COMPENG 2DX3

Final Project Report

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

Aegir Wang – wanga158 – 400453657

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 our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by [**Aegir Wang, wanga158, 400453657**]

Contents

1 Device Overview

The ROOMSCANNER400453657 is a device that scans rooms. It consists of a microcontroller, a stepper motor, and a time of flight sensor.

1.1 Features

The device consists of three components, specifically: a MSP432E401Y microcontroller, a VL53L1X time of flight sensor, and a 28BYJ-48 stepper motor.

The MSP432E401Y microcontroller has the following features:

- ARM Cortex-M4F core
- 120 MHz maximum clock speed, programmable
- 1024KB flash memory, 256KB SRAM, 6KB EEPROM
- 2 Onboard User Buttons, 4 Onboard User LEDs
- 4.75 VDC to 5.25 VDC Board Supply Voltage through USB 2.0 connectivity
- 90+ GPIO Pins
- 115200 bps Baud Rate UART external communication interface
- KEIL U-Vision IDE, supporting C/C++ and ARM Assembly development
- Approx 73.18 CAD

The VL53L1X time of flight sensor has the following features:

- 4 meters maximum range
- 2.8-3.5V operating voltage
- Up to 400 kHz I2C communication interface
- Approx 21.07 CAD

The 28BYJ-48 stepper motor has the following features:

- Unipolar 4-phase gear reduced stepper
- 64:1 gear ratio
- 2048 steps per revolution (full-step mode)
- 5V DC operating voltage
- ULN2003 interface PCB
- Approx 4.18 CAD

1.2 General Description

The ROOMSCANNER400453657 device is an adjustable resolution, simple set-up, straightforward interface room scanner. A distance sensor is mounted to a motor with the ability to turn the sensor 360°. This enables the sensor to measure a full 'slice' of a room. Taking multiple slices then enables the device to plot a room in its entirety.

The VL53L1X time of flight sensor is used to acquire distance information in the scan. To acquire this data, the sensor first sends out a pulse of light. Photons are then reflected from the wall, which the sensor detects. The sensor then calculates the time differential between the sent pulse and the received pulse. Noise, such as from ambient light, is filtered from the signal in the preprocessing stage. The sensor then converts the analog value into a digital value, and are sent to the microcontroller through the I2C interface.

The MSP432E401Y microcontroller is used to send acquired sensor data to the user's laptop. The microcontroller passes along distance measurements from the sensor to the laptop through the UART communication protocol, which the laptop receieves and stores. The UART communication protocol relies on both transmitter and receiver sharing the same baud rate as a clock signal is not sent. Instead, two transmission lines send a start bit, the data, and a stop bit through the two required wires.

The user's laptop is used to transform acquired sensor data into a 3-D visualization of the data. The visualization of the data is done through a python script, using Open3D, an open source python library. The script takes the point cloud previous generated through the distance measurements and computes a convex hull. The hull is then coloured appropriately for user-friendliness before it is visualized for the user.

