COMP ENG 2DX3
Final Project Report

L02

Sarujan Baheerathan, bahees1, 400453475

Device Overview

Features

MSP432E401Y – Texas Instruments Microcontroller

- Operates at approx. 5V at a 30MHz bus speed
- Processor – Cortex-M4F processor, Harvard Architecture
- Baud Rate – 115200bps – between microcontroller and PC

VL53L1X – Time of Flight(ToF) sensor

- 3 distance measuring modes – short, medium, long range
- Maximum measuring distance – 4m
- Communication – I2C communication protocol with 400KHz maximum frequency
- Operating voltage – 2.6-3.5V

28BYJ-48 & ULN2003 – Stepper Motor and Control Board

- Operating voltage – 5V
- Rotation – 512 steps

Visualization – Software

Python 3.8

- Used to perform calculations to determine distance measurements in the y-z plane
- Used to process incoming data from TOF sensor

Open3D

- Used to plot the processed distance data from TOF sensor to create a 3D representation for the user

General Description

The goal of this embedded systems project was to design and build an embedded spatial measurement system using a TOF sensor and stepper motor. The system should be able to take multiple scans of a hallway/small room and plot the received distance measurements to create a 3D model of the scanned space. This design is a custom, compact, and lightweight alternative to commercial LIDAR systems that allow students to measure spaces easily. The system produces its 3D representation with the help of a stepper motor that provides 360-degree measurement capability, as well as a Time of Flight (TOF) sensor that takes distance measurements every 22.5-degrees providing 16 measurements per frame. The number of frames scanned to recreate a space is set by the user, this setting affects the accuracy of the scan providing a more detailed scan if more frames are measured. Since the x plane measurements are set by the user, the device focuses on scanning in the y-z plane to accurately recreate the space. After analog measurements are taken they are sent to the microcontroller using the I2C communication protocol, which are then serially transmitted to the users desktop using the onboard UART communicator. The data received by the UART is then processed using a python script, that takes the distance measurements and calculates the y-z co ordinates using trigonometry. The new coordinates are then written to a .xyz file that will be then read by the Open3D library to plot the data and finally create a 3D representation of the scanned space.

