your specific COM number (e.g., COM4).
3. **Wire the sensor circuit according to the pin mapping used in the code.**
    - Refer to (6) and wire accordingly
4. **Compile and flash the code to the board using Keil:**
    - Open the project in Keil
    - Click `Translate` → `Build` → `Load`
    - Hit the flash/reset button on the board to run the code
5. **Begin scanning procedure:**
    (a) Press enter as prompted by MATLAB to begin receiving data.
    (b) Press SW2 to start scanning - LED3 will turn off indicating data collection is active.
    (c) Line 253 in the C code defines a variable `measured_steps`. Default is 64 but this number can be changed for more precise models.
    (d) Move forward 100mm or the defined z value.
    (e) Press SW1 on the board after final scan is completed and LED3 is on.
6. **After all slices are collected, MATLAB will render the final 3D plot of the scanned environment.**

Axis Definition

This system defines its spatial axes as follows:

- **X-axis:** Horizontal sensor Sweep
- **Y-axis:** Vertical sensor sweep
- **Z-axis:** non-measured displacement (depth) from the intial sensor location

Coordinate Definition

In this implementation, the coordinate system is defined as follows:

- The **z-axis** represents *displacement* from the origin. Each full sweep assigns a new constant z value.
- The **s** and **y** axes are derived from polar to Cartesian conversion of angular ToF data defined in (3.1)

4.4 Expected Output

The collected scan data, visualized in MATLAB, forms a 3D point cloud resembling the actual scanned space. Table 3 is a side-by-side comparison of the physical environment and the rendered 3D model.

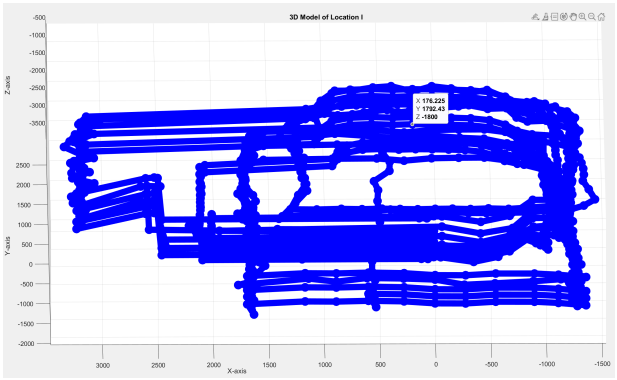
Actual Campus Location (I)	3D Model from Scan
	

Table 3: Comparison Between Real Scene and Reconstructed 3D Model

The rendered model captures architectural features such as walls, angles, and depth transitions, validating the system's spatial fidelity.

5 Limitations

This system employs rounding for floating point numbers. This raises error with highly precise modeling after distance measurements. The primary logic for the coordinates as:

Listing 2: Core Visualization Parameters

```
sprintf(printf_buffer, %.3f %.3f %.3f \r\n",x,y,z);
```

5.1 FPU and Trigonometric Logic

The measurements being in millimeters means the precision of the model will be up to 1 micrometer (0.001 mm), since the format %.3f rounds the floating-point values to three decimal places. This limits the spatial resolution and and thus is unreliable for extreme precision modeling. Also, despite the microcontroller including a single-precision FPU, this supports up to 32-bit precision, which would translate to around 7 decimal digits of accuracy. As the FPU is not optimized for high-volume calculations, the model accuracy is not meant for exact sizing.

Moreover, the trigonometric functions used were software-based implementations which implies PI was a defined constant of 6 decimal precision and cos/sin are truncated taylor series expansions. Although small, this will introduce non-negligible error for users expecting highly precise models.

5.2 Quantization Error

The VL53L1X uses an SAR (Successive Approximation Register) ADC to convert its analog time-of-flight measurements into digital values. The ADC quantizes the continuous distance values into discrete steps, each corresponding to a fixed resolution. Since the module's resolution is 1 mm, it means that any measured value is truncated to lowest 1 mm interval. Consequently, the maximum quantization error - defined as the largest possible deviation between the true distance and its quantized value - is 1 mm. This error is inherent in the ADC conversion process, as the sensor can only distinguish increments of 1 mm, setting the highest quantization error at that step size. In the worst-case scenario, the actual value could be almost 1 mm higher than the truncated value. Thus, the maximum quantization error - that is, the difference between the true distance and the digitized value - remains 1 mm.

