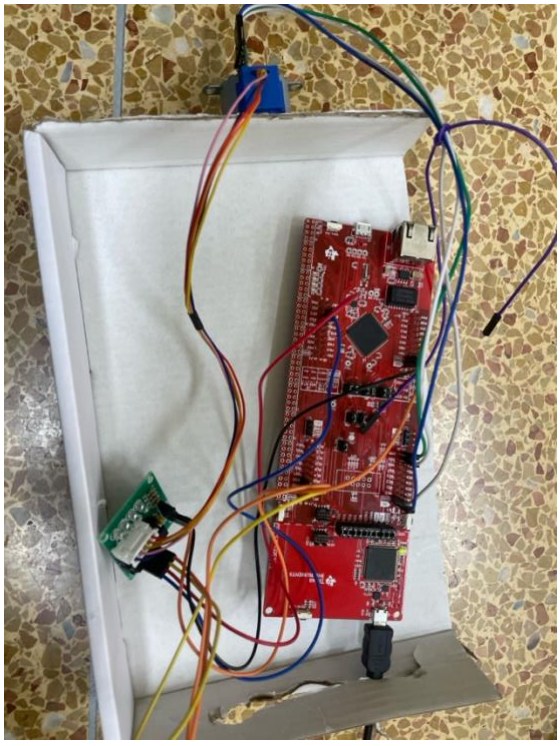
COMP ENG 2DX3

Final Project Report

Instructor: Dr. Haddara/Athar/Doyle

Date Submitted: 17th – April - 2024

Mustafa Shahid– 400440384 – shahim45



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 **[Mustafa Shahid, 400440384, shahim45]**

Table of Contents

Device Overview *3*

Features 3

General Description 3

Block Diagram 4

Device Characteristics Table *5*

Detailed Description *5*

Distance Measurement 5

Visualization 6

Application *6*

Note 6

Instructions for user 7

Expected Output 8

Limitations *9*

Circuit Schematic *10*

Flowchart *11*

References *12*

Device Overview

Features

Texas Instruments MSP432E401Y Microcontroller:

- Processor: ARM Cortex-M4F
- Operating Frequency: 24 MHz
- Operating Voltage: 2.5 to 5.5 V

VL53L1X Time-of-Flight Sensor:

- Operating Range: up to 4 meters
- Operating Frequency: up to 50Hz
- Operating voltage: 2.5 to 3.5 V

28BYJ-48 Stepper Motor and ULN2003 Driver Board:

- Mode of operation: full-step mode
- Steps per rotation: 2048 steps
- Operating voltage: 5 to 12 V

Software used

- Microcontroller: C language in Keil uVision
- Open3d: creating visual graphs from spatial data.
- Python 3.9 and libraries like PySerial and Open3D

