2DX4: Microprocessor Systems Project

Final Report

Instructor: Drs. Bruce, Haddara, Hranilovic, and Shirani

Haroon Janjua – L05 – Janjuh1 – 400196693

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 [Haroon Janjua, janjuh1, 400196693]

Table of Contents

Interview Video Questions

Question 1

1. How did you communicate the raw sensor data from your microcontroller to your computer?

https://drive.google.com/file/d/172st_CdA08bc-Cdk9B8gG_OG76hdsvyj/view?usp=sharing

Question 2

2. What data processing did you perform on the raw data?

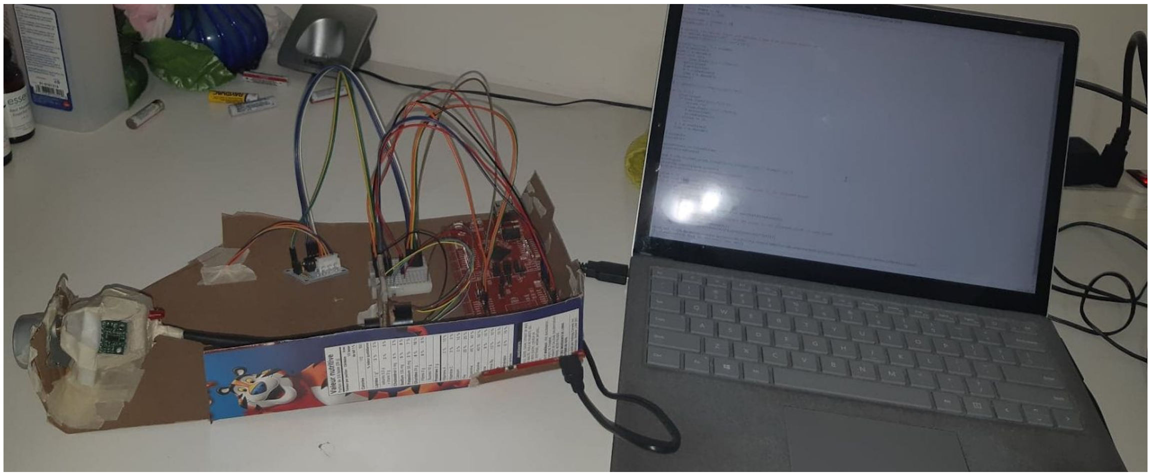
https://drive.google.com/file/d/1lg5w01cZ4jb_oB2W0AqJP9E2MyXNHMOY/view?usp=sharing

Question 3

3. How did you visualize the processed data using 3D rendering?

https://drive.google.com/file/d/1zh9jZkMT37q5Pi4oOi5XJRCxYdL5G8_t/view?usp=sharing

Device Overview



Features

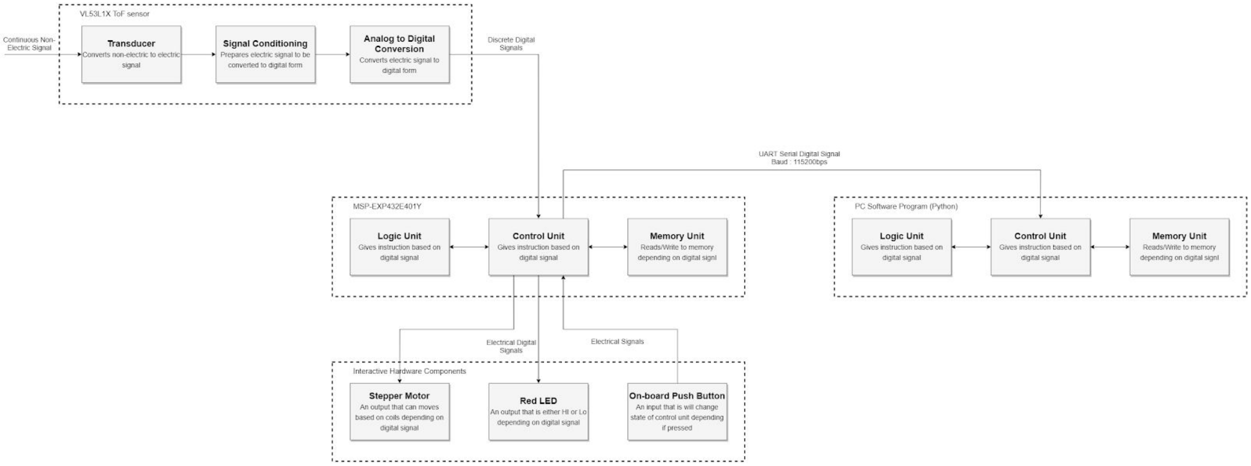
- Gets a 3D 360 scan of the environment and opens up a 3D render which can be interacted with
- Provides a 360-degree scan of the y-z plane for every 20cm
- Does a detailed degree sweep on the y-z plane for every 45 degrees
- Generates 8 points for every 360-degree sweep
- Stores all points in order into “points2DX4proj.xyz”
- Stores initialization results separately in another file called “ToFInitDetails.txt”
- Is used to track the device’s surrounding environment and it 3D renders it for the user
- The device is used to map out the places it has been
- Includes a distance status external LED which tells the user when the next slice is ready to be taken
- Takes 5 slices totaling 100cm of distance traveled
- Includes a displacement status on-board LED1 which tells the user when ToF sensor is operating
- Includes onboard LED3&4 flashing for I²C successful reading and writing.
- Flashes all onboard LED once to confirm initializing is successful and keeps flashing to show there is an error.
- Interacts with the PC using python applications provided
- storePoints python application will take 5 slices of ToF measurements, store info and 3d render it
- render3D python application can render the “points2DX4proj.xyz” file in a 3D space without scanning for new points, so now you can save files and render them later
- 120MHz Arm Cortex M4F Processor (120MHz clock speed)
- Set Bus Speed at 48MHz
- 4.75V to 5.25V Operating Voltage
- Cost around $120 CAD for all components including IMU
- 1024KB of Flash Memory, 256KB of SRAM, and 6KB EEPROM
- Two 12-bit SAR Based ADC Modules
- Each ADC has a sampling rate of 2 Mega samples/second
- Uses C++ Language (uVision IDE)
- Uses 115200 bits/sec Baud Rate for UART Serial Communication

General Description

Components	Basic Description
Texes Instrumental MSP432E401Y Microcontroller	Does most of the processing of raw data and is used to connect all other components together. This thing will communicate with every component and process all types of data to allow for a smooth digital network. Also the onboard push-button starts up the process of getting ToF data measurements to send to the PC. There are also on-board LEDs which will flash for various reasons, one of them being LED 1 (PN1) wil flash every 45 degrees the motor spins.
VL53L1X Time of Flight Sensor	Gets distance measurements which are in mm (upto 4m) and is an analog signal. Then it performs signal conditioning and analog-to-digital conversions before sending a distance to the microcontroller
MPU 9250 IMU	This should have been calculating the distance traveled by the car, and it would have used I^2C just as the ToF sensor does. Since the IMU will give you acceleration you would have to record its values for the time the car was moving and integrate everything twice two get distance.
ULN2003A Stepper Motor	Will rotate in counter-clockwise direction when the on-board push button of the microcontroller is pressed.
Adafruit Slipring	Allows me to connect the ToF sensor to the motor since this allows the wiring of the ToF sensor to move 180 degrees.
External Red LED	This is the displacement status. It connects to PL5 and represents if the next slice is ready/motor is done moving 360 degrees.
Computer	The computer will use python to store information in the xyz file that it receives through UART serial communication with a baud rate of 115200bps.