Block Diagram

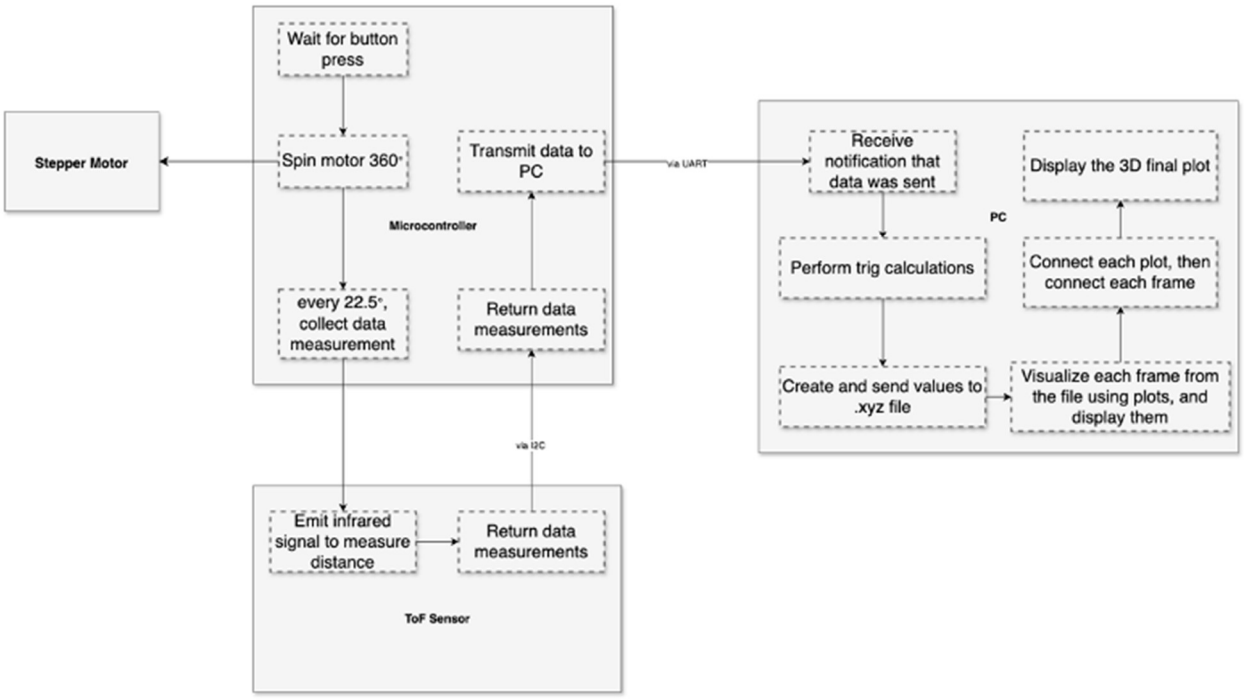


Figure 1: Logic flow block diagram

Device Characteristics Table

Table 1: Device Characteristic Table

Physical Component	**Details/Important information**
Time of Flight sensor	Vin – 3v3 source GND – ground SDA – PB3 on micro SCL – PB2 on micro
Stepper Motor	*All PM# connections are made on the micro Vin - 5V GND - GND IN1 – PM0 IN2 – PM1 IN3 – PM2 IN4 – PM3
Push Button (onboard & interrupt based)	PJ1 on micro
Status LED (onboard)	LED 1 – PF0 on micro LED 2 – PN0 on micro
External Applications	**Details/Important Information**
Keil uVision5	C based Keil project code to run the script to control the stepper motor and TOF sensor
Python 3.8	Used to serially transmit data from microcontroller to PC and plot TOF distance data

Detailed Description

Distance Measurement

The Time of Flight sensor is the device used in this project to gather the distance measurement. The sensor works by emitting a light source such as an LED/laser and receives the reflected light using a sensor that is calibrated to the incoming light's wavelength. The distance is measured by comparing the time delay between sending and receiving a light's wavelength. Therefore, by using the formula below the distance can be calculated by multiplying the time delay with the speed of light and dividing the result by two since the delay is for sending and receiving the signal.

$$distance = \frac{1}{2}(time\ delay\ x\ speed\ of\ light) = \#\ mm$$

The second device that works in combination with the ToF sensor is the stepper motor which has the ToF sensor directly mounted on it. This mounting solution allows the system to perform a full 360-degree scan of a single frame for a space. For this project a total of 16 measurements that were 22.5-degrees apart were taken to fully capture a single frame for the scanned space, for a total of 10 frames to fully capture the assigned small space.

In order to collect the data from the ToF sensor and process it into data that can be plotted, the sensor must communicate with the microcontroller to transmit the data. To collect the data functions such as SensorInit(), StartRanging(), GetDistance() were used to first initialize the ToF sensor so that it was active and ready to measure. The other two functions were used to ensure that the ToF sensor was ready to receive data and capture distance measurements whenever they were requested by the program. This was done using the I2C communication protocol, by connecting the Serial clock line (SCL) as well as the Serial Data Line (SDA) pins between the ToF sensor and the assigned PB2 and PB3 pins as shown in Figure 4. The data that the microcontroller receives must then be serially transmitted to the user's desktop for further calculations and processing. This was done by using the UART communication protocol at a baud rate of 115200 bits per second under COM Port 3.

Once the desktop receives the analog distance measurements a separate python script reads the incoming data and prepares a new file for the calculated xyz coordinates to be written into. The python script reads the data points line by line and after each line is processed it performed trigonometric calculations to determine the y-z coordinates for that point in the scan. Since the x coordinate is predetermined by the user before the scanning performance begins, the python script focuses on determining the remaining coordinates in the y-z plane. The calculations were done using the following equations:

Y = distance * cos(radians)

Z = distance * sin(radians)

X = incremented by 500mm/scan for the room scan, 100mm/scan increments for live demonstration

As the coordinates are calculated per iteration in the python script they are also written into a .xyz file for the plotting stage.

