COMPENG 2DX3: Final Report

Jane D'Souza (400366436)

C01 – L02

Term: Winter 2024

Submitted: April 16th, 2024

Table of Contents

Device Overview 3
1.1 Features 3
1.2 General Description 4
1.3 Block Diagram 4
Device Characteristics Table 5
Detailed Description 6
2.1 Distance Measurement 6
2.2 Visualization 7
Application 9
3.1 Application 9
3.2 Instructions 10
3.3 Expected Output 11
Limitations 13
Circuit Schematic 15
Programming Logic Flowcharts 16
References 17

Table of Figures

Figure 1: Connection Block Diagram 4
Figure 2: Stepper Motor Wiring 6
Figure 3: Python - Iteration Initialization 7
Figure 4: Coordinate calculations 8
Figure 5: ETB hallway 8
Figure 6: Open3D Visualization 8
Figure 7: Hallway G in ETB 9
Figure 8: Hardware Setup 10
Figure 9: COM4 10
Figure 10: 3D Rendering of Hallway G 12
Figure 11: Bus Speed Change 14
Figure 12: Circuit Schematic 15
Figure 13: Logic Flowchart 16

Table of Tables

Table 1: Feature Overview 3
Table 2: Device Characteristics 5

Device Overview

1.1 Features

Feature Specification	Description	Cost
Microcontroller (MSP432E401Y)	The Cortex-M4F processor is built on a high-performance processor core with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The processor delivers exceptional power efficiency through an efficient instruction set and extensively optimized design, providing high-end processing hardware [1]. Specifications: - 32-bit Cortex-M4F architecture - 120MHz operation - SysTick: A 24-bit count-down timer that can be used as a Real-Time Operating System (RTOS) tick timer or as a simple counter [1]. - NVIC: An embedded interrupt controller that supports low latency interrupt processing [1].	$254.99
Time-of-Flight (ToF) Sensor (VL52L1X)	- Used to measure the planar spatial distance using laser-ranging sensor [2]. - Fast and accurate long-distance ranging Up to 400 cm distance measurement and 50 Hz ranging frequency [2]. - Distance mode: Long mode - 3.3 V	
Stepper Motor	- Unipolar stepper motor - 5V	
Serial Communication: (ToF to micro): I2C (MCU to PC): UART	- Serial communciation between components using communication protocols using UART and I2C.	
Languages: (Collection and Visualization): Python (MCU Communication): C	- Python 3.9 IDLE - C used in Keil	
Baud Rate: 115200 bps Bus Speed: 12 MHz Port: COM4 Memory: SRAM	- Serial COM4 port for UART communciation	
Total Cost:		$254.99

Table 1: Feature Overview

1.2 General Description

This product is an embedded spatial measurement system utilizing a time-of-flight (ToF) sensor to gather data about the surrounding environment. [3]. The system uses a rotating mechanism to measure distances in a full 360° sweep within a single vertical plane (such as the y-z plane) [3]. The spatial data collected is then stored in the system's onboard memory and subsequently transferred to a PC or web application for visualization and graphical reconstruction [3].