The device contains a MSP432E401Y Board (MSP), a VL53L1X Time of Flight Sensor (ToF sensor), MPU 9250 IMU (IMU), 5V DC Stepper Motor (stepper motor), Adafruit Slip-Ring (SlipRing), and a red L.E.D (LED). The ToF sensor is on a breadboard which is connected to the stepper motor. This allows the ToF sensor to spin 360 degrees in the y-z plane. The external red LED (displacement status) is also located on the same breadboard as the ToF sensor and when lit, means a slice in the y-z plane is ready to be sampled and press any side onboard button to proceed. When either of the two onboard buttons is pressed, the onboard LED 1 (PN1, distance status) will indicate a rotational movement of the motor and when ToF measurement is being taken. The stepper motor rotates the ToF sensor and gets 36 XYZ coordinate points during the 360-degree sweep. Then it sends the coordinates via UART serial communication to the computer where the python program called storePoints is used to read and save the coordinate points in an XYZ file. This process will be repeated 5 times while incrementing x by 20cm until where python will have enough data points to make a 3D render and it will stop reading the microcontroller's data.

Block Diagram


VL53L1X ToF sensor
Continuous Non-Electric Signal
Transducer
Converts non-electric to electric signal
Signal Conditioning
Prepares electric signal to be converted to digital form
Analog to Digital Conversion
Converts electric signal to digital form
Discrete Digital Signals
UART Serial Digital Signal
Baud : 115200bps
MSP-EXP432E401Y
Logic Unit
Gives instruction based on digital signal
Control Unit
Gives instruction based on digital signal
Memory Unit
Reads/Write to memory depending on digital signl
PC Software Program (Python)
Logic Unit
Gives instruction based on digital signal
Control Unit
Gives instruction based on digital signal
Memory Unit
Reads/Write to memory depending on digital signl
Electrical Digital Signals
Electrical Signals
Interactive Hardware Components
Stepper Motor
An output that can moves based on coils depending on digital signal
Red LED
An output that is either HI or Lo depending on digital signal
On-board Push Button
An input that is will change state of control unit depending if pressed

Device Characteristics

To get the device running on your own first you need to have both the hardware and software part working.

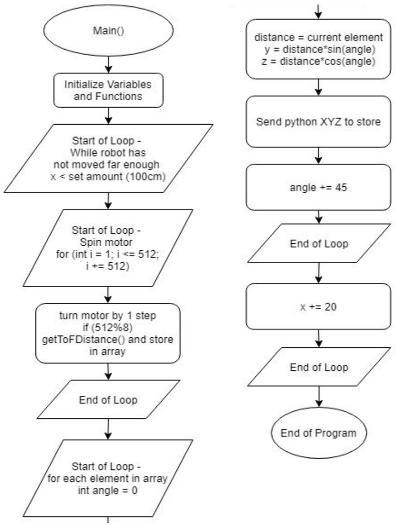
For the hardware part, you will need to connect the microcontroller with the same pin configuration as listed in the chart below. Also, you must connect the ToF sensor to the stepper motor in a way where none of the other components come in the way of the sensor. Lastly do not forget to wait for the external red LED to take ToF measurements.

For the software portion, you first need to check if the hardware is connected to the software.
To connect the microcontroller to the software you must connect a USB cable that can send and receive data with the microcontroller and computer. After that ensure the right port is set within python. This can be checked by cmd where you type in “python -m serial.tools.list_ports -v”, before USB is connected then after to see which new ports were added. From there use real terms to see which port from the new ports gives you initialization information of hardware when the reset push button is pressed. To set to the known port in python just change where it says “COM5” to “COMX” where X is your port number. After setting your port you must install and import the open3D python library to your python version (3.6 to 3.8) properly.
To install open 3D do the following:
Open CMD with administrative rights
Type “pip install open3d” and wait for it to install
Then type “python -c "import open3d" and see if there is no error you are done
However, if there is an error and you are on version 3.6 install Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019 x86 and x64 from this website:
https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads

Pins setup of MSP432E401 Board	
PL4	External Red LED with a 330-ohm resistor (displacement status)
PN1	Internal LED1 (distance status)
PL0 - PL3	IN1 - IN4 (Stepper Motor)
5V	VIN (Stepper Motor)
GND	GND (Stepper Motor)
5V	VIN (ToF Sensor)
GND	GND (ToF Sensor)
PB3	SDA (ToF Sensor)
PB2	SCL (ToF Sensor)
Bus Speed	48MHz
Serial Port	COM5
Baud rate	115200bps

Detailed Description

Distance Measurement


Main()
Initialize Variables and Functions
Start of Loop - While robot has not moved far enough x < set amount (100cm)
Start of Loop - Spin motor for (int i = 1; i <= 512; i += 512)
turn motor by 1 step if (512%8) getToFDistance() and store in array
End of Loop
Start of Loop - for each element in array int angle = 0
distance = current element y = distance*sin(angle) z = distance*cos(angle)
Send python XYZ to store
angle += 45
End of Loop
x += 20
End of Loop
End of Program

First I set up the ToF sensor, to do this I call a method named ToF_Init() and what this does is keeps on trying to boot the ToF sensor until it succeeds (when sensor state is HI). This process uses I^{2}C communication to access certain parts of the memory of the ToF which will store ToF sensors status. After that, It will flash all onboard LEDs to represent the ToF sensor done booting. Then I will use I^{2}C communication again to send data to the computer using UART serial communication about whether initializing sensors is successful and if we can start ranging. Then I will turn off the distance LED and spin the motor every 45 deg and store the distance result in an array. After 8 spins I will send the data to the computer as X Y Z. To send X Y Z I need to run a for-loop which will run for each element of the array while incrementing angle by 45 degrees each time. Within this for-loop, Y = distance*cos(angle) and Z = distance*sin(angle) where distance will be the current distance of array and angle will be changing corresponding to for-loop. Then I will send "X Y Z\r\n" where x is a variable that is set to zero and only increments when side push buttons are pressed (when the next slice begins). After that, it is sent to the computer where python will take care of reading and storing it.