Communication

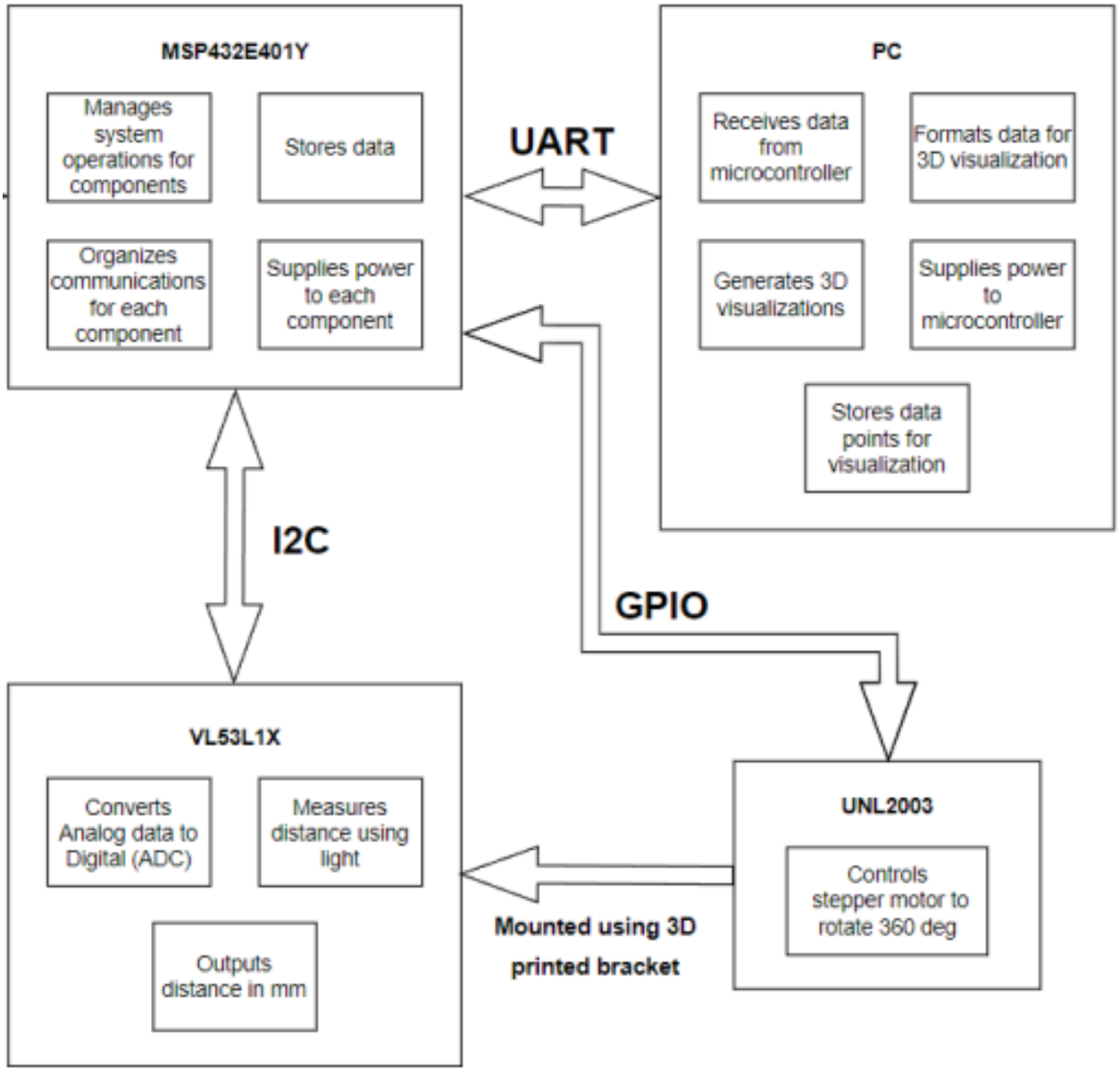
- Baud rate: 115200 bps
- MSP432E401Y and VL53L1X – I2C protocol
- MSP432E401Y and PC (Macbook) – UART protocol

General Description

This system for spatial mapping provides an economical and adaptable way to visualize indoor environments in three dimensions. The system is primarily made up of several parts, such as a microcontroller, stepper motor, and a ToF sensor. The microcontroller coordinates the movement and exchange of information between these parts. It is possible to facilitate communication with protocols like UART and I2C. The ToF sensor measures distances precisely using infrared light, and it communicates these measurements to the microcontroller via the I2C protocol. Complete 360-degree rotation of the sensor is accomplished by the stepper motor when paired with it, allowing for thorough spatial data mapping. Microcontroller onboard pushbuttons (PJ1 and reset) facilitate user interaction by making it simple to start and pause for scans, respectively.
The system gathers data points in the x-y plane at a predetermined step angle (11.25 degrees) for

a total of 32 scans in one rotation when PJ1 is pressed. After a scan is finished, the system returns to the starting position. At this point, the user can press PJ1 to start a new scan after moving to the next position along the z-axis. Python libraries like Pyserial and Open3D enable interaction with Python through data transmission to the PC through UART communication. The open3d python library is used to convert the various slices, along the z axis, of the data points gathered on the xy plane into 3D visualizations, giving viewers a visual understanding of the mapped environment.

Block Diagram



Device Characteristics Table

Component	Specification	Description
MSP432E401Y	Microcontroller	• **Bus Frequency**: 24 MHz • **Buttons:** Reset and PJ1 • **LEDs**:PF4 and PN1 • **Baud Rate**:115200 BPS
VL53L1X	Time-of-Flight Sensor	• **Ground**: GND Pin on microcontroller • **Supply voltage**: 3.3V • **SCL**: Pin PB2 on microcontroller • **SDA**: Pin PB3 on microcontroller
ULN2003	Stepper Motor Driver board	• **Ground:** GND pin on microcontroller • **IN1 – IN4**: Pins PH0 – PH3 on microcontroller • **Supply Voltage**: 5V
Software	Programming and Visualizing tools	• **Microcontroller**: Keil uVision • **Python**: Version 3.9 • **Library**: PySerial • **Mapping**: Open3D

