Final Report - Datasheet

COMPENG 2DX3
Instructors: Dr. Haddara, Dr. Doyle, Dr. Athar

April 9th, 2025
Edison He (400449140)

HE400449140 LiDAR Room Scanner

Summary of Device Overview

The HE400449140 is a hallway scanner making use of LiDAR. A Texas Instrument MSP-432E401Y microprocessor controls all components consisting of a VL53l1X Time-of-Flight sensor, 28BYJ-48 Stepper Motor and ULN2003 Driver Board. The ToF sensor is mounted to the Stepper Motor to capture 360° frames of a hallway using distance and angle.

Features

- **115200 BPS Baud Rate**
- **12 MHz Operational speed**
- **Status LEDs D1 (start), D2 (stepper motor rotation), D3 (measurement verification from ToF)**
- **Counter rotation for wire care on reset**
- **Automatic 3D scan visualization via Open3D**
- **I2C communication link between ToF and Microprocessor**
- **UART communication link between Microprocessor and PC**
- **Dual push button operation for start and end**
- **Device status and operation on PC terminal using UART**

MSP-432E401Y Microprocessor

Parameter	**Value/pin**
Internal Clock Speed	12 MHz
Baud Rate	115200 BPs
Serial Port	COM10
Push Button	PJ0 & PJ1

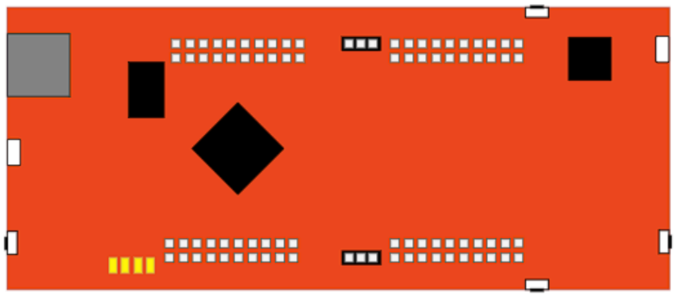
Time-of-Flight Sensor

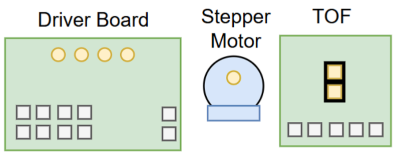
MSP-432E401Y Pin	**VL53l1X Sensor Pin**
PB2	SDA
PB3	SCL
3.3V	Vin
GND	GND

Driver Board/Stepper Motor

MSP-432E401Y Pin	**ULN2003 Driver Board Pin**
PH0	In1
PH1	In2
PH2	In3
PH3	In4
5V	V_+
GND	V_-

MSP-432E401Y





1. Full Device Overview

Below are the full feature sets, general descriptions and block diagrams of the Texas Instruments MSP-EXP432E401Y, VL53L1X Time-of-Flight Sensor, 28BYJ-48 Stepper Motor, 3D Visualization, and Serial Communication components.

1.1 Features Set

Texas Instruments MSP-EXP432E401Y

- 32-bit Cortex M4F CPU
- 256 kB of static random access memory
- 1024 kB of Flash Memory
- Up to 120 MHz bus speed, default 12 MHz
- Onboard Push Buttons
- Onboard LEDs
- General Purpose Input/Output Pins (GPIO)
- $73.18
- 1.4V - 3.6V operating range

VL53L1X Time-of-Flight Sensor

- Max range of up to 3m
- Adjustable Field of View from 15° to 27°
- I2C interface, up to 400 kHz
- 50Hz operating speed
- $14.95
- Laser Class 1, safe for eyes
- 2.6V - 3.5V operating range

3D Visualization

- Python programing language with Numpy and Open3D libraries
- Visual Studio Code terminal for status
- Full Interactive 3D model of the scan
- Instantaneous generation

28BYJ-48 Stepper Motor

- Full step, Half step, and wave drive modes
- 4096 steps per revolution in half-step and 2048 steps for full-step mode
- 64:1 gear reduction ratio
- $4.19
- 5V DC

Serial Communication

- I2C protocol for communication between ToF and microprocessor
- UART protocol for communication between microprocessor and PC
- 115200 BPs UART baud rate

