Spatial Mapping Using Time-of-Flight

Final Project Report

COMP ENG 2DX3 Microprocessor Systems

Instructors: Dr. Athar, Dr. Doyle, Dr. Haddara

Brent Menheere - menheerb – 400362843 – L03

April 17th, 2023

Contents

Device Overview

Features:

Texas Instruments MSP432E401Y

- 120 MHz Clock/Bus Speed
- 2.5 V – 5.5 V Operating Voltage Range
- 115200 Baud Rate
- 1024 KB Flash Memory
- C/C++ Capable
- 15 GPIO Blocks

VL53L1X ToF Sensor:

- 400 cm Maximum Distance Measurement
- I2C Interface
- 940 nm Invisible Laser Emitter
- 50 Hz Ranging Frequency
- Single Power Supply

General Description:

This built LiDAR system utilizes the Texas Instruments MSP432E401Y microcontroller in collaboration with the VL53L1X time of flight (ToF) sensor. When operating this system, the user starts by enabling the rotation of a stepper motor, in which the ToF sensor is mounted on, by pressing button 0. Once the user decides that they are ready to start taking measurements, button 1 can be pressed. The stepper motor rotates to capture a vertical slice of the hallway. Every 11.25 degrees a measurement is taken, and every 360 degrees captured, the user will be prompted to take a step forward of about 70 cm. At this step forward point the stepper motor will also switch directions to stop the cables connected to the ToF from being twisted and tangled. The user is in full control of when to stop collecting data and can be controlled with button 1. All the above was programmed in C coding language and flashed onto the microcontroller.

The ToF sensor transmits the data collected in real time via I2C communication to the microcontroller, which in-turn transmits the data to the users PC via UART communication. The UART data is intercepted by a Python program which converts all data transmitted into XYZ coordinates which can then be plotted by the Open3D software. Once plotted the user is presented with a 3D diagram of the hallway of interest.

Block Diagram:

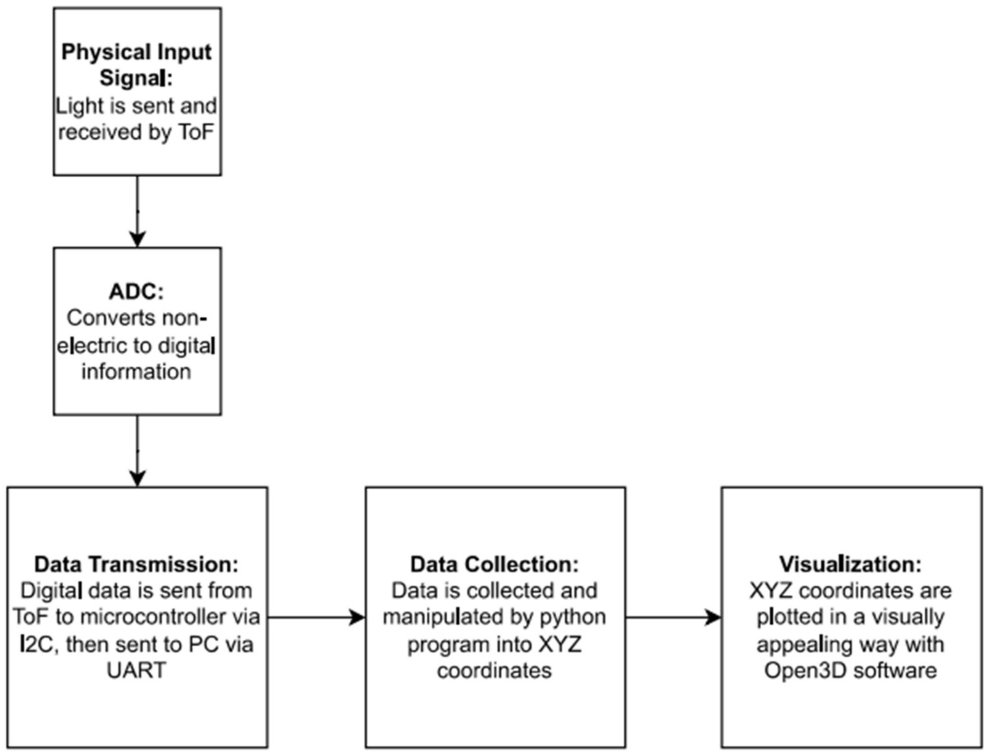
Physical Input Signal: Light is sent and received by ToF
ADC: Converts non-electric to digital information
Data Transmission: Digital data is sent from ToF to microcontroller via I2C, then sent to PC via UART
Data Collection: Data is collected and manipulated by python program into XYZ coordinates
Visualization: XYZ coordinates are plotted in a visually appealing way with Open3D software

