2DX: Microprocessor Systems
Final Project

Instructor: Drs. Boursalie, Doyle, Haddara

Grace Xing – xingg – 400327772
– 2DX3 – Monday Evening – L06

Device Overview

Features

- Embedded spatial measurement system used as a cost effective and more compact alternative to LIDAR
  - Suitable for indoor exploration and navigation
  - Generates a 3d map of surroundings using the Python library Open3D
  - Overall Cost: $95.29
- Texas Instrument MSP432E401Y SimpleLink™ Ethernet Microcontroller
  - Arm Cortex-M4F Processor with FPU
  - Default Bus Speed of 120 MHz (bus speed used for this system is 30 MHz)
  - 1024 KB of Flash Memory,
  - 256 KB of SRAM
  - 6KB of EEPROM
  - Operating Voltage: 4.75-5.25 V
  - Two 12-bit SAR-Based ADC Modules, each can support 2 Msps
  - Programmed in C
  - Cost: $73.40
- 28BYJ-48 Stepper Motor with ULN2003 Driver
  - 512 Steps per rotation
  - Operating Voltage: 5-12 V
  - Cost: $6.99
- VL53L1X Time of Flight Sensor
  - Up to 400 cm distance measurement
  - Up to 50 Hz ranging frequency
  - I^{2}C interface (up to 400 kHz)
  - Operating Voltage: 2.6-3.5 V
  - Cost: $14.99
- Serial Communication
  - UART communication between microcontroller and PC with a baud rate of 115200 bps
  - I2C communication between TOF sensor and microcontroller
- Data Visualization
  - Programmed in Python3
  - Utilizes Open3D for data graphing to form a 3D map of the surrounding

General Description

The embedded spatial measurement system generates a 3D map of the surrounding indoor area. The system is able to collect distance measurements through the VL53L1X Time of Flight sensor. It emits pulses of infrared light and measures the time it takes for each pulse of light to reach the nearest object and reflect back to the detector. The sensor has an internal transducer, precondition circuit and ADC that converts the continuous analog signal to discrete data which is then sent to the microcontroller via I2C communication.

By attaching the VL53L1X Time of Flight sensor to the stepper motor, 32 data measurements are collected at every 11.25-degree rotation of the motor.

A single vertical geometric plane (yz) is produced for each 360-degree rotation of the motor. The system is physically moved every 10cm along the orthogonal axis (x-axis) for each vertical slice.

The data is then transmitted serially from the TOF sensor to the microcontroller through I2C communication. Within the microcontroller program the raw data is processed. The distance measurements (cylindrical) are converted into cartesian coordinates and sent to the PC through UART via USB at a baud rate of 115200 bps.

The data is sent byte by byte to the PC where it is written into a xyz file in the Python3 program. After all the measurements have been written into the file, the data is parsed for 3D visualization using Open3D.

Every measurement in a single vertical slice is connected to create multiple separate planes. These individual planes are then joined together to form a 3D visualization of the surrounding area.

Within the system, there is also an external push button that uses polling to control the start/stop of the data collection.

