2DX4: Microprocessor Systems Project

Final Project

Instructors: Dr. Bruce, Dr. Haddra, Dr. Hranilovic, Dr. Shirani

Sawoud Al-los - L03 - alloss - 400188404

Submitted by: John Wyeth, 400171677

Video

https://drive.google.com/file/d/1vx4q4xat17Xc5pWZcpNugo7-4-DLbr9Z/view?usp=sharing

Device Overview

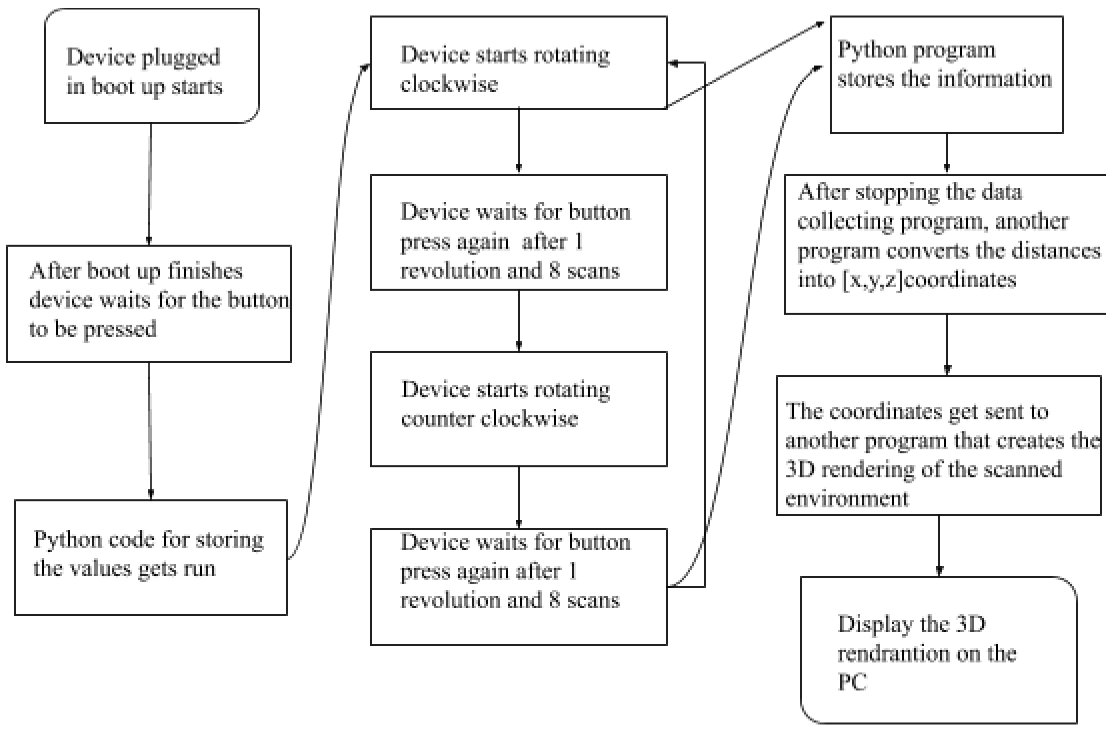
Features

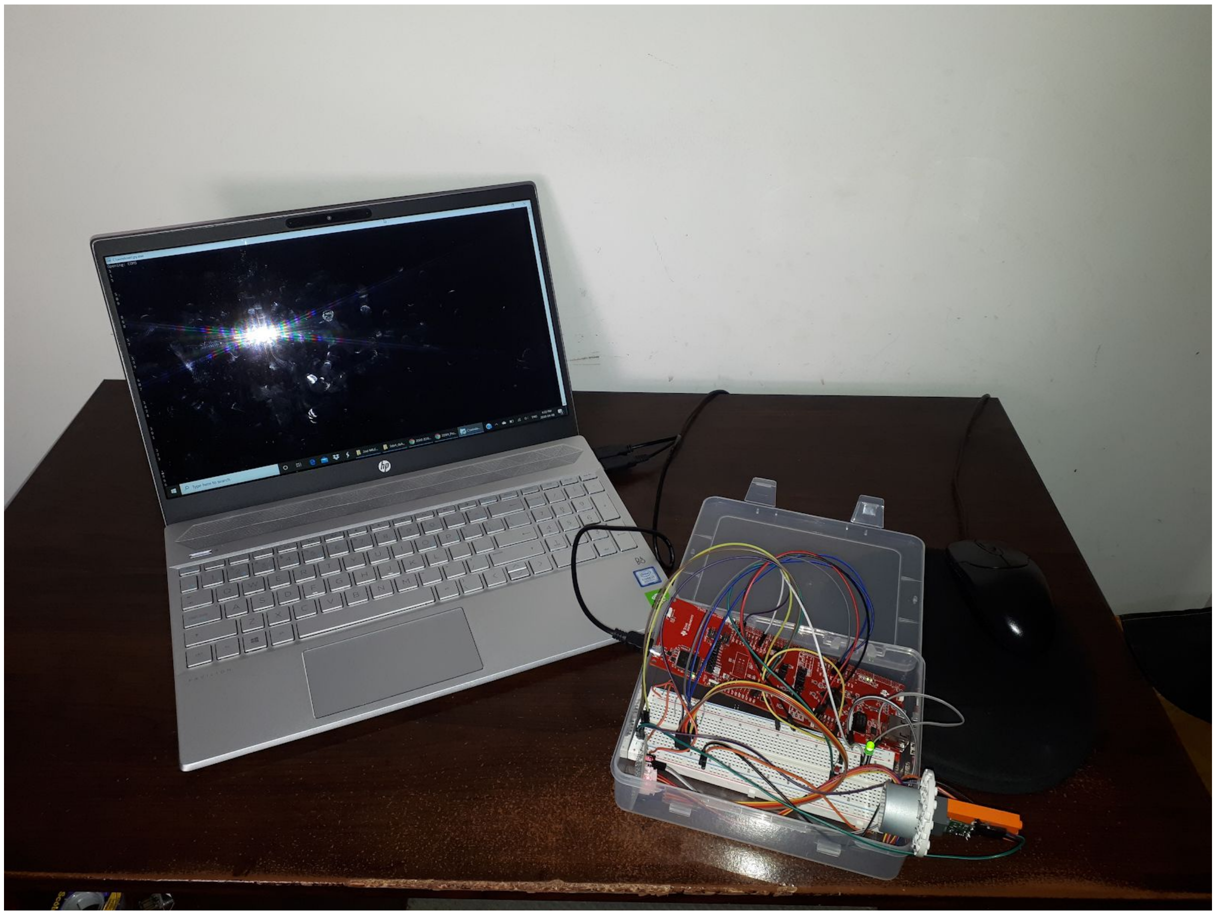
- Device uses a velleman stepper motor that alternates between clockwise and counterclockwise rotation.
- Device capable of collecting 8 measurements at every 45 degrees.
- The measurements get stored in a file which is then used to draw the location.
- An MSP-EXP432E401Y microcontroller, Operating voltage: 4.75V - 5.25V [1].
- Microcontroller uses a clock speed of 30MHz.
- It uses a VL53L1X ToF sensor.
- ToF also does the ADC and returns the distance in mm.
- It uses a UART communication protocol with a baud rate of 115200 bits/s.
- It requires 1 stop/start bit.

General Description

This system is designed to take distance measurements at 45 degree intervals using a Time of Flight Sensor (ToF), it rotates via a stepper motor clockwise and counterclockwise alternatingly every time an external button is pressed, while this is happening a python program starts running just before data collection starts to store data in a folder, then the data is used to create an image of the area swept by the device.

Block Diagram


Device plugged in boot up starts
After boot up finishes device waits for the button to be pressed
Python code for storing the values gets run
Device starts rotating clockwise
Device waits for button press again after 1 revolution and 8 scans
Device starts rotating counter clockwise
Device waits for button press again after 1 revolution and 8 scans
Python program stores the information
After stopping the data collecting program, another program converts the distances into [x,y,z]coordinates
The coordinates get sent to another program that creates the 3D rendering of the scanned environment
Display the 3D rendrantion on the PC



(On the PC screen it shows the measurements with a new line separating each one)

Device Characteristics Table

To run this device the following must be done:

- PL01 is used to show that the device is currently taking distance measurement instead of the onboard LEDs so it looks clear.
- For the stepper motor it was connected to the 5V and ground where the positive and negative voltage ports were located, and pins PM0, PM1, PM2, PM3 to IN 4, IN 3, IN 2, IN1, on the stepper motor respectively.
- On the ToF sensor port B02 and B03 are connected to the SCL and SDA ports respectively while the 3.3V and ground are connected to VIN and GND.
- The bus speed of the microcontroller is set to 30MHz, and the System.tick10ms() function gets its NOP number changed from 120,000,000 to 30,000,000 to maintain the 10 ms delay. The communication system is UART baud rate is set to 115200 bits/s

Detailed Description

Distance measurements

The ToF sensor used in this project does a lot of the work by doing the ADC conversion and preconditioning the data as soon as it comes out, to briefly explain how it works, it emits a class one laser, which is safe for the eyes, of wavelength 940 nm from its sensor and it calculates how long it took for it to come back, then it returns the result in millimeters. When the device is plugged in it will take on average 10 seconds, sometimes a maximum of 20 seconds, to boot up the ToF, after that is done when the provided red button is pressed the stepper motor starts rotating in the clockwise direction while the ToF collects data, and since no slip ring was used when the button is pressed again, the stepper motor will again collect data but it will do it in the counterclockwise direction, it in both cases the stepper motor will stop at every 45 degrees to take a measurement. Once a measurement has been taken it gets sent to a python program in which it collects the distance measurement via UART, but since UART sends each digit of a number individually and the program does not know where the next number starts or stops, the device also sends an "x" character so that the python program will know where the current number ends and the next begins, all of this is done so that the distances can be stored in a file in which later they are used.

Then another python program is used to convert the distance measurements into cartesian coordinates using the formulas; $y = distance * sin\,(n)$, $z = distance * cos\,(n)$, $n = +/- \frac{\pi}{4}$

The 'n' variable is increased/decreased at every iteration of the loop to get all the 8 coordinates of each sweep using the correct angles, it is said that it could be +/- since the motor both rotates clockwise and counterclockwise, additionally to adjust for that fact, that some sets are recorded CW and some CCW, coordinates of the CCW rotation are inputted in the opposite order, except for the first coordinate. To briefly summarize what each program does please check figures 2 and 3.

Displacement Measurement

To implement the displacement measurement, an IMU sensor would have been used with an I2C serial connection, when the ToF stops collecting data, the IMU would have started collecting data and the external LED would have been lit (in the final project the external LED was used for the distance measurement instead to make it clear), then when the button in pressed the IMU would stop measuring while the stepper motor does its y-z sweep. In the actual project a ruler was used to calculate a 25 cm displacement and in the python code it was incremented manually and added to the file.

Visualization

To achieve the visualization a 64-bit Windows 10 Laptop, 12 Gigabytes of ram, with an 8th generation intel i7 core processor and a GForce graphics card was used, and Python version 3.6.8 was installed alongside with NumPy (64-bit) and Open3D (64-bit) were installed. At the start, the program reads the data and and it stores it as a point cloud multidimensional array, also the number of lines in the file is determined and used to determine how many sets of measurements exist. After all of that is done the lines.append function, which is basically an array that stores the coordinates of a line connection, creates the line boundaries for each plane, is used to connected the vertices of each plane together, after that is done lines.append is again used but this time is to connect the appropriate vertices to the appropriate vertices on the other planes.

Application Example

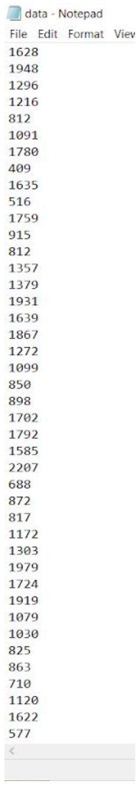
To setup this device to working order the following must be done:

1. Plug the device into the computer via the USB port.
2. Wait until bootup of the ToF sensor is done, it can be known that this operation has been done when the onboard LED's stop flashing.
3. In the first python program choose which communication port you are using and input its number in the "serialport" variable.
4. Run the first python program that takes the distance measurements and puts them in a file.
5. Press the external button to start data acquisition.
6. Wait until the first sweep is done.
7. Move the device's box to the desired position, as long as a constant displacement is used.
8. Again press the button to start data acquisition.
9. Keep repeating step 4 until all the measurements are gotten.
10. Once done close the python program and unplug the device.
11. Change the "x" variable to whatever the displacement is in mm in the second python code.
12. Run the second python code to get the .xyz format.
13. Run the last python code to get the 3D rendering.