1.3 Block Diagram

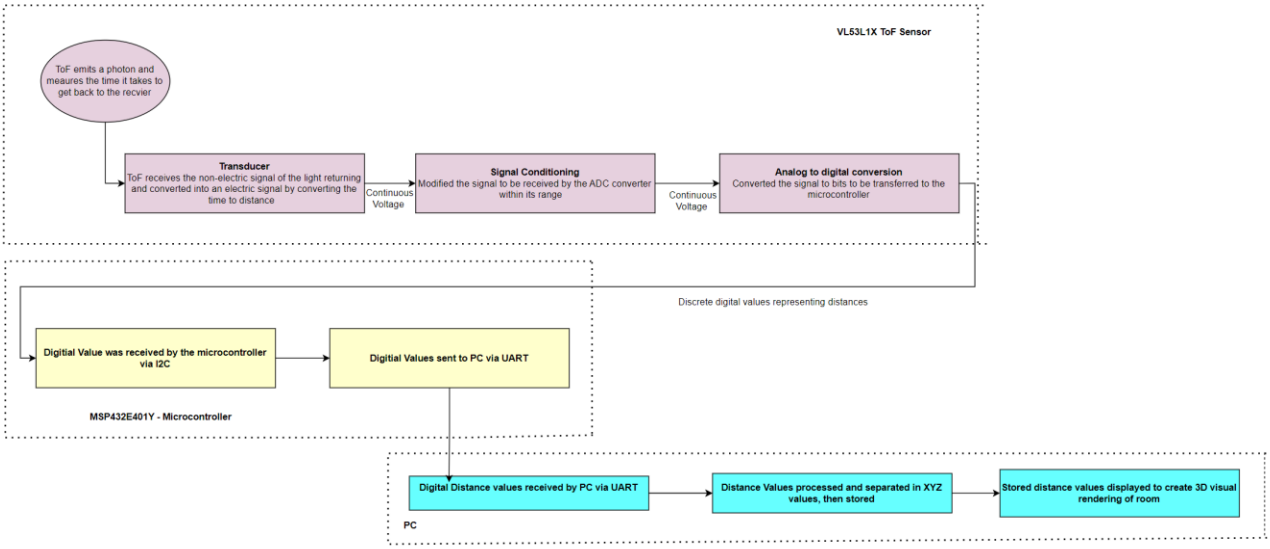


Figure 1: Connection Block Diagram

Device Characteristics Table

The characteristics of the device which are important for users' knowledge are displayed in Table 2 below.

Device Characteristics	**Pins Used**	**Values / LED Assigned**
Measurement Status LED	PF0	D4
Additional Status LED	PF4	D3
Stepper Motor	PH[0:3]	
Start/stop Push Button	PJ1	
I2C - ToF (SDA)	PB3	
12C – ToF (SCL)	PB2	
Bus Speed		12 MHz
Communciation Speed		115200 bps
Serial Port		COM4

Table 2: Device Characteristics

Additional Information:
Libraries used: Serial, Math, time, numpy, open3d (Python 3.9)
Software: Python IDLE, Keil uVision5

Detailed Description

2.1 Distance Measurement

Measurements are obtained by the ToF sensor. The ToF emits (transmits) light until it hits an object and then returns back to the receiver sensor. The sensor measures how long it takes for the emitted light (photons) to come back to the receiver sensor. This time is used in the equation below to get the measurement.

$$Distance = \frac{(Travel\ time of\ light)(Speed\ of\ light)}{2}$$

The microcontroller sends voltage signals to a stepper motor while receiving data from the ToF, which is attached to the stepping motor. The stepper motor does an 11.25° turn for a total of 360° full rotation. With a total of 32 turns for a full rotation, it makes sure the ToF sensor captures the data coordinates of the surrounding environment. The greater the number of measurements per rotation, the more detailed and accurate the final representation of the environment will be. The user can identify whether the data collection is working and the direction (CW or CCW) by identifying the LEDs that are blinking on the microcontroller. Referring to Table 2, the measurement status every 11.25° would flash D4. This LED signifies the data collection at every turn. When the ToF sensor rotates CCW, LEDs D3 and D4 flash on the MCU. This indicates that the stepper motor is rotating CCW at 11.25°. The stepper motor direction is determined by the MCU by applying voltages in a precise sequence to the pins PH [0:3], which are connected to the input pins of the motor IN [1:4]. Figures 2 and 12 provide a visual for the wiring and pin placements. These rotations help the sensor capture the data (x, y coordinates) needed to create the visualization using open3D. The chart below shows the default MCU pin placements from PM [0:3]. Depending on pin initialization and wiring, these pins can change.

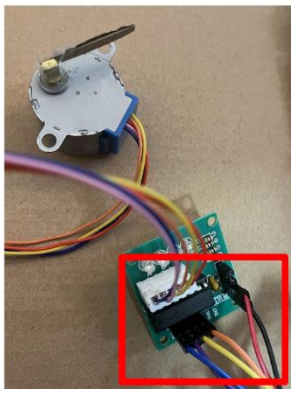
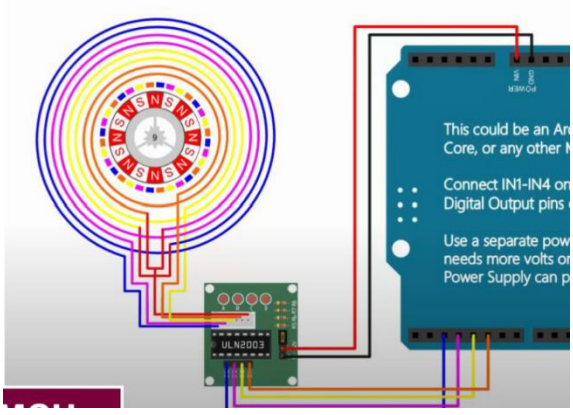