Displacement

Current implementation:

For this, I had a local variable called x in Keil. I would increment it after every slice is done by 200 (20cm). From there I would send data with Y and Z when sending distance measurements to the computer. After that I will have to manually push the device 20cm in the x-direction for the next slice.

Past implementation:

What we were supposed to do for this was to use the IMU to find out how far the device moved and increment our x variable proportionately. From there we would stop our program after it reached a certain distance. The IMU would work in combination with another robot that would move this device a certain distance and stop for measurements to happen. New code would be needed too, for example, IMU_Init() and getIMUMeas(). To use IMU like this, is possible since the IMU would give you acceleration which you could use to find out the distance travelled if you keep track of time and integrate it twice. Also the process to get IMU measurement would use the similar processes (I^2C) when getting ToF measurements.

Visualization

This step happens in the software section using python. So after python has stored all the values of XYZ into an XYZ file I will run the visualization par.t To do this I can use either of the 2 python applications, however one will get new data every time while the other one will use saved data. The first thing I do is use the open3D library to convert the XYZ file to a point cloud and store it in a variable. Then I would make a list called lines which will store all the lines I want to connect as a list of two points. I made lines for each point in a slice that will connect to the next point within that slice, then I did that for every slice. Then I made lines for each point connecting to its adjacent point on the next slice for one whole slice, then I did that for every slice except the last one. After this, I used another open3D method to make a lines_set variable which would make actual lines replacing the points of lines to their corresponding points. (What this means is lines might have a point like this [0,1] which means connect a line from point 0 from the point cloud to point 1 from the point cloud. This will be turned into [actual value of point 0, actual value of point 1]). Now with all the lines configured, I render it using open3D lib again to visualize all the lines. This gives me a 3D render of my points with a coherent shape.

PC Info:
Microsoft Surface Laptop Windows 10
Intel Core i7-7660U 2.5GHz CPU
8GB DDR4 Ram

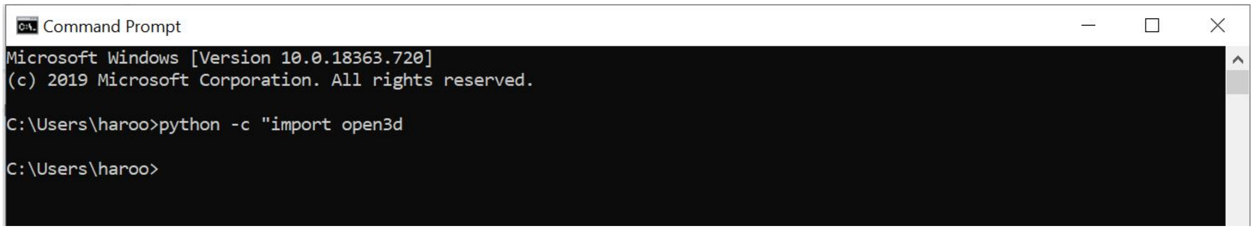
Program Used:
Python 3.6.8

Libraries Used:

- pySerial
- Open3d
- NumPy

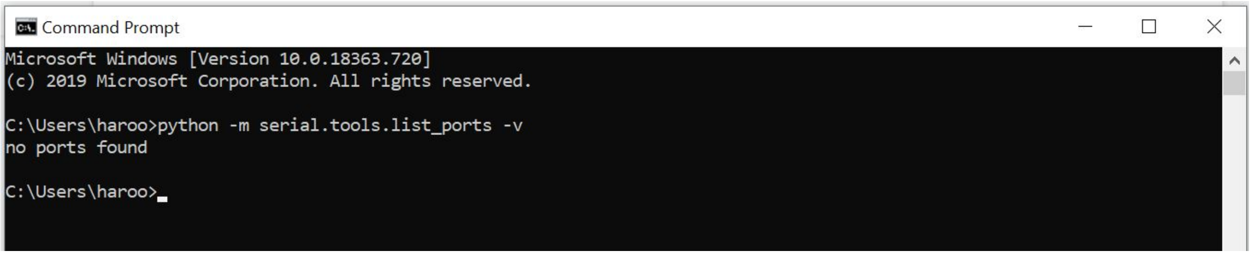
Application Example

1. First, make the circuit following the circuit schematic. This can be found in Table of Contents. Also, connect the ToF sensor to the motor.
2. Upload the Keil program to the microcontroller.
3. Download python 3.6.8 or latest version (more steps with 3.6.8) with environmental path selected in advance settings
4. Install open3D Python library by opening Command Prompt and run as administrator, type “pip install open3d” and wait for it to install. (This will install NumPy and PySerial as well)
5. To confirm if open3D was installed properly type “python -c "import open3d" into the command prompt and see if you get an error. (If everything goes right then no error will happen and it should look something like this)



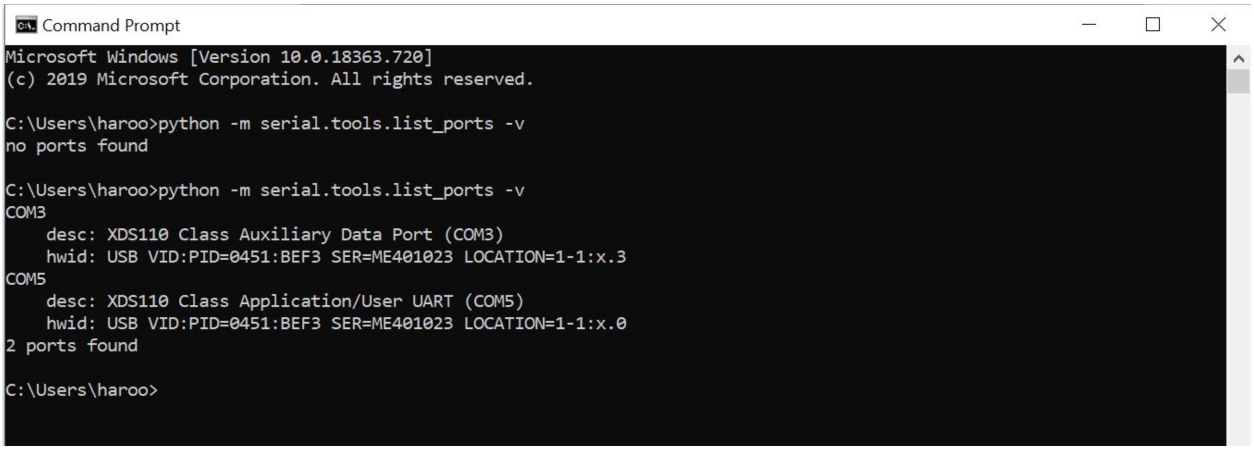
```
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\haroo>python -c "import open3d

C:\Users\haroo>
```

6. If you did get an error trying on python 3.6.8 then install Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019 x86 and x64 from this website: https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads
7. Now restart your compute and type “python -c "import open3d" into the command prompt again. This time no error should happen.
8. Now you need to find the port to communicate on. To do this unplug USB connecting the microcontroller and the PC, and type “python -m serial.tools.list_ports -v” to Command Prompt.