1.3 Block Diagram

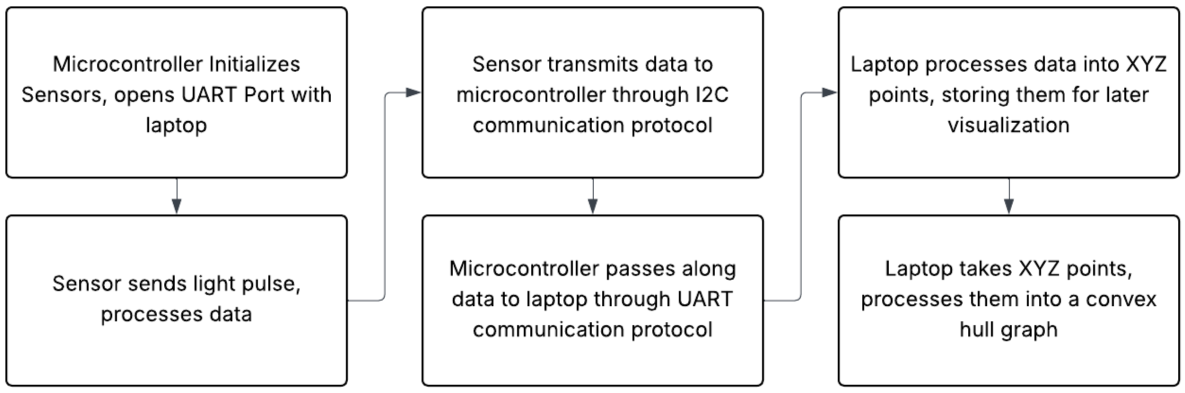

Microcontroller Initializes Sensors, opens UART Port with laptop
Sensor sends light pulse, processes data
Sensor transmits data to microcontroller through I2C communication protocol
Microcontroller passes along data to laptop through UART communication protocol
Laptop processes data into XYZ points, storing them for later visualization
Laptop takes XYZ points, processes them into a convex hull graph

Figure 1: Data Flow Block Diagram

2 Device Characteristics Table

Table 1: Characteristics of Microcontroller, ToF Sensor, Stepper Motor

Microcontroller Info		**VL53L1X PINOUT**		**ULN2003 PINOUT**	
Bus Speed	10 MHz	V_{in}	$3.3V$	V_+	$5V$
Onboard Buttons	PJ[0 : 1]	GND	Gnd	V_-	Gnd
Onboard LEDs	PN[0, 1], PJ[0, 4]	SDA	PB3	IN[0 : 4]	PH[0 : 4]
Baud Rate	115200 bps	SCL	PB2	-	-
Measurement Status	PF0	-	-	-	-
UART Tx Status	PN0	-	-	-	-
Ready-to-transmit Status	PN1	-	-	-	-
Serial Port	COM3	-	-	-	-

3 Device Detailed Description

3.1 Distance Measurement Acquisition

The VL53L1X measures distance through a LiDAR system, as mentioned above. By measuring the amount of time a laser pulse takes to reflect off an object and return to the sensor, the sensor is able to calculate the distance of the object to the sensor. The simplest formula relating distance, speed, and time is:

$$d = s \cdot t$$

where d refers to distance, s refers to speed, and t refers to time. Since the time for the pulse to be received can be measured and the speed of the pulse is known, the distance can be measured. However, since the time measurement is the time taken for the pulse to travel from the sensor to the object and the time to travel from the object to the sensor, the final formula must be modified into

$$d = \frac{\Delta t \cdot c}{2}$$

where c refers to the speed of the laser, or the speed of light.

The VL53L1X detects photons through a SPAD (Single Photon Avalanche Diode). A SPAD is a type of highly sensitive semiconductor. It is named as such as photons entering the active device area can trigger an 'avalanche' of current, which the sensor can then detect.

For the microcontroller to gather data, it must first initialize the sensor. This consists of initializing ports PB2 and PB3 to be used as an I2C interface, then waiting for the sensor to finish its boot process. For the microcontroller to gather a distance measurement, it must first send a request to the sensor to take a measurement. Then, the sensor responds by transmitting the data. To transmit the data, the ToF sensor converts the analog value from the photon diode to a digital value, which is then transmitted to the microcontroller through the I2C protocol. The distance data is transmitted as a 16 bit unsigned integer, in two packets of 8 bits each (following the I2C protocol). As the data is transmitted in millimeters, the range is therefore $[0, 4000] * mm$.

