2DX: Microprocessor Systems Final Project

Instructors: Drs. Boursalie, Doyle, and Haddara

Adam Poonah – poonaha – 400338309 – 2DX3 – Wednesday Evening – L03

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 **[Adam Poonah, poonaha, 400338309]**

Device Overview

Features:

This spatial mapping system is a complex system integrated with the VL53L1X Time-of-Flight sensor alongside the UNLN2003 stepper motor to reconstruct a 3D graphical representation from 360-degree measurements of distance. The device has various applications ranging from transmitting measurements directly to the user's PC all the way to advanced reconstruction of hallways in a presentable format.

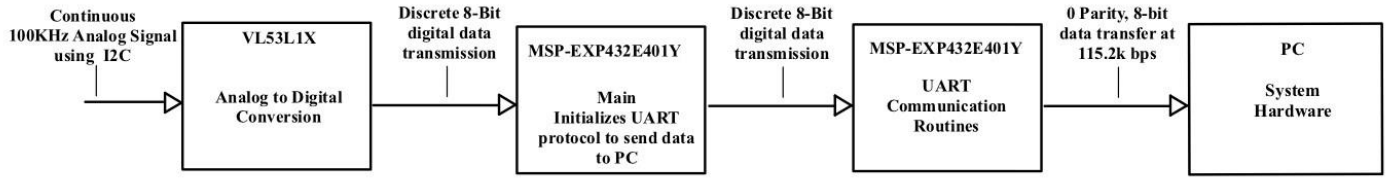
- Compact and Portable module
  - Size: 10.5cm × 20cm ×6cm
  - Secure motor placement secured with two screws and brackets
  - Sleek design for easy storage
  - Dedicated USB channel for ease of use
  - Open top design for microcontroller visibility and access
  - Extended wires for sensor readjustments
- Eases of use
  - Software is simply plug and play with very little set up
  - Minimal components with single pushbutton for start and stop
  - Use of a single power supply
  - Both I2C and UART integration
- Hardware specifications
  - 20MHz Bus speed
  - Clock speed up to 120MHz
  - Operating voltage of 4V for the MSP-EXP432E401Y Microcontroller
  - Operating voltage of 2.6V – 3.5V for the VL53L1X Time-of-Flight sensor
  - Operating voltage of 5V for the UNLN2003 stepper motor
  - 1024KB of flash memory
  - 256KB of single-cycle SRAM
  - 20 ADC channels with 2 Msps sample rate
  - Cost of approximately $101.30 CAD
  - Programmed in C++ and Python
  - Uses I2C and UART communication protocols
  - Baud rate of 115200bps

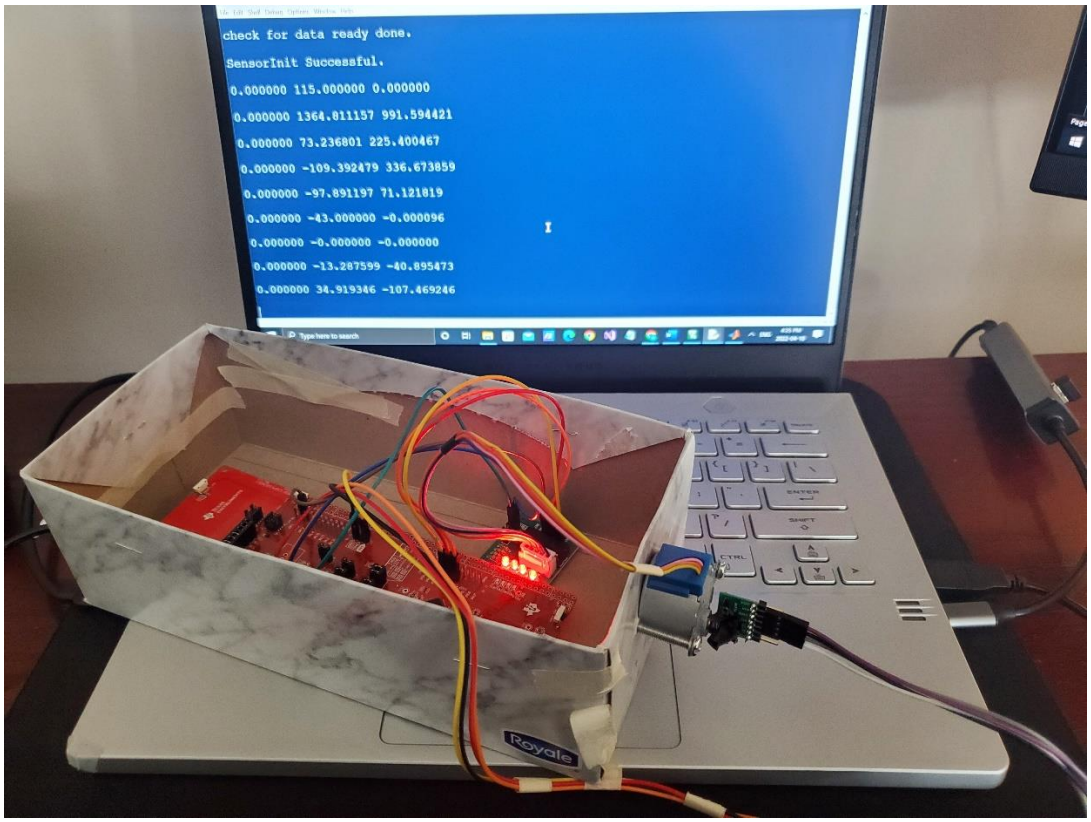
General Description:

This system is comprised of a MSP-EXP432E401Y Microcontroller, VL53L1X Time-of-Flight sensor, and UNLN2003 stepper motor. The microcontroller used in this system is a powerful microcontroller designed by “Texas Instruments” and utilizes the Arm Cortex-M4 microprocessor [1]. The device is equipped with several GPIO ports for input and output with onboard pushbuttons and reset buttons, all of which are utilized to create a fluent embedded system. This choice of microcontroller is ideal for a system like this as it provides top performance with the advantage of support from multiple developer tools allowing for easy customization [1]. This system also uses a Time-of-Flight sensor which is equipped with a fast and miniature sensor that can detect distances up to 4m away with precise accuracy [2]. This is done by emitting light, and when it returns back it measures how long it took to receive the pulse back, therefore calculating the distance. This data is communicated with the user’s PC using the I2C protocol.

Lastly, this system is equipped with a stepper motor capable of spinning 360 degrees at various speeds [3]. The way the system uses these parts is by having the time-of-flight sensor mounted onto the stepper motor, and by clicking the onboard button it will rotate, capturing several measurements. These measurements are then transferred to the user’s PC using the UART protocol, which it is plotted using 3D graphical software in Python.

Block Diagram:






```
Python 3.7.7 (tags/v3.7.7:d7c567b08f, Mar 10 2020, 10:41:24) [MSC v.1900 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\AdamA\Desktop\2Nd Year Labs\1. Winter 2022\2DX3 Notes\Labs\Final
Project Milestone 2\2DX_2022_Studio_9_E2_Python.py
Opening: COM3
Press Enter to start communication...
Program Begins

2DX4 Program Studio Code 1

(Model_ID, Module_Type)=0xeacc

ToF Chip Booted!

 Please Wait...

writing configuration failed with (-1)

check for data ready done.

SensorInit Successful.
```

Device Characteristics Table

Weight	- 1.34 kg
Pins	- PM0-PM3 - PB2-PB3 - PJ1 - GND Pin ×2 - 5V Pin - 3V Pin
Bus Speed	- 20MHz Bus speed
Baud Rate	- 115200bps
Clock Speed	- Up to 120MHz
Libraries	- cmath - standard python libraries - Open3D
Dimensions	- 10.5cm × 20cm ×6cm
Voltage Requirements	- 5 volts for the UNLN2003 stepper motor - 2.6 volts – 3.5 volts for the VL53L1X Time-of-Flight sensor - 4 volts for the MSP-EXP432E401Y Microcontroller
Serial Parts	- VL53L1X Time-of-Flight sensor - UNLN2003 stepper motor - MSP-EXP432E401Y Microcontroller
Input/Output	- Micro USB port - USB 3.0 output
Software Requirements	- 1.5 GHz 64-bit processor [4] - 2GB of RAM [4] - 1GB of Disk space [4] - Python 3.6-3.9. (3.10 NOT supported)

Detailed Description

Distance Measurements:

The distance that the device captures is done through an analog to digital conversion process within the time-of-flight sensor. The time-of-flight sensor utilizes I2C to communicate to the microcontroller in which it sends an analog signal at a frequency of 120KHz. The physical device is equipped with various pins; however, this system utilizes the VIN, GND, SDA, and SCL pins.

