Electrical and Computer Engineering

Project Report

COMPENG 2DX3: Microprocessor Systems Project

Abdalla Mahdy, 400411114

1. Device overview

The developed device is a highly efficient and cost-effective solution for capturing 3D environments using a time-of-flight (ToF) sensor and a Texas Instruments MSP432E401Y microcontroller. The ToF sensor emits light, which reflects off the surface and returns to the sensor, enabling distance measurement using the formula:

$$Distance = \ Speed\ of\ light * (\frac{photon\ travel\ time}{2})$$

To capture the x and z axes in 2D and the fixed y-axis in the third dimension, the device uses a rotating unipolar stepper motor. The microcontroller is programmed to control the stepper motor and collect distance measurements from the ToF sensor every 22.5 degrees. These measurements are then sent to a computer via the serial UART port.

With a bus/clock speed of 120 MHz, the microcontroller comes with 4 LEDs, 2 interacting switches, a reset switch, and 40 different functionality input/output pins. It has a wide operating voltage range of 2.5-5.5 V, 1024 KB flash memory, 256 KB SRAM, and 6 KB EEPROM. Additionally, it has 2 12-bit SAR-based ADC modules, 16 digital comparators, 8 UARTS each with an independent transmitter and receiver, and 10 I2Cs with high-speed support.

The ToF sensor, on the other hand, has an operating voltage range of 2.6-3.5 V and uses a single power supply (2v8). Emitting a 940 nm invisible laser, it has a SPAD receiving array with an integrated lens. It can measure a maximum distance of 400 cm and has a maximum ranging frequency of 50 Hz. The sensor has an I2C interface for up to 400 kHz.

After collecting the distance measurements, a Python code processes the data to combine the 2D plots for the x and z axes with the changing values of the y-axis to generate the final 3D model as the output. The data is saved in a '.xyz' file format that can be displayed graphically. The user can move the device in the x-direction and press the onboard button to repeat the capturing process until the desired number of planes is achieved. The program then uses the Open3D Python add-on to create a point cloud and connect lines between the points for a more easily interpreted 3D visualization of the scanned environment.

The device's cost-effective and accurate solution for capturing 3D environments makes it useful in various industries such as construction, engineering, and design. The use of a stepper motor and a ToF sensor enables the device to capture highly precise and accurate distance measurements, making it a suitable alternative to complex and expensive LIDAR devices. Overall, the developed device is an excellent example of the innovative and practical use of technology in the industry.

2. Device Characteristics Table

Characteristic	Description
Stepper Motor	V_{in} = 5V, GND, Ports H0:H3
Time of Flight sensor	V_{in} = 3.3V, GND, SCL = PB2, SDA = PB3
On-Board Push Button	Port J1 (Configured with Interrupts)
Computer Port Input	COM4 (user-specific)
Serial Communication Speed	115200 Baud rate (bps)
Bus Speed	12 MHz (student-specific)
ToF–Microcontroller communication	I2C protocol (serial)
Microcontroller–Computer communication	UART protocol (serial)
Special Python Libraries	Math, pyserial, NumPy, open3d
C Header Files	PLL, SysTick, tm4c1294ncpdt, vl53l1x_api, uart, stdint

3. Device description

Distance Measurement Process Report:

This report outlines the process of measuring distance using a ToF (Time of Flight) sensor in conjunction with a microcontroller and a stepper motor. The system is designed to measure distances in the positive x-direction, with the aid of a push-button to initiate the measurement process. The purpose of this report is to describe the design and implementation of this distance measuring system, as well as the data visualization method.

System Setup:

The system is set up as seen in the circuit schematic. The first step in the distance measurement process is initializing and configuring all required ports. The system utilizes Port H to output signals to the stepper motor, Port N for the LED indicator light, Port B for I2C, and Port J for the on-board push button. Interrupts are enabled for Port J to sense the button press. The required steps for configuring interrupts are taken, including arming the source of the interrupt, enabling the interrupt in the NVIC, enabling global interrupts, setting the priority level, and writing an appropriate interrupt service routine (ISR).

