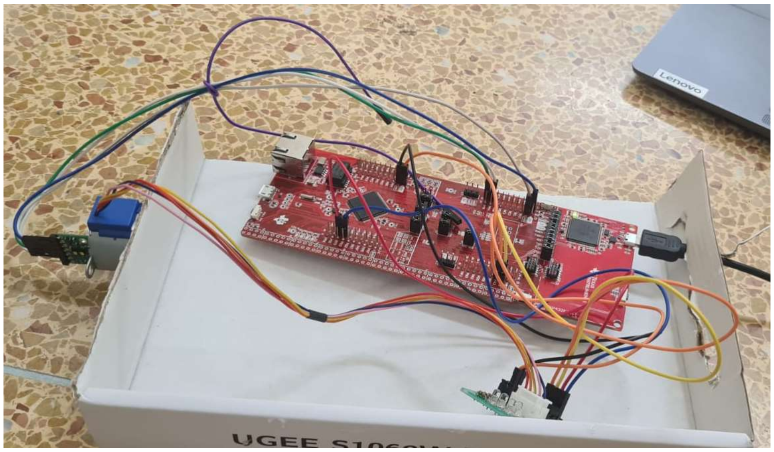
COMP ENG 2DX3
Final Project Report

SPATIAL MAPPING USING TIME-OF-FLIGHT

Instructor: Drs. Haddara, Athar, Doyle

Date Submitted: 2024 – 04 – 17

Someshwar Ganesan – ganesans – 400430923



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 my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by **[Someshwar Ganesan, ganesans, 400430923]**

Table of Contents

Device Overview

Features

The overall system is driven by a **Texas Instruments MSP432E401Y Microcontroller** loaded with C programming language code written in Keil uVision software. The microcontroller has the following specifications [1]:

- ARM Cortex M4 Microprocessor
- 60 MHz internal bus clock speed
- 2 status LEDs
- Onboard push-button
- GPIO pins to connect with time-of-flight sensor and stepper motor driver board

The time-of-flight sensor is a **VL53L1X Time-of-Flight Sensor** with the following specifications:

- Distance measurements in mm, with a maximum range of 4000mm (4 meters)
- 3.3V operating voltage
- Up to 50 Hz ranging frequency

The time-of-flight sensor is mounted onto a **28BYJ-48 stepper motor** to enable 360-degree measurement of a single plane. The stepper motor and its **ULN2003 driver board** have the following specifications: [2]

- The motor has two center-tapped coils to control an internal gear, which in turn controls an outer gear. (Internal to external gear ratio of 64:1)
- Motor operates in full-step mode (2048 steps of internal gear for 1 rotation of outer gear).
- Motor controlled by driver board connected by 6 wires (4 to control each end of the coils, 1 supply, 1 ground)
- Driver mode has 5V operating voltage

The above devices interact with each other using the following communication protocols:

- **I2C** serial communication between VL53L1X Time-of-Flight Sensor and Texas Instruments MSP432E401Y Microcontroller
- **UART** serial communication between Texas Instruments MSP432E401Y Microcontroller and PC
    - Baud rate of 115200 bps

Finally, Python3 modules **pyserial** and **open3d** are used for 3D visualization.