Detailed Description

Distance Measurement

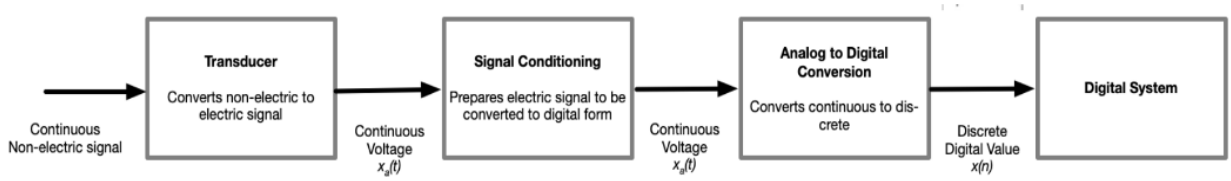
The stepper motor-mounted VL53L1X time-of-flight sensor is in charge of the distance measurement process. One push of the microcontroller's onboard push button PJ1 starts each motor rotation that allows the ToF sensor to scan a single vertical plane. When PJ1 is pressed, an interrupt is set off that causes the motor to rotate clockwise, stop at 11.25 degrees (every 64 steps out of 2048), and allow the ToF sensor to record one measurement.
The formula used,

$$distance = \frac{travel\ time}{2} \times speed\ of\ infrared\ laser\ light$$

Since the travel time represents the total amount of time photons of light take to reach the object and return to the receiver, it is divided by two.

The ToF sensor measures the distance and then performs transduction, conditioning, and ADC internally, to create a digital representation of the analog distance measurement. The transducer process, which turns the non-electrical signal into an electrical signal, is the first step the sensor

conducts. After that, the signal undergoes conditioning, which turns the continuous signal into a discrete digital signal. I2C protocol transfers this digital data to the microcontroller.



Figure#1: Analog to Digital Signal Conversion Process

The ToF sensor boot function is called in the Keil C program to turn on the ToF sensor and allow it to start ranging. The sensor is now ready to take distance readings. Now, the microcontroller uses the interrupt method to wait for a trigger from its peripherals. This is accomplished through the use of an onboard push button, PJ1, which, when pressed, causes the stepper motor to rotate 360 degrees, stopping every 11.25 degrees, to give time for the ToF sensor to take a reading. The TOF's API functions are used to retrieve distance measurements. The microcontroller receives these 32 distance measurements via I2C, and the data is stored in the onboard memory of the microcontroller.

Visualization

The visualization process was done on a Macbook Pro M1 and OS system Sonomo 14.2.1 with 8GB RAM. However, this can be changed since it is not as crucial for the operation of the system. The computer uses a USB-A port and USB-A to USB-Convertor since Mac doesn't have a USB-A port, Python and some common libraries mentioned above. Once the PC receives the measurements, the data is stored in an array in memory and then
written to a .xyz file. Since the data we have is in polar form, where we know the magnitude and the angle.
These two formulas are used to calculate the x and y coordinates.

x-coord = measurement * cos(angle)

y-coord = measurement * sin(angle)

In the Python code, the z coordinate is manually set. After that, the xyz coordinates must be transferred into a xyz file so that the features of the Python open3d module can be used to map them in a 3D plane. This is how each component of a scan is made. Next, open3D packs all of these components together along the z axis to provide us with a full scan.

Application

Note

The 3D Spatial Mapping System has applications in a wide range of fields. By enabling accurate 3D modeling of buildings, it revolutionizes planning and renovation in architecture and can greatly improve design accuracy and efficiency. The system's precise environmental maps, which make it

easier for autonomous robots to navigate through intricate environments like hospitals and warehouses, are also very helpful in the field of robotics.

Furthermore, virtual reality (VR) offers enormous potential for utilizing technology to create immersive environments. The comprehensive 3D maps generated in emergency planning facilitate scenario simulation and evacuation strategy optimization, improving readiness and safety.

The system expands its applications and improves each field's capabilities by combining accurate data collection and sophisticated visualization, which raises the bar for technological integration.

