2DX3 Final Report

April 9, 2025

Jason Baik

Baikh1

400529068

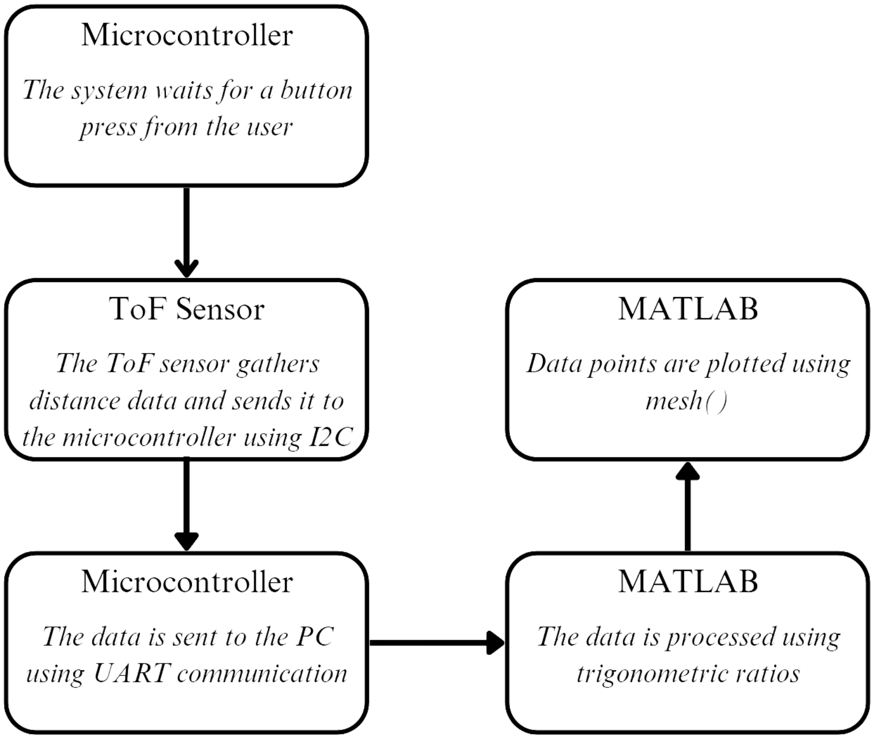
As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

Device Overview

The final implementation of the embedded spatial measurement system features 360° room mapping capabilities in three different axes with only the x axis being a manual step using a MSP432E401Y microcontroller. The specifications for this implementation consist of:

Bus Speed	26 MHz
Operating Voltage	
Cost	$254.99
Serial Communication	UART COMMUNICATION PORT: COM5 BAUD RATE: 115200
Receiver Side Programming Language	MATLAB
Memory Information	

The microcontroller is started with an initial button press. It is able to create a mapping of the room with respect to the y and z axis by obtaining the data from the time-of-flight sensor, which emits a 940nm wavelength infrared light and calculates the distance using the time that the beam took to bounce off the object and back to the sensor. This is done using by halving the total time it took and by multiplying by the speed of light. The process is done 64 times, each time at a different angle by mounting it onto a stepper motor. This data is sent to the micro through I2C communication. Afterwards, this data is communicated to the PC using UART serial communication at a baud rate of 115200 and is received by MATLAB. MATLAB then processes the information, taking each point and multiplying it with the appropriate trigonometric ratios, cosine for the 'y' value and sine for the 'z' value. This information is put into an array with set 'x' value increments, which was defined by the user prior to starting the scan and plots the data using the mesh() function.