Block Diagram

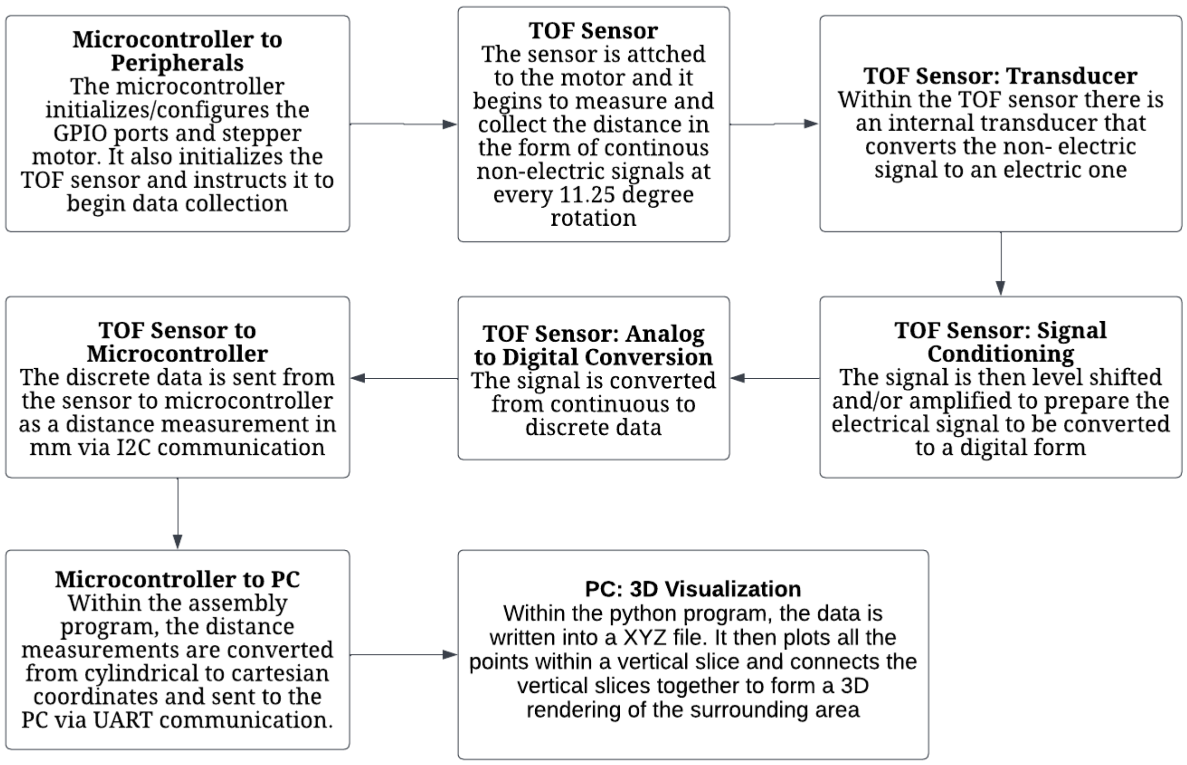

Microcontroller to Peripherals
The microcontroller initializes/configures the GPIO ports and stepper motor. It also initializes the TOF sensor and instructs it to begin data collection
TOF Sensor
The sensor is attched to the motor and it begins to measure and collect the distance in the form of continous non-electric signals at every 11.25 degree rotation
TOF Sensor: Transducer
Within the TOF sensor there is an internal transducer that converts the non- electric signal to an electric one
TOF Sensor: Signal Conditioning
The signal is then level shifted and/or amplified to prepare the electrical signal to be converted to a digital form
TOF Sensor: Analog to Digital Conversion
The signal is converted from continuous to discrete data
TOF Sensor to Microcontroller
The discrete data is sent from the sensor to microcontroller as a distance measurement in mm via I2C communication
Microcontroller to PC
Within the assembly program, the distance measurements are converted from cylindrical to cartesian coordinates and sent to the PC via UART communication.
PC: 3D Visualization
Within the python program, the data is written into a XYZ file. It then plots all the points within a vertical slice and connects the vertical slices together to form a 3D rendering of the surrounding area

Figure 1: Block Diagram of System

Images of System

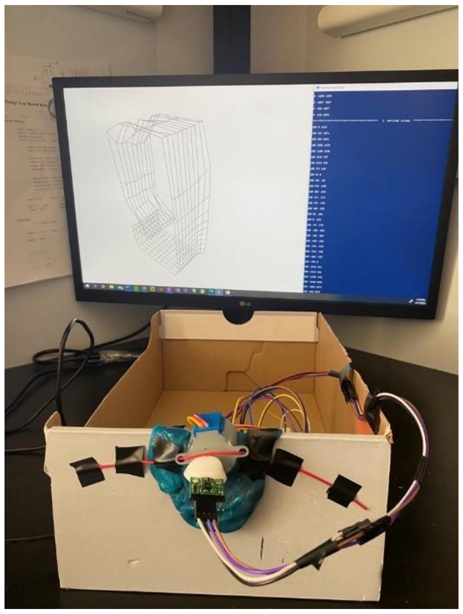


Figure 2: The system displaying data

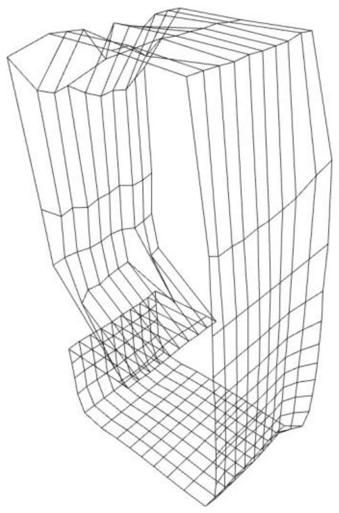


Figure 3: A screenshot of the PC Display

Device Characteristics Table

Device	Characteristic	
VL53L1X Pin	Device Pin	MCU Pin
	VDD	---------------
	VIN	3.3 V
	GND	GND
	SDA	PB3
	SCL	PB2
	XSHUT	----------------
	GPIO1	----------------
ULN2003 Driver Board Pin	Device Pin	MCU Pin
	+(5 V)	5 V
	-(5 V)	GND
	IN1	PH0
	IN2	PH1
	IN3	PH2
	IN4	PH3
Computer Software	Software	Version
	Python	3.7.3 (64 bit)
	Numpy	1.21.5
	Open3d	0.15.1
	Pyserial	3.5
Bus Speed	30 MHz	
Baud Rate	115200 bps	
Serial Port (UART)	COM5	
External Button Input	PM0	

Table 1: Device Characteristics Table

Detailed Description

1) Distance Measurement

The system uses the VL53L1X Time of Flight sensor to collect data on its surrounding environment. The sensor is attached vertically on top of the stepper motor. This allows the sensor to collect 32 distance measurements every 360 degree rotation of the motor. Thus, forming a geometric vertical slice in the yz plane of the surrounding area.

Initially when the microcontroller is powered and the code is flashed, the appropriate GPIO ports, clock, communication protocols and peripherals are configured and initialized. The TOF sensor is also booted. Once the python program is run, the user is prompted to enter the number of vertical slices they desire. The value that the user inputted is sent byte by byte (encoded) from the PC to the microcontroller via UART. It is then stored in the count variable and will be used to determine the number of times the overall data collection process is run.

The microcontroller continually checks for the external button to be pressed to instruct the TOF sensor to start collecting data. The microcontroller uses polling to check if there is an input through PM0 which is connected to an active low button. Each time the button is pressed, the microcontroller sends the character “A” to the PC through UART. This indicates to the PC that the data collection is ready to begin and to start writing the data into the XYZ file. The button is pressed each time a new vertical slice is being measured.

The sensor is able to collect distance measurements every 11.25 degree rotation of the stepper motor. The onboard LED of the microcontroller (D4) blinks every time the distance is being measured. After the motor is rotated 360 degree (32 data measurements) clockwise, the motor rotates a full cycle counterclockwise to prevent the wires from being tangled. No data is being collected during this period. Once this is completed, the microcontroller returns back to its polling state, waiting for an input from the external push button for the process to be repeated again.