General Description

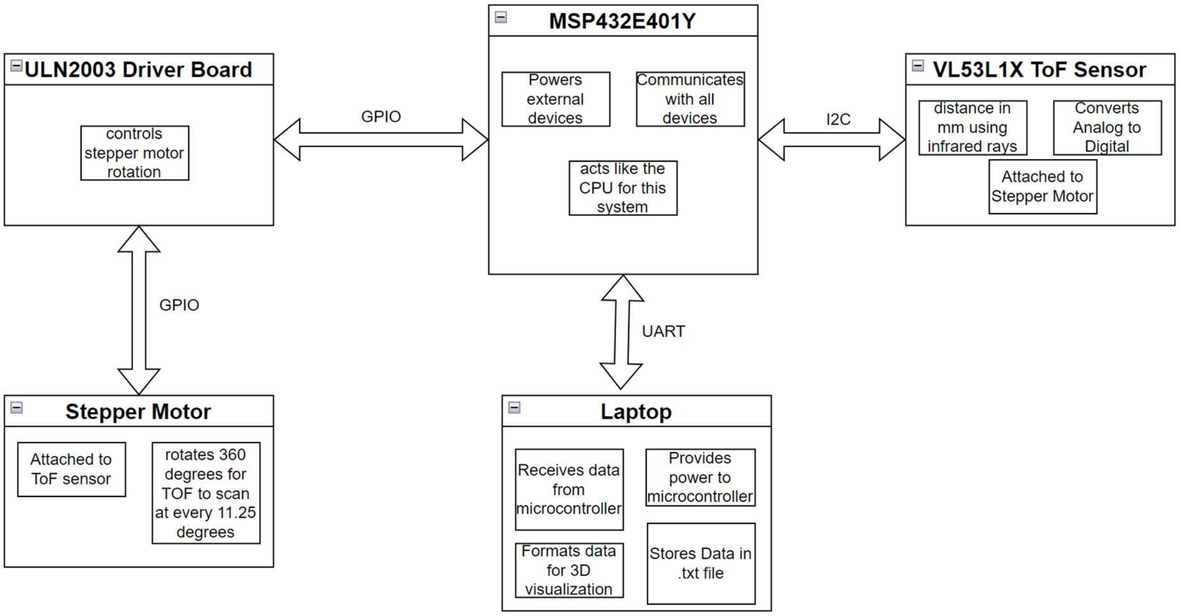
Commercial Light Detection and Ranging (LIDAR) equipment is expensive and usually bulky. This application is relatively cheap and smaller, making it suitable for indoor mapping and navigation [3].

This spatial measurement system is implemented using the following main components; a PC, the microcontroller, a stepper motor with its driving board, and the time-of-flight sensor. These components are connected to one another using GPIO pins and serial communication protocols which are 12C and UART.

The system uses the VL53L1X time-of-flight sensor mounted on a stepper motor using hot glue to acquire 360-degree measurements of distance within a single vertical geometric plane (say x-y). Each plane is formed using 32 points of measurements taken at 11.25 degrees each. The set of measurements is transmitted from the time-of-flight sensor to the microcontroller using 12C serial communication protocol over GPIO pins and then to a text file in a PC using UART communication over a USB cable.

Multiple such planes of distance measurements are arranged at a predefined distance along the axis perpendicular to the planes (z-axis) to create a 3D visualization of the environment scanned. This 3D mapping is generated and visualized on a PC using Python and Open3D. Using the 'open3d' module in Python, the measurements from the text file is obtained and plotted.

Block Diagram (Data flow graph)



MSP432E401Y
Powers external devices
Communicates with all devices
acts like the CPU for this system
ULN2003 Driver Board
controls stepper motor rotation
GPIO
VL53L1X ToF Sensor
distance in mm using infrared rays
Converts Analog to Digital
Attached to Stepper Motor
I2C
GPIO
UART
Stepper Motor
Attached to ToF sensor
rotates 360 degrees for TOF to scan at every 11.25 degrees
Laptop
Receives data from microcontroller
Provides power to microcontroller
Formats data for 3D visualization
Stores Data in .txt file

Device Characteristics

Texas Instruments MSP432E401Y Microcontroller	
Bus Clock Speed	60 MHz
Measurement Status LED	Onboard LED 4 (PF0)
Additional Status LED (Motor Running)	Onboard LED 3 (PF4)
Onboard pushbutton	PJ1
Baud Rate	115200 bps

VL53L1X Time-of-Flight Sensor [4]	
Supply Voltage (VIN)	3.3V from microcontroller
Ground (GND)	Ground (GND) pin on microcontroller
SDA	Pin PB3 on microcontroller
SCL	Pin PB2 on microcontroller
ULN2003 stepper motor driver board	
Supply Voltage (VIN)	5V from microcontroller
Ground (GND)	Ground (GND) pin on microcontroller
IN1 – IN4	Pin PH0 – PH3 on microcontroller
Software	
Microcontroller code	Keil uVision (C language)
Python	Version 3.9 (can use any between 3.6 – 3.9)
Python module for serial communication	Pyserial
Python module for 3D visualization	Open3D

Detailed Description

Distance Measurement

The Distance Measurement process is headed by the VL53L1X time-of-flight sensor which is mounted on the stepper motor. Each rotation of the motor to enable the ToF sensor to scan one vertical plane is initiated by one press of the microcontroller onboard-push button PJ1. Pressing PJ1 triggers an interrupt that runs a routine to spin the motor in the clockwise direction and stop at every 11.25 degrees (every 64 steps out of 2048) and lets the ToF sensor take one reading.