Figure 1 Data Flow Graph

Device Characteristics Table

In the table below the user can find the corresponding ports and other specifics relating to the main characteristics of this system.

Characteristic	**Correspondence**
ToF Sensor	V_{IN} (3.3 V), GND, SDA (PB3), SCL (PB2)
Button 1 (Motor Control)	PM0, polling configured, active low
Button 2 (Sensor Control)	PM1, polling configured, active low
Bus Speed	30 MHz
Stepper Motor	5 V, GND, IN1 (PH0), IN2 (PH1), IN3 (PH2), IN4 (PH3)

Detailed Description

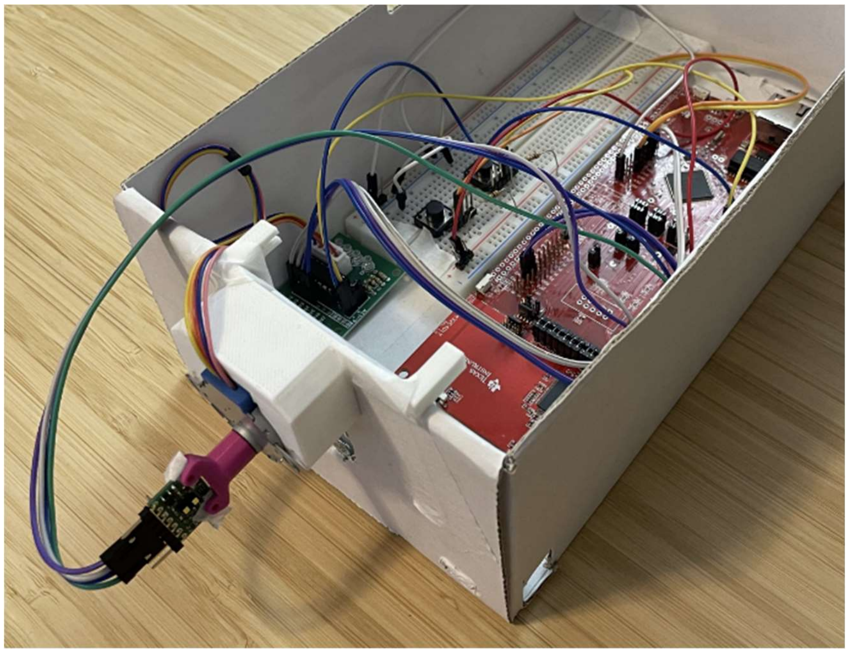

Figure 2 Final Project

Distance Measurement:

To take distance measurements, this system utilizes the VL53L1X ToF sensor. This sensor uses a 940 nm invisible laser transmitter to send a photon in its surrounding environment. This photon will reflect off of the nearest object in it's path and return to the sensor via the receiver. Internally, on the ToF chip, the sensor determines the distance between the nearest object and itself. To do this the sensor utilizes the known speed of light and the time it took for the photon to be emitted and then received.

Measured Distance = (Photon travel time / 2) * Speed of light

After this calculation is done, it is then manipulated to be displayed in mm by multiplying the result by 10^-3. The data is then ready to be transmitted via I2C communication to the Texas Instruments MSP432E401Y microcontroller which in turn transmits the data via UART communication to the user's PC.