The TOF sensor measures distance using LIDAR. It emits a 940 nm pulse of infrared light. The sensor measures how long it takes for the emitted pulses of light to reach the nearest object and then reflect back to the detector. It uses the following formula to measure the distance:

$$Distance = \frac{Photon\ Travel\ Time}{2} \times Speed\ of\ Light$$

Within the TOF sensor there is an internal transducer that takes in the photon travel time as a continuous non-electric signal and converts it into an electrical signal. The signal is then amplified and/or level shifted in preparation for the analog to digital conversion where the continuous analog signal is converted to discrete data. The distance is then calculated using the equation above by passing in the discrete time data. These stages all happen within the TOF sensor. The distance measurements are then transmitted to the microcontroller via I2C communication.

Within the C program, the data (distance) received from the TOF sensor is converted from cylindrical to cartesian coordinates through the following formulas:

$$y = Distance \times \sin(angle) \qquad z = Distance \times \cos(angle)$$

$$where\ angle = i \times 11.25\ degrees$$

The angle is determined by multiplying the number of 11.25 degree increments by 11.25 degrees. The angle is converted from degrees to radians by multiplying it with pi/180 before being used to calculate the y and z coordinates of the distance measurement.

To measure the displacement in the orthogonal (x-axis), after every 360 degree rotation of the motor, the x variable is incremented by 100mm. The 100 mm is an arbitrary value that should be changed according to the area being measured. For this specific project, painters tape was taped onto the hallway and 10cm increments were marked onto the tape. The system was then placed onto a stool to ensure that it remains at a constant height above the ground. After each vertical slice, the system would be moved 10cm in the x direction. This was repeated a total of ten times.

After each vertical slice, the appropriate x/y/z coordinates are concatenated into a string with a \n character at the end. This string is sent byte by byte from the microcontroller to the PC via UART at a baud rate of 115200 bps through COM5. Within the PC, the python program preforms the visualization from the received data. This will be discussed in more detail below.

2) Visualization

The data visualization is performed on a HP ENVY X360 convertible with a Intel(R) Core(TM) i5-8250U processor with 8 GB of RAM and 250 GB SSD. The PC is currently running on Windows 10 and the version of Python installed is 3.7.3 (64 Bit). Open3d, NumPy and Pyserial are additional libraries used in the python program. Open3D (version 0.15.1) is used to map the data to form a 3D rendering of the surrounding area, while NumPy (version 1.21.5) is used to format the data into arrays. Pyserial (version 3.5) allows for serial (UART) communication between the PC and microcontroller. PyCharm was the IDE used to edit the Python program.

Once the data is sent to the microcontroller byte by byte, the data is read until it encounters the “\n” character. The data is then decoded and written into the XYZ file. This process repeats itself until all the data points are written into the file. The port for UART communication is then closed. The data points are read and stored into a array that contains the xyz coordinates. A 3D point cloud is generated using Open3D. However, to produce the actual 3D shape, the python program gives each point (vertices) a unique number. It then connects the vertices (data points) of a single vertical slice. Each of the individual vertical slices are then joined together to form a 3D rendering of the surrounding area. An isometric view of the hallway where the visualization took place can be seen below.

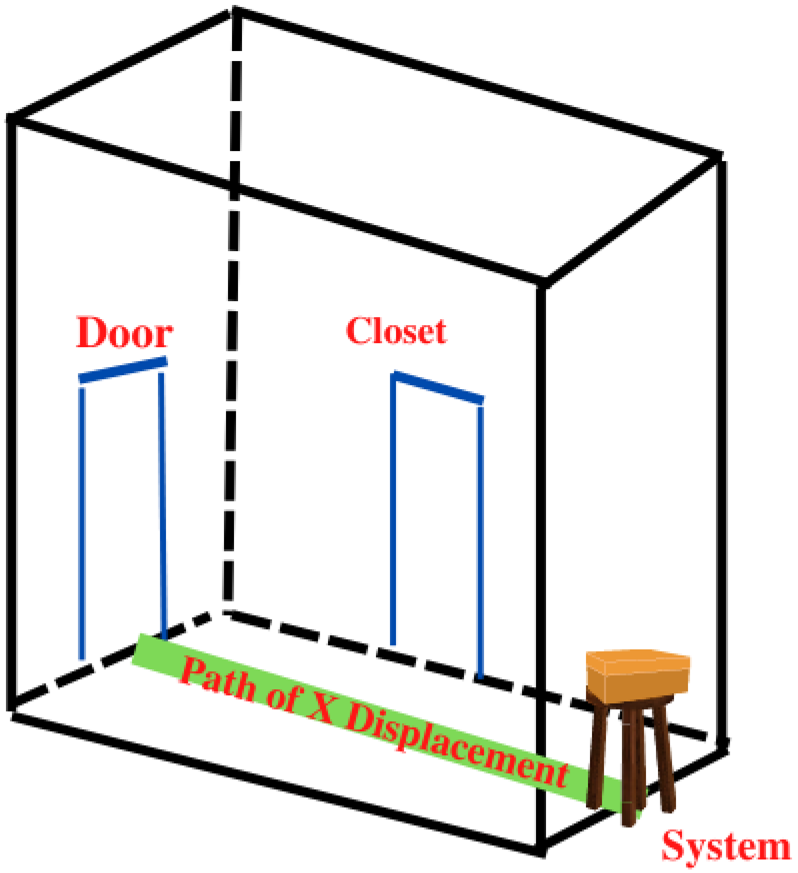


Figure 4: Isometric View of hallway where the 3D visualization was preformed

Application Example

This system can be used as a cheap and more compact alternative to LIDAR. It is important to note that ideally this system is to be used indoors. The following steps are executed once the system is in action:

1. The user will be prompted to enter the number of vertical slices they desire. Once entered, this number will be sent to the microcontroller via UART. The user UART port is COM5 and the baud rate is 115200 bps for this specific system.