The microcontroller takes 32 equally spaced measurements per rotation of the stepper motor. Because the provided python script has this information, it can calculate the angle between two measurements. Since the measurements are equally spaced, each successive measurement must be at an angle of $\frac{2\pi}{32}$ further than the previous measurement. Knowing that the starting angle is zero, the python program can then determine the exact XY coordinates through simple trigonometry:

$$X = \cos(\theta) \cdot d$$

$$Y = \sin(\theta) \cdot d$$

Because the Z coordinate only changes when one full rotation is finished, the provided python script can increment the Z coordinate by 20 centimeters after each set of measurements are taken. In this manner, the first slice is kept at a Z coordinate of $0cm$, the second slice is kept at a Z coordinate of $20cm$, and so forth. In this manner, the full XYZ data can be generated.

3.2 3D Visualization

Open3D, an open source python library was used to produce the final 3D visualization of the previously acquired points. Three critical functions were used to transform the points to a useful graph. The first is $o3d.io.read\_point\_cloud()$, which converts a set of 3D XYZ points to a point cloud. The point cloud is then further processed with the function $open3d.geometry.compute\_point\_cloud\_convex\_hull()$. This computes a convex hull from the previously generated point cloud. The points are then connected with the function $o3d.geometry.LineSet.create\_from\_triangle\_mesh()$. The final graph is then visualized with the function $o3d.visualization.draw\_geometries()$, manifesting the output graph for the user to inspect.

3.2.1 Convex Hull

A convex hull, in this context, is the smallest convex 3-D shape which covers a set of points. The method in which the convex hull is found is out of the scope of this paper and will not be covered.

4 Application Notes, Instructions, and Examples

4.1 Examples

At the core of the system is a LiDAR sensor, a widely used technology that has large potential to be used in futuristic systems.

Mapping out an area using LiDAR can be helpful in a variety of scenarios. Self-driving cars utilizing LiDAR can create a constantly updating 3D mapping of objects around itself, crucial to 'observing' its environment. Augmented reality devices require depth sensing, crucial in seamlessly integrating its various functions into a unser's point of view.

Scanning an area, what the device demonstrates, is highly applicable to many various fields. Agriculture, archaeology, forestry, and other various fields may need to map out survey sites frequently. Using an autonomous LiDAR system can reduce this time needed.

4.2 Instructions

The steps for operation are provided below.

To set-up the device, the user must connect the various components of the device according to the circuit schematic, shown in Figure 4. The user must also have a python environment installed, as well as a serial port available to connect the microcontroller to.

To start a scan, the user must first connect the microcontroller to their laptop of choice, and run the given python script. After running the python file, the user must press enter to allow the laptop to receive transmitted data. Additionally, the serial port used in the python script must be changed to match the user's port of choice, as various motherboards may implement various different port numberings.

The user must then reset the microcontroller. The microcontroller will acknowledge the reset by flashing all onboard LEDs, then proceed to initialize all necessary components. When the microcontroller is finished initializing, it will turn on LED 1, corresponding to Port N1. This signals that the microcontroller is ready to start scanning.

Scans must performed with the motor facing parallel to a wall (vertical). To start scanning, the onboard button button 2, corresponding to PJ1, must be pressed. The motor will then complete one full revolution, taking 32 equally spaced distance measurements with the ToF sensor. When the motor completes one section, LED 4, corresponding to Port F0, will flash. When the microcontroller sends a data transmission to the connected laptop, LED 2, corresponding to Port N0, will flash. The user should ensure the motor stays static in position. To complete the full scan, the user should move the device 20 centimeters forward in the chosen Z-axis (horizontal), and press button 2 once more, until the end of the room is reached.

After the scans are finished, the onboard button 1, corresponding to PJ0, must be pressed.

This signals to the microcontroller that the end of scanning has been pressed. The laptop will then visualize the scanned data, producing a 3-D point visualization.

To adjust the resolution, the number of points that the device scans must be changed. To achieve this, change lines 146 and 176 of the file *main.c*, and lines 19 and 28 of *vis.py*. To adjust the depth between measurements, adjust line 29 of *vis.py*.