Visualization

After the final coordinates are calculated and written into the .xyz file, the visualization stage begins where the script begins by reading the previously written xyz coordinates. To transform the coordinates into something that the user can visualize and comprehend, the point cloud functionality must be utilized to create the final 3D model. In order to make use of the point cloud function the script must first append all the raw coordinates into a single array. This allows the use of the o3d.io.read_point_cloud() function which reads the points into the function. Next, the points must be connected to create the individual frames/slices of the scan. This step is achieved by creating a separate array for the slices, and then appending new line segments that connect the first 16 points together since they are all organized in order within the array. After the individual frames are linked and appended to the array, the script proceeds to attach the points of each frame to the next frame. This is done by drawing segments that are at an index 16 points apart from each other. At the end of creating all these line segments connecting the frames together the o3d.geometry.LineSet() function is used to map the lines to the 3D coordinate vertices. Finally to see the 3D model the o3d.visualization.draw_geometries function is used to see what the point cloud looks like graphically.

Application Note

Instructions

1. Download the provided Python file, Keil project code, ensure Python 3.8 is downloaded as well as the pyserial and open3d libraries as well

2. Proceed to connect all the hardware as stated in the circuit schematic in Figure 4, and the device characteristic table in Table 1.

3. Important note for the hardware, ensure that the TOF sensor is securely mounted onto the stepper motor using any method that does not cover the front of the sensor to prevent any errors in measurement. At this stage the hardware component is complete!

4. Next, the user must setup the software component, this consists of opening the provided Keil project as well as the provided Python code.

    4.a The user must ensure that they identify their COM port for the UART and modify the corresponding line in the provided python code.

    4.b User must set the number of scans and the array lengths in both the Keil project as well as the python code at the specified lines.

    With that the software portion is setup and the user is ready to start scanning their space.

5. With the entire system setup and ready to go the user must position the device at the starting point of the scan and ensure there are no obstructions on the TOF sensor itself, and to double

check all connections with hardware as well as software settings to ensure the desired output is achieved.

6. When the user is ready to go and stationed at the starting point, they can flash the Keil code onto the board and get ready for the scan. To start the user must press the RESET button on the microcontroller to start the first scan.

7. Once the first CW scan is finished a second CCW rotation will occur to return the sensor back to the home position. An LED will remain ON for the duration of the return phase. While the device is returning to user should move the device to the next measuring position.

8. Once the user continues this CW and CCW phase ensuring that they move the device to the next point of their scan in their space during the CCW phase, when the number of scans set by the user has been reached the device will stop its rotations and the data is ready to be plotted.

9. The user can now open the Python script where they will run the module and press Enter to print the xyz coords onto the screen where the user can inspect the values. After all the values are printed the 3D model from Open3D should be generated where the user can inspect the 3D model of their scanned space.