2. Press the button on the breadboard to start the data collection process. This will send the character “A” to the microcontroller via UART, indicating to the MCU that the TOF sensor will start the distance measurement.

3. After every 11.25 degrees CW (16 steps) rotation of the stepper motor, the TOF sensor collects a distance measurement. The data will be sent from the TOF sensor to the microcontroller via I2C. It will then be converted to xyz coordinates and be sent to the PC via UART where it will be written into the XYZ file.

4.The appropriate onboard led (PF0) will also blink each time the data is being collected

5. After the motor has rotated 360 degrees clockwise, 32 data measurements should be acquired. The data measurements should be outputted to the screen from the python program.

6. The motor then proceeds to rotate counter clockwise 360 degrees to untangle the wires. During this rotation, no data collection is being done

7. Steps 2-6 are repeated until the desired number of vertical slices are completed.

8. Within the Python program, Open3d is used to generate a point cloud of all the coordinates.

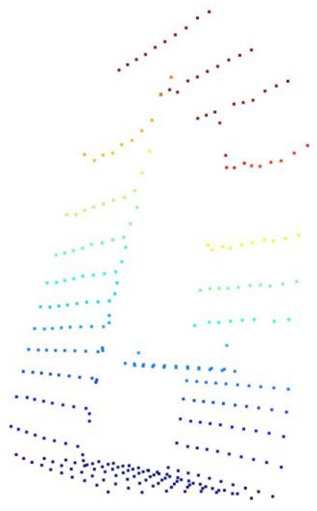


Figure 5: Point cloud image of hallway reconstruction

9. Lines are used to connect all the points to produce a 3D rendering of the surroundings

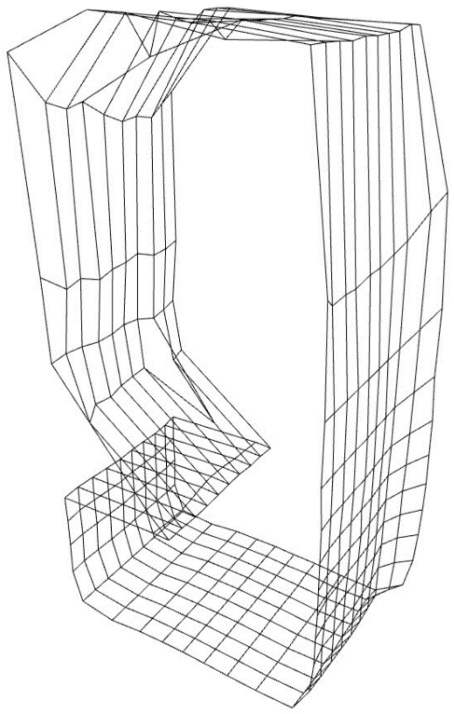


Figure 6: 3D visualization of hallway

User Guide

In the system, the yz plane is for the vertical slices while the x axis is the orthogonal/forward displacement. An example of the coordinate system can be seen below.

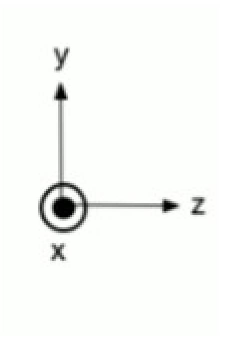


Figure 7: Defined Coordinate System For Distance Measurement

It is important to first ensure that the appropriate software is installed in order to run the visualization program.

1. Install a Python Version between 3.6-3.9 (64-bits): https://www.python.org/downloads/ .

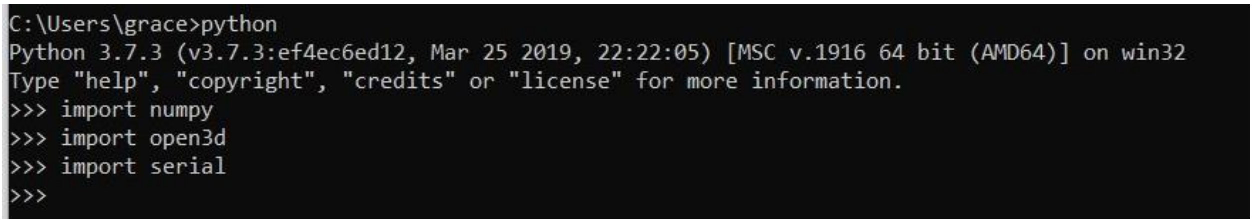
2. Run python in the command prompt. If it was installed correctly, Python will show the current version installed. Type “exit()” to leave.