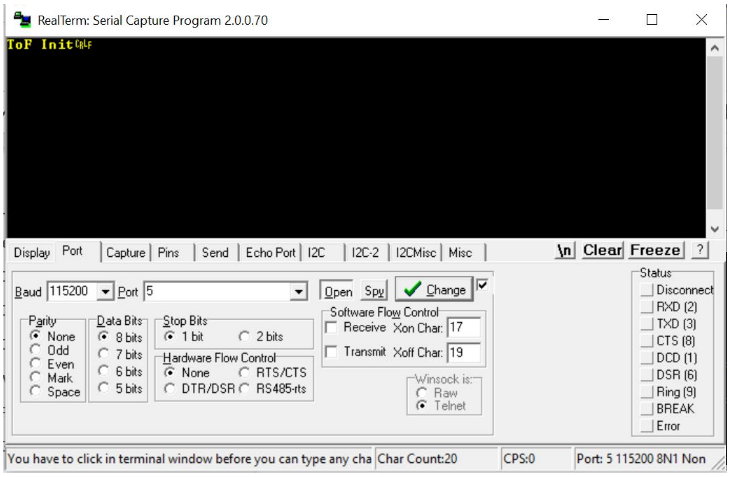
```
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\haroo>python -m serial.tools.list_ports -v
no ports found

C:\Users\haroo>_
```

9. Now do the same thing again but with the USB plugged in. Check which ports have been added.



```
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\haroo>python -m serial.tools.list_ports -v
no ports found

C:\Users\haroo>python -m serial.tools.list_ports -v
COM3
    desc: XDS110 Class Auxiliary Data Port (COM3)
    hwid: USB VID:PID=0451:BEF3 SER=ME401023 LOCATION=1-1:x.3
COM5
    desc: XDS110 Class Application/User UART (COM5)
    hwid: USB VID:PID=0451:BEF3 SER=ME401023 LOCATION=1-1:x.0
2 ports found

C:\Users\haroo>
```

10. Now we will need to test the new ports to see which one the microcontroller sends data. To do this keep USB connected and open up Realterm. Now select one of the ports and set Baud to 115200.
11. Now restart the microcontroller with the onboard push button and you should immediately get the message “ToF Init”. If you don’t change the port and restart the microcontroller.



```
ToF Init␍␊
```

Baud 115200 Port 5

Parity: ● None ○ Odd ○ Even ○ Mark ○ Space

Data Bits: ● 8 bits ○ 7 bits ○ 6 bits ○ 5 bits

Stop Bits: ● 1 bit ○ 2 bits

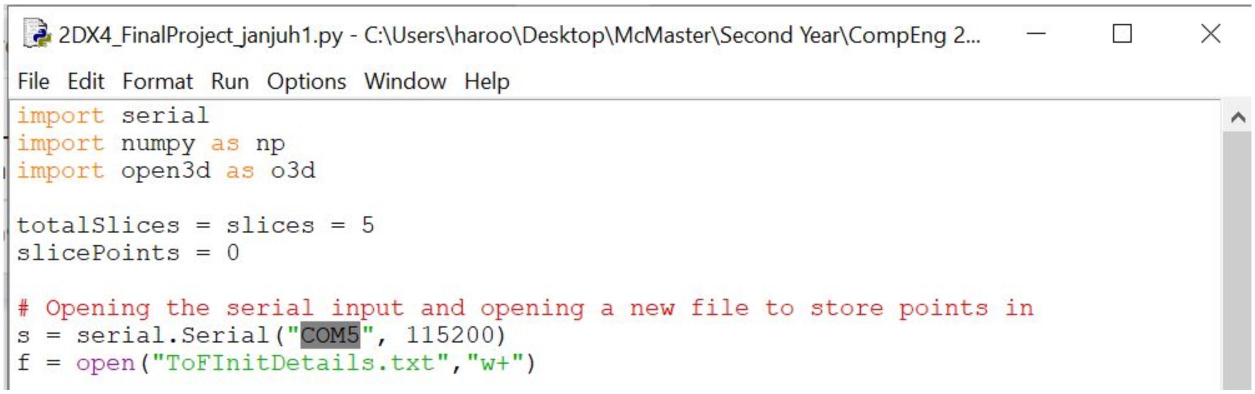
Hardware Flow Control: ● None ○ RTS/CTS ○ DTR/DSR ○ RS485-rts

Software Flow Control: □ Receive Xon Char: 17 □ Transmit Xoff Char: 19

Winsock is: ○ Raw ● Telnet

You have to click in terminal window before you can type any cha | Char Count:20 | CPS:0 | Port: 5 115200 8N1 Non

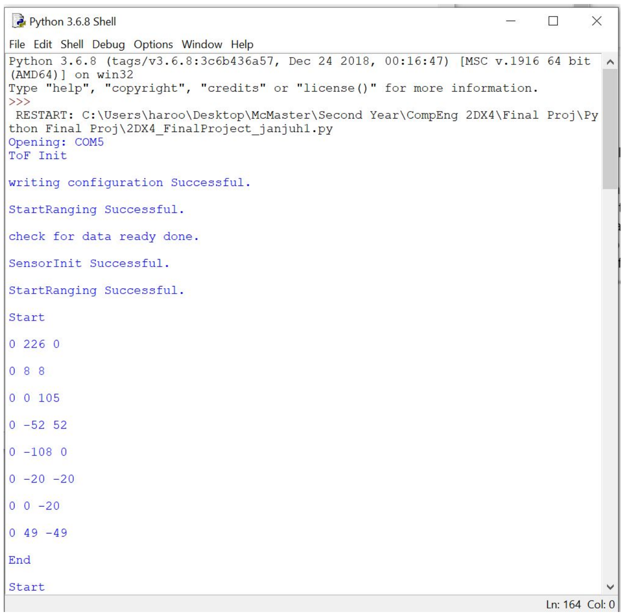
12. Now all you need to do is open up python code in IDE and change “COM5” to “COM#” where # is your port.