The x-axis is the displacement axis while the y-axis is the 'y' axis of the cross section, and the z-axis is the 'x' component of the cross section.
Also the below pictures show the expected output in my case, to briefly summarize the location:

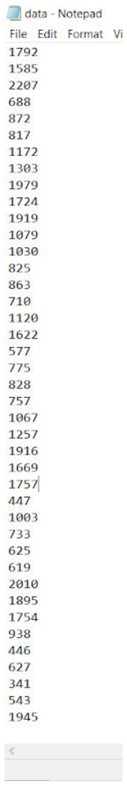
- Room with the door closed
- Bookshelf followed by a table and chair on the right side
- Bed on the right side

- Light Bulb in the middle of the ceiling
- Room dimensions approximately: 3 m by 4m by 2.5 m
- Measurements were taken approximately 1.5 m from each wall and 1.2 m from the floor



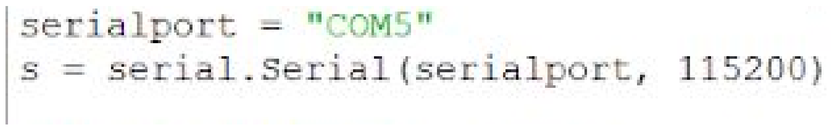
data - Notepad
File Edit Format View

```
1628
1948
1296
1216
812
1091
1780
409
1635
516
1759
915
812
1357
1379
1931
1639
1867
1272
1099
850
898
1702
1792
1585
2207
688
872
817
1172
1303
1979
1724
1919
1079
1030
825
863
710
1120
1622
577
```



data - Notepad
File Edit Format Vi

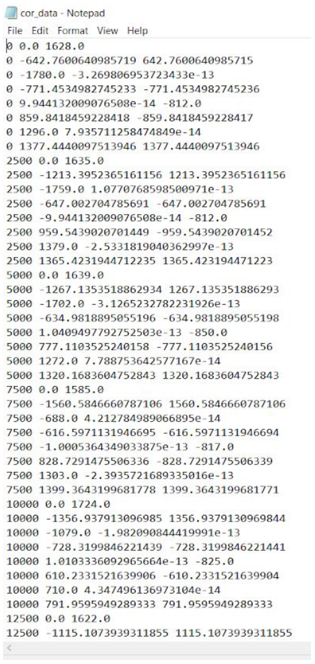
```
1792
1585
2207
688
872
817
1172
1303
1979
1724
1919
1079
1030
825
863
710
1120
1622
577
775
828
757
1067
1257
1916
1669
1757
447
1003
733
625
619
2010
1895
1754
938
446
627
341
543
1945
```



```
serialport = "COM5"
s = serial.Serial(serialport, 115200)
```

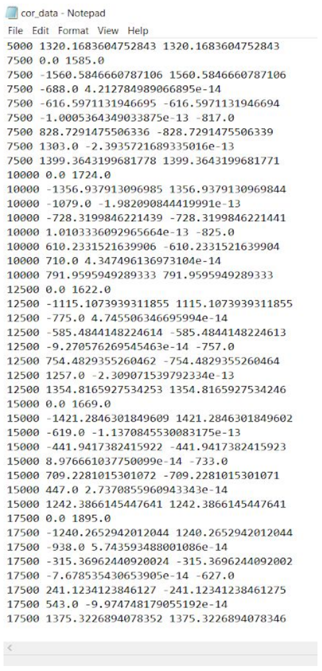
The picture above is the only section that changes in the code to whatever port is required, everything else may remain the same.

The pictures on the left show that acquired data. As it can be seen there exists 64 measurements since 8 sweeps of the room were done, the measurements are stored as strings and the file is of a .txt format.