3. In the command prompt, install Pyserial (the python serial communication API) by typing in: **pip install pyserial.**

4. Remain in the command prompt. Type in **pip install numpy** to install Numpy.

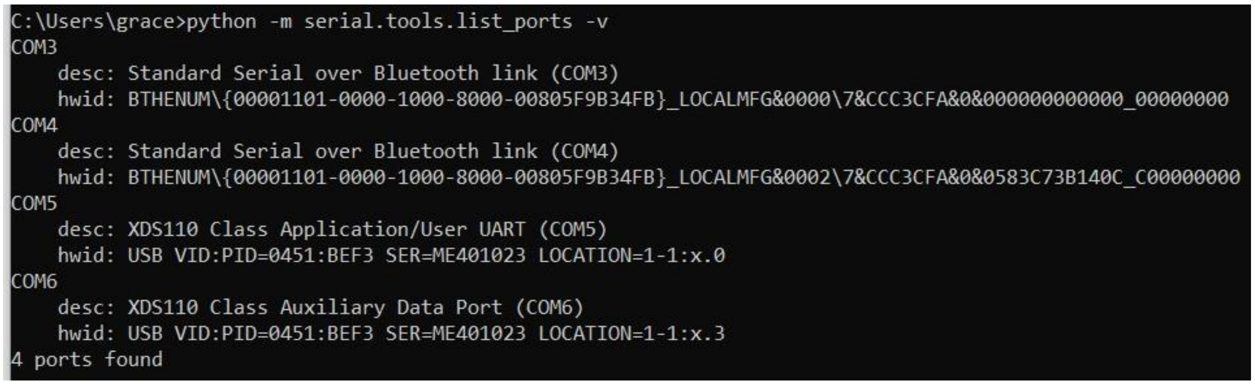
5. To install Open3d (the visualization software), type in **pip install open 3d open3d-python** in the command prompt.

6. To verify that these python libraries were correctly installed, in the command prompt run python. Then type in import numpy, import serial, import open3d. If no issues arise, the modules were installed correctly. A screenshot of this can be seen below:



```
C:\Users\grace>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> import open3d
>>> import serial
>>>
```

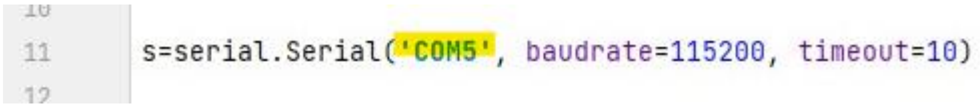
7. Power the system (microcontroller) through the USB port. In the command prompt, type in the following line: **python -m serial.tools.list_ports -v.** This will provide a detailed list of the available ports, as seen below:



```
C:\Users\grace>python -m serial.tools.list_ports -v
COM3
    desc: Standard Serial over Bluetooth link (COM3)
    hwid: BTHENUM\{00001101-0000-1000-8000-00805F9B34FB}_LOCALMFG&0000\7&CCC3CFA&0&000000000000_00000000
COM4
    desc: Standard Serial over Bluetooth link (COM4)
    hwid: BTHENUM\{00001101-0000-1000-8000-00805F9B34FB}_LOCALMFG&0002\7&CCC3CFA&0&0583C73B140C_C00000000
COM5
    desc: XDS110 Class Application/User UART (COM5)
    hwid: USB VID:PID=0451:BEF3 SER=ME401023 LOCATION=1-1:x.0
COM6
    desc: XDS110 Class Auxiliary Data Port (COM6)
    hwid: USB VID:PID=0451:BEF3 SER=ME401023 LOCATION=1-1:x.3
4 ports found
```

Look for the port named XDS110 Class Application/User UART. In the screenshot above, this would be COM5.

8. Open the data_collection.py file. Change line 11 to the appropriate user UART port of the PC. In line 11, the baud rate is also set as 115200 bps, but can be changed if needed.



```
10
11      s=serial.Serial('COM5', baudrate=115200, timeout=10)
12
```

9. Ensure that the microcontroller is powered. Open the Keil project and flash the code onto the microcontroller.

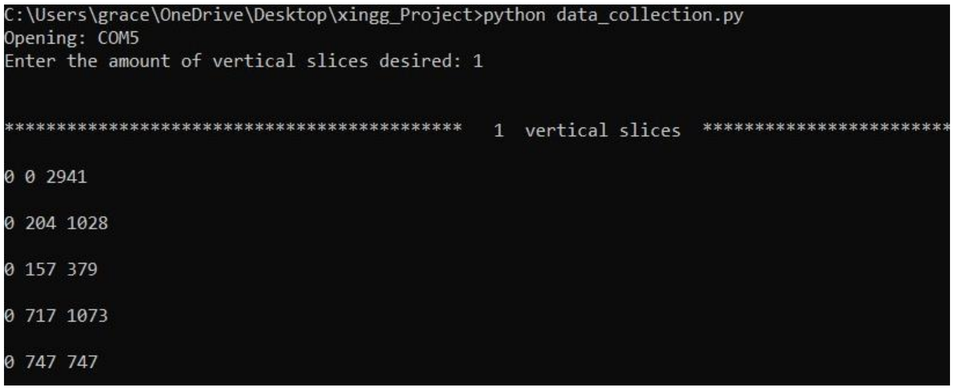
10. Press the restart button on the top side of the microcontroller. It is along the same side as the power cord.

11. Open the command prompt, and cd into the directory in which the data_collection.py file is stored. Then type in the following line: **python data_collection.py** to start the data collection.

12. Ensure that the following message is displayed in the terminal: “Opening: COM5”. This means that the serial data port is open and ready to start collecting distance measurements.

13. In the command prompt, type in the number of vertical slices desired and then press enter.

14. Press the external button on the breadboard to start the data collection. Once the button is pressed, the motor should start rotating clockwise and the xyz coordinates should be outputted to the screen as seen below:



```
C:\Users\grace\OneDrive\Desktop\xingg_Project>python data_collection.py
Opening: COM5
Enter the amount of vertical slices desired: 1


********************************************   1  vertical slices  ************************

0 0 2941

0 204 1028

0 157 379

0 717 1073

0 747 747
```

15.After the motor finishes rotating 360 degrees, move the system 10 cm forward (in the x direction).