Distance Measurement Process:

Once the initialization is complete, the system waits for the interrupt created from the push of the PJ1 on-board button. When a button press occurs, the VL53L1X ToF sensor boots up and is configured with the default settings. The ToF then utilizes its 940 nm invisible laser transmitter to send a wavelength of light into its environment. The light reflects off the closest object and eventually returns to the ToF, which then determines the distance between itself and the object by calculating the time it took for the signal to return and using the known value for the speed of light. The distance measurement is obtained in millimeters and saved to a long integer variable to later be sent to the PC via UART. The onboard LED PN0 (student specific) flashes to signal a distance measurement has been observed. The system then checks that the PC is ready to accept the data via the UART port and then sends the data over at the specified baud rate. A Python program running on the PC receives and stores the data on the hard drive for later use. After the data transmission, the microcontroller calls a function to rotate the uni-polar stepper 32 steps or 22.5 degrees. It utilizes the full-step method and activates the proper coils in the stepper motor by outputting HI to the correct pairings of Port H pins in order. After the rotation is complete, the microcontroller signals the sensor and obtains a new distance measurement using the same process outlined above. Once a full rotation is complete, 16 distance measurements have been taken and transmitted to the PC, the motor does a full rotation in the opposite direction to avoid wire entanglement. The system is designed to wait for the user to move forward the specified increment, which is in the positive x-direction. The user can press the push button again, and the entire above process is repeated. This process is repeated until the intended number of cross-sections has been obtained.

Data Visualization:

The data visualization process is carried out by a Python program running on the PC. The program imports required additional libraries, including serial, math, open3d as o3d, and NumPy as np. The program takes the serial port ‘COM4’ (user specific) configured with a baud rate of 115200 bps, equal to that of the microcontroller. This port is then opened and cleared of all potential data remaining. A ‘.xyz’ file is then opened and configured for writing. If the file exists, it will be overwritten, and if it does not exist, it will be created. Variables for the number of storing this value as a float. The x and y coordinates of the point are then calculated using trigonometry, with the distance measurement obtained from the ToF sensor being the hypotenuse and the angle of rotation being the reference angle. These coordinates are then written to the ‘.xyz’ file in the format: X-coordinate Y-coordinate Z-coordinate After all 16 measurements have been taken, the program creates a point cloud using the open3d library and visualizes it. The program then waits for the user to input a value to move forward in the X direction, and the process repeats until the desired number of scans has been taken.

4. System Setup and Usage Guide

This guide outlines the steps required to set up and use the 3D scanning system, which utilizes the MSP432E401Y microcontroller, VL531X ToF sensor, and stepper motor. By following these instructions, you will be able to use the system to create a 3D model of a location (or assigned campus location).

Hardware Setup:

1. Connect the MSP432E401Y microcontroller, VL531X ToF sensor, and stepper motor as shown in the circuit schematic in Figure 3. Make sure to secure the motor and sensor to the motor, as shown in the Built System in Figure 4.
2. Flash the C code to the microcontroller using an IDE such as Keil. Keil will translate, build, and then flash the C code to the microcontroller.
3. Load the code onto the Texas Instruments MSP432E401Y by pressing the onboard reset button.

UART Setup:

1. Check your device manager and search for the COM port using UART. To do this, go to Device Manager > Ports (COM & LPT) > XDS110 Class Application/User UART (COM#). The number value in the # position is what is required. Modify the Python code to have this number instead of the current value 4.
2. Ensure the baud rate is set to the designed 115200 bps.

Scanning Process:

1. Run the Python program using IDLE or your IDE of choice or using the command prompt and type in 'cd (python file location),' then type in 'python (python file name).' Then, answer the question in the console “How many full scans are you going to take?” with an integer value.
2. Begin scanning by going to the beginning of the area you wish to scan and pressing the onboard PJ1 push button. The system will begin scanning and stop automatically when the full rotation is completed, and the motor has completely stopped moving and is in its initial resting position.
3. After each full rotation, move forward the pre-set amount in the forward direction (positive X direction).
4. Repeat steps 2 and 3 until the desired number of scans has been reached. The program will open the point cloud, and you can manipulate the viewpoint of the point cloud by clicking and dragging the cursor in the window.
5. Close the point cloud window to view the final 3D interpretation with connecting lines. You can also manipulate the viewpoint of this image in the same way.

Application Example:

Using the 3D scanning system, you can create a 3D model of any location. This model can be used to analyze the building's structure, create virtual tours, or for other applications.

Instructions:

Follow the steps outlined above to set up and use the 3D scanning system. Make sure to connect the hardware components correctly and configure the UART settings and baud rate as required. Then, run the Python program and follow the prompts to begin scanning.