1.2 General Description

The HE400449140 LiDAR Room Scanner provides a low cost solution and a simple learning curve for the user. Driven by the programmable Texas Instruments MSP-EXP432E401Y microprocessor, the system is able to capture a full 360° degree frame in the yz plane. The VL53L1X Time-of-Flight Sensor is attached to the 28BYJ-48 Stepper Motor in 32 increment measurements of 11.25°. To start the scan, PJ0 is pressed and D1 will blink once. When active, D2 will blink for the travel duration. D3 blinks in the ToF scanning process including transmission via I2C. This is repeated until the full 360° degree frame is completed. The microprocessor transmits the data to PC via UART while ToF rotates in the opposite direction for wire management. Pressing PJ0 again will initiate another frame scan. When the user is content with the number for frames, PJ1 may be pressed to initialize point cloud visualization.

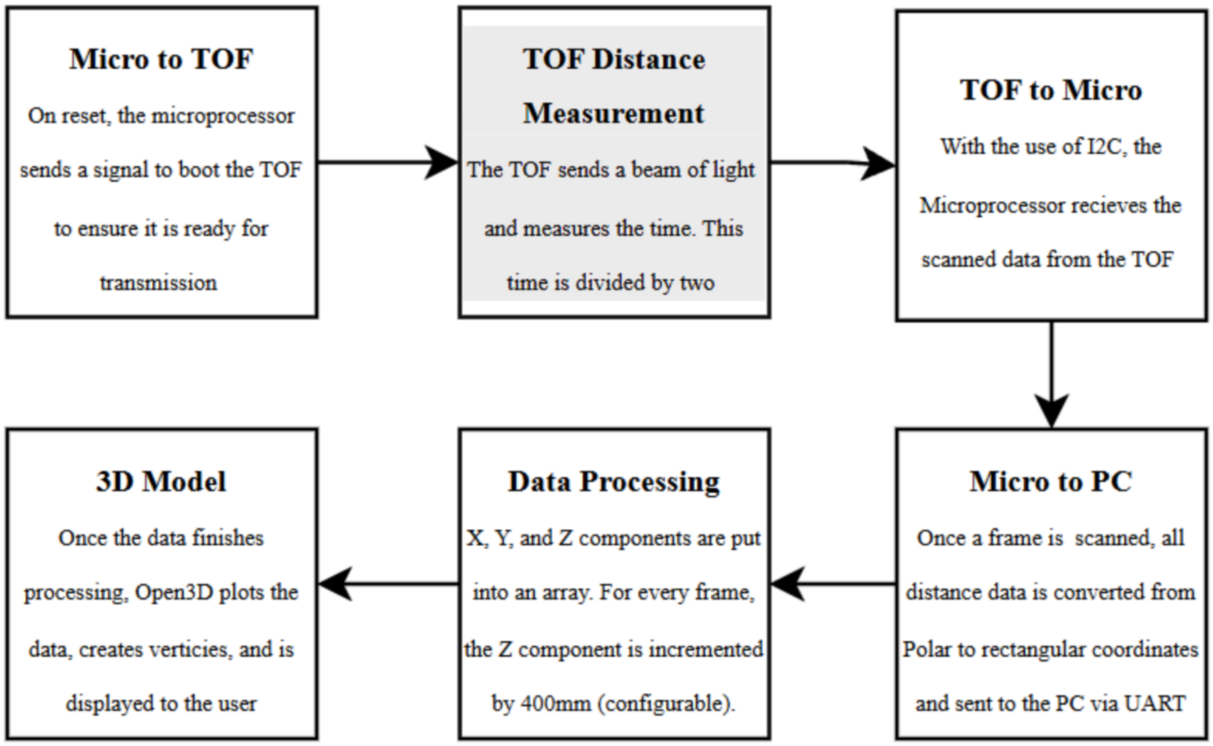
The microprocessor is kept at a bus frequency of 12 MHz, but can be configured to up to 120 MHz. The ToF sensor emits a beam of light which bounces off the object and back to the sensor. Given distance by the ToF and angle tracking with the microprocessor, the microprocessor converts polar coordinates to rectangular. These equations are given below.

$$Distance = \frac{Photon\,Travel\,Time}{2} \times Speed\ of\ Light$$

$$x = Distance \times cos(rad)$$
$$y = Distance \times sin(rad)$$

The data is sent to the PC, where Open3D plots the pointcloud and constructs vertices between each point. The Z component is incremented 400mm (configurable) every frame and is configurable in the Python code. Once these steps are finished, a 3D model is displayed to the user.