16. Press the external button to start the data collection for a new vertical slice

17. Repeat steps 14-16 until the desired number of vertical slices is achieved

18. Once the data collection is done, a new window titled “Open3D” should pop up. This contains the 3d point clouds. Once that window is exited, the 3D rendering of the surrounding area will be displayed on a new window.

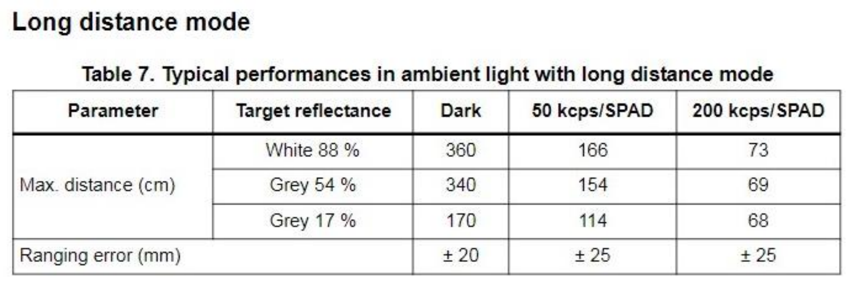
Limitations

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

The core processor of the MSP432E401Y is the 120-MHz Arm® Cortex®-M4F Processor Core with Floating-Point Unit (FPU). The FPU supports single precision addition, subtraction, multiplication and division. It provides 32-bit instructions for single-precision (C float) data-processing operations. The trigonometric functions are calculated within the Keil code using the C math library(<math.h>). This library takes in a double (64 bit) as an argument and will return a double as the result. However, these operations may also be applied to float variables. In order for this to occur, data conversion must happen which will lose the accuracy/precision of the final result.

2) Calculate the maximum quantization error

The maximum quantization error is between ±20 to ±25 mm. This information can be found in the datasheet of the VL531X Time of Flight sensor.



Long distance mode

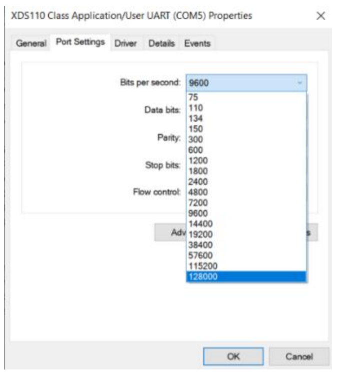
Table 7. Typical performances in ambient light with long distance mode

Parameter	Target reflectance	Dark	50 kcps/SPAD	200 kcps/SPAD
Max. distance (cm)	White 88 %	360	166	73
	Grey 54 %	340	154	69
	Grey 17 %	170	114	68
Ranging error (mm)		± 20	± 25	± 25

Figure 8: Quantization Error

3) What is the maximum standard serial communication rate you can implement with the PC. How did you verify?

The maximum standard serial communication rate that can be implemented with the PC is 128000 bits per second. The was verified by going to the device manager and looking up the properties of the user UART port.



XDS110 Class Application/User UART (COM5) Properties

Bits per second: 9600

Figure 9: Screenshot of the properties of the User UART port

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

I2C communication was used between the microcontroller and the TOF modules. Within the assembly program, the speed is configured to 100KHz and thus the data transfer rate is 100 kbps. The TOF sensor has a maximum I2C speed of 400 KHz.

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

The primary limitation on the speed is the ranging on the TOF sensor. The ranging duration (timing budget) can be programmable. Increasing the timing budget will increase the maximum distance the device can range. However, it will decrease the overall speed of the system. The sensor requires the greatest delay (140 ms) to reach the maximum distance of 4m in Long Distance Mode. This is the current mode the sensor is set at for the system. As the number of data measurements increase, the total sensor delay will build up, ultimately slowing down the entire system. In order to test this, I attempted to make the delay smaller. After the change in delay, the sensor was no longer producing any data. Thus, this delay cannot be reduced or else the sensor can no longer measure data from a long range. In addition, according to the data sheet of the sensor, it has a maximum ranging frequency of 50Hz. Compared to the other peripherals in the system, this is a relatively small frequency and dramatically limits the speed of the system as well.