The ToF sensor consists of an infrared laser light source and a matching sensor. Distance is measured by determining the *Time* taken for the matching sensor to detect an infrared laser light after it is emitted by the source, reaches the closest object (typically a wall or ceiling), and bounces back onto it. The distance is calculated using the following formula:

$$\textit{Measured Distance} = (\textit{Time} / 2) \times \textit{Speed of Infrared Laser Light} \qquad [5]$$

The ToF sensor then does signal conditioning and analog-to-digital conversion internally, resulting in a digital reading of the analog distance measurement (in mm). This data acquisition follows the steps below.

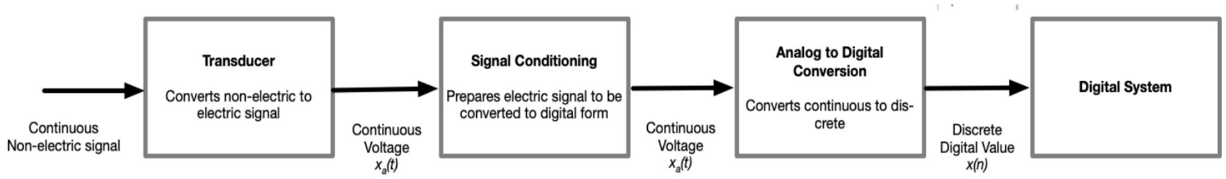

Continuous
Non-electric signal
Transducer
Converts non-electric to electric signal
Continuous
Voltage
$x_a(t)$
Signal Conditioning
Prepares electric signal to be converted to digital form
Continuous
Voltage
$x_a(t)$
Analog to Digital Conversion
Converts continuous to dis-crete
Discrete
Digital Value
$x(n)$
Digital System

Figure 1: ADC Process [3].

Each distance reading, along with other values such as Range Status*, Signal Rate**, Ambient Rate***, etc. [4], is obtained and transmitted to the microcontroller using I2C serial communication. Out of these values, only the distance measurement is transmitted to the PC using UART serial communication.

After 32 such transmissions of distance measurements, one rotation is complete. The motor now spins in the anti-clockwise direction to bring itself and the ToF sensor back to its home position, mainly to prevent tangling of wires.

Visualization

The visualization process is done on a laptop which runs Windows 11 and has the required software installed (any device that can run Keil uVision and Python, must have a USB-A data port, and must be connected to the microcontroller during the entire process). The distance measurements are sent from the microcontroller to the laptop using UART serial communication at a baud rate of 115200 bps.

These values are received by Python using the ‘pyserial’ module and saved onto a text file (.txt). This file is then opened by the same Python program and the values are used to calculate an x-y cartesian coordinate pair with the help of trigonometric function as follows.

$$x \text{ - } coordinate = distance \times cos\Theta$$
$$y - coordinate = distance \times sin\Theta$$

Where Θ is the angle which starts from 0 degrees and increases by 11.25 degrees for every 32 readings (1 rotation). These coordinates are plotted along with a predefined z-coordinate onto an empty space using the ‘open3d’ module. Necessary points are connected using line segments and the final 3D mapping is produced.

Application

Note

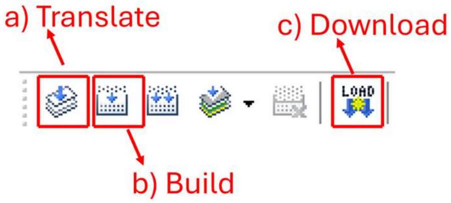
This spatial measurement system offers a solution for various applications where an accurate 3D mappings of an enclosed environment is required. Potential applications include mapping buildings or warehouses for navigation and security, and visualization for Augmented / Virtual Reality. Overall, this system offers an efficient and cost-effective solution, empowering users to acquire, analyze, and visualize spatial data with precision and efficiency. Its use of different hardware components, programming, and modules enables seamless operation and provides the opportunity to be included into bigger systems to suit more diverse application needs

Instructions

1) Ensure that all the connections are made according to the circuit schematic in Figure.5.
2) Ensure that the required software (Keil uVision, Python) and modules (pyserial, open3d) are downloaded and installed (Keil uVision, Python) .
3) Open the file ‘ganesans_2dx3_project’ which is of file type ‘uVision5 Project’ using the Keil uVision software.