For the user to control the collection of data, 2 active low push buttons were implemented and connected to ports PM0 and PM1. Within the Keil code, using a polling method, the C code is constantly checking whether button 0 or button 1 is pressed. In the project specifications we were tasked with implementing 2 buttons to control the stepper motor and then collection of data separately. To start this system, the user pressed button 0 which will start the rotation of the stepper motor. When the user is ready to start taking distance measurements button 1 should be pressed. The working principle of these controls is that when the system detects a low signal from the respective button, a loop is entered and an enable variable is toggled which is allows access into other loops which control either the rotation of the motor or scanning. When button 1 is enabled, the system enters scanning mode. Every 11.25-degrees of rotation a measurement is taken and transmitted to the PC. Once 360-degrees of measurement have been acquired for that specific depth, on the user's PC they will be promoted, "Take a step forward", and the stepper motor with switch rotation direction to prevent cables from being tangled. Once the user wants to stop data measurement and collection, button 1 can be pressed again. This will toggle the ToF enable in the C program, scanning mode will be turned off, and the original stepper motor state will be returned and can be turned off by pressing button 0 again.

Visualization:

The above section described the physical actions and background processes that take place onboard the MSP432E401Y microcontroller. After the data is transmitted to the users PC it is handled by a python program. When the user runs the Data Grab python program, their COM port is opened and an '.xyz' is created/opened to store the data for Open3D. When the scanning button on the device is pressed, the PC is sent a start message which tells the program to start expecting valid data to be used in the plot. After this point the program enters a loop which takes each iteration of data passed and converts the single distance value into (x,y,z) format. The y and z values make up the vertical slice of our hallway, while x represents the depth of the hallway. To convert to y and z coordinates the following operations are used:

```
y = distance * math.cos(math.radians(angle))

z = distance * math.sin(math.radians(angle))
```

where angle is incremented by 11.25-degrees after each measurement. To add the x coordinate, x is assigned a variable depth. Since we want to create a full 360-degree slice of our hallway for each step, every 32 measurements the depth variable is increased by our step size, currently 700 mm (70 cm). This in turn creates 32 measurements all at the same x coordinate and gives us our series of slices. Once the user has achieved their ideal number of slices and pressed button 1 again, the C program sends a stop message to the PC which breaks the python program out of the while loop. The program then closes the COM port, closes/saves the '. xyz' file, and displays the generated image via Open3D. The corresponding image should consist of a cloud of points with each point in a slice connected by a line and every point with the same angle in all the slices connected by a line. The result is a frame like structure which resembles a hallway.

Instructions & Application Example

Setup:

1. Connect system hardware, microcontroller, ToF sensor, and push buttons, in respect to the device characteristic table and circuit schematic (Figure 5). Note that both push buttons are in an active low configuration. A schematic of this configuration can be observed in Figure 6.
2. ToF sensor should be mounted to the stepper motor using mounting hardware of choice, and stepper motor should be mounted to a rigid body. An example of a mounting option can be seen in Figure 2**.**
3. After connecting the microcontroller to the users PC via micro-USB, the C program can be flashed onto the microcontroller using Keil IDE. Once flashed the user should reset the device.
4. To setup the python program, the user must open the Data Grab file on IDLE (or any python IDE). To ensure this program functions, the user must find their COM port which allows for UART communication. This can be done in Device Manager > Ports (COM & LPT) > XDS110 Class Application/User UART (COM '#'). In line 61 of the code, 'COM4' should be changed to the user's specific port.
5. At this point all setup has been complete. The user can click run on the python program and should be prompted with "Press Enter to start communication...". If not, please thoroughly check steps 1 – 4 to ensure proper setup.
6. When the user is ready to start scanning, press enter on the keyboard, and then start the rotation of the stepper motor by pressing Button 0. When ready the user also must press Button 1 to start taking scans. It is recommended that the user presses Button 1 close to the moment the stepper changes rotation direction.
7. At this point the user should observe the points being printed on their PC. Every 360-degree scan, the user will be prompted to take a step forward. The default step size is 70 cm; however, it can be changed in line 99 of the Data Grab program.
8. Once the user has completed their choice of scans. Button 2 can be pressed again, which will stop the scanning, stop the python program, and open the 3D image of their scan. It is recommended that the user presses Button 1 to stop after the program prompts them to take a step forward, as this ensures the most recent 360-degree frame is completed and properly plotted.