5.3 Serial Communication Rate - PC

The maximum standard serial communication rate implemented with the PC is 115200 baud. This value is specified in the system's design and firmware configuration. Defined in section 6.5.6.4 in the MSP432E401Y data sheet it defines the UART baud rates to be configured up to 115.2 kbps. To ensure operation this was verified by configuring MATLAB to initialize the receiving port with a 115200 baud. Test strings were sent from the testbench in keil and it appeared in MATLAB successfully without error - this verified a consistent working baud rate. Together, these steps confirmed that the maximum standard serial rate achievable in the system is indeed 115200 baud.

5.4 Serial Communication Rate - ToF

The communication between the microcontroller and the VL53L1X ToF module is established via I2C. In the initialization code, the I2C peripheral is configured to operate at a clock speed of 10kbps though can be changed to 100kbps. This specific rate is set by programming the I2C0_MTPR_R register - in the C code this can be found on kube 72. The I2C protocol, using dedicated SDA and SCL lines, facilitates the leader-follower communication where the microcontroller (leader) sends commands and retrieves distance measurements from the VL53L1X (follower).

5.5 Primary Limitation

The system's speed is primarily constrained by two factors: the stepper motor speed and the ToF sensor measurement rate. The stepper motor controls the sensor's rotation, taking measurements at discrete intervals.. The motor's speed directly impacts the time needed for a full rotation and, consequently, the rate at which new data is captured. To test this, the delay values between each change in the stepping sequence was addressed. It was found that the lowest value before hindering the consistency in rotation was 36000 which implies a delay of 3ms.

The ToF sensor, VL53L1X, measures distances based on light pulse time-of-flight, and its speed is limited by both the sensor's measurement process and the I2C communication rate. The sensor provides measurements in a few milliseconds, but the I2C communication operates at 10 kbps, further slowing data transfer. The testing, using the AD3 as a spy on the SCL line, confirmed the 10kpbs clock rate. It confirmed that the sensor's measurement rate is constrained by both this communication speed and the inherent delay in the sensor's operation.