1.3 Block Diagram



Micro to TOF
On reset, the microprocessor sends a signal to boot the TOF to ensure it is ready for transmission
TOF Distance Measurement
The TOF sends a beam of light and measures the time. This time is divided by two
TOF to Micro
With the use of I2C, the Microprocessor recieves the scanned data from the TOF
Micro to PC
Once a frame is scanned, all distance data is converted from Polar to rectangular coordinates and sent to the PC via UART
Data Processing
X, Y, and Z components are put into an array. For every frame, the Z component is incremented by 400mm (configurable).
3D Model
Once the data finishes processing, Open3D plots the data, creates verticies, and is displayed to the user

2. Device Characteristics

This section covers the Device Characteristics table of the Texas Instruments MSP-EXP432E401Y, VL53L1X Time-of-Flight Sensor, 28BYJ-48 Stepper Motor.

MSP-432E401Y Microprocessor	
Parameter	**Value/pin**
Internal Clock Speed	12 MHz
Baud Rate	115200 BPs
Serial Port	COM10
Push Button	PJ0 & PJ1

Time-of-Flight Sensor I2C	
MSP-432E401Y Pin	**Vl53l1X Sensor Pin**
PB2	SDA
PB3	SCL
3.3V	Vin
GND	GND

Driver Board/Stepper Motor	
MSP-432E401Y Pin	**ULN2003 Driver Board Pin**
PH0	In1
PH1	In2
PH2	In3
PH3	In4
5V	V_+
GND	V_-

3. Detailed Description

This section contains a detailed description of the distance measurement via the Vl53l1X Time-of-Flight sensor and visualization of the point cloud via PC, Python and Open3D.

3.1 Distance Measurement Via Vl53l1X ToF

The VL53L1X Time-of-Flight (ToF) sensor, developed by STMicroelectronics, operates by emitting a beam of infrared light from its onboard emitter. This beam bounces off objects in proximity and returns to the sensor's receiver. The time measured from the emission and reception process is is used to compute the distance to the object using the following formula:

$$Distance\ = \frac{Photon\ Travel\ Time}{2} \times Speed\ of\ Light$$

After capturing the time-of-flight measurement, the sensor performs a series of operations. These include transduction, signal conditioning, and analog-to-digital conversion(ADC). The transduction of the signal is from the ToF, which is conditioned and converted by the microprocessor.This produces a digital distance reading. This data is then transmitted to a microprocessor using the I2C serial communication protocol.

3.2 Microprocessor Integration and Logic

Upon initialization, the ToF sensor is activated via the boot function in the Keil C program. The sensor does not measure data immediately and is on standby. The microprocessor awaits user interaction via interrupts on Port J, the onboard pushbuttons.

- **Push Button PM0:** Controls the rotation of a stepper motor. When pressed, Start is set to 1, *"Scan_Started\r\n"* is sent to the PC, D1 blinks, and the motor rotates a full 360 degrees.
- **Push Button PM1:** Toggles a variable named End. The system is now ready to transmit the data to the PC via UART and sends *"Scan_End\r\n"*.