Application Example:

To test the functionality of my system, I was assigned a hallway on the second floor of ITB (Location D) to scan and model. The hallway provided was simple enough to test the basic functionality of my system, however it is also seen in Figure 3, that near the beginning of my scan path, there is a cut-out in the wall for a water fountain. For my final design to be a success the 3D image acquired must show the basic ‘rectangular prism’ features and must be able to show the change in dimensions as the water fountain area is scanned.

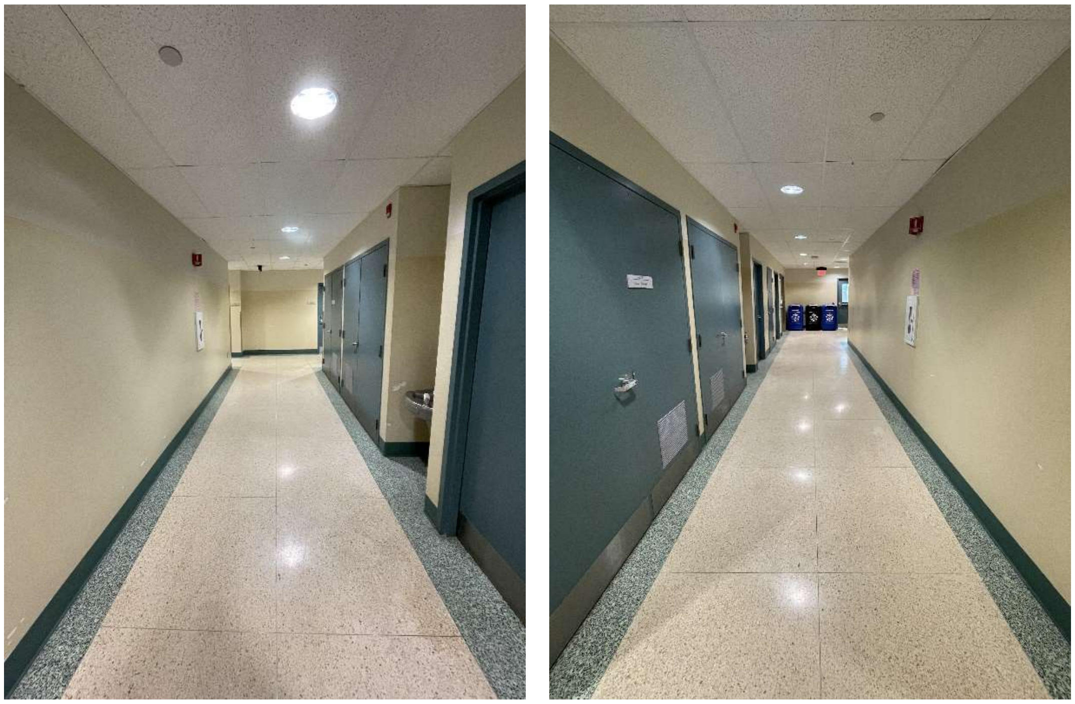

Figure 3 Hallway Scanned

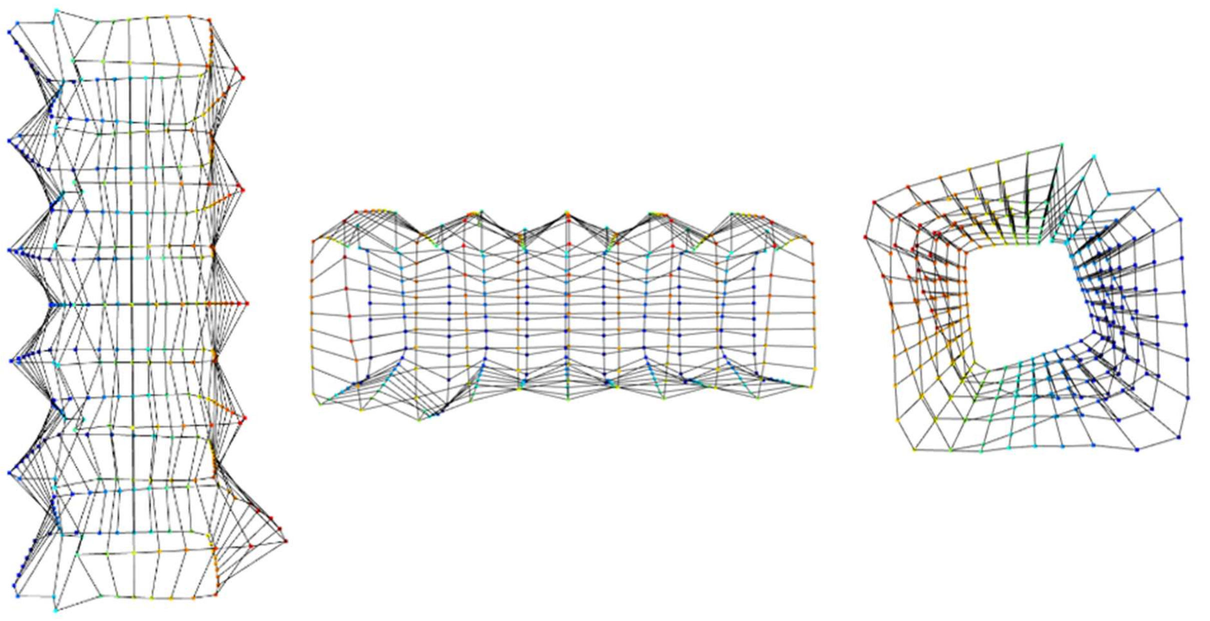

Figure 4 Generated Images from Scan

Shown above is a side-by-side comparison of my assigned campus location vs. my 3D model based on the scanned data. The 3D generated images above reflect the ‘top’, ‘side’, and ‘front’ of the hallway, respectively. It is clear in the top view that there is one section that protrudes out drastically more than any other spot on the image. This point reflects the water fountain area, which was one step after the starting point of my scan. The side and front view of the scan reflect the general shape of a hallway. The two factors classify my system as a success, however its also apparent that these scans don’t precisely reflect the shape and features of the hallway. The is mainly due to human error while taking the scan, such as not walking it an exact straight line, or not always holding the sensor complete level.

Limitations

1. Texas Instruments MSP432E401Y is equipped with the Cortex-M4F 32-bit processor which includes a FPU, floating-point unit. This means that any calculations made was limited to the degree that the numbers being manipulated were only as accurate as the 32-bits total available. Specifically in terms of this project, when using trigonometric functions, such as sine and cosine which usually produce numbers with various number of decimal points, we are limited to our FPU.
2. Maximum Quantization Error for the ToF Module:

   Max Quantization Error = [Max Reading)]/ [2^(# of ADC bits)]

   Max Reading of ToF = 4000 mm

   VL53L1X stores values in 16-bit format

   Max Quantization Error = [4000 mm]/[2^16] = 0.0610

3. The maximum standard communication rate that we can implement with the PC is 115200 bps, which was also the speed that was implemented into this system. To verify that it could not be increased any more, the next highest baud rate available on Realterm, 230400 bps, was tested and resulted in an error.

4. The communication method used between the microcontroller and the ToF module was I2C. I2C is a synchronous communication protocol which provides bidirectional data transfer through a two-wire design made up of the serial data line (SDA) and serial clock line (SCL). This transmission was done at 400 kHz.