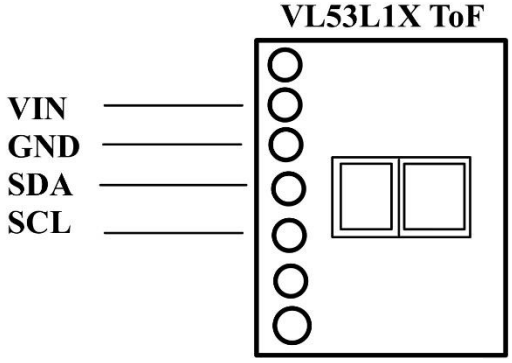


Figure 1

The input voltage must be supplied with a voltage between 2.6V and 5.5V, and the purpose behind the SDA and SCL pins is for I2C data and clock lines respectively. The ToF sensor is equipped with two miniature peripherals', a sensor, and an emitter. The method in which the device collects data is by the emitter sending a pulse of light and once the object hits its target, the light is then projected back to the sensor. The sensor then measures how long it took to travel to that object, and by using the following equation distance is measured:

$$Measured\ Distance = \frac{Photon\ Travel\ Time}{2} * speed\ of\ light$$

This distance measurement is then communicated to the microcontroller via I2C, then the stepper motor is rotated 36 degrees where the time-of-flight sensor once again calculates a distance measurement. This process is repeated for a full cycle, with as many scans as the user programs. Once a cycle is completed, the stepper motor will automatically begin to rotate in the reverse direction to bring the wires back to their original position to prevent any damage.

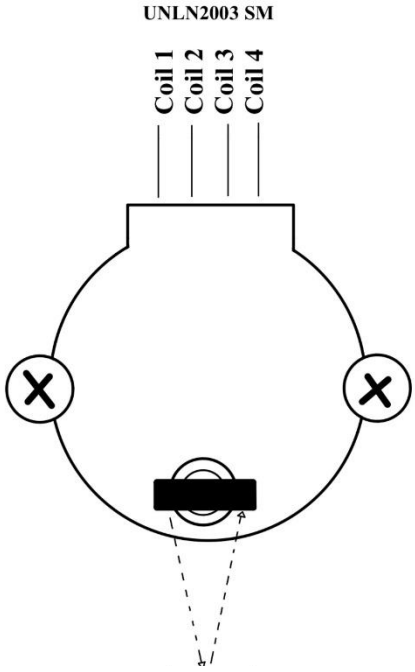


Figure 2

As illustrated in figure 2, the stepper motor is mounted onto a secure casing which will rotate while the time-of-flight sensor collects measurements. Once these measurements have been collected, they are sent to the microcontroller using I2C, and then manipulated within "Kiel". The distance measurement is broken down into three dimensions, the x, y, and z components. The x component is a constant measurement that depicts the action of the user taking a step forward at a constant distance. The y and z components reflect our axis of measurement, found by the equation below.

$$y = z * \cos(angle)$$

$$x = z * \sin(angle)$$

Both of these measurements are calculated in milliammeters, with the "angle" being calculated based on how far the stepper motor rotates clockwise from its starting position.

After each successful scan, the x, y, and z coordinates are then transmitted to the user's PC using UART communication protocol at a baud rate of 115200bps. The corresponding values can be found by launching the Python file included in which the user can see the values listed once they are received. Once the final rotation is completed, the Python program will visualize the received values using the Open3D library and plot the corresponding values with connected vector lines for a complete spatial render.