The main control loop monitors the position of the stepper motor. The stepper motor requires 512 steps to complete one full rotation (360°). Each step traverses around 0.7031°. The motor traverses 11.25° while toggling LED 2, 32 times (since 32*11.25 = 360°). After every 11.25° the ToF scans for data which if successful sets the dataReady flag to 1. In addition, it increments the angle multiplier “i” by 1 to keep track of the stepper motor’s position with respect to angle. The microprocessor waits for the dataReady flag to toggle while flashing LED 3. From the API function, **RangeStatus**, **Distance**, and **SpanNum** are retrieved. After every scan, the Polar coordinates (distance and degree) are converted to Rectangular with trigonomic relationship seen in the equation below. These points are saved in “x” and “y”, and sent to the PC via UART.

$$x = Distance \times cos(rad) \qquad\qquad y = Distance \times sin(rad)$$

3.3 Data Processing and Visualization in Python

On the host computer, Python and Visual Studio Code handles serial communication which is used to retrieve the data. It connects to COM10 and parses incoming data to an array. After every frame of data received, the z axis is incremented by 400mm, but can be altered by the user. At this stage, the xyz coordinates have been established and can be plotted using Open3D. Vertices are traced between every point and the final output is visualized to the user with the *“draw_geometries”* function.

4. Application Note, Instruction and Expected Output

This section includes a user guide and setup of the system, including an example of a hallway and expected output.

4.1 Application

LiDAR mapping devices take distance measurements and create 3D environments. It has the ability to create high resolution and models which are beneficial to many applications. This includes mapping terrain, autonomous vehicles for an obstacle and collision detection, and forestry, where it may be used to identify fire hazards or unsafe terrain leading to landslides.

4.2 Instructions

This instruction manual assumes all necessary software is downloaded with the correct version on a Windows 10 or 11 system. Keil Development environment, Visual Studio Code, Python 3.12, Numpy, Open3D are necessary. The Texas Instruments MSP-432E401Y Microprocessor, VL53L1X Time-of-Flight Sensor, 28BYJ-48 Stepper Motor, and circuit components must also be configured properly.

1) Plug the microprocessor into any USB type-A port on PC and a USB type Micro B to the microprocessor and determine the COM port. To do this, search for *"Device Manager"* in start or hit the Windows key. In device manager, scroll to *"Ports (COM & LPT)"* and expand the submenu. Take note of the COM port beside *"XDS110 Class Application/User UART"*.
2) Open *"2DX3_Python_Code"* using Visual Studio Code and change "*COM#"* to the proper value determined in step 1. Run the program and press the ENTER key on the PC. Serial Communications are now set up.
3) If the circuit has not been set up, connect the corresponding pins found in Section 2 of this manual, **Device Characteristics.**
4) Open "*2DX3_Keil_Code"* and in sequential order, Translate, Build and Load the project.
5) Reset the project code by pressing the button beside the USB micro B port. The Visual Studio Code terminal will receive startup prompts.
6) To start the scan, press PJ0. Wait until one frame of the scan completes and the ToF sensor returns to its original position. If more frames are required, repeat this step.
7) To end the scan, press PJ1. A 3D visualization of the scan will appear on the PC.

4.3 Expected Output

An example scan location is present in Figure 1. This is Location A in the Information Technology Building at McMaster University. Figure 2 is a multiview of the Final 3D model of the scan. With 32 scan segments every frame, the accuracy of the system can be seen. In the Figure 2 front view, it can be seen that the corners of the roof were detected and angled in Figure 1. In addition, the cubby, and the open doors out of view on the left side were out of the 3m range and were not detected. Finally in the side view facing down in Figure 2, the back can be seen at the top of the image. This back area also has a missing segment in it. This represents the hallway in the right of Figure 1, which was not detected as it was also out of range.