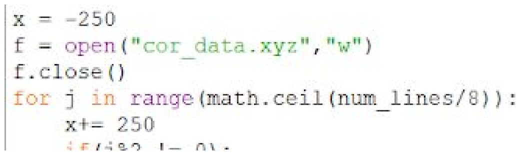
cor_data - Notepad
File Edit Format View Help

```
0 0.0 1628.0
0 -642.7600640985719 642.7600640985715
0 -1780.0 -3.269806953723433e-13
0 -771.4534982745233 -771.4534982745236
0 9.944132009076508e-14 -812.0
0 859.8418459228418 -859.8418459228417
0 1296.0 7.935711258474849e-14
0 1377.4440097513946 1377.4440097513946
2500 0.0 1635.0
2500 -1213.3952365161156 1213.3952365161156
2500 -1759.0 1.0770768598500971e-13
2500 -647.002704785691 -647.002704785691
2500 -9.944132009076508e-14 -812.0
2500 959.5439020701449 -959.5439020701452
2500 1379.0 -2.5331819040362997e-13
2500 1365.4231944712235 1365.423194471223
5000 0.0 1639.0
5000 -1267.1353518862934 1267.135351886293
5000 -1702.0 -3.1265232782231926e-13
5000 -634.9818895055196 -634.9818895055198
5000 1.0409497792752503e-13 -850.0
5000 777.1103525240158 -777.1103525240156
5000 1272.0 7.788753642577167e-14
5000 1320.1683604752843 1320.1683604752843
7500 0.0 1585.0
7500 -1560.5846660787106 1560.5846660787106
7500 -688.0 4.212784989066895e-14
7500 -616.5971131946695 -616.5971131946694
7500 -1.0005364349033875e-13 -817.0
7500 828.7291475506336 -828.7291475506339
7500 1303.0 -2.3935721689335016e-13
7500 1399.3643199681778 1399.3643199681771
10000 0.0 1724.0
10000 -1356.937913096985 1356.9379130969844
10000 -1079.0 -1.982090844419991e-13
10000 -728.3199846221439 -728.3199846221441
10000 1.0103336092965664e-13 -825.0
10000 610.2331521639906 -610.2331521639904
10000 710.0 4.347496136973104e-14
10000 791.9595949289333 791.9595949289333
12500 0.0 1622.0
12500 -1115.1073939311855 1115.1073939311855
```



cor_data - Notepad
File Edit Format View Help

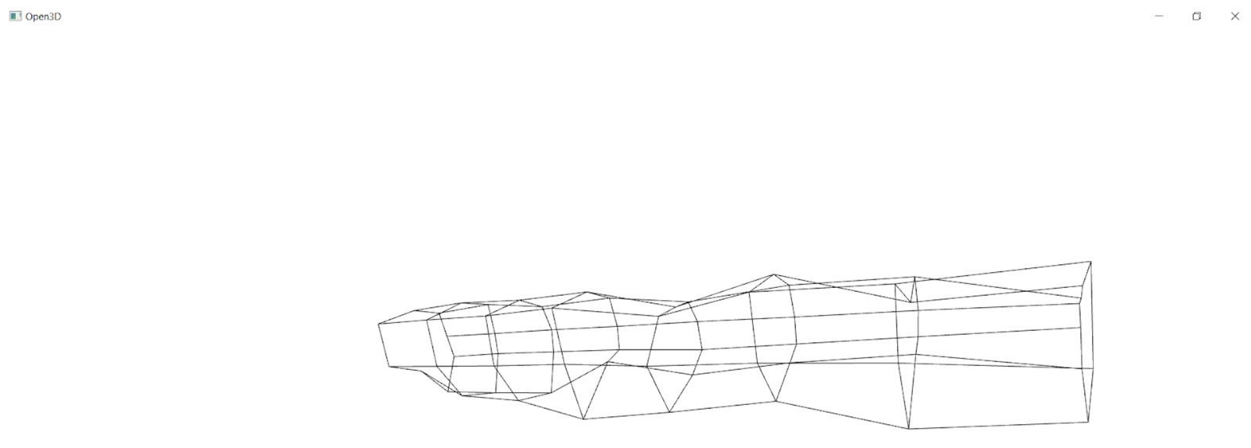
```
5000 1320.1683604752843 1320.1683604752843
7500 0.0 1585.0
7500 -1560.5846660787106 1560.5846660787106
7500 -688.0 4.212784989066895e-14
7500 -616.5971131946695 -616.5971131946694
7500 -1.0005364349033875e-13 -817.0
7500 828.7291475506336 -828.7291475506339
7500 1303.0 -2.3935721689335016e-13
7500 1399.3643199681778 1399.3643199681771
10000 0.0 1724.0
10000 -1356.937913096985 1356.9379130969844
10000 -1079.0 -1.982090844419991e-13
10000 -728.3199846221439 -728.3199846221441
10000 1.0103336092965664e-13 -825.0
10000 610.2331521639906 -610.2331521639904
10000 710.0 4.347496136973104e-14
10000 791.9595949289333 791.9595949289333
12500 0.0 1622.0
12500 -1115.1073939311855 1115.1073939311855
12500 -775.0 4.745506346695994e-14
12500 -585.4844148224614 -585.4844148224613
12500 -9.270576269545463e-14 -757.0
12500 754.4829355260462 -754.4829355260464
12500 1257.0 -2.309071539792334e-13
12500 1354.8165927534253 1354.8165927534246
15000 0.0 1669.0
15000 -1421.2846301849609 1421.2846301849602
15000 -619.0 -1.1370845530083175e-13
15000 -441.9417382415922 -441.9417382415923
15000 8.976661037750099e-14 -733.0
15000 709.2281015301072 -709.2281015301071
15000 447.0 2.7370855960943343e-14
15000 1242.3866145447641 1242.3866145447641
17500 0.0 1895.0
17500 -1240.2652942012044 1240.2652942012044
17500 -938.0 5.743593488001086e-14
17500 -315.36962440920024 -315.3696244092002
17500 -7.678535430653905e-14 -627.0
17500 241.1234123846127 -241.12341238461275
17500 543.0 -9.974748179055192e-14
17500 1375.3226894078352 1375.3226894078346
```



```
x = -250
f = open("cor_data.xyz","w")
f.close()
for j in range(math.ceil(num_lines/8)):
    x+= 250
```