4.3 Expected Outputs

As an example, a hallway in ETB at McMaster has been scanned, and a sample output has been provided. The hallway, the point cloud, and the convex hull are shown below.

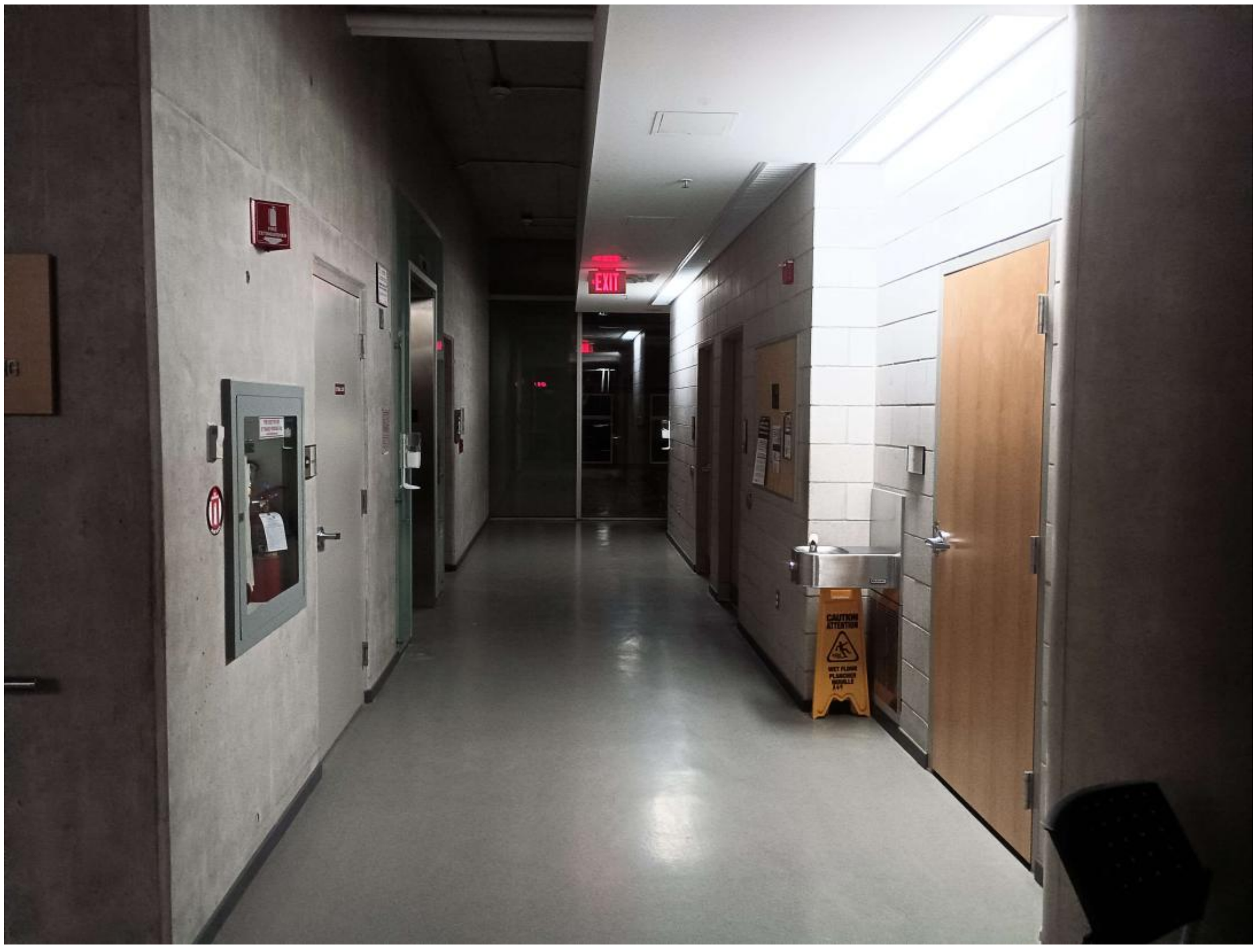
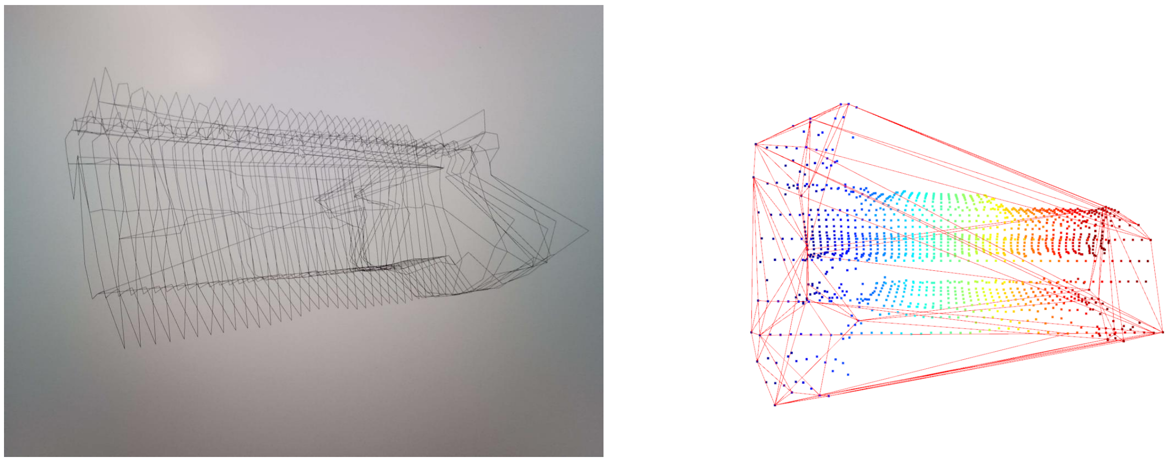