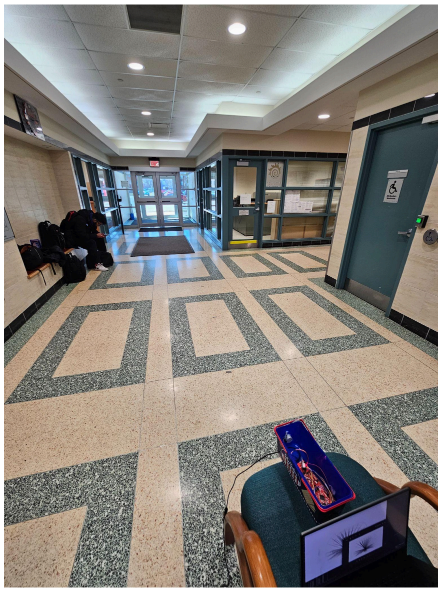


Figure 1: Hallway Located in ITB, Location A

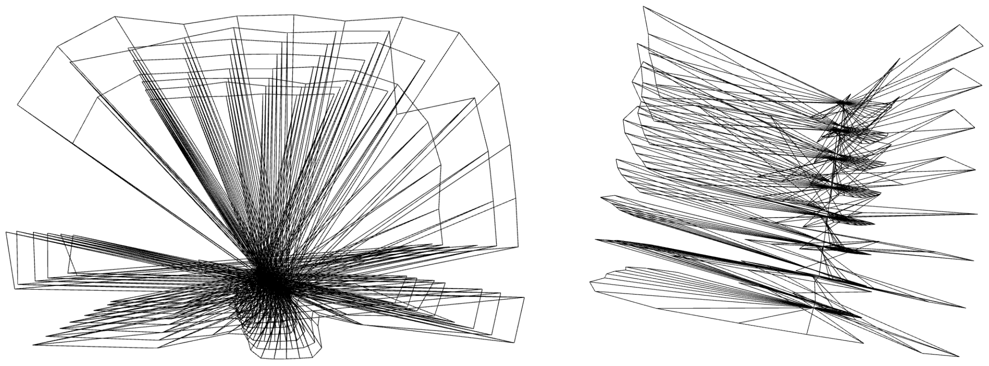


Figure 2: System Scan Front and Right View Facing Down

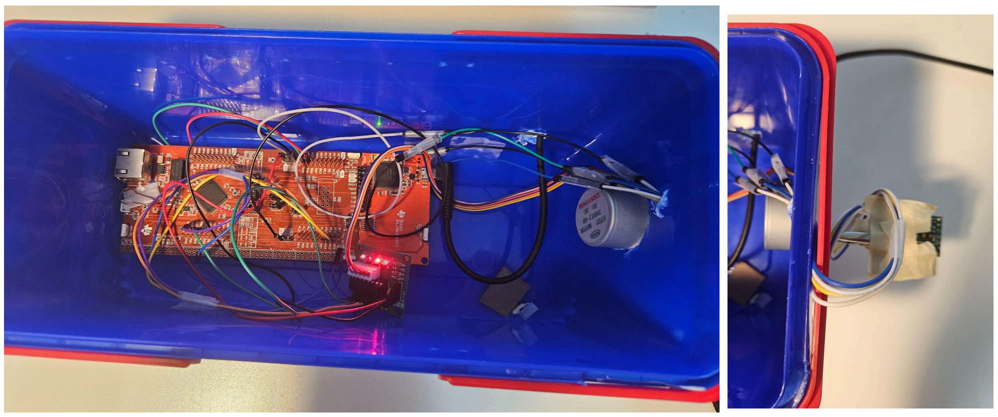


Figure 3: HE400449140 LiDAR Room Scanner

5. Limitations

This section reviews limitations of the system including quantization error, and transmission speeds.

5.1 Quantization Error

Maximum quantization error is the maximum difference between the real-life analog input and the closest digital representation the ADC. Max Quantization error is calculated with the formula given below. For context, the maximum range of the ToF is 3m (3000mm) and uses a 16 bit format.

$$Max\ Quantization\ Error = \frac{Max\ Reading}{2^{\#\ of\ ADC\ bits}}$$

$$Max\ Quantization\ Error = \frac{3000}{2^{16}}$$

$$Max\ Quantization\ Error = 0.04577$$

5.2 Transmission Speeds

The maximum transition speeds may be determined by the host PC or the set maximum baud rate. This can be verified in *Device Manager* under the ports section. BPS is found in XDS110 Class Application/User UART port. On the test computers, the maximum speed is 128 000 BPs, while the max speed implemented in the project is 115200. The max transition speed between the microprocessor and ToF sensor is set by the ToF sensor itself at 50Hz.