The picture above shows that only the "x" variable in those two locations needs to be changed (the sign on the first variable must not be changed), otherwise nothing needs changing.

The pictures on the left show what the .xyz file contents should look like (decimal points were kept to ensure accuracy)



This is the visualization of the room (the displacement wan increased just to clearly show what is happening), and as it can be seen it is fairly accurate it shows on the far left how the bed starts to decrease the area of the planes, and in the middle plane the light bulb causes a significant decrease in elevation of the room.

Limitations

1. For the MSP-EXP432E401Y Microcontroller its limitations do not have much effect for this purpose, since python is implementing the trigonomic functions the users PC's is the true limitation

2. The ToF sensor's data sheet states that it uses an 8-bit register and an input voltage of 3.3 and a ground voltage [2], while The IMU uses a 16 bit register and a 2.5 V input and a ground [3] which means that the quantization errors are:

$$resolution = V_{FS}/2^n$$
$$resolution_{ToF} = 3.3/2^8$$
$$resolution_{ToF} \approx 1.29 * 10^{-2}$$

$resolution_{IMU} = 2.5/2^{16}$

$resolution_{IMU} \approx 3.81 * 10^{-5}$

3&4. The maximum standard serial communication rate is 115200 bits/s, this was verified by using realterm and seeing that its output screen showed the expected messages from the microcontroller without any character errors, and the communication method is UART. for the IMU it only accepts an I2C serial communication port.

5. The bus speed of the microcontroller is the main limitation on the system, it was noticed that when this speed was higher before it operated much faster, but when it was changed to the student specific 30 MHz, it became significantly slower.