4) On the ‘Project’ pane on the left, go to Target 1 -> Source Group 1 and open the .c file which is called ‘project_main’.

5) On the build toolbar, click on the following options in the given order.



6) Click the ‘Reset’ pushbutton on the microcontroller.

7) Open the .py file ‘2dx3_room_scan’ using IDLE. Change the port in line 13 from ‘COM3’ to whatever port on the user’s laptop supports UART. Run the file.

8) When asked to enter number of rotations, enter the number of 360 degree scans you would need to cover the room or hallway being scanned. Press Enter. **Note that each scan needs to be taken at 60cm apart. The distance between each scan is mentioned as 600mm in line 72, this can be changed to any value of the user’s choice in mm. Save and close the file. Repeat from step 7.**

9) Stand in position for the first scan.

10) Press lowercase ‘s’ initiate communication between the laptop and microcontroller.

11) The screen will prompt to press PJ1 in the microcontroller. Press PJ1. The motor will begin to rotate slowly in clockwise direction and stop at every 11.25-degree angle. Whenever the motor stops, you will see a distance reading (in mm) on the python output window. You will see 32 of these readings before the motor stops rotating.

12) The motor will rotate back in anticlockwise direction to untangle the wires. During this time, move the system forward by 60 cm to the next scanning spot.

13) Repeat steps 11 and 12 until the entered number of rotations are completed and the python program no longer asks to press PJ1. An array of coordinates will be displayed, and a point-cloud window will appear, showing all the recorded distances plotted on a 3D space.

14) Close this window, another window will open with the necessary points connected using a line segment. This is the final result.

Expected Output

Below is an image of the hallway that was scanned for the purpose of this demonstration. Observe that it is a narrow hallway with closed doors on either side. The hallway is uniform throughout, except for another hallway opening at the end towards the right.

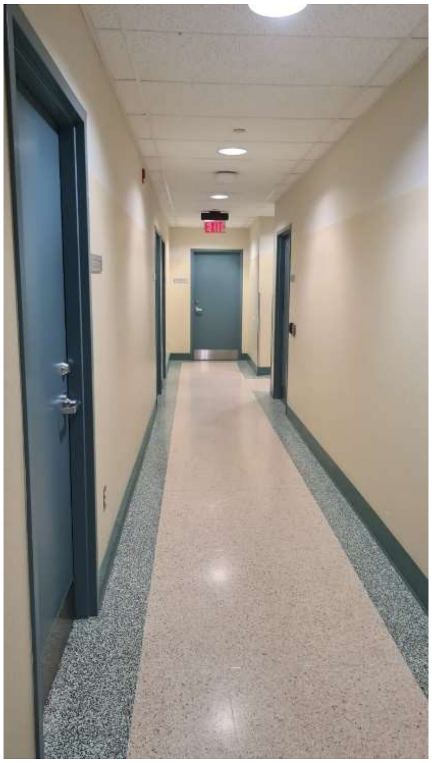


Figure 2: Hallway scanned for this demonstration.

The following image shows the point-cloud output for the hallway scan. It is a mapping of all the recorded points in an empty 3D space.