2DX4_FinalProject_janjuh1.py - C:\Users\haroo\Desktop\McMaster\Second Year\CompEng 2...
File
Edit
Format
Run
Options
Window
Help

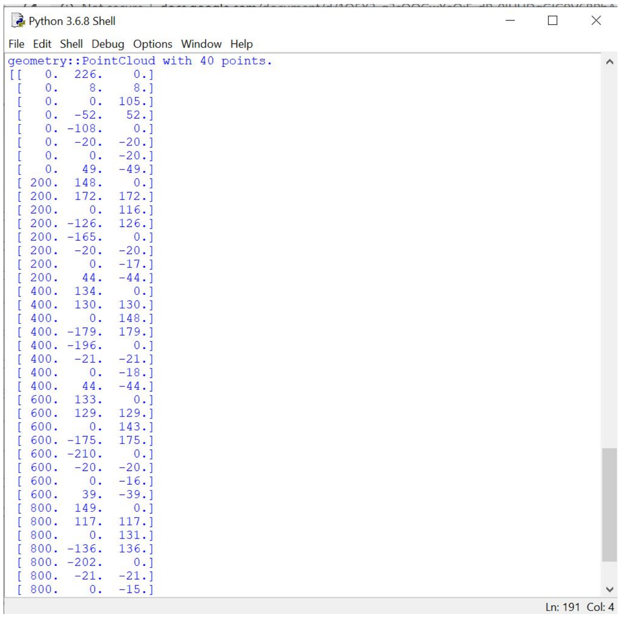
```
import serial
import numpy as np
import open3d as o3d

totalSlices = slices = 5
slicePoints = 0

# Opening the serial input and opening a new file to store points in
s = serial.Serial("COM5", 115200)
f = open("ToFInitDetails.txt","w+")
```

13. Now all that’s left is to run python code and reset the microcontroller. All the initializing messages will be stored in a separate file called “ToFInitDetails.txt”, so you are allowed to start the microcontroller first and let it set up and then turn on python. Just make sure python is opened and running before you start to get ToF measurements from that slice.
14. To start recording ToF measurements press any of the two sides onboard push buttons (perpendicular to the starting one). Wait for the external LED to light up again and repeat 5 times. After you are done the code will automatically save info in the file and load a 3D render. From there play around with it. If python crashes on you just open other python code called graphXYZFile which will 3D render the saved data set.

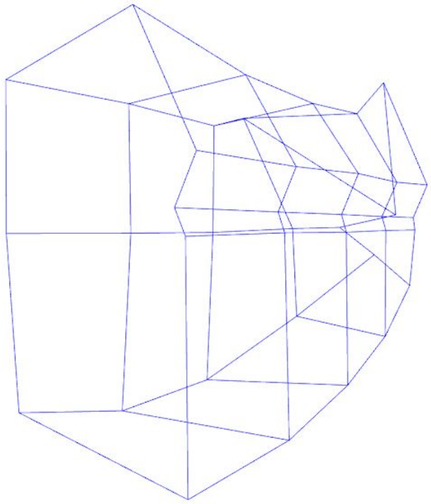
Sample Output



```
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\haroo\Desktop\McMaster\Second Year\CompEng 2DX4\Final Proj\Py
thon Final Proj\2DX4_FinalProject_janjuh1.py
Opening: COM5
ToF Init

writing configuration Successful.

StartRanging Successful.

check for data ready done.

SensorInit Successful.

StartRanging Successful.

Start

0 226 0

0 8 8

0 0 105

0 -52 52

0 -108 0

0 -20 -20

0 0 -20

0 49 -49

End

Start
```



```
geometry::PointCloud with 40 points.
[[   0.  226.    0.]
 [   0.    8.    8.]
 [   0.    0.  105.]
 [   0.  -52.   52.]
 [   0. -108.    0.]
 [   0.  -20.  -20.]
 [   0.    0.  -20.]
 [   0.   49.  -49.]
 [ 200.  148.    0.]
 [ 200.  172.  172.]
 [ 200.    0.  116.]
 [ 200. -126.  126.]
 [ 200. -165.    0.]
 [ 200.  -20.  -20.]
 [ 200.    0.  -17.]
 [ 200.   44.  -44.]
 [ 400.  134.    0.]
 [ 400.  130.  130.]
 [ 400.    0.  148.]
 [ 400. -179.  179.]
 [ 400. -196.    0.]
 [ 400.  -21.  -21.]
 [ 400.    0.  -18.]
 [ 400.   44.  -44.]
 [ 600.  133.    0.]
 [ 600.  129.  129.]
 [ 600.    0.  143.]
 [ 600. -175.  175.]
 [ 600. -210.    0.]
 [ 600.  -20.  -20.]
 [ 600.    0.  -16.]
 [ 600.   39.  -39.]
 [ 800.  149.    0.]
 [ 800.  117.  117.]
 [ 800.    0.  131.]
 [ 800. -136.  136.]
 [ 800. -202.    0.]
 [ 800.  -21.  -21.]
 [ 800.    0.  -15.]
```



As you can see it’s a little messed up near the end because my ToF sensor fell off the motor. Also, this is a 3D render of my upstairs hallway. I have to say it did a good job.

Limitations

1)

The main limitation of floating numbers in the microcontroller, in fact, any digital device is that they can't store decimal numbers accurately since they operate in the base of 2, whereas decimals operate in the base of 10. This causes some inaccuracy when converting between decimal numbers to floating numbers, and if we do calculations with these inaccuracies we can get farther and farther from the absolute answer. The inaccuracy only affects the final answer by a little bit but, if enough arithmetic calculations are repeated with this inaccuracy it may start to add up and may cause a programming error.

2)

ToF $\Delta = V_{IN}/2^m = V_{IN}/2^{12} = V_{IN}/4096$
V_{IN} can be from 2.6V-5.5V but for our case, we can set V_{IN} = 3.3V or 5V
ToF Δ = 3.3/4096 or 5/4096 = 0.00080566406 or 0.00122070312

IMU $\Delta = V_{IN}/2^m = V_{IN}/2^{16} = V_{IN}/65536$
V_{IN} can be from 1.9V-3.6V but for our case, we can only set it to 3.3V
IMU Δ = 3.3/65536 = 0.000050354

3)

The fastest we can communicate with the PC is 115200 bps. This is because the serial port we use for communication relies on the UART chip which uses a crystal oscillator. The crystal oscillator produces frequency up to 3579545Hz. From there the sample rate must at most be half the frequency, making the sample rate of 1789773 bps. However, to reliably sample you must divide by 16 to get a sub-sampling clock for the data. This means our actual highest sample rate is 111861 bps, however, this is not an ideal baud rate and so it gets estimated to 115200 bps, giving it around a 2% error from an ideal baud rate. You can verify this by opening up realterms and selecting the highest available baud rate. The highest baud rate is 115200 bps which matches the calculations.