Expected Output

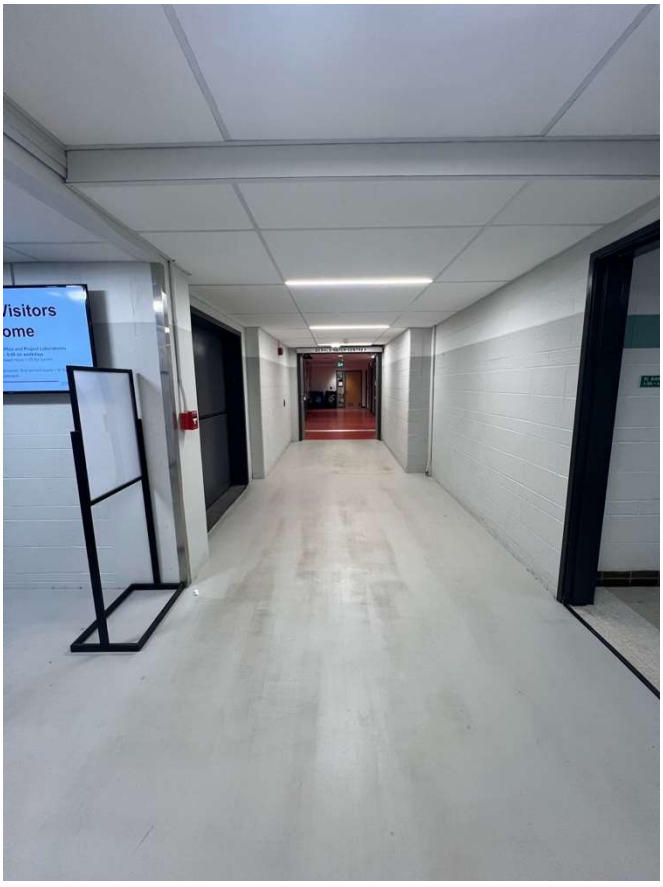


Figure 2: Assigned Scan Location

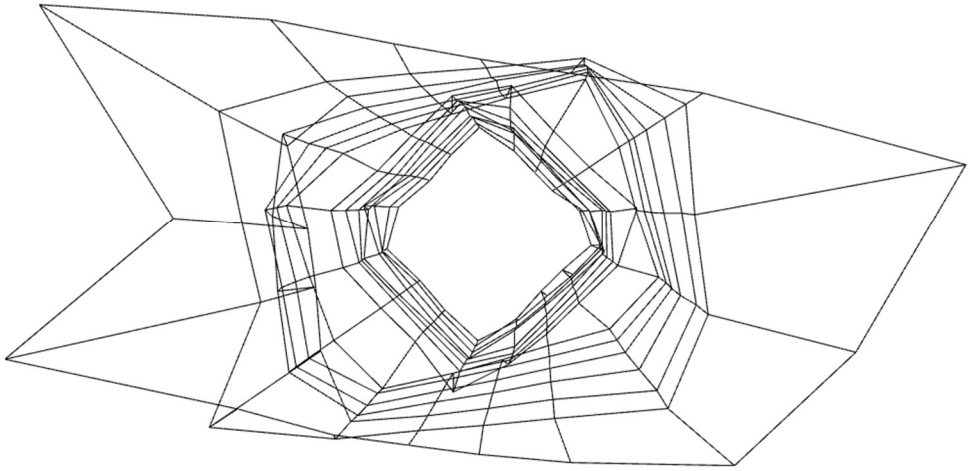


Figure 3: Final 3D model of scanned space

The assigned space for the scan above was for the second floor hallway of the John Hodgins Engineering building (Figure 2). The hallway is a simple long corridor that features two entrances to different spaces on the left and right side of the space. Further down is an elevator door on the left side of the wall, followed by a pertrusions that make the general dimensions of the hallway narrower. In figure 3 the same features listed previously can be identified. The large measurements that start the scan indicate the separate entrances on the left and right, followed by a larger indent on the left side of the wall representing the elevator door. Further down the dimension of the scan gets narrower describing the same pertrusions in the walls as seen in Figure 1. However during the scanning process there was an error with the stepper motor that caused the starting position to change, which caused a rotation in the model of the scanned space. This problem can be easily fixed to represent the hallway more accurately if the scan was done once more. Details could also be improved if the angle at which measurements were taken was decreased to take measurements more frequently.

Limitations

Although the goal of this project was to design and build an embedded system that is able to accurately recreate a 3D mapping of a small space, the components used still pose limitations. For the physical components the stepper motor's spin rate directly affects the length of the scans, therefore limiting the efficiency of the scan time. The other limitations come from the ToF sensor that has a max scanning distance of 4m at a frequency of 50Hz, which limits the spaces the user can scan. This constraint makes the device only usable for relatively small rooms/hallways which were the objective of this project, however for rooms larger than 4m a more powerful ToF sensor is required. In addition the accuracy of the distance measurements collected by the ToF sensor is also limited due to the quantization error of the sensor. The error can be calculated by using the following formula:

$$Max\ Quantization\ Error = \frac{Max\ distance\ measurement}{2^{\#of\ bits\ in\ ADC}}$$

$$\frac{4000mm}{2^{16}} = 0.0610$$

When it comes to the microcontroller there are some limitations with bus speed and communication protocols. A design constraint for this project set by the coordinators was that the bus speed must be restricted to 30MHz, which restricts the overall performance of the microcontroller which can operate at a higher bus frequency at a max of 120 MHz. There are also further limitations with the I2C and UART communication protocols. The transmission speeds of the I2C are maxed out at approximately 395 kHz and the UART is limited to a baud rate of 115200 bps. These restrictions further decrease the speed and increase the length of data transmission during the scanning process.