5.3 Peripheral Limitations

In addition, scan speeds are also determined by two limitations, the ToF sensor and the stepper motor. The ToF sensor requires a bootup sequence and time allocated for a scan. However, the biggest speed limitation of the scanning sequence is the stepper motor due to its larger delay by nature. Each movement phase requires a delay between all 4 phases. This was set to 10ms, or 40 ms total. Although decreasing the delay between phases increases the rotation speed, an insufficient delay does not allocate enough time for rotation. After testing and decreasing the delay, the lowest delay that can be set per phase is around 3ms, or 12 ms total which can be set which may be different across other devices. However, for simplicity and reliability, 10ms per phase was used.

5.4 Cortex M4F Limitations

The Texas Instruments MSP432E401Y has an ARM cortex M4F cpu. It has a FPU (Floating Point Unit) that assists with mathematical calculations. On the microprocessor, the FPU is limited to 32-bit operations in a single precision. Because of this, it cannot perform trigonometric calculations. This is the reason the calculations are performed in the Keil Code based on C++, rather than onboard. It does not have the correct number of decimals to be used for trigonometric calculations. In this case, it is unable to calculate sine and cosine functions.

6. Circuit Schematic

Below is the circuit schematic of the system. The left and right images are zoomed in segments of the connection.

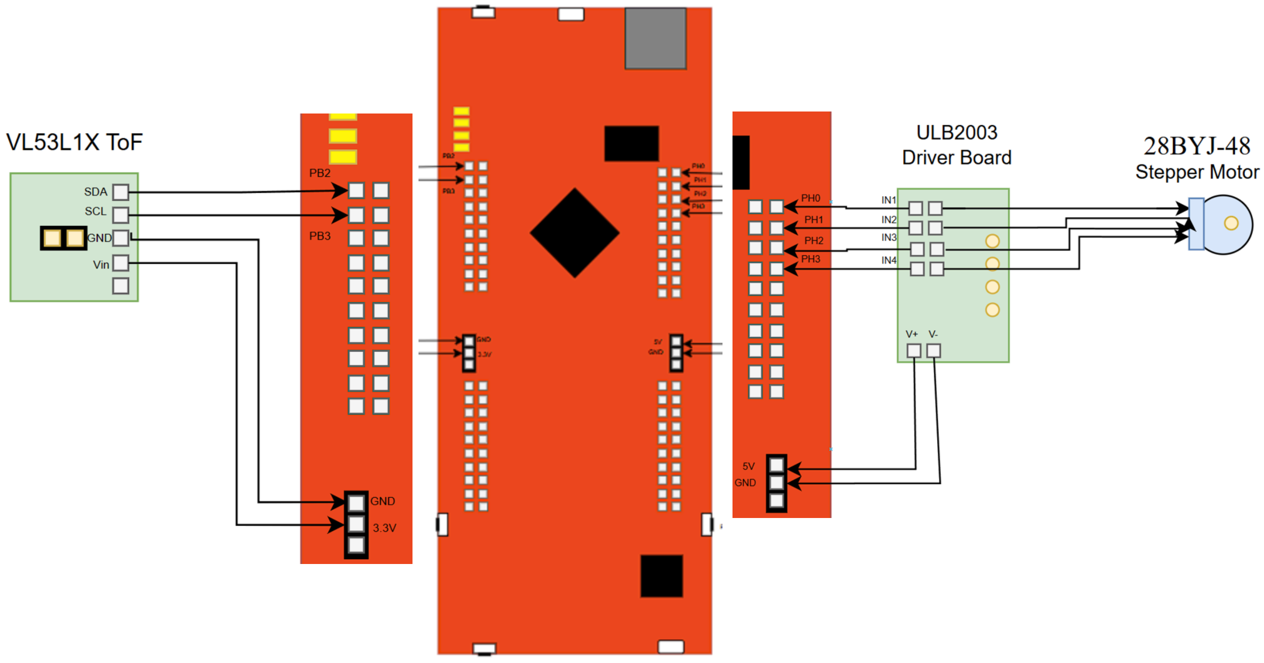


Figure 4: Circuit Schematic

7. Programing Logic Flowchart

Below is the Programming Logic Flowchart in Keil for the Onboard System and the Python Flowchart..