Expected Output:

After completing the scanning process, the program will open a point cloud of the scanned area, which you can manipulate to view the 3D model. You can also view the final 3D interpretation with connecting lines. You can use this output for further analysis or visualization of the scanned area.

Side-by-Side Comparison:

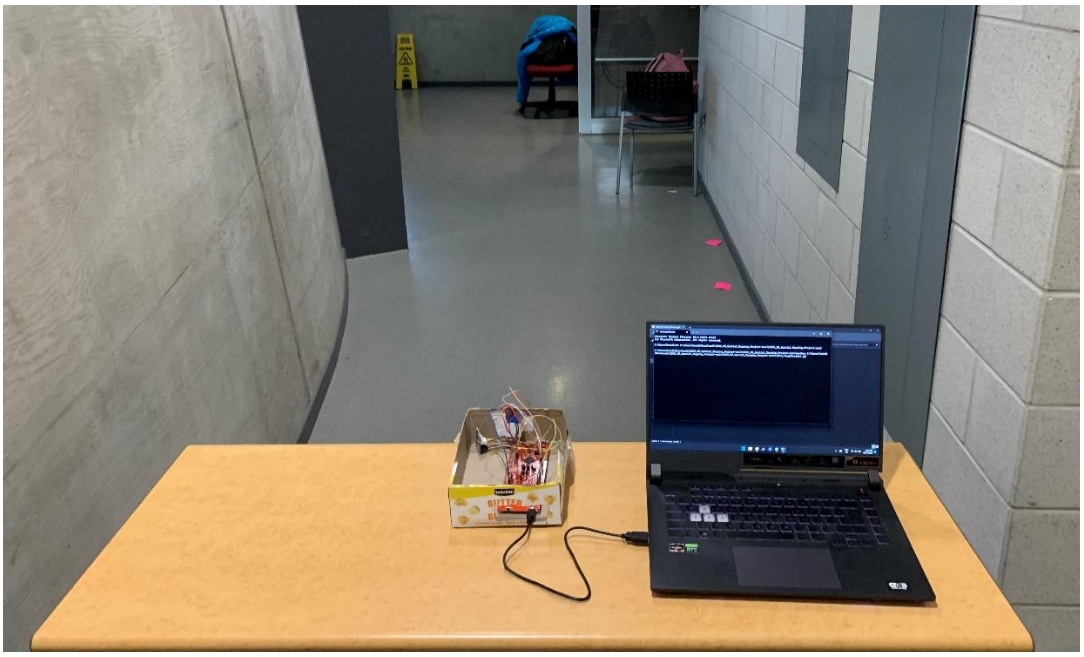

Figure 1: Hallway that was scanned

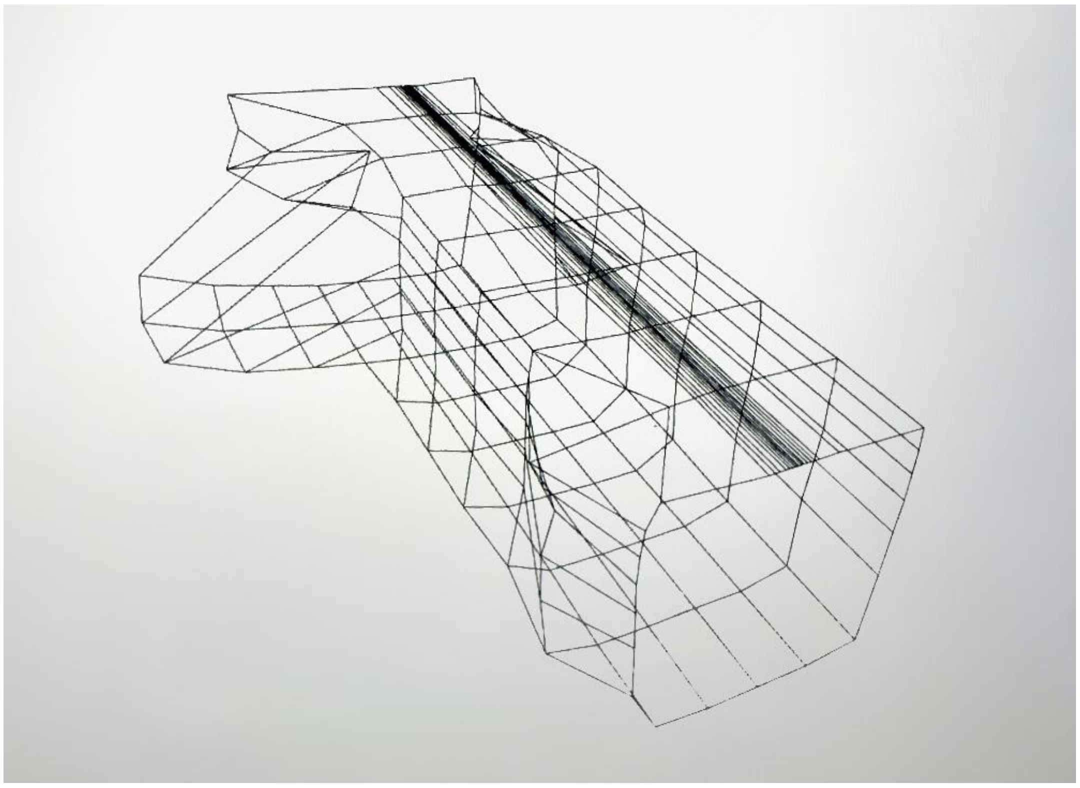

Figure 2: 3D of scanned hallway

5. Limitations

1. The microcontroller's floating-point capability is limited to 32 bits due to its 32-bit Arm Cortex-M4F processor core. This affects floating-point calculations as they are limited to the available 32-bits. Trigonometric functions also have limitations due to this.
2. To calculate the maximum quantization error for each of the Time-of-Flight (ToF) modules, we need to determine the resolution using the formula:
$$Resolution = \frac{VFS}{2^m}$$
where VFS is the Full-Scale Voltage and m is the number of bits. For the Texas Instruments MSP432E401Y microcontroller, VFS is 5V and m is 12 bits, resulting in a resolution of 1.22mV. For the VL53L1X ToF, VFS is 3.3V and m is 8 bits, resulting in a resolution of 12.89mV.
3. The maximum standard serial communication rate that can be implemented with the PC is 115200 bps. This was confirmed through testing by attempting to increase the baud rate beyond 115200 bps, which resulted in an error due to the microcontroller's inability to handle speeds beyond 115200 bps.
4. The communication method used between the microcontroller and the ToF sensor was I2C, which consists of a data line and a clock signal line. The transmission speed between the two devices is equivalent to the ToF's maximum transmission speed of 400 kbps.
5. The primary limitation on the system speed is the baud rate of the UART transmissions between the microcontroller and the PC, which is limited to 115200 bps. This speed is significantly slower than the 400-kbps communication speed between the microcontroller and the ToF. To test this limitation, we can measure the time it takes for the system to transfer data from the microcontroller to the PC and compare it to the theoretical maximum transfer time at the baud rate of 115200 bps.