The use of Python also has its own limitations for example the floating point variables in Python are capable of holding 64-bits which can cause issues with the accuracy of trigonometric calculations which are the chosen method in calculating the distance measurements in this project. Another restriction comes from the fact that the Open3D library is only available to users that have Python versions between 3.8 to 3.10.

Circuit Schematic

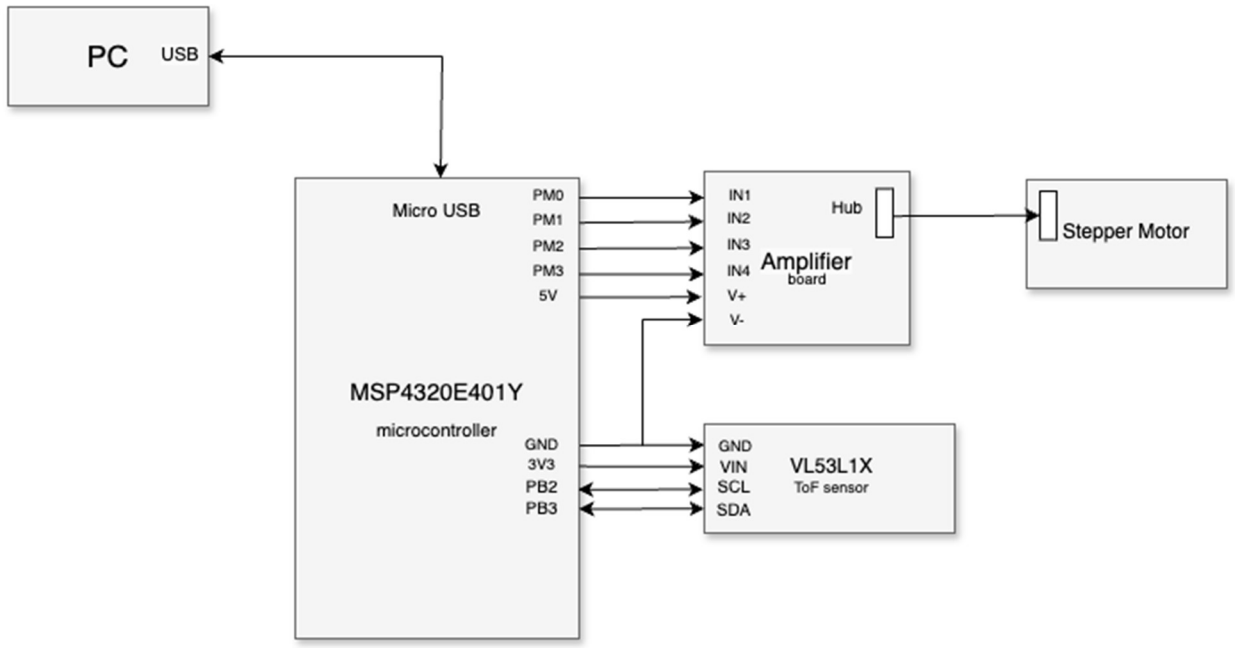
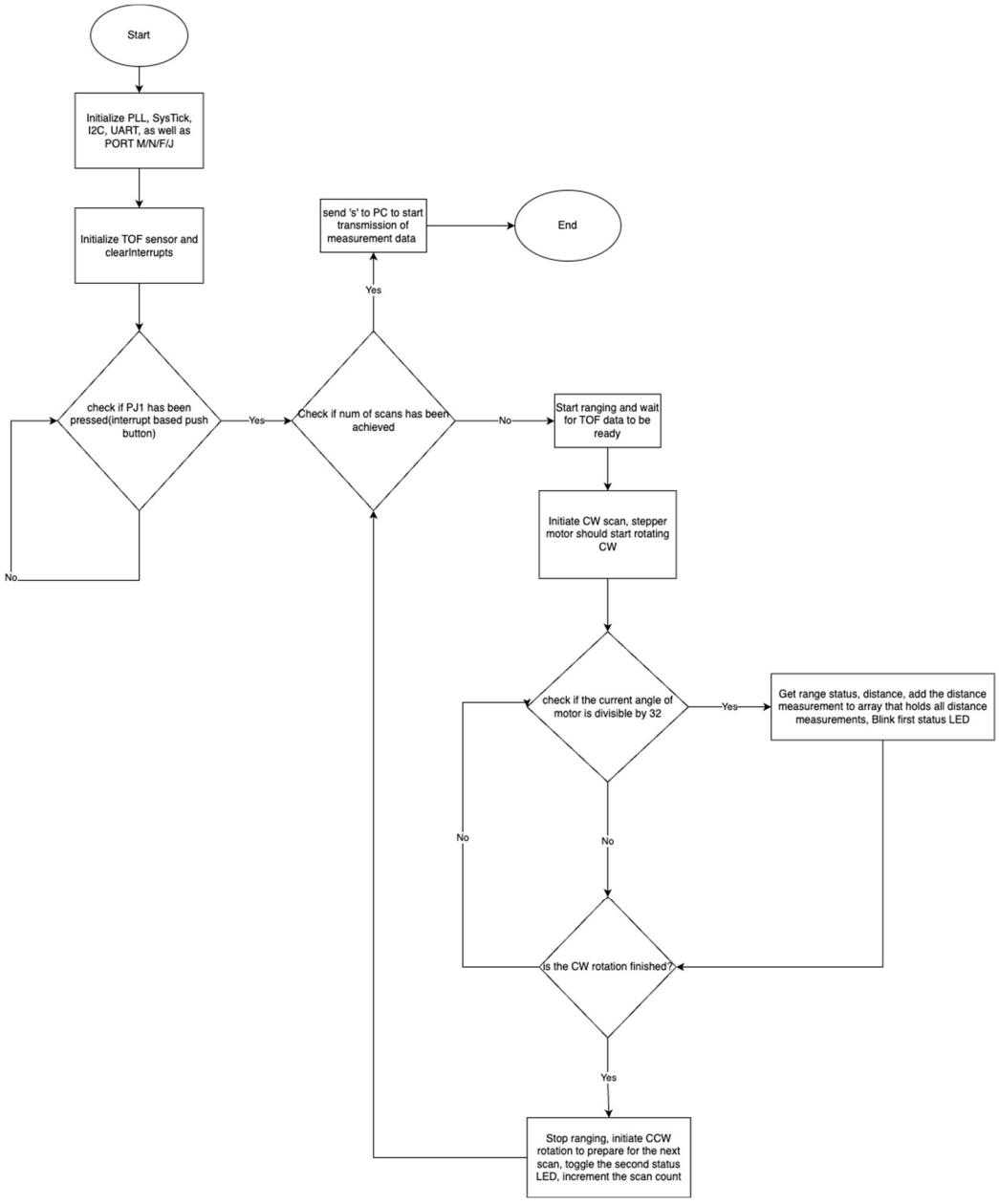