Instructions For User

1. Build the circuit according to the schematic in Figure#6.
2. Download and install Keil uVision, Python v3.9 with library (PySerial) and Open3d.
3. Open the file ‘shahim45_2dx3_project’, type of file ‘uVision5 Project’ using the software Keil uVision.
4. Now to pack click translate, build, and download.
5. Click the ‘Reset’ pushbutton on the microcontroller.
6. Open the .py file ‘2dx3_room_scan’ using IDLE. Change the port in line 13 from ‘COM4’ and then run the file.
7. Enter the number of 360-degree scans you would need to complete the room or hallway being scanned when prompted to enter the number of rotations. Hit the Enter key. Keep in mind that there must be a 60-cm gap between each scan. Line 72 states that the distance between each scan is 600 mm; however, the user is free to enter any other number in millimeters. Close and save the file. Repetition of this step is necessary.
8. To start communication between the laptop and microcontroller, press lowercase "s." and the Enter key.
9. Press ‘PJ1’ on the microcontroller to start the scanning process. Motor rotates in clockwise direction with an angle 11.25 deg.
10. After scanning the motor will rotate anti-clockwise, and go back to its home position (ensure that wires don’t tangle). Move forward to the next scanning spot.
11. Repeat steps 9 and 10 until the scanning process is complete.

Expected Output

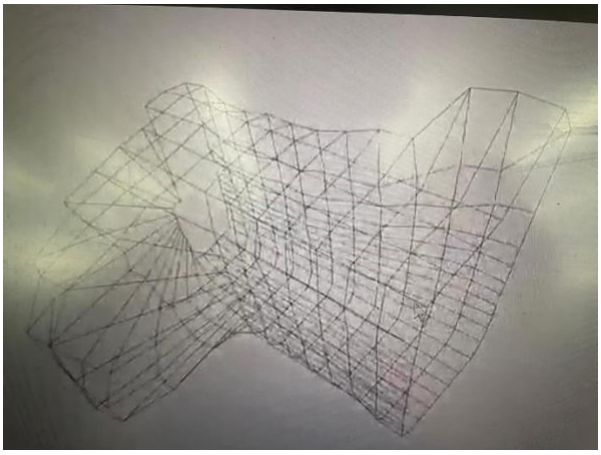
The hallway features a uniform plane wall on the left with a glass door towards its end, and a curved wall on the right, as seen in the screenshots below. As seen in the screenshot, in order to maintain distance contact between each measurement and rotation and obtain accurate readings, we have marked distances using objects. We can see the expected output of the device as follows by comparing the shape of the designated scanning location with the 3D visualization. Below is a comparison of the 3D scan produced by the finalized circuit design with the actual image of the hallway. The system's ability to accurately capture the details of the scanned hallway is demonstrated by the completed scan; this is primarily due to the small step angle of 11.25 degrees that was selected.

Figure#2: Images of the Hallway

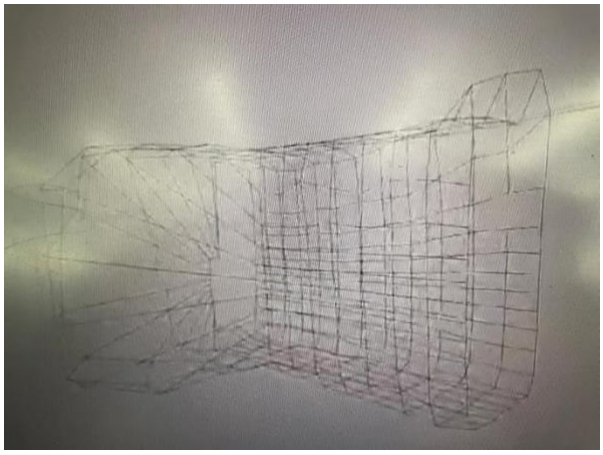
The open3D software generates a point cloud visualization first which can be seen in the screenshots below, which provides a quick overview of how distance points were organized in the 3D plane, before creating the mesh visualization.

Figure#3: Data points visualized used open3d

After joining all of the distance points in a single slice of the distance measurement data and then joining all of the slices together along the z-axis, open3D created the mesh 3D visualization shown below. When we compare the scan to the general layout of the designated hallway, we can see that a high-quality scan was obtained. As we can see below in figure#5, the scan captured the irregular bell-shaped curve wall on the right and the uniform wall to the left.