A flowchart of what the microcontroller is doing can be seen below in figure 3:

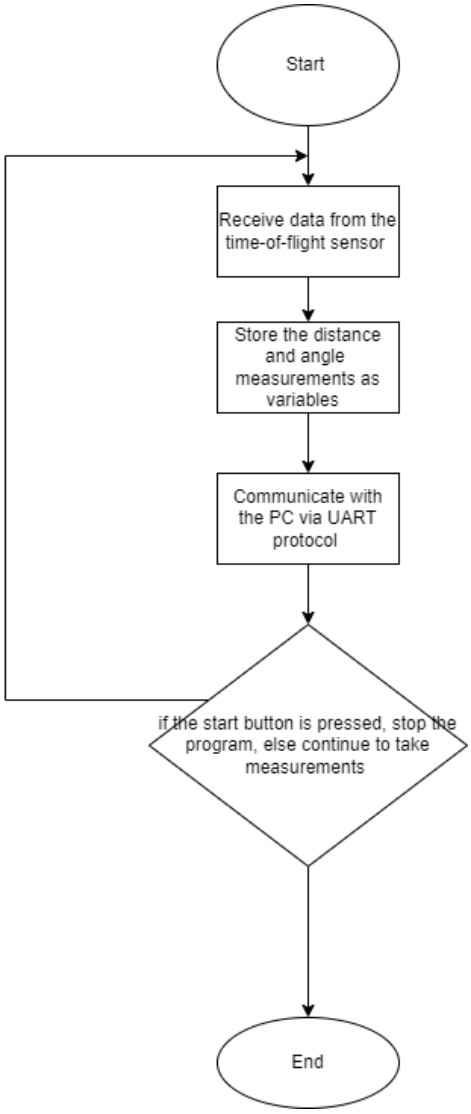


Figure 3

Visualization:

The computer is used to receive the transmitted data from the microcontroller and utilizes Python and Open3D libraries to convert these data points into a 3D spatial map. For this system to function correctly, Python versions 3.6-3.9 (3.10 NOT supported) must be used for Open3D to function correctly. The following hardware components that have been tested for this system can be seen below.

Tested Hardware Configurations:	
Operating System	Windows 64-bit
RAM	16gbs
Disk Space	1TB
CPU	AMD Ryzen 9 4000 Series
GPU	GeForce RTX 2080

The following functions/libraries were also used:

Libraries	**Functions**
Numpy	Spin()
Open3D	ReverseSpin()
Serial	Stop()
Cmath	UART_printf()
	decode()

The method that which the data is stored is by creating the specified dimensions within the Keil program by utilizing the “cmath” C++ library to use the trigonometric functions. Once the microcontroller receives the data from the time-of-flight sensor, it is then used as a parameter for the equations above to find the y and z values of the measurements. These values are converted to a floating-point variable which is then sent to the PC with UART. The data is then displayed onto the Python terminal by using the decode () function and then finally printing it with “printf”. Once all the values are displayed, open3D visualizations are called where the vertexes are created and finally, the 3D mapping is produced.

The process to visualize the data starts by first loading the point cloud from the data collected by the sensor into Open3D. The visualization function is called which will result in a point cloud to be displayed to the user. Next, a for loop is run which connects the planes of the mapping to each of the vertices. Lastly, to complete the process the vertices are connected to the planes to produce a final 3D spatial mapping of the environment. As seen in the figure below, an accurate reconstruction of the hallway outside of ITB 143 can be seen using the system. (Note: Wire interference near the end of the mapping process).