4)

The communication method used between ToF modules and microcontrollers is I^2C which has a speed of 400kHz.

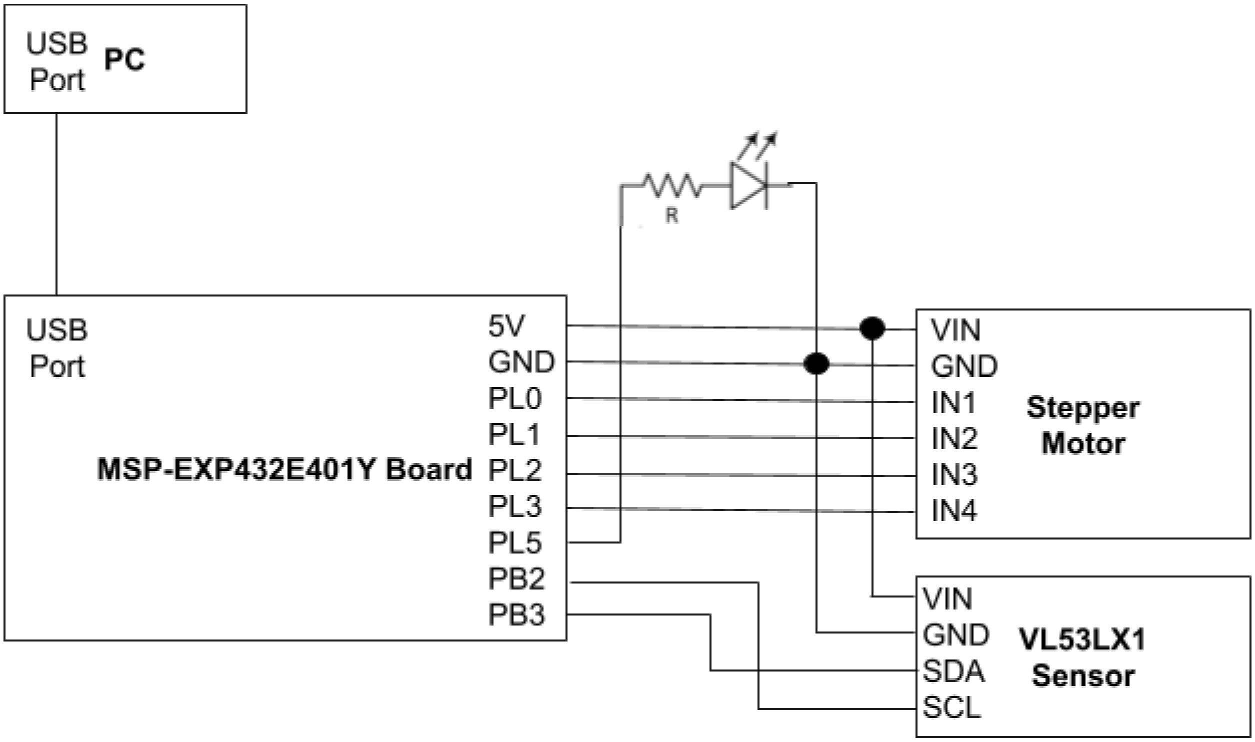
5)

When looking at the entire system the thing that takes the longest is to initialize the ToF sensor and spinning motor, however, initializing only happens once so it is negligible so therefore the primary limitation on speed comes from the motor. This is because the motor needs time to spin itself and cannot be speeden up. After a certain speed (6ms for me) the motor won't spin anymore. The fastest I can spin is 10ms. This means that the frequency of the motor would be 100Hz per spins which is by far the slowest component of the whole system. The reason why it is the slowest is because each coil within the motor must be magnetized for a certain time to allow the motor to spin, and doing this upto to 2048 times adds up. To test this I tried to make the motor go faster but the limit of the motor was 7ms for me, going any lower did nothing.

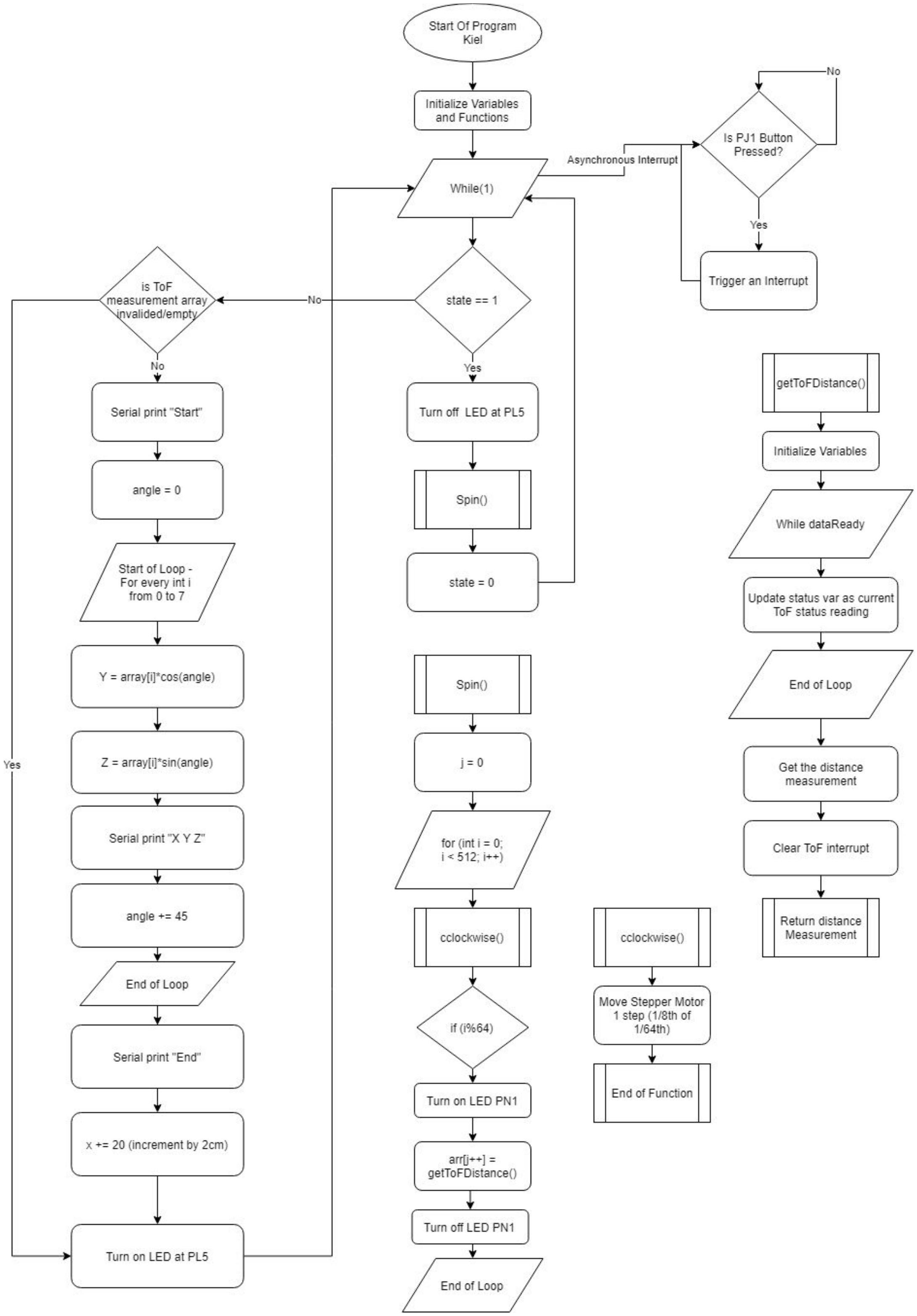
6)