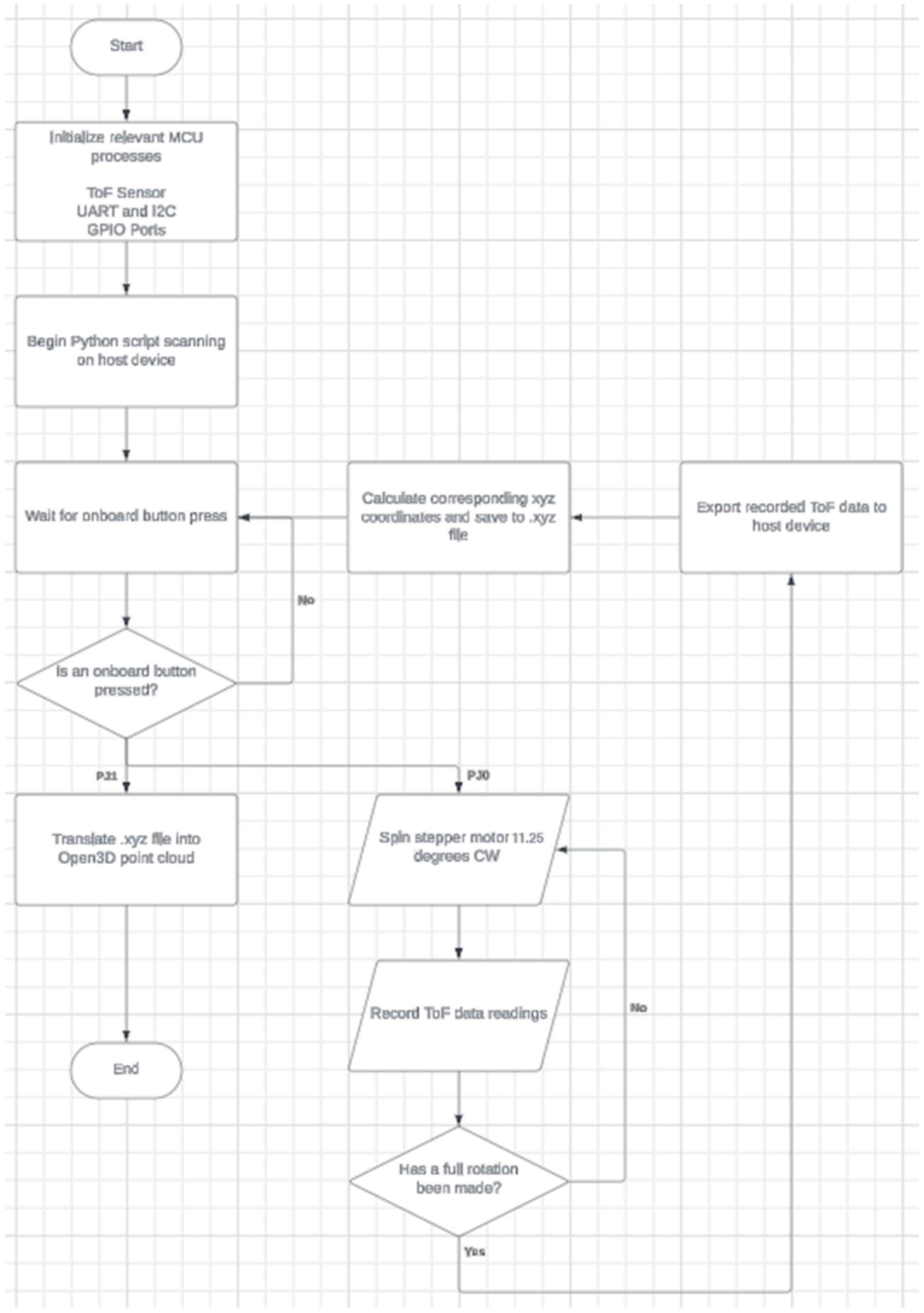

Start
Initialize relevant MCU processes
ToF Sensor
UART and I2C
GPIO Ports
Begin Python script scanning on host device
Wait for onboard button press
Calculate corresponding xyz coordinates and save to .xyz file
Export recorded ToF data to host device
No
Is an onboard button pressed?
PJ1
PJ0
Translate .xyz file into Open3D point cloud
Spin stepper motor 11.25 degrees CW
Record ToF data readings
No
End
Has a full rotation been made?
Yes