6. Since we are working at 30MHz the Nyquist sampling rate would be

$S = f/2$

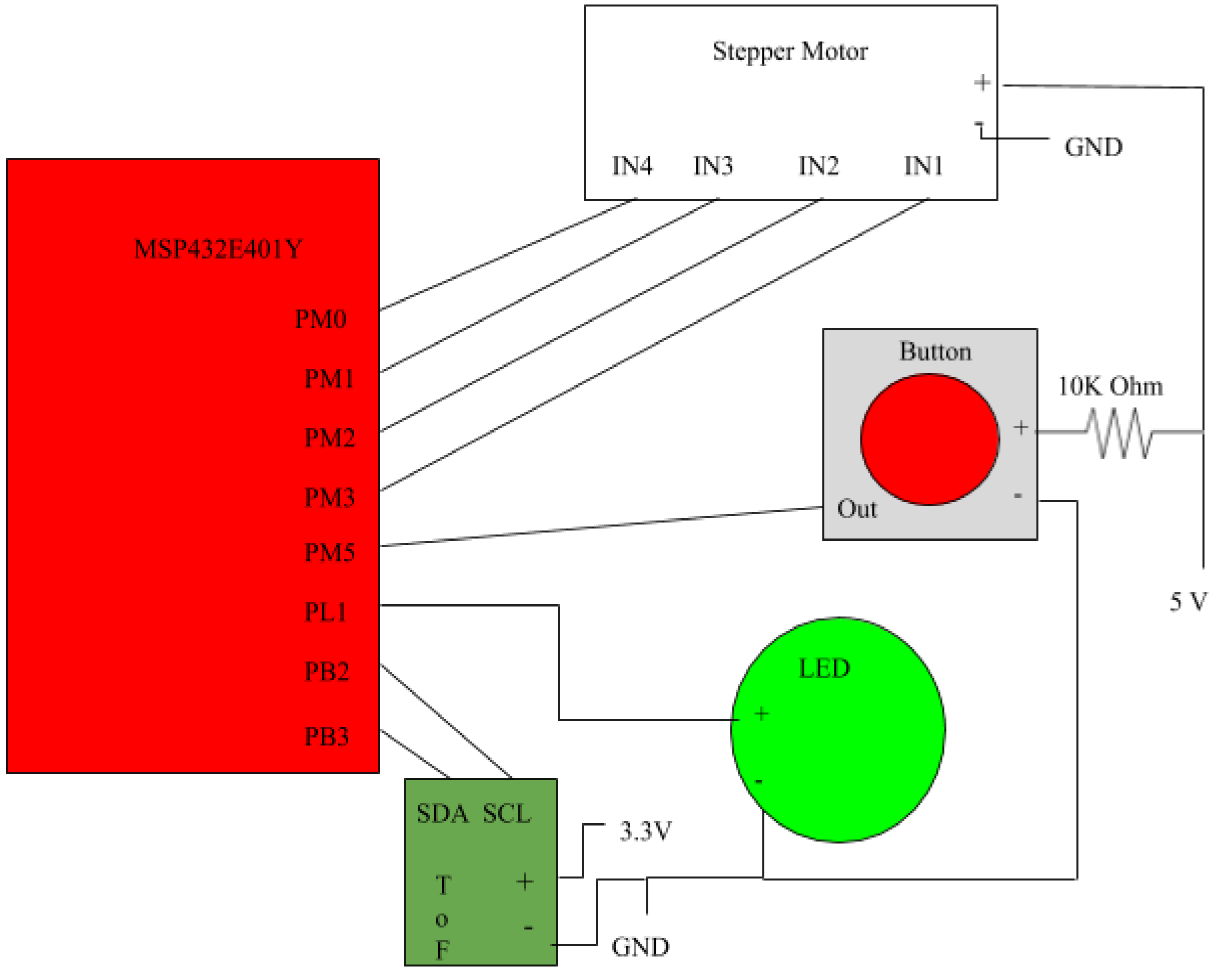
$S = 30MHz/2$

$S = 15MHz$

If we start to exceed this frequency, then the device will not sample data at relatively meaningful locations, it might give us an output that looks like it makes sense but in reality it is not showing what is truly happening.

Circuit Schematic

Figure 1



Programming Logic Flowchart(s)