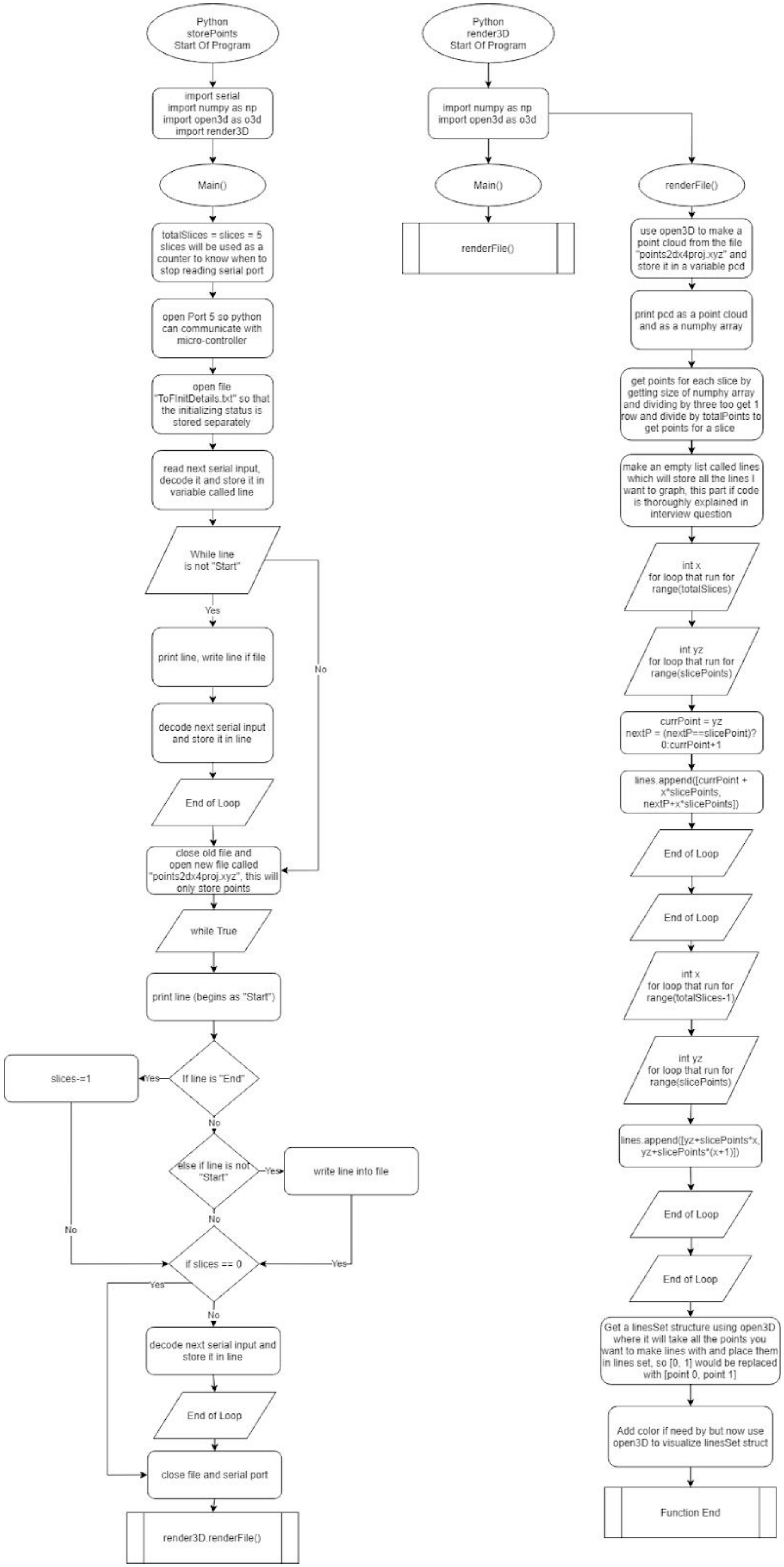
Displacement module is talking about the IMU's sampling speed. Since my bus speed **frequency is 48Mhz** the maximum sampling rate for IMU's **sampling speed would need to be 24MHz** according to the Nyquist Rate. However since this uses I^2C and that has an integrated buffer of 1024 times my actual frequency would be 48MHz/1024 which would mean my max sampling speed for IMU should be 23438bps.