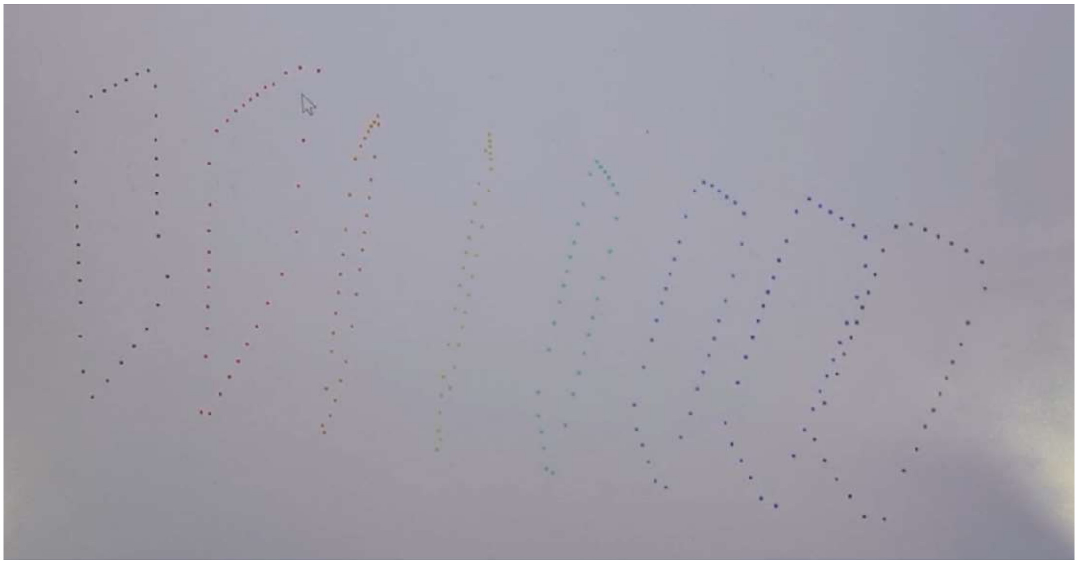


Figure 3: Point Cloud Visualization

The following image shows the final output for the hallway scan, with all points in one plane, and the point at the same location in the next plane, connected by a line segment.

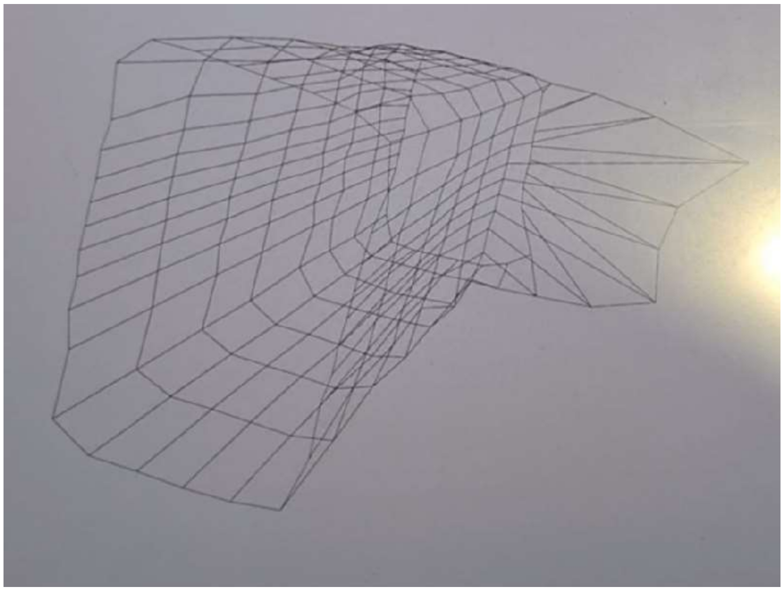


Figure 4: Connected Point Cloud using Lineset.

Limitations

1) **Limitations of the MSP432E401Y microcontroller floating point capability and use of trigonometric functions:** Although the microcontroller features a Floating-Point Unit (FPU) capable of handling single-precision 32-bit operations, Python and its math module was used to perform trigonometric calculations. While this decision made the development process faster, it disregards potential optimizations offered by onboard processing.

2) **Maximum quantization error for each of the ToF module** [8]**:**

$$Max.\,quantization\ error = \frac{Maximum\ distance\ reading}{2^{no.of\ ADC\ bits}} = \frac{4000\ mm}{2^{16}} = 0.061\ mm$$

This quantization error, although relatively small, introduces a level of uncertainty into distance measurements that may impact applications that may prioritize absolute precision.

3) **Maximum standard serial communication rate you can implement with the PC. How did you verify?** Serial communication between the microcontroller and PC is constrained by the PC's maximum standard serial communication rate, set at 128,000 bits per second. This can be checked for any PC from **Device manager > Ports > Right click on port > Properties > Port settings**. Despite the microcontroller's capability of supporting speeds up to 15 Mbps, the final rate is based on the PC's limitations. This constraint highlights the importance of compatibility considerations in system design. The baud rate that we used was 115200 bps.

4) **Communication methods and speed used between the microcontroller and the ToF modules:** The device uses I2C protocol for communication between the microcontroller and ToF sensor, offering a maximum transmission speed of 3.33 Mbps. However, the UART communication between the microcontroller and PC, utilizing COM3 on this design (different for different PCs) runs on a baud rate of 115,200 and imposed bus speed of 60MHz.

5) **Element which is the primary limitation on speed**: The primary roadblock in system speed is attributed to the ranging time required by the ToF sensor since the ToF has a maximum ranging frequency of 50Hz. While other delaying factors such as the motor's rotation speed, contributes to minor delays, it remains relatively fast in comparison to the ranging process. The ToF needs a considerable amount of time to calculate the distance as well as to send and receive infrared rays. The results showed too many errors while testing with reduced delays for this. As a result, before completing the full 360-degree turn, the motor must temporarily stop rotating. This need is the main restriction that slows down the system's overall speed.