Figure 5: Keil Code Flowchart

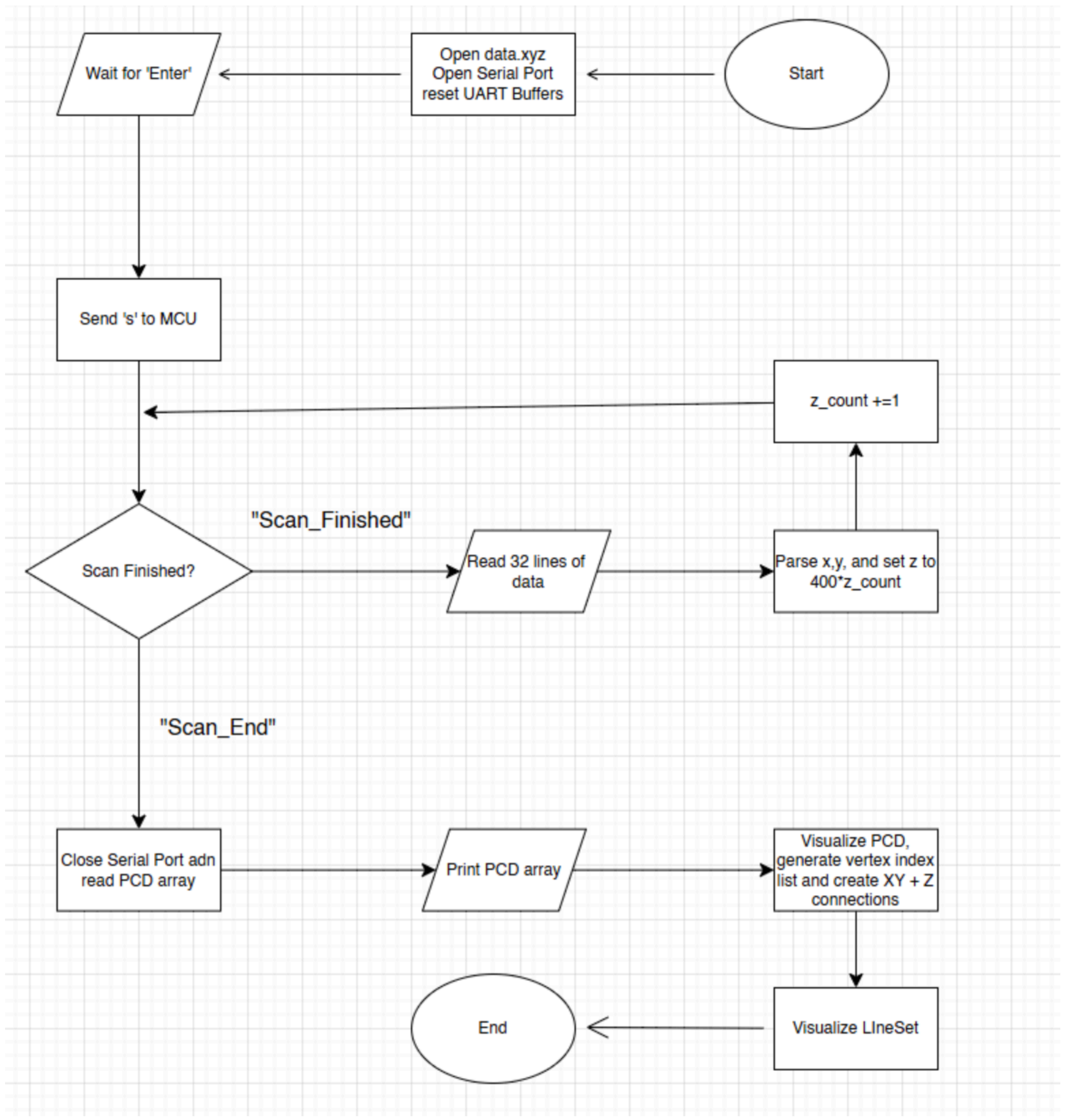


Figure 10: Python Code Flowchart