Circuit Schematic



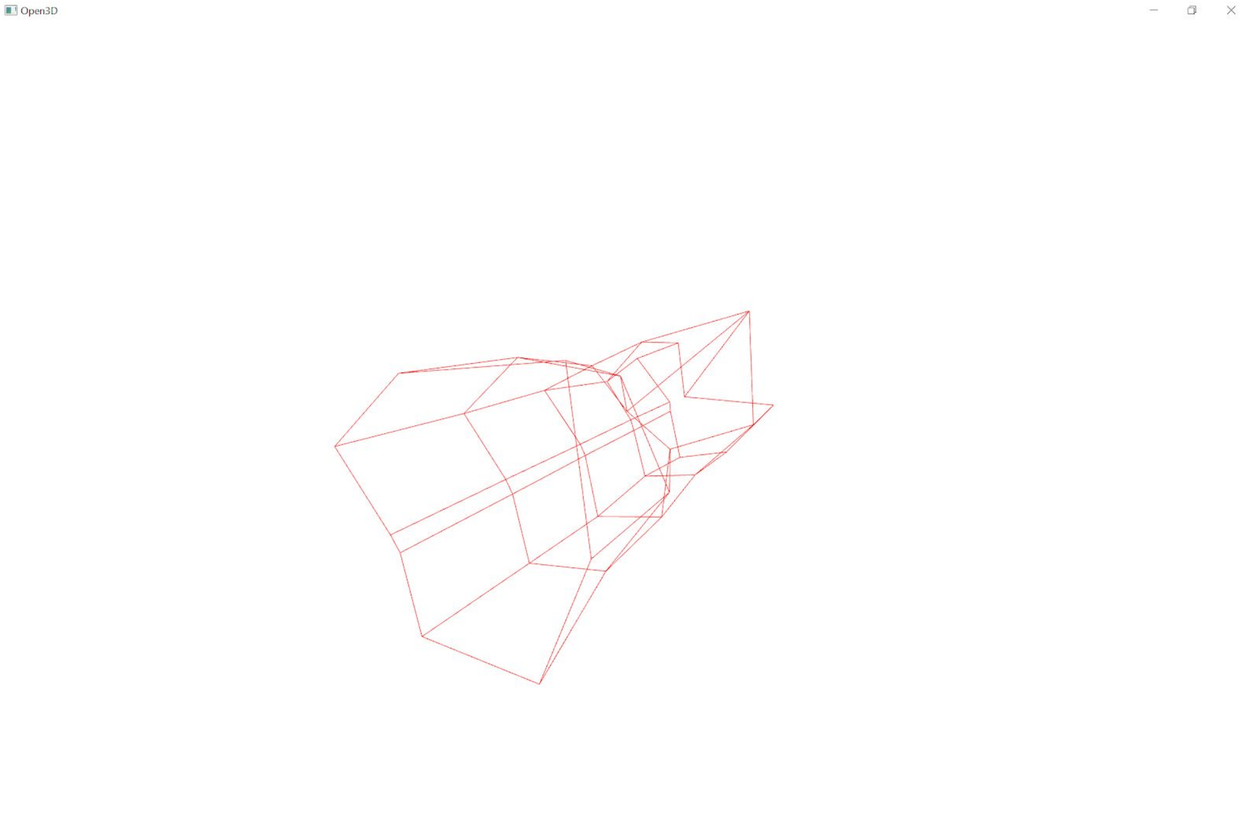
Programming Logic Flowcharts



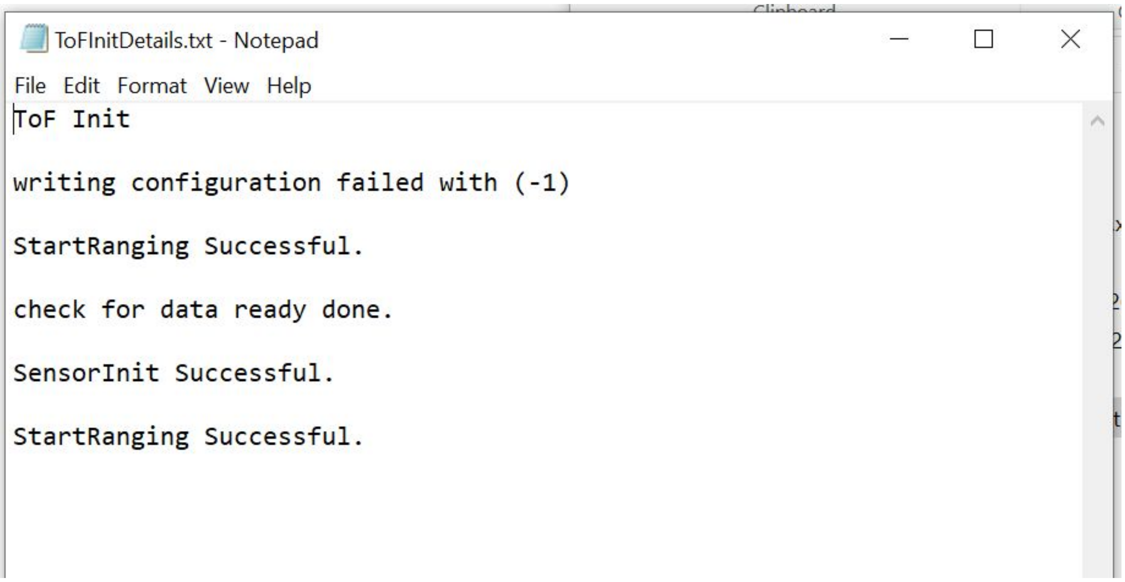

Python
storePoints
Start Of Program
import serial
import numpy as np
import open3d as o3d
import render3D
Main()
totalSlices = slices = 5
slices will be used as a
counter to know when to
stop reading serial port
open Port 5 so python
can communicate with
micro-controller
open file
"ToFInitDetails.txt" so that
the initializing status is
stored separately
read next serial input,
decode it and store it in
variable called line
While line
is not "Start"
Yes
No
print line, write line if file
decode next serial input
and store it in line
End of Loop
close old file and
open new file called
"points2dx4proj.xyz", this will
only store points
while True
print line (begins as "Start")
If line is "End"
Yes
slices-=1
No
else if line is not
"Start"
Yes
write line into file
No
No
if slices == 0
Yes
Yes
No
decode next serial input and
store it in line
End of Loop
close file and serial port
render3D.renderFile()
Python
render3D
Start Of Program
import numpy as np
import open3d as o3d
Main()
renderFile()
renderFile()
use open3D to make a
point cloud from the file
"points2dx4proj.xyz" and
store it in a variable pcd
print pcd as a point cloud
and as a numphy array
get points for each slice by
getting size of numphy array
and dividing by three too get 1
row and divide by totalPoints to
get points for a slice
make an empty list called lines
which will store all the lines I
want to graph, this part if code
is thoroughly explained in
interview question
int x
for loop that run for
range(totalSlices)
int yz
for loop that run for
range(slicePoints)
currPoint = yz
nextP = (nextP==slicePoint)?
0:currPoint+1
lines.append([currPoint +
x*slicePoints,
nextP+x*slicePoints])
End of Loop
End of Loop
int x
for loop that run for
range(totalSlices-1)
int yz
for loop that run for
range(slicePoints)
lines.append([yz+slicePoints*x,
yz+slicePoints*(x+1)])
End of Loop
End of Loop
Get a linesSet structure using open3D
where it will take all the points you
want to make lines with and place them
in lines set, so [0, 1] would be replaced
with [point 0, point 1]
Add color if need by but now use
open3D to visualize linesSet struct
Function End

Final Project Demo

Not my best result but definitely not my worst.
https://drive.google.com/file/d/14cYadXg6SAR4LKwgZOWWfXLAzb2Wj6Gv/view?usp=sharing



ToF initialization details:



```
ToF Init

writing configuration failed with (-1)

StartRanging Successful.

check for data ready done.

SensorInit Successful.

StartRanging Successful.
```