6 Circuit Schematics

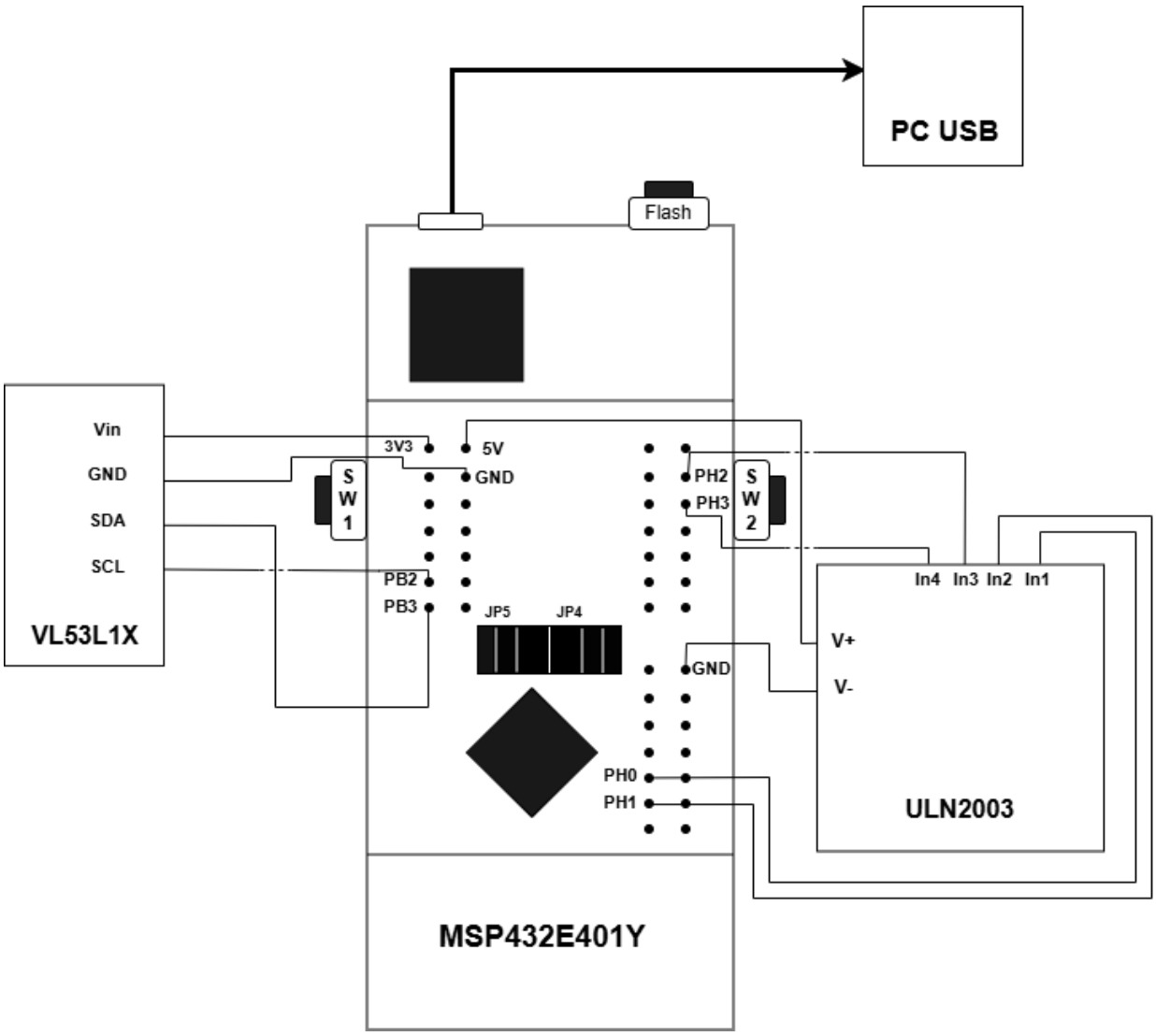


Figure 2: Complete system schematic diagram

7 Firmware Flowchart

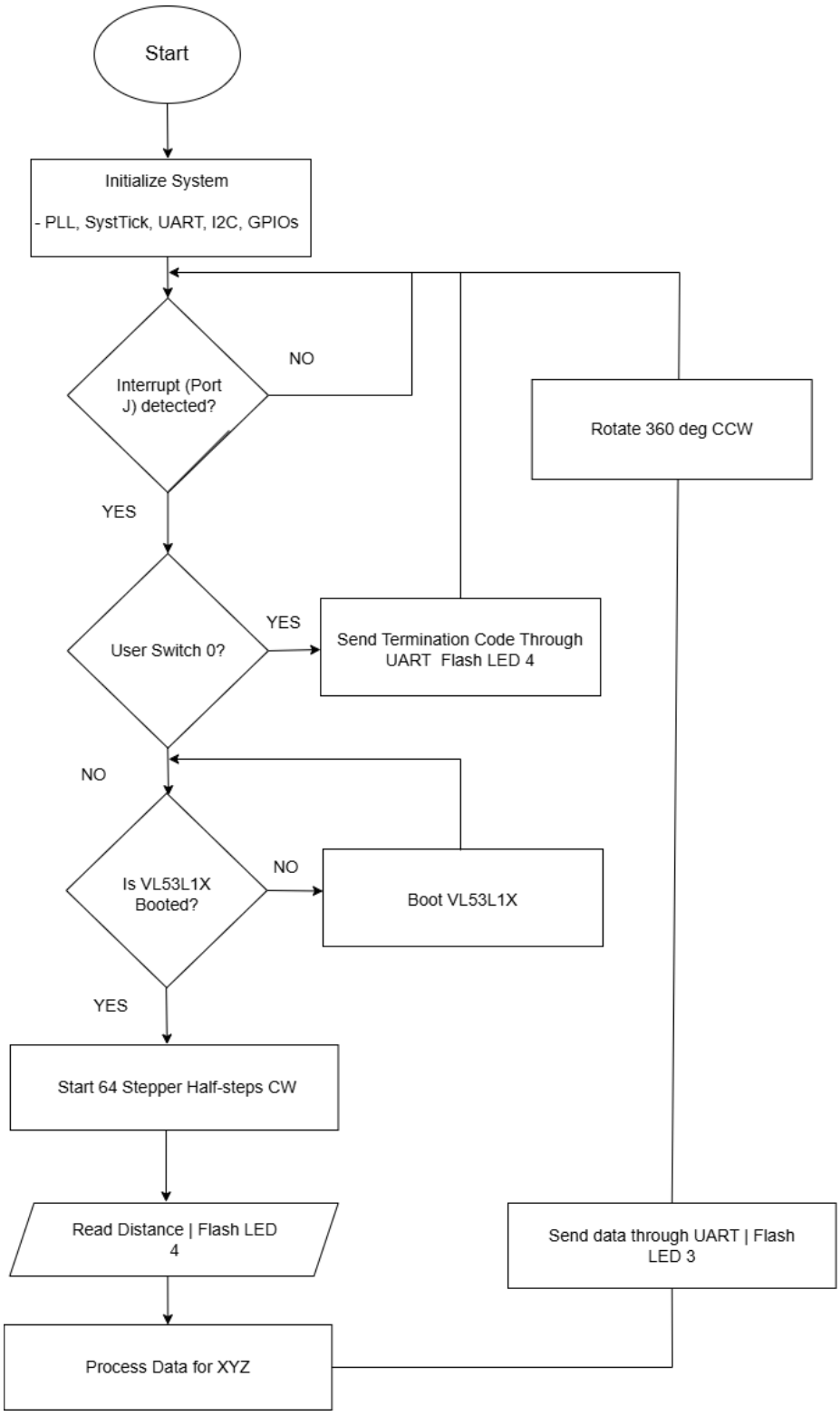