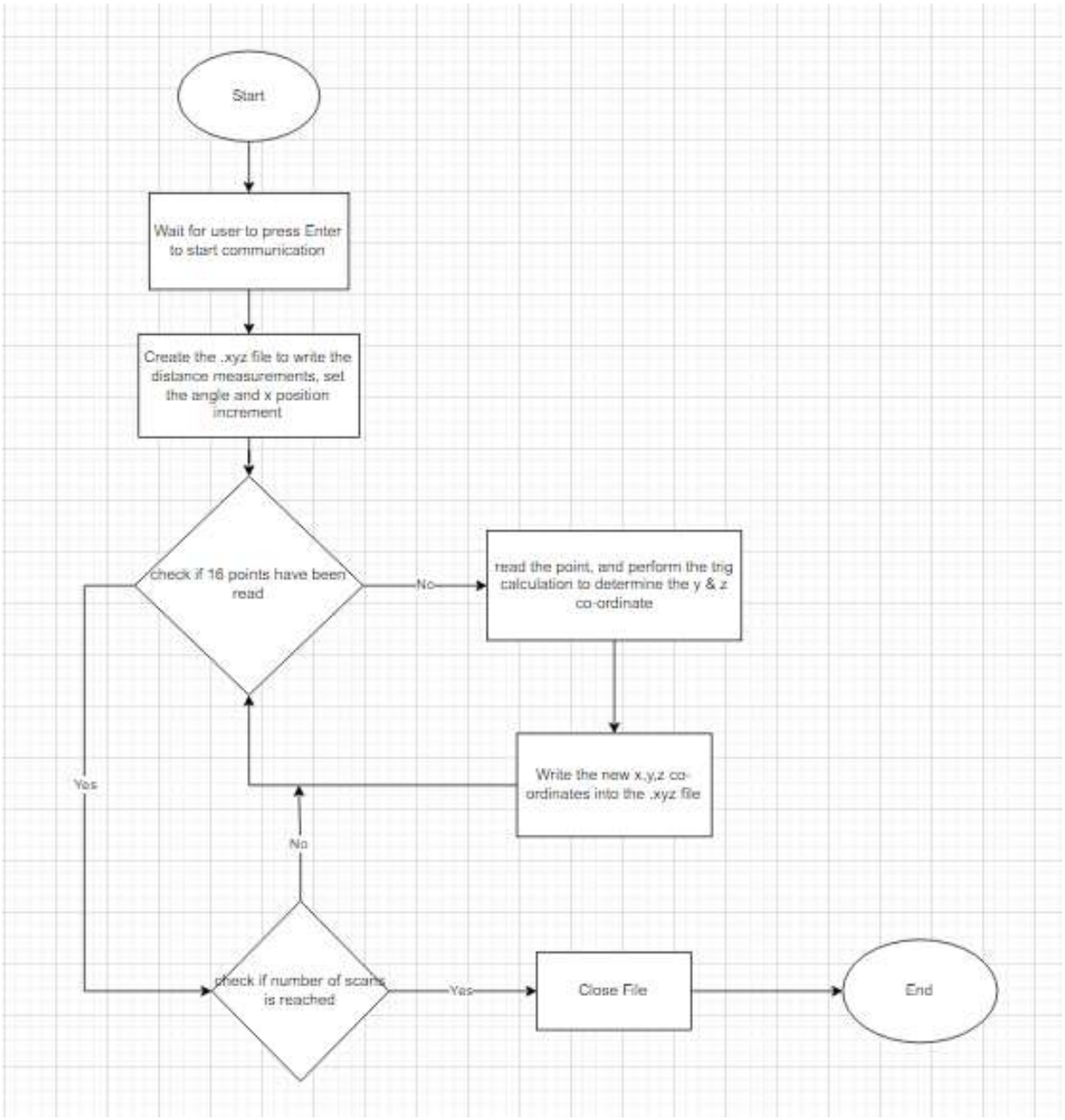
Figure 4: Embedded System circuit schematic