5. Reviewing the entire system, the primary limitation on the system speed is microcontroller. As mentioned above the maximum baud rate of the microcontroller was determined to be 115200 bps. When looking at the other elements in this system that have a known transmission speed, we can look at the ToF module. This module is transmitting at 400 kHz, which is equivalent to 400 kbps. I2C has the capability to run in select modes that can run even faster than this rate. Since 115200 bps is significantly less than 400000 bps, we can determine that the microcontroller is the primary limitation on the system speed.

Circuit Schematic

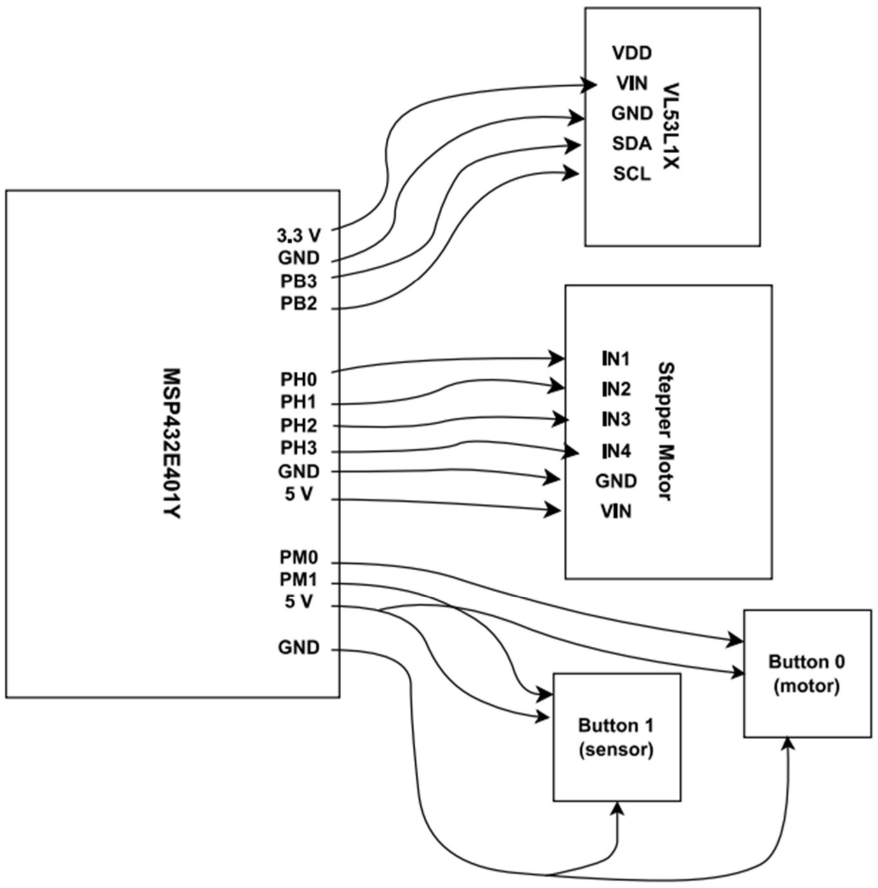