Figure 2: Stepper Motor Wiring

Wire Color	ULN2003 Driver Board	MCU
Red	+ (5 V)	5 V
Black	- (5 V)	Gnd
Blue	IN1	PM0
Purple	IN2	PM1
Yellow	IN3	PM2
Orange	IN4	PM3



After the configuration of the stepper motor and ToF sensor is complete, the next part of retrieving the distance measurement would be to assign the stop/start button. For this device, the button is assigned to PJ1. Once the MCU is loaded with the code and the button is pressed, the device starts

collecting data from the environment. When the MCU detects a low voltage on pin PH0, it interprets this as the button being pressed. Consequently, this triggers the program that controls the motor to spin and initiates the measurement process once more. The next cycle begins after the user moves the system a set distance (ex. 50cm). This setup allows the system to capture measurements throughout the environment (hallway). After repositioning the device forward 50cm, the button needs to be pressed again to switch the motor CCW. This cycle continues the number of times set by the user on the Python data collection file, as seen in Figure 3.

```
# num of measurements per layer and the toal num of layers (REMEMBER: layers start at 0, L-1)
measurements_per_layer = 32
total_layers = 10 #for hallway G (ETB)
#total_layers = 3  #for demo (box)
distance_between_layers = 100 #distance between consectuvie layers
```

Figure 3: Python - Iteration Initialization

Figure 3 shows that iterations of 360° rotations occur 10 times. In the 10 iterations, the user should manually move the system forward 50cm (or the desired distance). The user can determine the number of iterations depending on how many data points are desired and the length of the hallway. After the iterations have been completed, the Python output will close the serial port (COM4), and it will notify the user. The visualization Python module should then run and output a 3D visualization representation of the data points collected by the ToF sensor. The visualization platform is Open3D, which is compatible with Python 3.7–3.9.

2.2 Visualization

PC Specifications –
Host Name: DESKTOP-76OVARU
Processor: Intel64 Family 6 Model 142 Stepping 12 GenuineIntel ~1502 Mhz
Product ID: 00325-81746-55366-AAOEM
System Type: x64-based PC
Windows 11

Libraries sued:
SysTick.c, PLL.c, UART.c, onboardLEDs.c, VL53L1X_api.c, PySerial, math

For visualization, the MCU gets distance measurements from the ToF sensor using the I2C protocol. After each measurement is obtained, it is transmitted to the connected PC through UART. The measurements collected by the PC are compiled into an array. The values collected are converted into (y,z) coordinates with the equations below.

$$y_{coordinate} = Distance * \sin(Motor\ rotation\ angle)$$

$$z_{coordinate} = Distance * \cos(Motor\ rotation\ angle)$$

The program determines the rotation angle of the motor by tracking the number of scans completed during the current cycle. Given that the sensor records a measurement every 11.25° ($\pi/16$), the angle can be calculated by: $(\pi/16) * Scan\ number$. Since the system is moving forward every 50cm, we can calculate the x-coordinate by getting the product of $500mm * cycle\ number$. Now that the (x,y,z) coordinates have been established, the file for the visualization gets generated as "output.xyz". This file contains all the data points collected by the ToF sensor.