Figure 2: Data Flow Block Diagram



(a) Point Cloud (b) Convex Hull

Figure 3: Open3D Visualization

5 Device Limitations

The microcontroller has an ARM Cortex-M4F core, which does support floating point arithmetic, as specified by the F. As the ARM Cortex-M4F core is a 32-bit processor. According to the IEEE 754 standard, the sign bit takes one bit, the exponent width takes eight bits, and the mantissa takes twenty three bits. This leads to an approximate seven decimal point precision, which can frequently lead to round errors through repeated floating point calculations, as real numbers have a maximum decimal precision of infinity. Additionally, trigonometric functions are expensive (timewise) and are typically implemented using LUTs (look up tables) in hardware. Because of the above factors, floating point calculations were performed using the python script in a 64-bit environment offering higher precisions and (hopefully) faster speeds.

The VL53L1X sensor sends its distance measurements through a 16-bit value. The maximum measurement is 4000. Therefore, its maximum quantization error is $\frac{4000}{2^{16}}$, which is $.061 * mm$.

The ToF sensor sends its distance measurements through the I2C communication protocol. The VL53L1X sensor is limited to a transmission speed of 400 kHz, listed on its manufacturer's official datasheet.

To determine the maximum baud rate, trial and error is needed. On a Linux system, supported baud rates include 115200 bps up to 1152000 bps. A list of supported baud rates of a system can be found in the /usr/include/asm-generic/termbits.h file; each baud rate can be set using the stty command. After narrowing down possible baud rates using binary search, the maximum baud rate of my chosen serial port was found to be 230400 bps.

The primary limitation on speed is most certainly the stepper motor. As the minimum delay between two full-steps of the stepper motor is approximately 2 milliseconds, a full rotation, consisting of one rotation counter-clockwise to measure the room, and one clockwise rotation to untangle the wires, will take approximately 30 seconds with the bus speed at its maximum; a longer delay results after bus speed modifications. As the depth of the room measuring increases, the speed of the stepper motor becomes more and more of a factor in determining the length of time needed to scan a room. The secondary delay is the ToF sensor's delay. A significant delay is in the initialization of the ToF sensor, as well as sensor measurements. Each sensor measurement requires for the microcontroller to wait for the sensor to finish processing data, another source of delays.