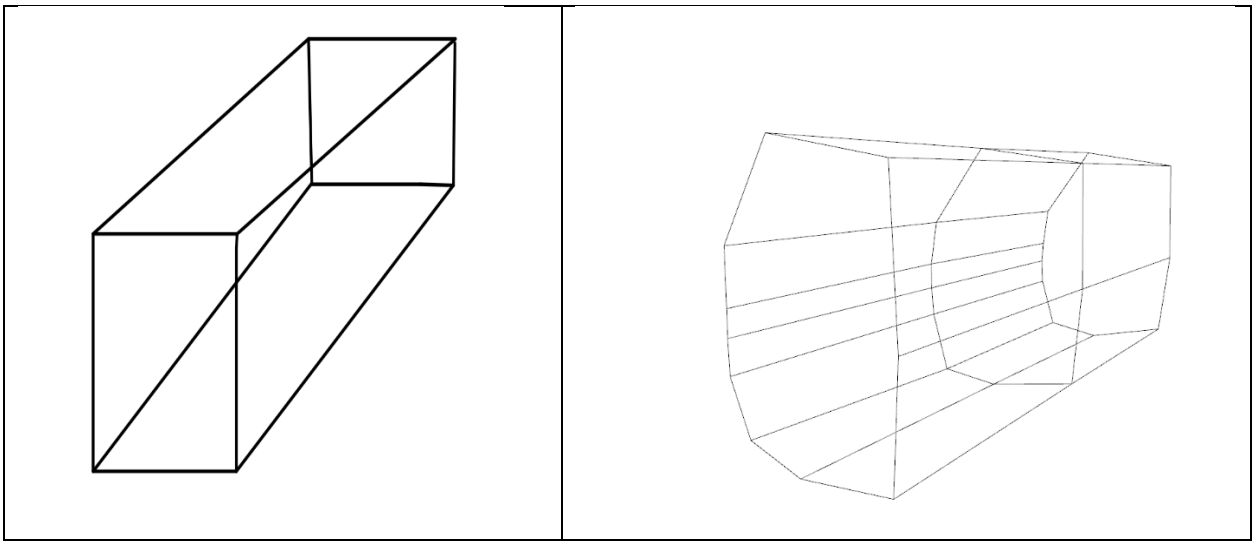


Figure 4

Application

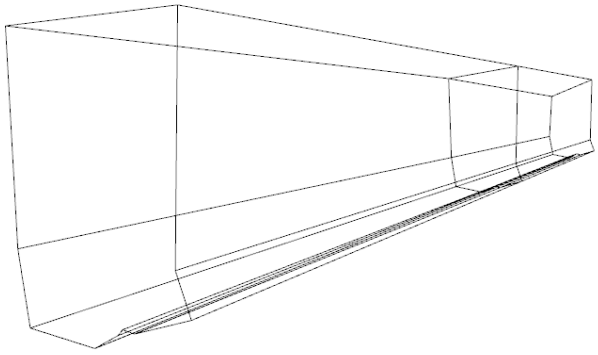
1. Insert the system into a computer via the USB 3.0 cable.
2. Hold the device in your hand at the centre of the room you wish to map
3. Launch Keil, open the 2dx_studio8c.c file
4. Press translates, build, and then load
5. wait for the LEDs on the microcontroller to being to flash
6. Click the restart button located near the micro-USB cable
7. Open up the attached Python file using IDLE/ the IDE of your choice
8. Run the python file and click enter once the communication is set up
9. Press the on-board pushbutton ladled as PJ1
10. Wait for a full 360-degree scan to complete
11. Once the motor begins to rotate in the opposite direction, take a step forward at about 30cm
12. Repeat steps 10 – 11 two times.
13. The 3D spatial mapping will now be displaced

Set Up- Guide:

1. Install the latest version of Kiel and any version of python between versions 3.6-3.9.
2. Install pyserial by searching for “Command Prompt” in the Windows search bar, once in type “pip install pyserial”.
3. Find the port on PC by opening “Device Manager”, and under “Ports(COM & LPT), write down the COM# of the UART port.
4. Navigate to the Python file included and change the “COM3” ladled at the top to your specific port.
5. To confirm the set-up is done correctly, run the Kiel program and restart the microcontroller
6. Run the Python file and click enter, distance measurements should now begin to display.