Figure#4: Top view



Figure#5: Side view

Limitations

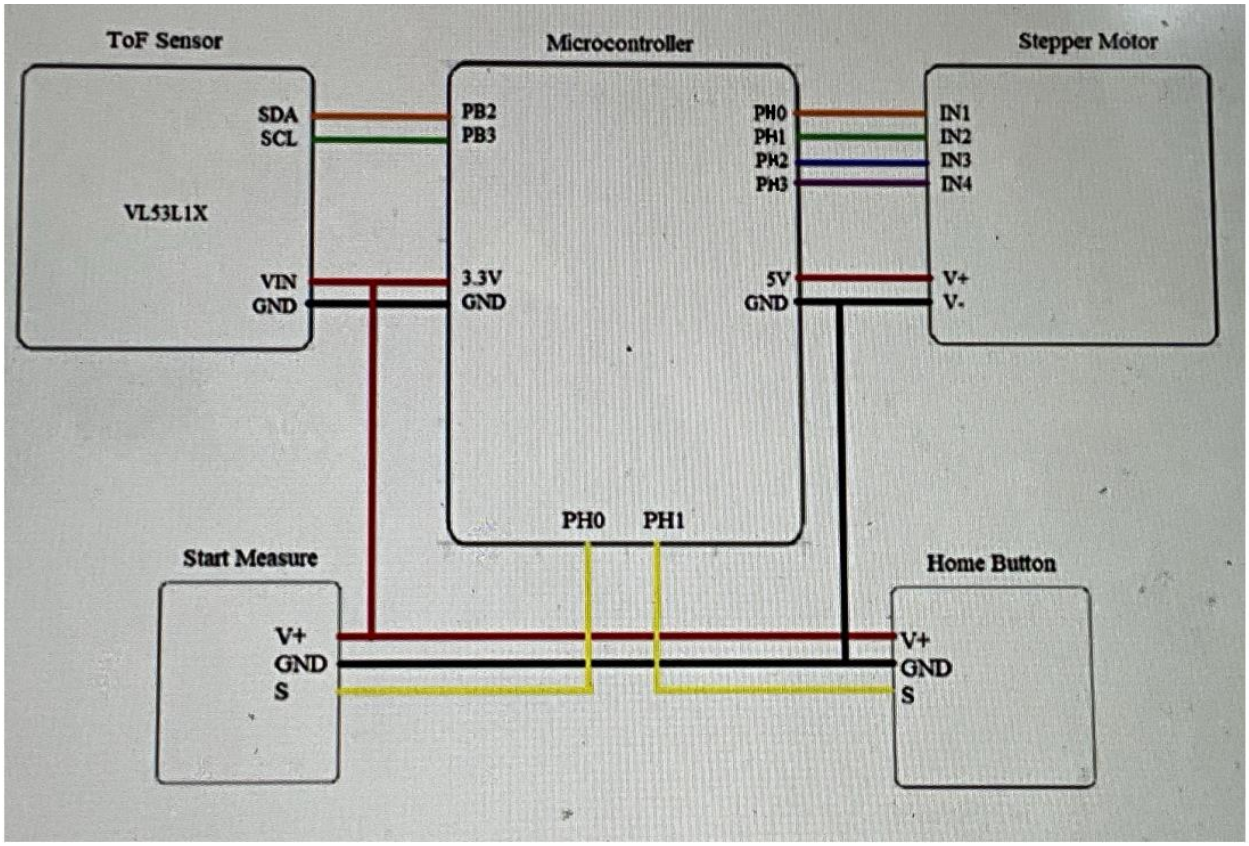
1) The maximum quantization error for the ToF sensor is

$$\frac{4000mm}{2^{16}} = 6.10 * 10^{-2} mm$$

2) I2C protocol is used to communicate between micro controller and TOF sensor. It has a max speed of 3.33 Mbps.