Circuit Schematic

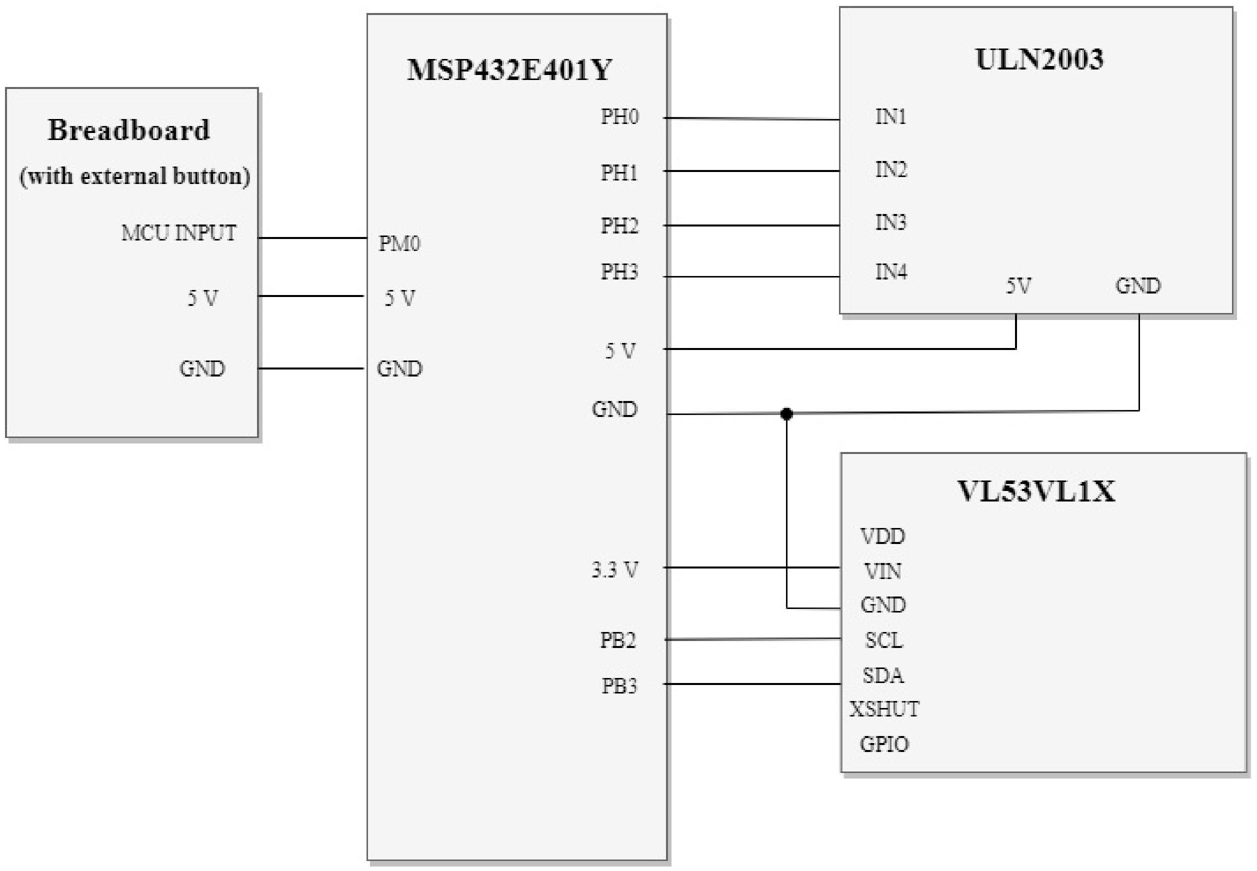


Figure 10: Circuit Schematic

Flow Chart

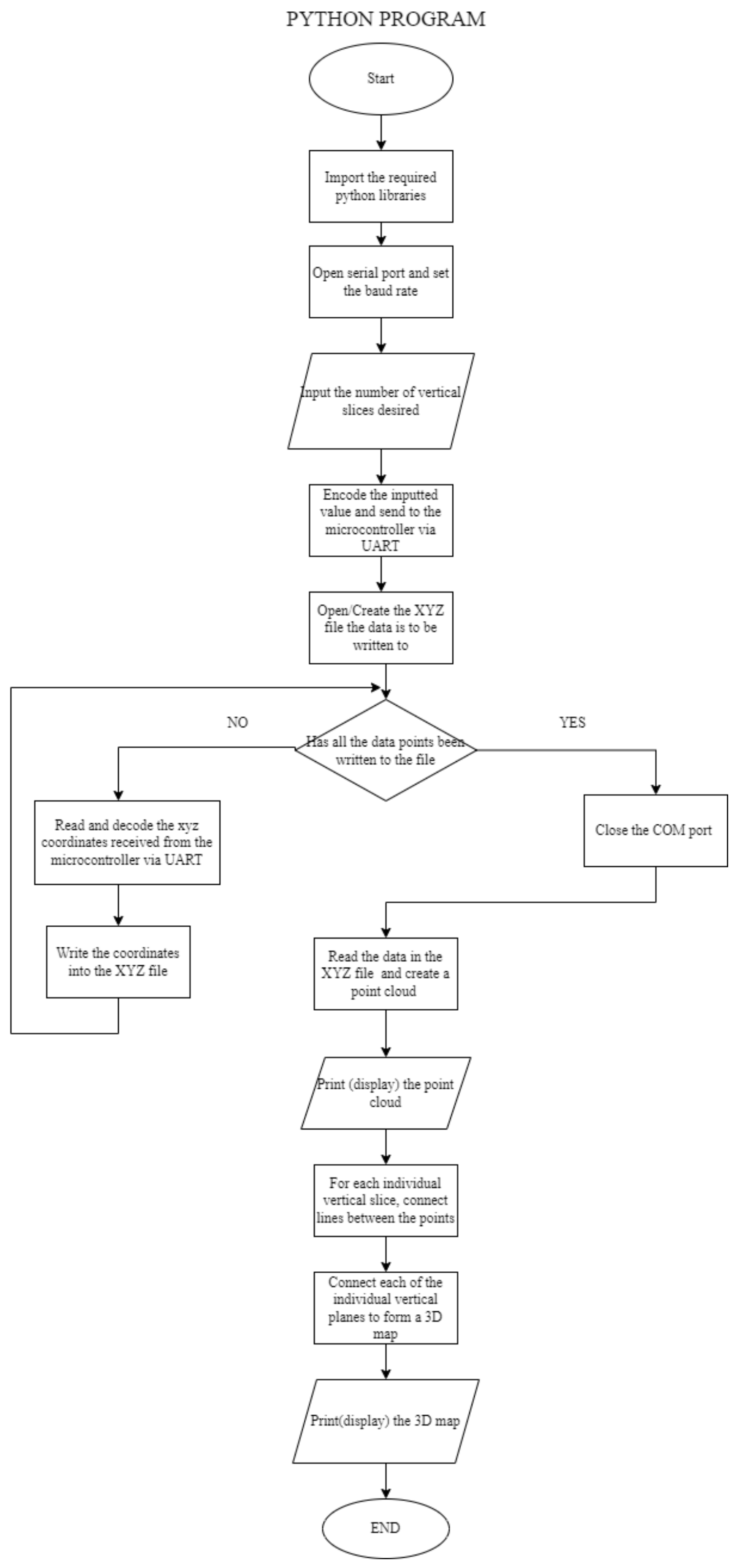

PYTHON PROGRAM
Start
Import the required python libraries
Open serial port and set the baud rate
Input the number of vertical slices desired
Encode the inputted value and send to the microcontroller via UART
Open/Create the XYZ file the data is to be written to
Has all the data points been written to the file
NO
YES
Read and decode the xyz coordinates received from the microcontroller via UART
Write the coordinates into the XYZ file
Close the COM port
Read the data in the XYZ file and create a point cloud
Print (display) the point cloud
For each individual vertical slice, connect lines between the points
Connect each of the individual vertical planes to form a 3D map