Programming Logic Flowchart

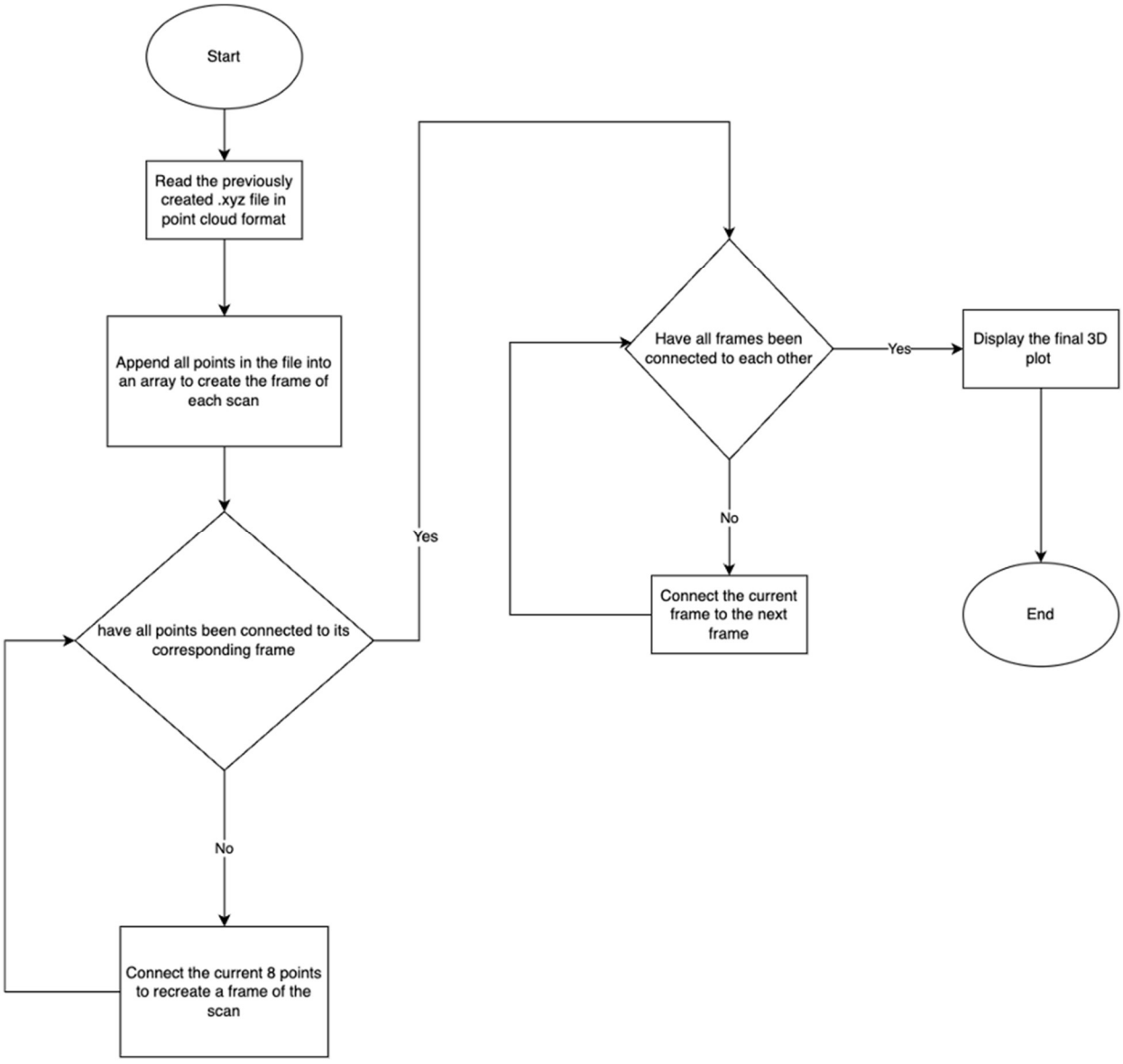
Keil Flowchart – Logic for ToF to take measurements and move stepper motor



Python flowchart – process distance measurements and write to file



3D Visualizer Flowchart – Plot the coordinates written to the file and create the 3D model



References

[1] “VL53L1X,” Avenue to Learn.
https://avenue.cllmcmaster.ca/d2l/le/content/557632/viewContent/4481182/View

[2] “MSP432E401Y - Microcontroller Data Sheet,” Avenue to Learn.
https://avenue.cllmcmaster.ca/d2l/le/content/557632/viewContent/4481116/View.

[3] “2023-2024 2DX3 PROJECT SPECIFICATION – OBSERVE, REASON, ACT: SPATIAL MAPPING USING TIME-OF-FLIGHT,” Avenue to Learn, Jan. 19, 2024.
https://avenue.cllmcmaster.ca/d2l/le/content/557632/viewContent/4481221/View