Expected Output:

A common method used to check if the system is operating as expected is to enter an empty room/hallway and begin measuring at the center of the room. This will make an accurate depiction of the hallway. Assuming the user uses the x dimension as a constant measurement for the forward movement, the expected output should be a rectangular prism that has a length of 60cm. As seen in the figure below, the completed spatial 3D render appears to be in the form of a hallway as expected. However, there are some minor errors at the bottom of the visualized image, which can be attributed to the wires connecting the time-of-flight sensor to the microcontroller getting in the way. If these wires were not in the way, the scan would have been near perfect to the real location.



demofile2dx - Notepad

File Edit Format View Help

```
0.000000 629.000000 0.000000
0.000000 419.879669 305.060760
0.000000 81.889252 252.030060
0.000000 -62.112675 191.162277
0.000000 -56.631271 41.144855
0.000000 -18.000000 -0.000040
0.000000 -9.708184 -7.053450
0.000000 -12.978585 -39.944416
0.000000 47.589195 -146.462509
0.000000 130.252136 -94.632874
3000.000000 620.000000 0.002769
3000.000000 482.172302 350.322510
3000.000000 80.961090 249.177246
3000.000000 -62.422623 192.113037
3000.000000 -57.440475 41.732384
3000.000000 -22.000000 -0.000147
3000.000000 -12.135185 -8.816875
3000.000000 -12.669378 -38.993420
3000.000000 46.971802 -144.560196
3000.000000 130.252563 -94.632294
6000.000000 609.000000 0.005440
6000.000000 499.968811 363.256348
6000.000000 81.577873 251.079773
6000.000000 -62.732506 193.063797
6000.000000 -61.485779 44.671009
6000.000000 -23.000000 -0.000279
6000.000000 -9.708117 -7.053543
6000.000000 -10.815174 -33.287113
6000.000000 46.972446 -144.559982
6000.000000 128.634933 -93.456161
```

Figure 4

User Guide

1. Plug the system into a computer using USB 3.0
2. Find the port on PC by opening “Device Manager”, and under “Ports(COM & LPT), write down the COM# of the UART port.
3. Open up the attached Python file and replace the COM3 with your specific port.
4. Download and open the Keil file
5. Press translates, build and then load
6. Open up the command prompt using Windows search
7. Install pyserial by typing “pip install pyserial”.
8. Open the attached Python file
9. Press the restart button on the microcontroller
10. Press the enter key on the Python application
11. Press the PJ1 on-board push button the microcontroller
12. Wait for the system to do three full scans, after each scan take 30cm forward.
13. Note that the program takes 2-3 minutes to generate the visualized graph once the distance measurements are completed
14. Done!

Limitation

(1) With the microcontroller program utilizing floating points to hold the values of the x, y and z dimensions, rounding errors will occur which can lead to inaccurate measurements. Also, the way in which computers use the trigonometric functions an error may occur where the output is calculated within the wrong quadrant, leading to a significant change in results.

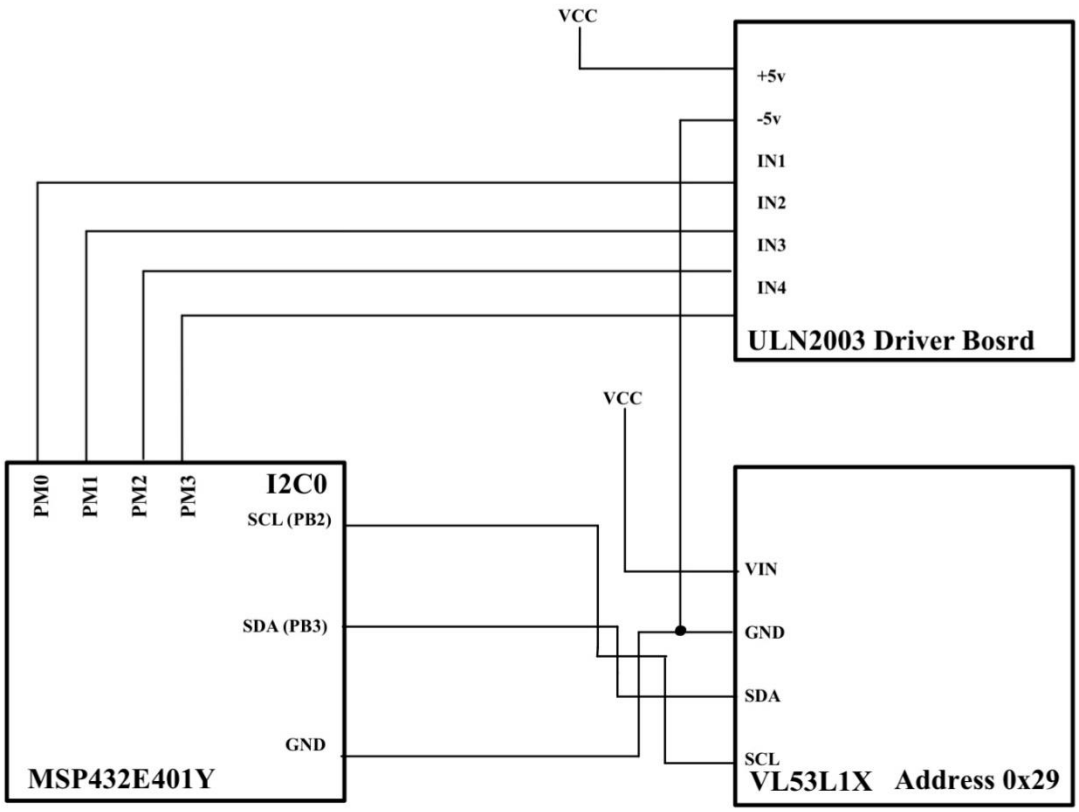
(2) Since Maximum Quantization Error can be modeled by: $\frac{V_{FS}}{2^n} = \frac{4000}{2^{16}}$, this is because we want to find out the quantization error of the ToF sensor's ADC conversion which only applies to its maximum distance which is 4000mm. Using the voltage as the numerator would not apply in this case as the ToF is not converting voltages, but instead distance.