Figure 5 Circuit Schematic

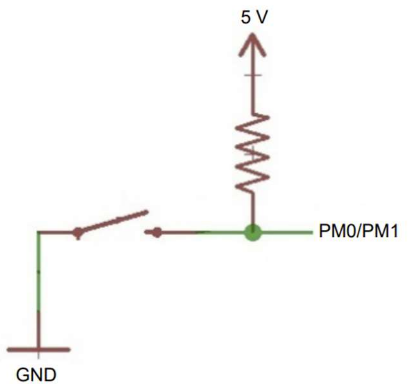

Figure 6 Active Low Configuration

Programming Logic Flowchart

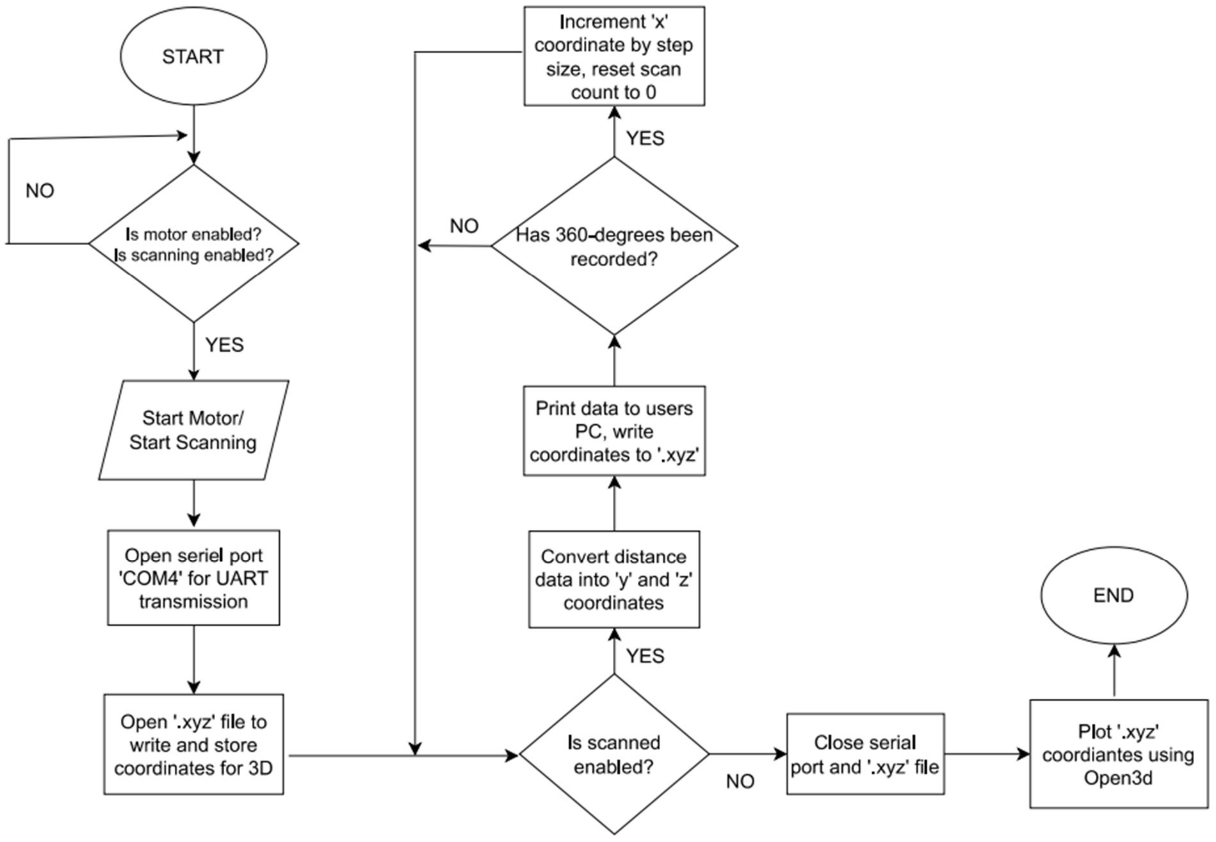
START
Is motor enabled? Is scanning enabled?
NO
YES
Start Motor/ Start Scanning
Open seriel port 'COM4' for UART transmission
Open '.xyz' file to write and store coordinates for 3D
Is scanned enabled?
YES
NO
Convert distance data into 'y' and 'z' coordinates
Print data to users PC, write coordinates to '.xyz'
Has 360-degrees been recorded?
NO
YES
Increment 'x' coordinate by step size, reset scan count to 0
Close serial port and '.xyz' file
Plot '.xyz' coordiantes using Open3d
END

Figure 7 Flowchart of System