Print(display) the 3D map
END

Figure 11: Flow chart of python program

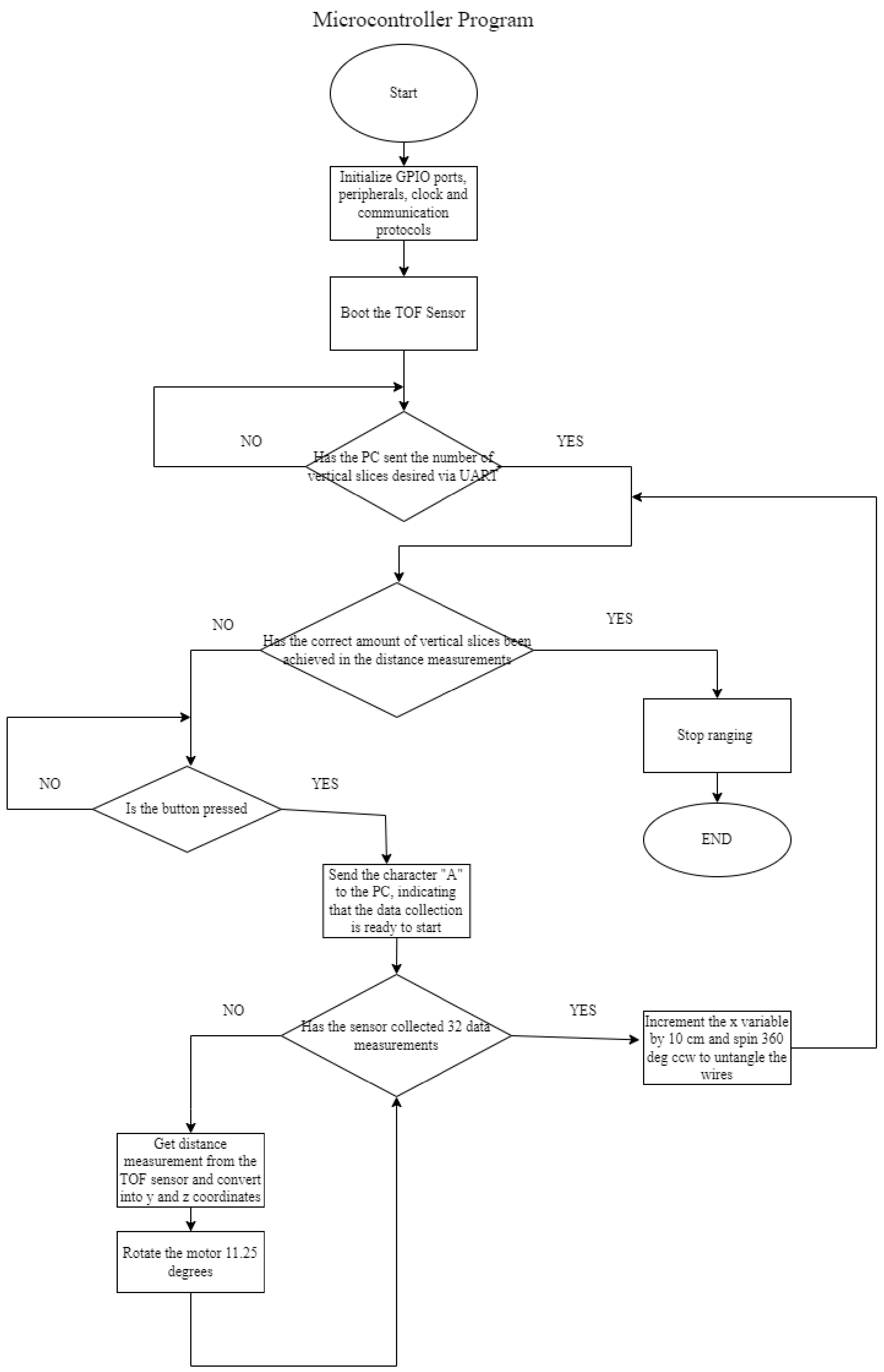

Microcontroller Program
Start
Initialize GPIO ports, peripherals, clock and communication protocols
Boot the TOF Sensor
NO
YES
Has the PC sent the number of vertical slices desired via UART
NO
YES
Has the correct amount of vertical slices been achieved in the distance measurements
Stop ranging
END
NO
YES
Is the button pressed
Send the character "A" to the PC, indicating that the data collection is ready to start
NO
YES
Has the sensor collected 32 data measurements
Increment the x variable by 10 cm and spin 360 deg ccw to untangle the wires
Get distance measurement from the TOF sensor and convert into y and z coordinates
Rotate the motor 11.25 degrees

Figure 12: Flow chart of Microcontroller Program