6. Circuit Schematic

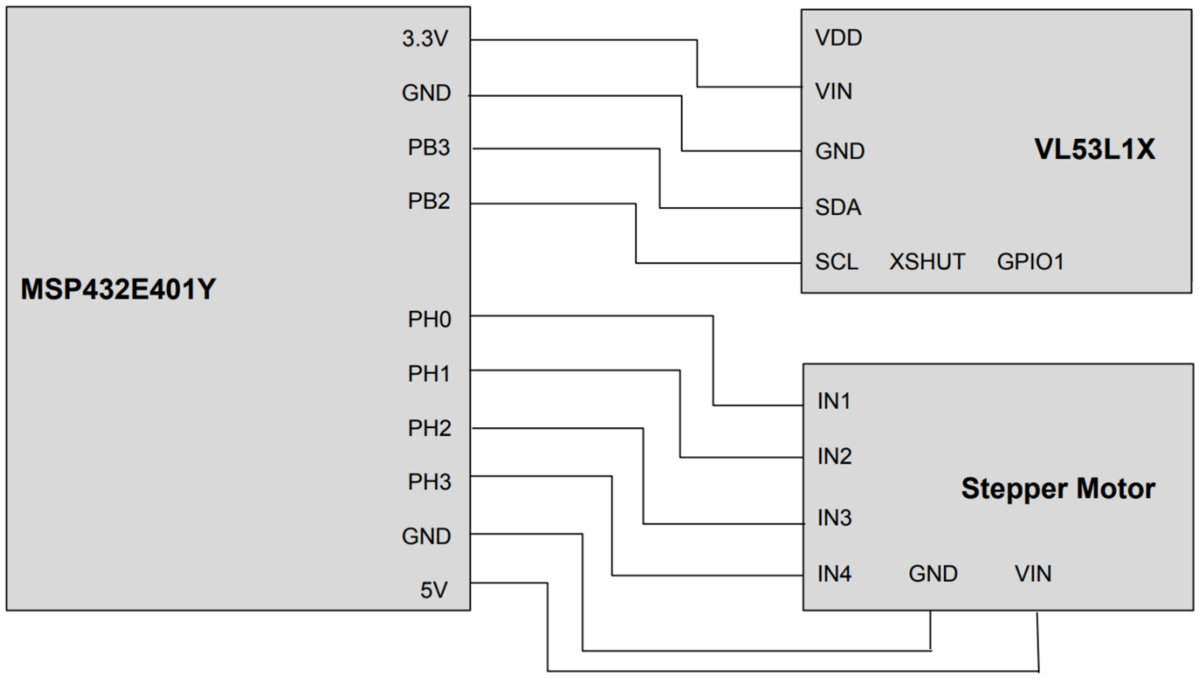

Figure 3: Circuit schematic

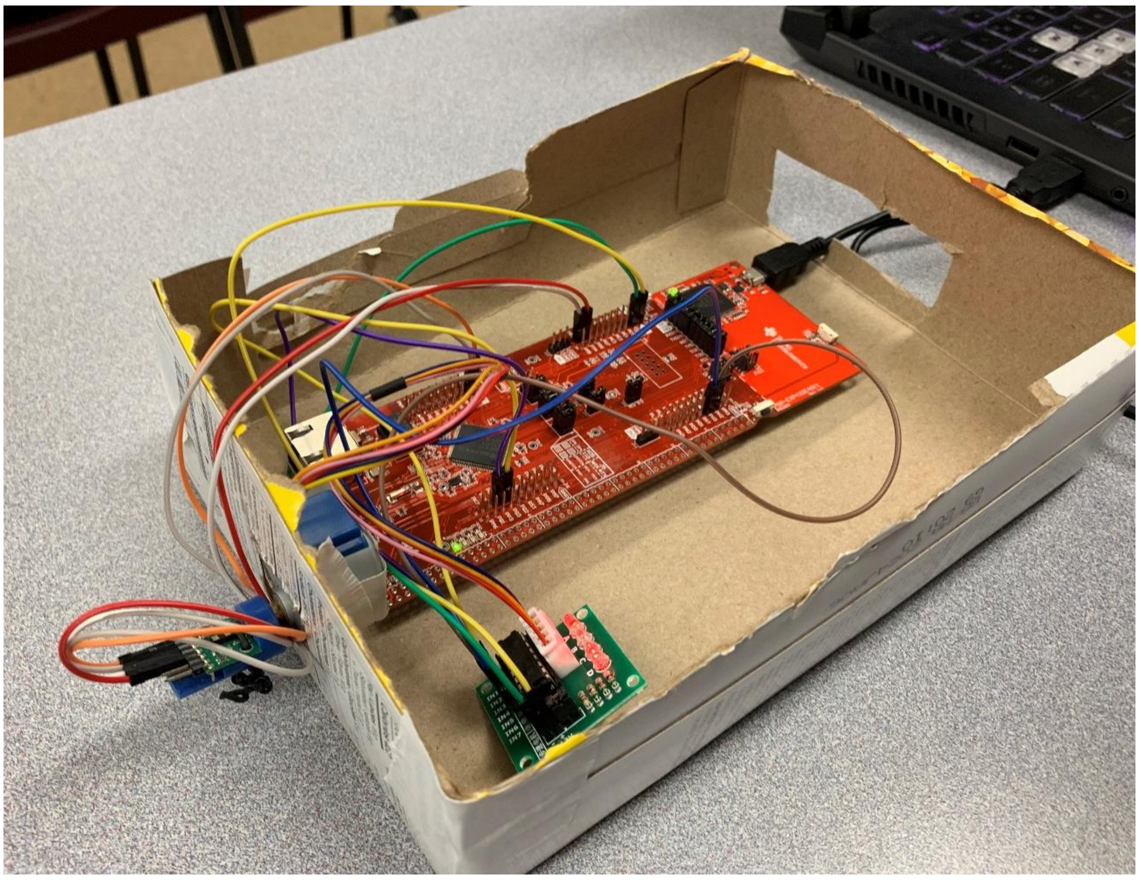

Figure 4: Built System

7. Programming Logic

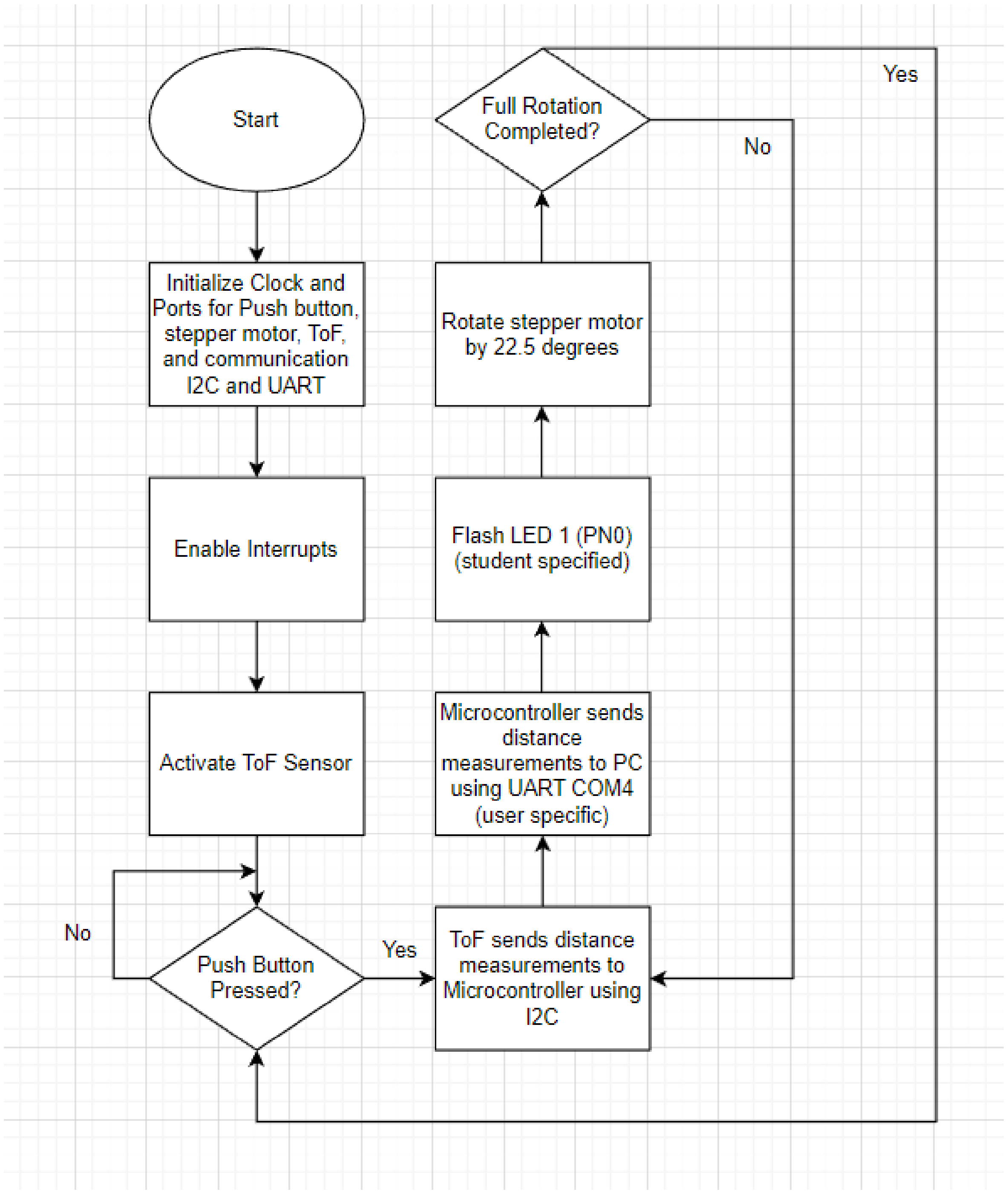
Start
Initialize Clock and Ports for Push button, stepper motor, ToF, and communication I2C and UART
Enable Interrupts
Activate ToF Sensor
Push Button Pressed?
No
Yes
ToF sends distance measurements to Microcontroller using I2C
Microcontroller sends distance measurements to PC using UART COM4 (user specific)
Flash LED 1 (PN0) (student specified)
Rotate stepper motor by 22.5 degrees
Full Rotation Completed?
No
Yes