Circuit Schematic

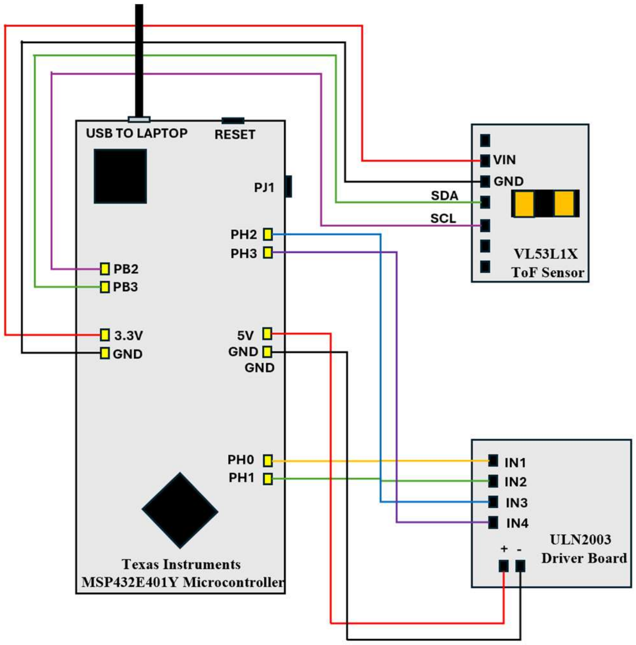
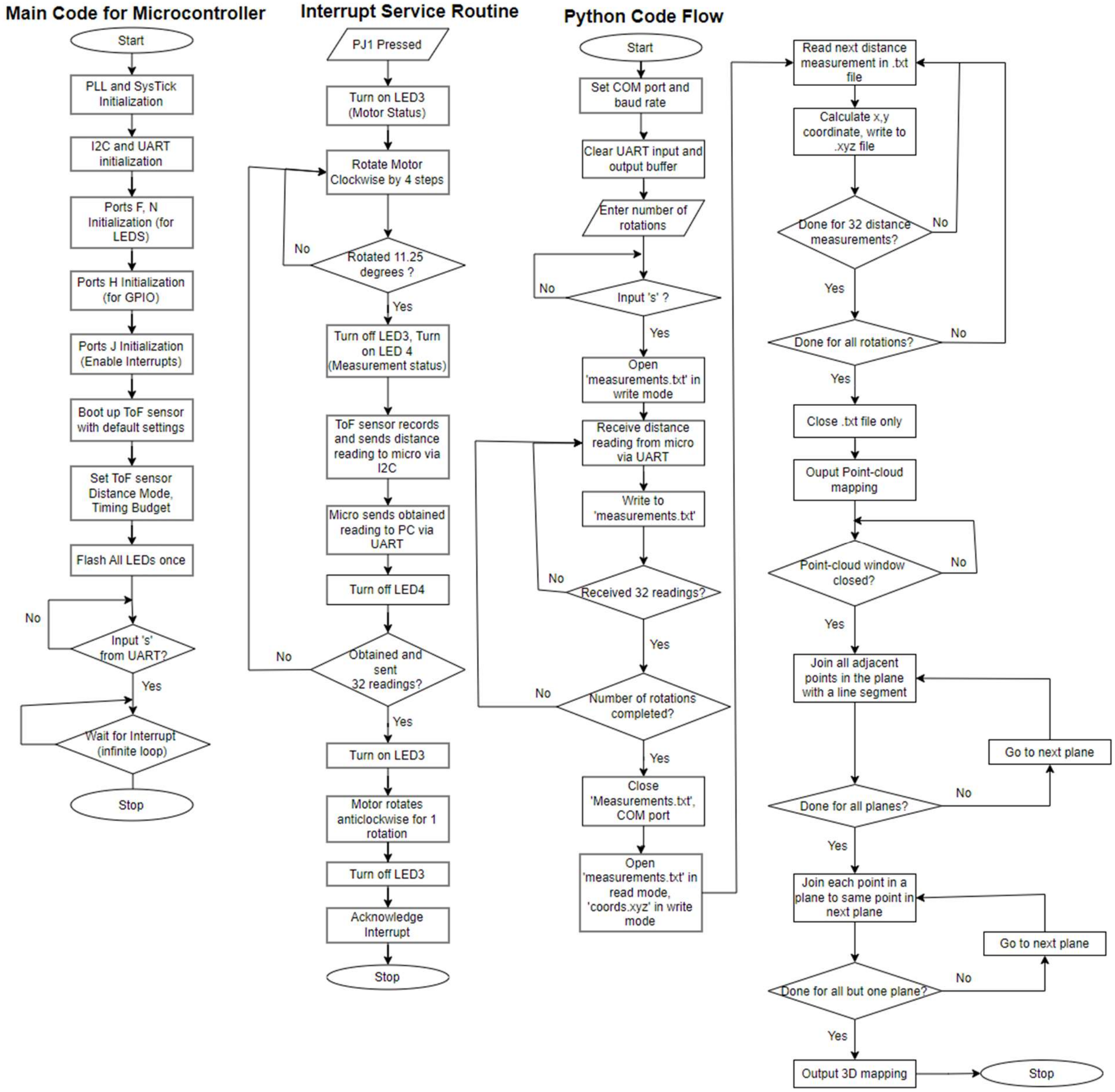