Once the cycles are finished, the output.xyz file is closed. It then proceeds to generate a 3D visualization of the gathered coordinates (x,y,z). Figure 4 shows the visualization Python file and the corresponding code for the calculation of coordinates.

```
# angle calcutions (rads)
angle_rad = 2 * math.pi * measurement / measurements_per_layer
y = distance * math.cos(angle_rad) #y coord based on cosin of angle
z = distance * math.sin(angle_rad) #z coord from sin of angle
x = layer * distance_between_layers  # x-coord based on layer num and serpation

# write coords to output file ( x y z format)
myfile.write(f"{x} {y} {z}\n")
```

Figure 4: Coordinate calculations

After completing all the measurement cycles, the .xyz file is closed. The visualization is then created using the stored coordinates. The data coordinates stored get constructed into the visual. Each point is stored as a vertex in an array. Lines are then drawn to connect all vertices within the same plane, so that each scan's coordinates are linked by lines. Additionally, the Python program links vertices across different lanes to stick together the entire shape. Normally, there are 32 vertices per plane, for a total of 10 planes, but this can be changed by the user's desired input. Finally, these connections are visualized in a 3D model that is presented to the user. Figures 5 and 6 show a side-by-side comparison of a hallway conducted in McMaster's Engineering Technology Building (ETB).

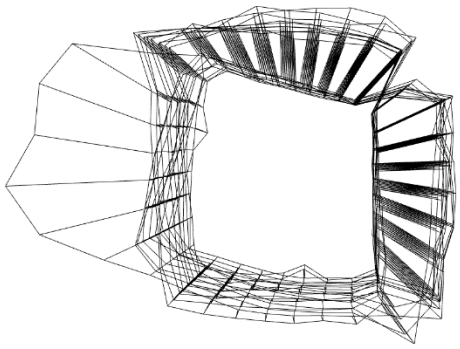


Figure 6: Open3D Visualization

Figure 5: ETB hallway

Application

3.1 Application

This system can be used in an enclosed place (room, hallway, etc.). For this application example, a hallway on the ETB second floor will be used as the environment. This hallway shall be called Hallway G.

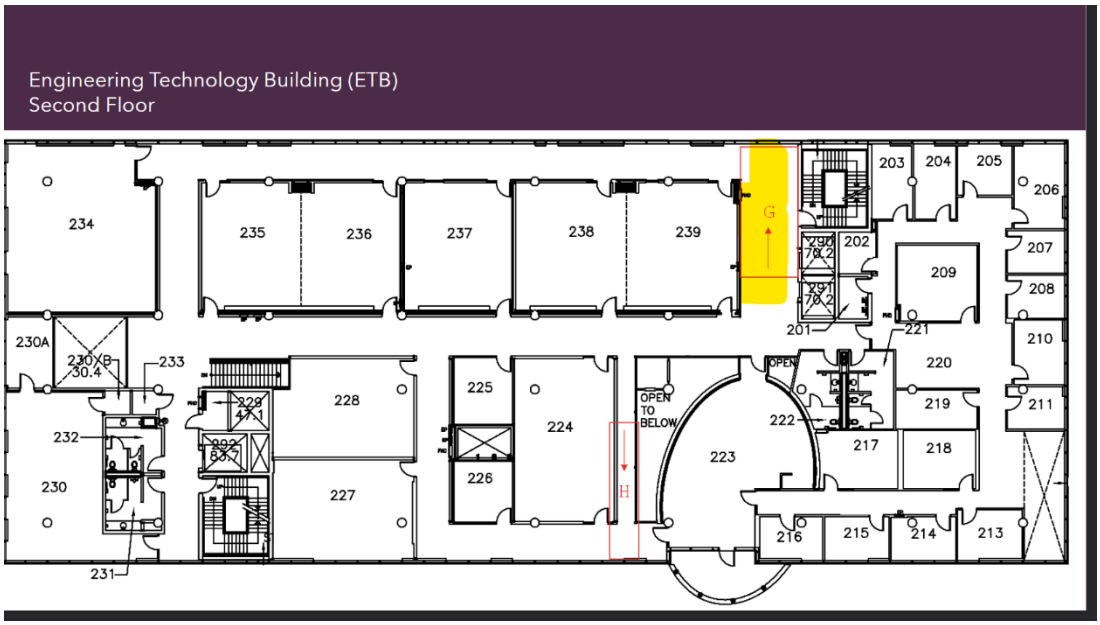


Figure 7: Hallway G in ETB

The length of this hallway is approx. 5 meters, resulting in 10 iterations (cycles). With 50cm between each cycle and a total hallway length of 5m, 10 scans will be sufficient to gather the most data points and conduct a valid 3D visualization of Hallway G. The device should be placed on an accurate movement system, such as a wheeled chair, which will be used for this application. For more accurate results, tape on the floor indicating each 50-cm increment can help the measurements be more precise. This application example will also be using Python IDLE with Python 3.9 to run the code modules, but users can choose other platforms to run the Python files, such as Visual Studio. Users can change values for layers in the DataCollection.py file in Figure 3.