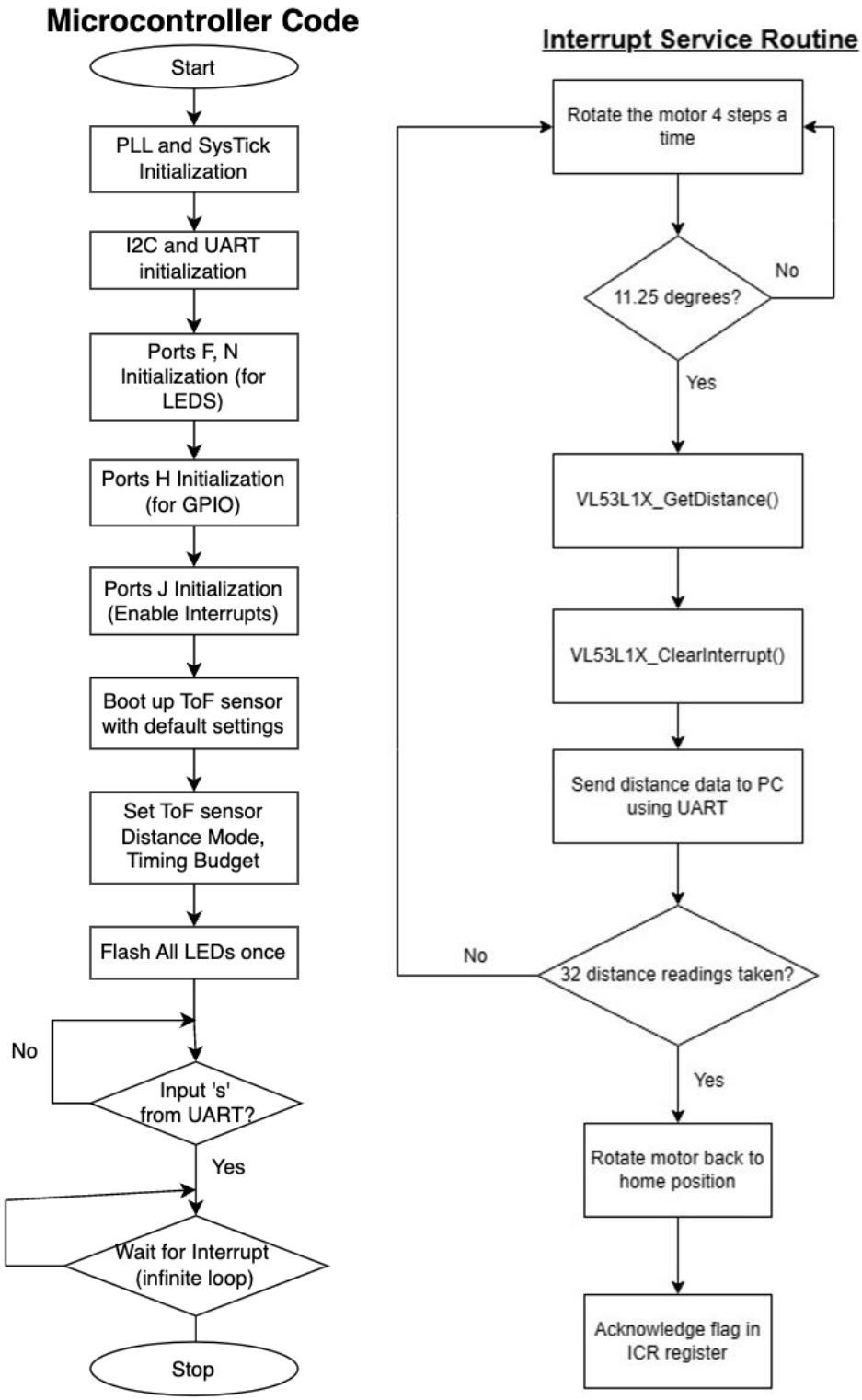
3) UART protocol is used to communicate between micro controller and macbook.It has a baud rate of 115200 and bus speed of 24MHz.
4) The ToF sensor's ranging time is the biggest factor limiting the system's speed. While there are some delays due to the motor's rotation speed, they pale in comparison to the ranging time. The ToF needs a considerable amount of time to calculate the distance as well as to send and receive the light signal. The results showed too many errors when testing with shorter delays for this. As a result, before completing the full 360-degree rotation, the motor must temporarily stop rotating. This requirement is the main restriction that slows down the system's overall speed.
5) Restrictions on the MSP432E401Y microcontroller's use of trigonometric functions and floating point capability: While the microcontroller has a Floating-Point Unit (FPU) that can process single-precision 32-bit commands, trigonometric computations were carried out using Python and its math module. Although this choice sped up the development process, it ignores onboard processing's potential for optimization.

Circuit Schematic



Figure#6: Complete circuit diagram

Flowchart



Figure#7: Microcontroller programming flowchart.

References

- Texas Instruments, “MSP432 SimpleLink™ Microcontrollers – Technical Reference Manual”, 2018, Ch. 4, 17, pp. 326 – 327, 1201 – 1252.
- Department of Electrical and Computer Engineering, McMaster University, "2023_2024_2DX3_Project_Specification", Winter 2022.