6 Circuit Schematic

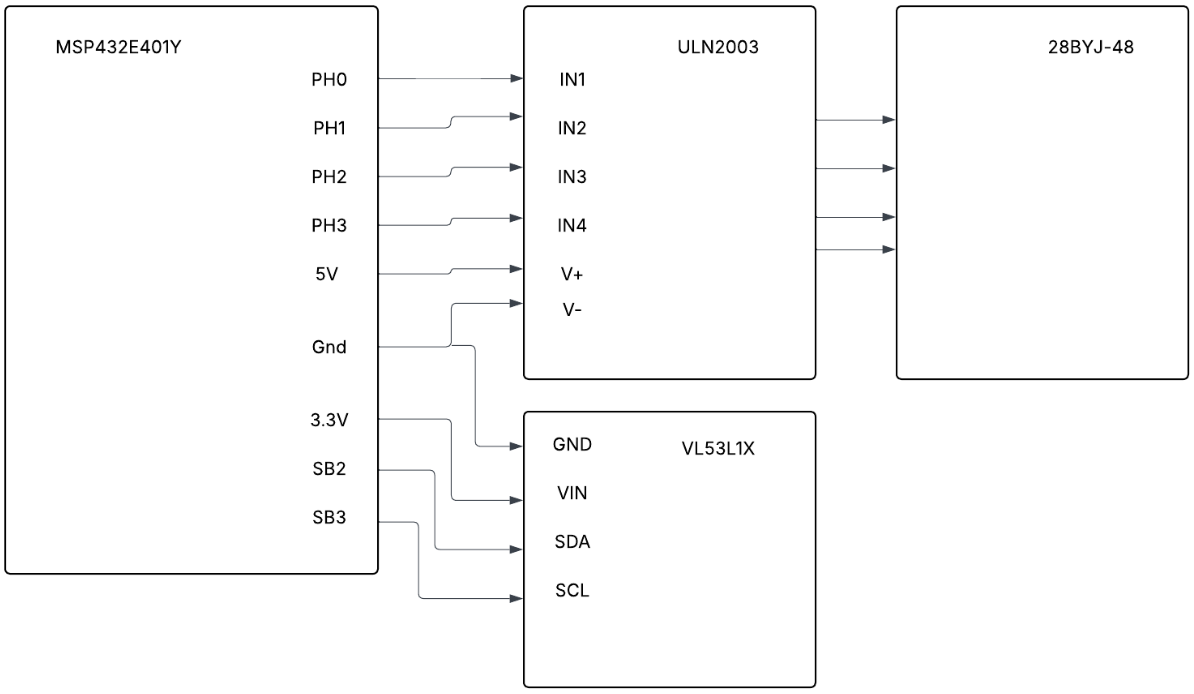


Figure 4: Circuit Schematic

7 Programming Logic Flowcharts

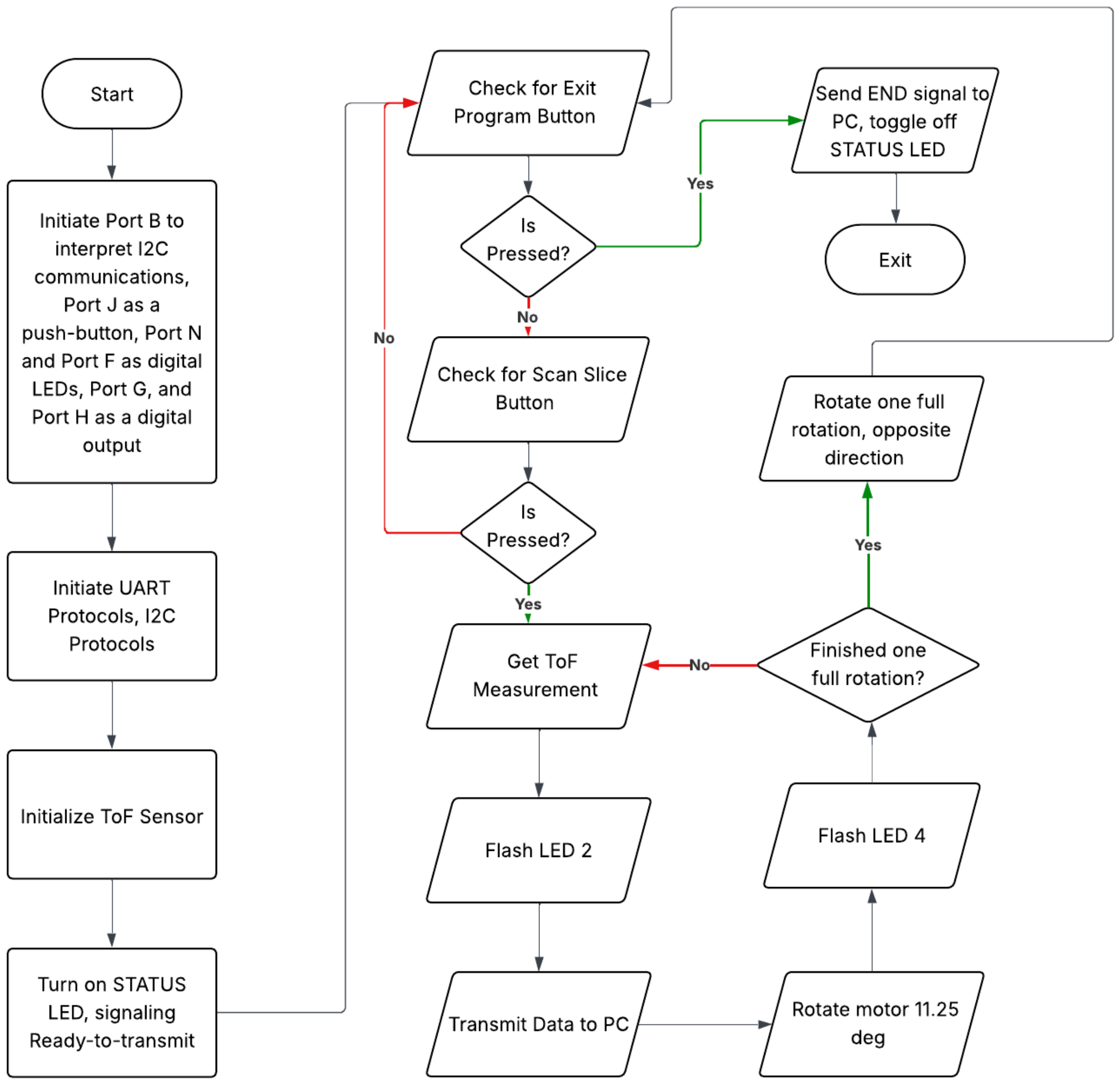

Start
Initiate Port B to interpret I2C communications, Port J as a push-button, Port N and Port F as digital LEDs, Port G, and Port H as a digital output
Initiate UART Protocols, I2C Protocols
Initialize ToF Sensor
Turn on STATUS LED, signaling Ready-to-transmit
Check for Exit Program Button
Is Pressed?
Yes
Send END signal to PC, toggle off STATUS LED
Exit
No
Check for Scan Slice Button
No
Is Pressed?
Yes
Get ToF Measurement
Flash LED 2
Transmit Data to PC
Rotate motor 11.25 deg
Flash LED 4
Finished one full rotation?
No
Yes
Rotate one full rotation, opposite direction

Figure 5: Program Flowchart