3.2 Instructions

A users guide for the system setup is as follows:

A. Hardware Setup

- Follow Table 2. Device Characteristics for pin arrangements (Figure 8)
- Place stepper motor on blue holder (can be attached multiple ways) as seen in figure 8
- Connect ToF sensor to white sensor holder (can be attached multiple ways) as seen in Figure 8
- Connect Microcontroller to PC using serial port COM4 which can be identified in device manager (Refer to B. Software Set-up below)
- After connection, build -> translate -> load Keil project (studio 8c file) to the microcontroller
-Press RESET on the microcontroller and run the Data collection python file.

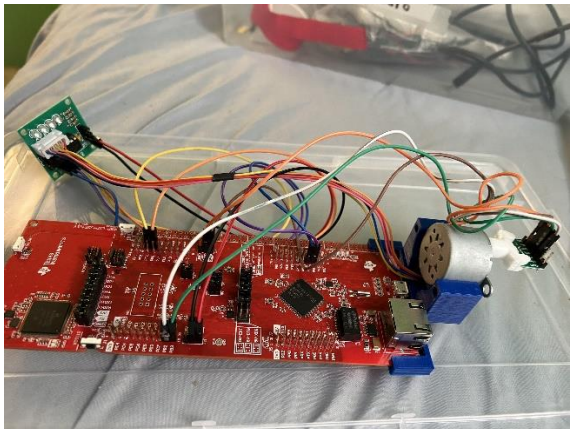


Figure 8: Hardware Setup

B. Software Set-up

- Open Device manager on your PC and confirm the port being used by the MCU. This application example will use COM4. (PC -> Ports-> UART)
- Update COM number on Python file as seen in Figure 9. And confirm if the baud rate is set to 115200.

```
 #setup serial pport conenction with COM port and baud rate with 10s timeout
s = serial.Serial('COM4', 115200, timeout=10) #Port confirmed on device manager as COM4
```

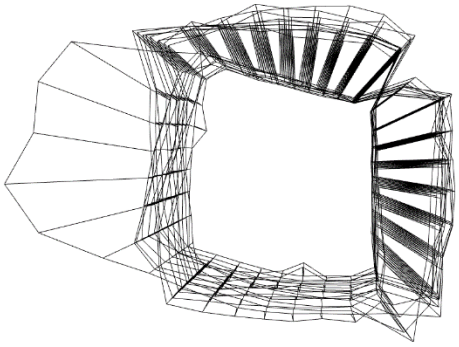
Figure 9: COM4

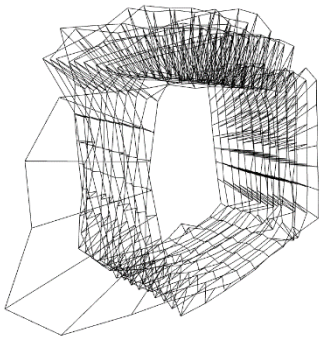
- Referring to Figure 3 above, the number of measurements and layers can be modified depending on user preference.
- With the corresponding bus speed and LEDs set, the user can now begin to use the system to scan the enirоment.

1. When RESET on the MCU is pressed, the user will be prompted to press ENTER to confirm the data collection process. Once RESET is pressed, the button PJ1 on the microcontroller should be pressed after to start the stepper motor rotation.
2. Keeping an eye on the measurements taken, which are displayed on the IDLE shell, once the measurement number hits 31 (starts at 0, therefore total turns are 32), press PJ1 again to switch to CCW, which also increases the layer count, and continue taking measurements until the iterations are done.
3. User must move the system 50cm forward to get the next set of measurements.
4. On the last cycle, the program will stop by itself without user input of pressing the button. The shell will output: “CLOSING COM4”, which indicated the data collection is done and no more measurements are being taken.
5. The user will then need to run the Visualization.py file on Python IDLE shell. Running this file will automatically open the Open3D.exe app and a generated representation (File name: Output.xyz) of the hallway which can be seen in Figure 10.