Figure 5: Physical Connections for Microcontroller, ToF sensor, and Motor driver board.

Programming Logic Flowcharts


Main Code for Microcontroller
Start
PLL and SysTick Initialization
I2C and UART initialization
Ports F, N Initialization (for LEDS)
Ports H Initialization (for GPIO)
Ports J Initialization (Enable Interrupts)
Boot up ToF sensor with default settings
Set ToF sensor Distance Mode, Timing Budget
Flash All LEDs once
No
Input 's' from UART?
Yes
Wait for Interrupt (infinite loop)
Stop
Interrupt Service Routine
PJ1 Pressed
Turn on LED3 (Motor Status)
Rotate Motor Clockwise by 4 steps
No
Rotated 11.25 degrees ?
Yes
Turn off LED3, Turn on LED 4 (Measurement status)
ToF sensor records and sends distance reading to micro via I2C
Micro sends obtained reading to PC via UART
Turn off LED4
No
Obtained and sent 32 readings?
Yes
Turn on LED3
Motor rotates anticlockwise for 1 rotation
Turn off LED3
Acknowledge Interrupt
Stop
Python Code Flow
Start
Set COM port and baud rate
Clear UART input and output buffer
Enter number of rotations
No
Input 's' ?
Yes
Open 'measurements.txt' in write mode
Receive distance reading from micro via UART
Write to 'measurements.txt'
No
Received 32 readings?
Yes
No
Number of rotations completed?
Yes
Close 'Measurements.txt', COM port
Open 'measurements.txt' in read mode, 'coords.xyz' in write mode
Read next distance measurement in .txt file
Calculate x,y coordinate, write to .xyz file
Done for 32 distance measurements?
No
Yes
Done for all rotations?
No
Yes
Close .txt file only
Ouput Point-cloud mapping
Point-cloud window closed?
No
Yes
Join all adjacent points in the plane with a line segment
Go to next plane
Done for all planes?
No
Yes
Join each point in a plane to same point in next plane
Go to next plane
Done for all but one plane?
No
Yes
Output 3D mapping
Stop

References

[1] Texas Instruments, "*MSP432 SimpleLink*^{TM} *Microcontrollers – Technical Reference Manual*", 2018, Ch. 4, 17, pp. 326 – 327, 1201 – 1252.

[2] Kattni Rembor, "*AdaFruit VL53L1X Time of Flight Distance Sensor*", 2021, https://learn.adafruit.com/adafruit-vl53l1x/overview , Accessed: Apr 17, 2024.

[3] Shahrukh Athar, Thomas E. Doyle, Yasser Haddara, "*2023 – 2024 2DX3 Project Specification – Observe, Reason, Act: Spatial Mapping Using Time-of-Flight*", McMaster University, Hamilton, Canada, 2024, pp. 1.

[4] Life.augmented, "*VL53L1X"*, 2018, pp. 1-2, 10-11.

[5] Shahrukh Athar, Thomas E. Doyle, Yasser Haddara, "*Computer Engineering 2DX3, Implementing Signal Transfer Functions & Calibrations* ", McMaster University, Hamilton, Canada, 2024, pp. 37.