Figure 5: Microcontroller C Code Flowchart

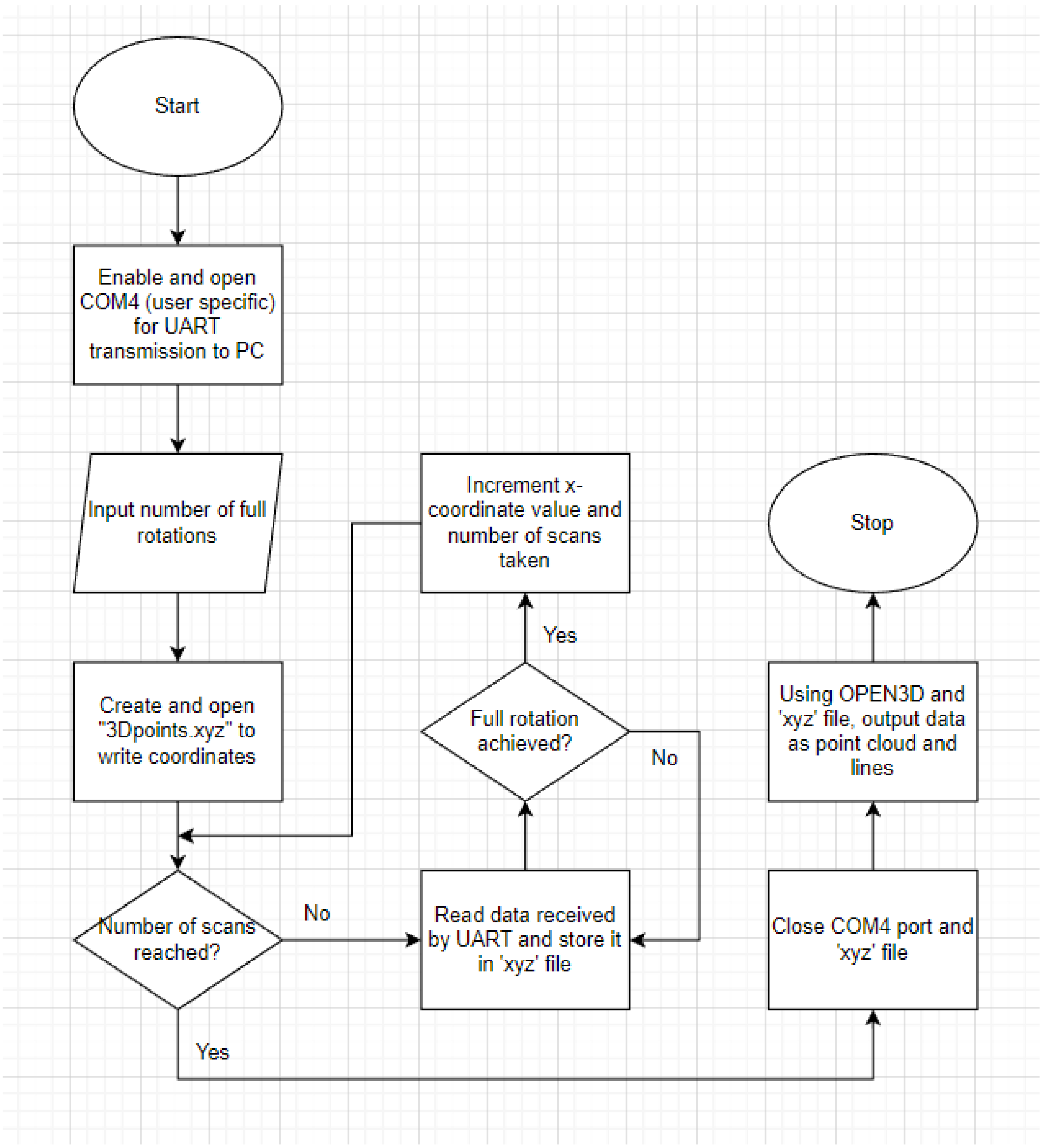
Start
Enable and open COM4 (user specific) for UART transmission to PC
Input number of full rotations
Create and open "3Dpoints.xyz" to write coordinates
Number of scans reached?
No
Yes
Read data received by UART and store it in 'xyz' file
Full rotation achieved?
Yes
No
Increment x-coordinate value and number of scans taken
Close COM4 port and 'xyz' file
Using OPEN3D and 'xyz' file, output data as point cloud and lines
Stop

Figure 6: Python Code Flowchart