3.3 Expected Output

Looking at figure 10, the dimensions and surroundings of the hallway are identifiable. The last couple of layers have a sudden spike to the side of the square shape (hallway). This is due to another hallway opening at the end of Hallway G. This can be seen in Figure 10 behind rooms 235-239. Some discrepancies in these images are due to the protruding glass casings that are fixed to the walls. The output is very similar to hallway G but has the potential to be more accurate by reducing the distance between measurements. This way, more data points can be collected that the user would be missing out on when moving up 50cm. The greater the number of data points, the more precise the scan will be.





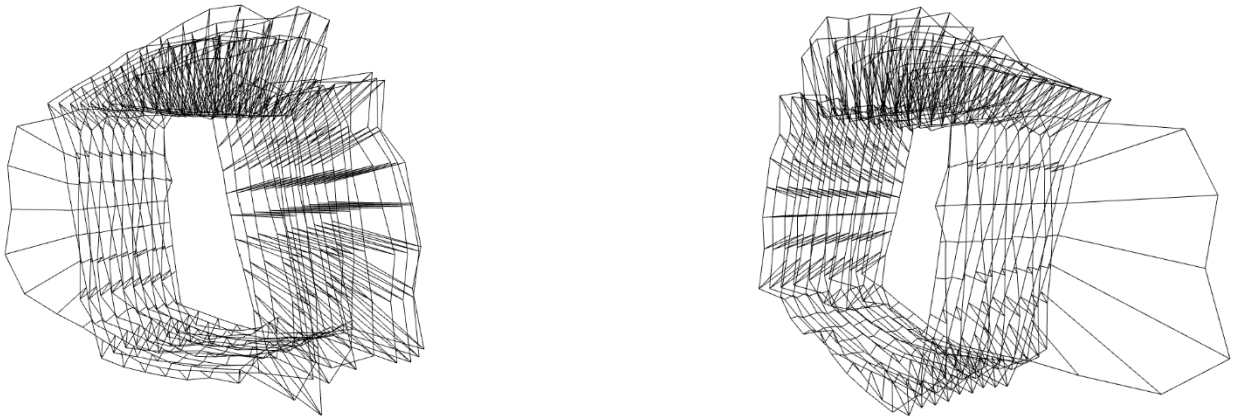


Figure 10: 3D Rendering of Hallway G

This device, and similar systems, can serve multiple purposes. For instance, it could assist individuals with visual impairments. By converting spatial data into auditory signals instead of visual displays, it can provide users with better awareness of their surroundings. Similar systems to this device can possibly emit sounds when the user approaches an object, aiding in spatial perception. This device can aid in measuring spaces for planning and arranging various elements within them (interior design, for example). It is useful in conveying information about areas that have limited lighting and are unsafe for people to access.

Limitations

(1) Summarize any limitations of the microcontroller floating point capability and use of trigonometric functions

The MCU floating point units can generally handle basic 32-bit operations. However, the trigonometric equations used require complex 64-bit operations. As a result, it has to be divided into two 32-bit segments. This can cause some slight discrepancies with rounding errors and can potentially diminish the precision of y-z plane results.

(2) Calculate your maximum quantization error for each of the ToF module.

$$Max\ Quantization\ error = \frac{\max range}{2^n} = \frac{4000mm}{2^{16}} = 0.061\ mm/bit$$

(3) What is the maximum standard serial communication rate you can implement with the PC. What speed did you implement and how did you verify?

115200 bps would be the maximum standard serial communication rate. This is verified by checking the device manager settings on my PC and verifying them on my UART properties. The baud rate can also be verified on Realterm since it is the maximum value available.

(4) What were the communication method(s) and speed used between the microcontroller and the ToF modules?

The communciation methods between the microcontroller and the ToF modules would be I2C where SDA and SCL is connected to PB3 and PB2 respectively. I2C is a serial communciation method which interfaces with systems and peripherals. The speed is about 100kbps with a clock pulse (bit-by-bit).

(5) Reviewing the entire system, which element is the primary limitation on system speed? How did you test this?