(3) The maximum standard serial communication rate you can implement with the PC is the baud rate which was 115200 which was verified by using the device manager to view the UART port user properties.

(4) The communication method between the microcontroller and ToF module is I2C. The speed used between these two is 100KHz.

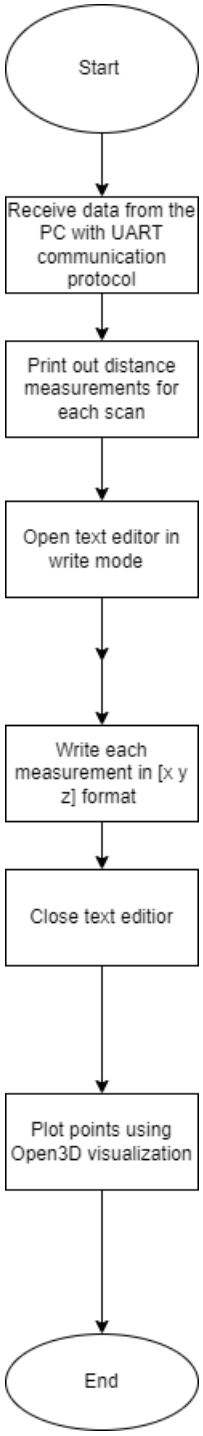
(5) Reviewing the entire system, the primary limitation on speed is the ToF, as it has the lowest speed of all of the devices used with 100kHz.

Circuit Schematic



Programming Logic Flowchart(s)

A flowchart of the Python code can be seen below:



Reference

[1] “Cortex - M4 Technical Reference Manual/ Instructions Set - COMPENG 2DX3:Microprocessor Systems Project.” https://avenue.cllmcmaster.ca/d2l/le/content/418585/viewContent/3472453/View (accessed Apr. 09, 2022).

[2] “VL53L1X Datasheet, PDF - Alldatasheet.” https://www.alldatasheet.com/view.jsp?Searchword=Vl53l1x%20datasheet&gclid=Cj0KCQjwgMqSBhDCARIsAIIVN1V7fnHAbqDNCxIoIYQLRB77Kj-EOxhhvR457wBa6sC81iszBUtIGpoaAv0hEALw_wcB (accessed Apr. 09, 2022).

[3] “2DX4 Lab 4 ULN2003A-PCB Stepper Motor Drivers Datasheet - COMPENG 2DX3:Microprocessor Systems Project.” https://avenue.cllmcmaster.ca/d2l/le/content/418585/viewContent/3593206/View (accessed Apr. 09, 2022).

[4] “System Requirements.” https://www2.keil.com/system-requirements/ (accessed Apr. 09, 2022).