Figure 2: C Program

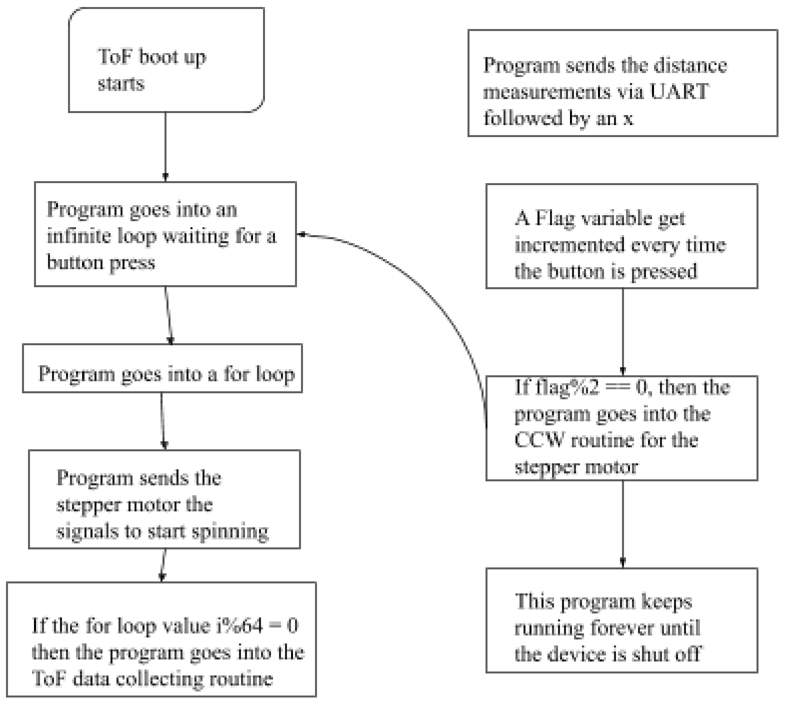
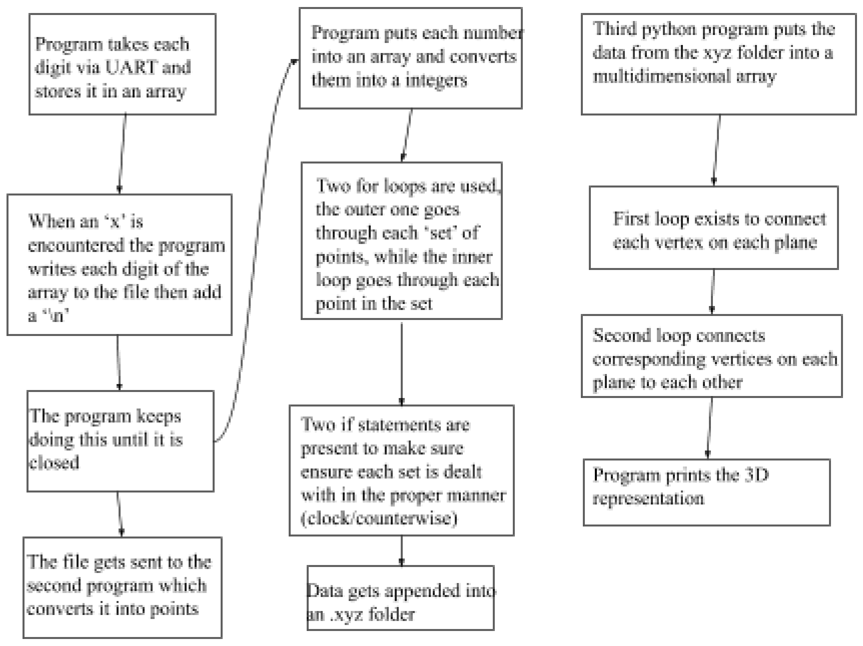

ToF boot up starts
Program goes into an infinite loop waiting for a button press
Program goes into a for loop
Program sends the stepper motor the signals to start spinning
If the for loop value i%64 = 0 then the program goes into the ToF data collecting routine
Program sends the distance measurements via UART followed by an x
A Flag variable get incremented every time the button is pressed
If flag%2 == 0, then the program goes into the CCW routine for the stepper motor
This program keeps running forever until the device is shut off

Figure 3: Python Program


Program takes each digit via UART and stores it in an array
When an 'x' is encountered the program writes each digit of the array to the file then add a '\n'
The program keeps doing this until it is closed
The file gets sent to the second program which converts it into points
Program puts each number into an array and converts them into a integers
Two for loops are used, the outer one goes through each 'set' of points, while the inner loop goes through each point in the set
Two if statements are present to make sure ensure each set is dealt with in the proper manner (clock/counterwise)
Data gets appended into an .xyz folder
Third python program puts the data from the xyz folder into a multidimensional array
First loop exists to connect each vertex on each plane
Second loop connects corresponding vertices on each plane to each other
Program prints the 3D representation

Citations

[1]
Texas Instruments, “MSP432E401Y SimpleLink™ Ethernet Microcontroller” MSP432E401Y datasheet, Oct. 2017 [Revised N/A].

[2]
Texas Instruments, “A new generation, long distance ranging Time-of-Flight sensor based on ST’s FlightSense™ technology” VL53L1X datasheet, Feb. 2018 [Revised Nov. 2018].

[3]
Texas Instruments, “MPU-9250 Product Specification Revision 1.1” PS-MPU-9250A-01 datasheet, Dec. 2013[Revised Jun. 2016].