After reviewing the entire system, the element that was a primary limitation to system speed was the ToF sensor. When we increase the number of measurements taken every iteration, we can see the increase in the time. Capturing the measurements took around 0.5s to collect and send to the array, but it was the most time-consuming task done by the MCU. Without the ToF, the program completion time would be cut by half.

(6) Show the steps and calculations to configure your assigned system Bus Speed from Table 1

The individualized bus speed based upon the least significant digit of my student number (6):

1. View Table 1 and get my assigned Bus speed which is 12MHz.
2. Open Keil-> PLL.h and find the corresponding PSYSDIV value (39)
3. Change the PSYSDIV value to 39 (Line 29 of PLL.h file)
4. Open Systick and change SysTick_Wait to 1200000 as seen in Figure 11

```
67 void SysTick_Wait10ms(uint32_t delay){
68   uint32_t i;
69   for(i=0; i<delay; i++){
70     SysTick_Wait(1200000);  // wait 10ms
71   }
```

Figure 11: Bus Speed Change

Calculation is as follows:

12MHz:

$$\frac{120{,}000{,}000}{10} = 1200000$$

Circuit Schematic

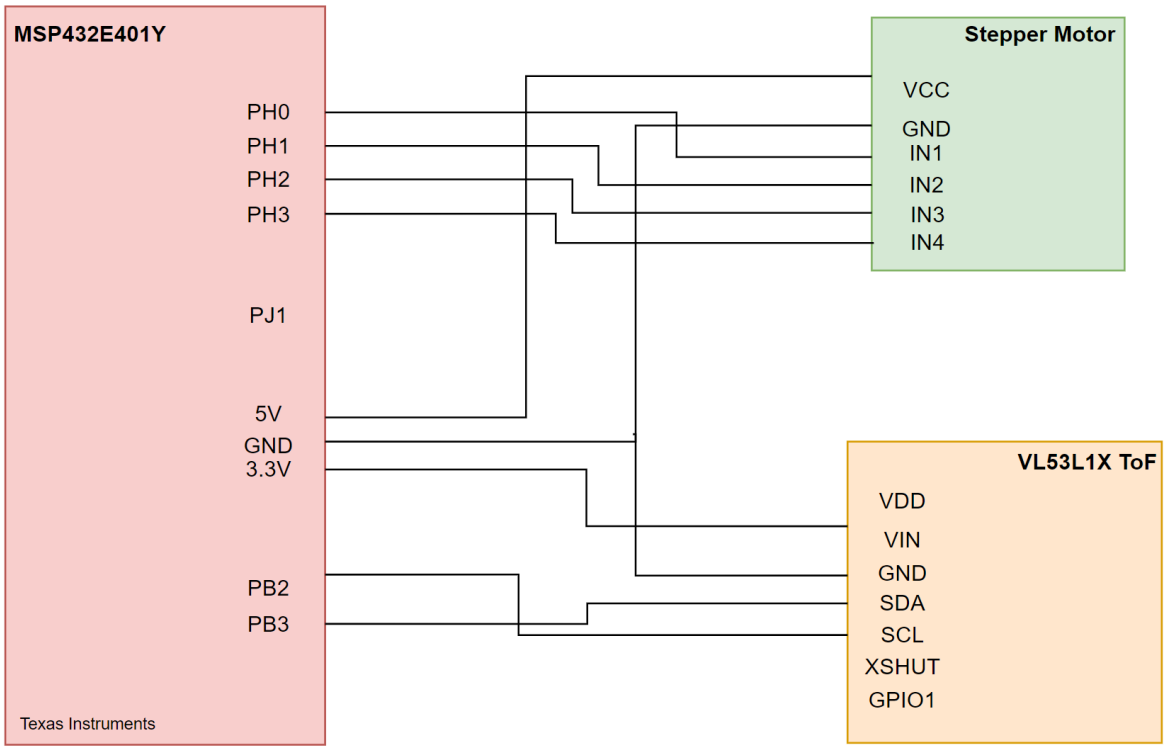


Figure 12: Circuit Schematic

Programming Logic Flowcharts

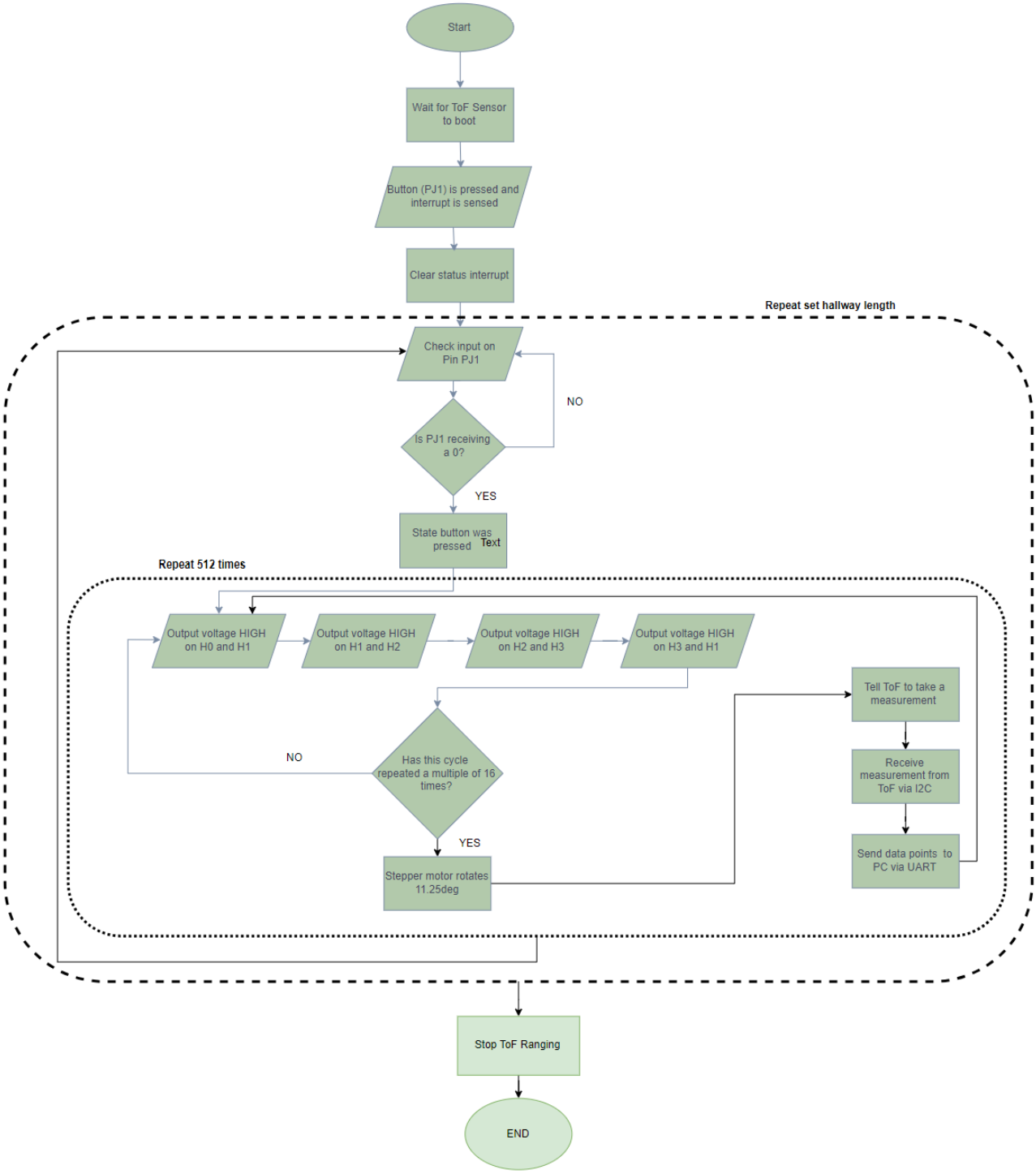


Figure 13: Logic Flowchart

References

[1] *MSP432E4 SimpleLink Microcontrollers - Technical Reference Manual,* Texas Instruments, October 2018. Accessed: April 18th, 2024

[2] *VL53L1X,* ST, Rev 3, November 2018. Accessed: April 18th, 2024

[3] S. Athar, T. E. Doyle, Y. Haddara*, Computer Engineering 2DX3 2023-Project Specification.* Hamilton: DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING, MCMASTER UNIVERSITY, 2024.