Figure 3: Flowchart for the microcontroller code

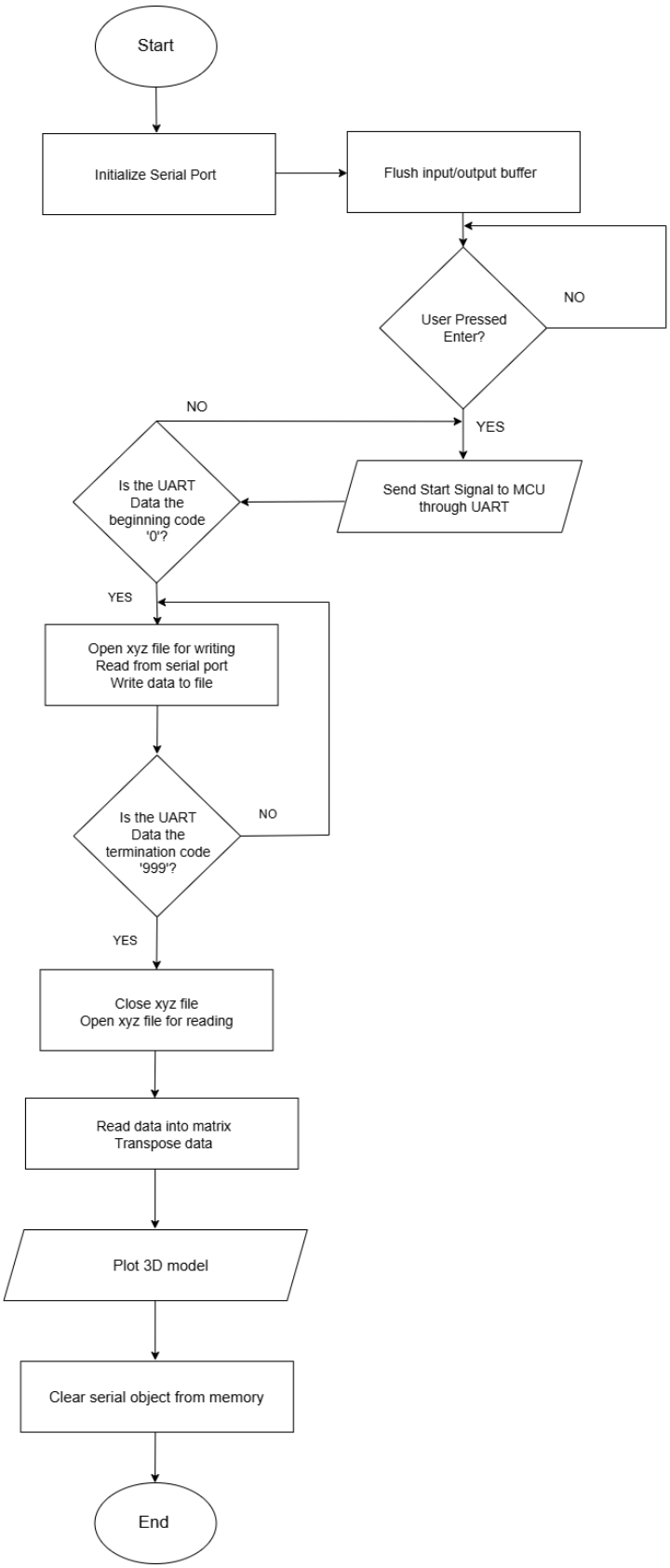


Figure 4: Flowchart for the MATLAB visualization code