Microcontroller

The system waits for a button press from the user

ToF Sensor

The ToF sensor gathers distance data and sends it to the microcontroller using I2C

Microcontroller

The data is sent to the PC using UART communication

MATLAB

The data is processed using trigonometric ratios

MATLAB

Data points are plotted using mesh( )

Device Characteristics Tables

Communication	
Start/Continue	PJ0
Bus Speed	26MHz
Baud Rate	115200
Port	COM5

ToF Sensor	
VIN	3V3
GND	GND
SCL	PB2
SDA	PB3

Stepper Motor	
IN1	PH0
IN2	PH1
IN3	PH2
IN4	PH3
+	5V
-	GND

LED	
Measurement	PF4 (D3)
UART Tx	PF0 (D4)
Scan Complete	PN0 (D2)

Detailed Description

The sensor that was used in this project was the VL53L1X time-of-flight sensor. This sensor can measure ranges up to four meters away and outputs the value in terms of millimetres. As briefly discussed earlier, the time-of-flight sensor emits a 940nm wavelength infrared light and calculates the distance using the time that the beam took to bounce off the object and back to the sensor. This is modelled by:

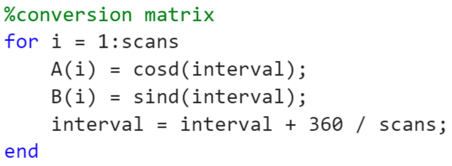
$$\text{Measured Distance} = \frac{\text{Photon Travel Time}}{2} * \text{Speed of Light}$$

This data is then transmitted to the microcontroller, which will flash D4 on board. The data is transmitted through the SDA line from the ToF sensor to the GPIO pin PB3, which is synchronized using the SCL which is connected to PB2. The microcontroller outputs a clock frequency of 100kbps. This process is repeated for the number of scans that are being taken. This is configured to be 64 scans, meaning that each step is going to be 5.625 degrees until a full rotation of the motor has been completed

This data in mm is then sent to MATLAB to be processed using UART communication, where it is sent in the form of:

1153,1343,1343,1343,1324,…, 323

The program then removes the commas and puts the numbers into an array. Then a loop is run to find the various trigonometric values that must be multiplied to obtain the y and z values. This is done by creating two more arrays that carry the values outputted by the cosine and sine function, being incremented, one step at a time as shown:



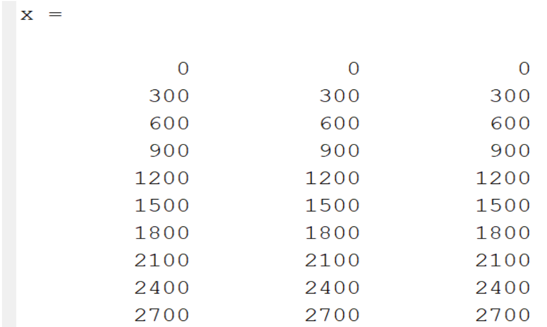
```
%conversion matrix
for i = 1:scans
    A(i) = cosd(interval);
    B(i) = sind(interval);
    interval = interval + 360 / scans;
end
```

In this case, the variable ‘interval’ is set to -270 degrees initially, as the ToF sensor is pointing downwards towards the ground. For example, to calculate the cosine ratio or the y value, the loop would be on its second iteration, with interval being -264.375, giving -0.098 at A(2).

The x coordinate of this program is a constant, with its value being 300mm. This means that the displacement for each step the system should be 30cm. In the code, the array is given as x = repmat((0:xOffset:(xOffset*(steps - 1)))', 1, scans);

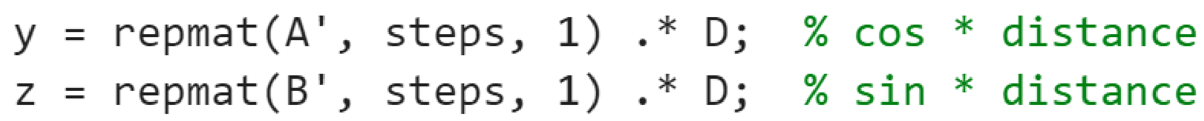
where xOffset is 300, steps is 10 and scans is 64. The output should look something like this:

The output matrix of x repeats 10 times*



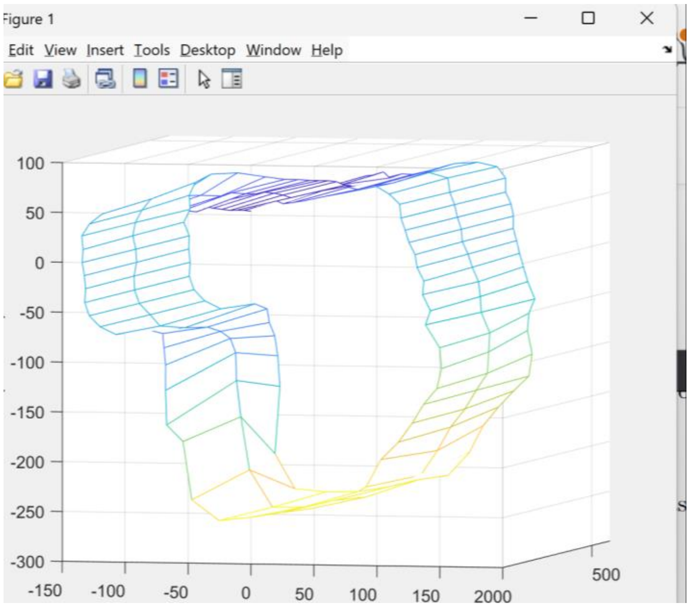
```
x =

            0           0           0
          300         300         300
          600         600         600
          900         900         900
         1200        1200        1200
         1500        1500        1500
         1800        1800        1800
         2100        2100        2100
         2400        2400        2400
         2700        2700        2700
```

Then the y and z axes are also set. Where repmat() creates a matrix where A' is repeated 10 times, creating a 10x64 matrix. Afterwards, each individual entry of A and B is multiplied by the distance data. At this point, D is a [steps, scans] sized matrix:



```
y = repmat(A', steps, 1) .* D;  % cos * distance
z = repmat(B', steps, 1) .* D;  % sin * distance
```

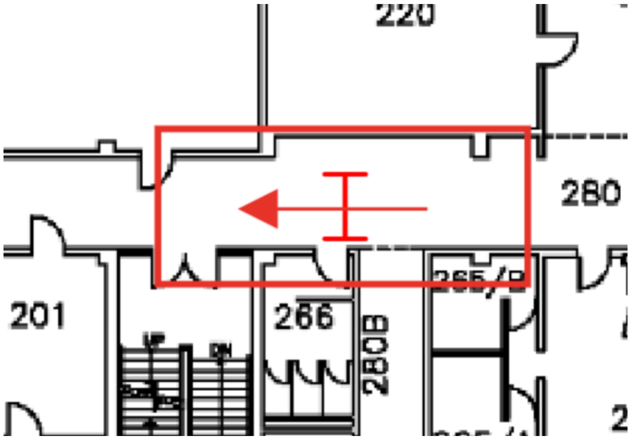
After running the previous code, the mesh() function is used, outputting a graph that looks like the following:

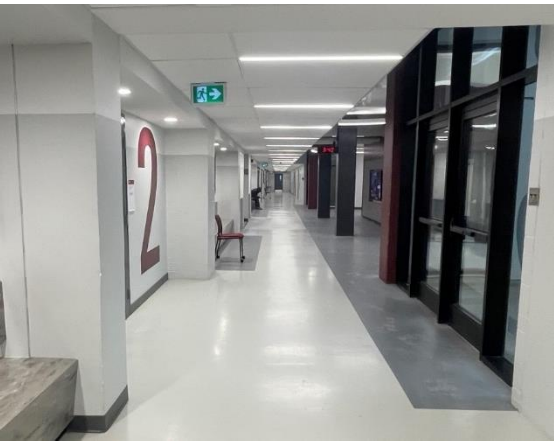


An image of a box scanned for deliverable two with three steps and sixty-four scans

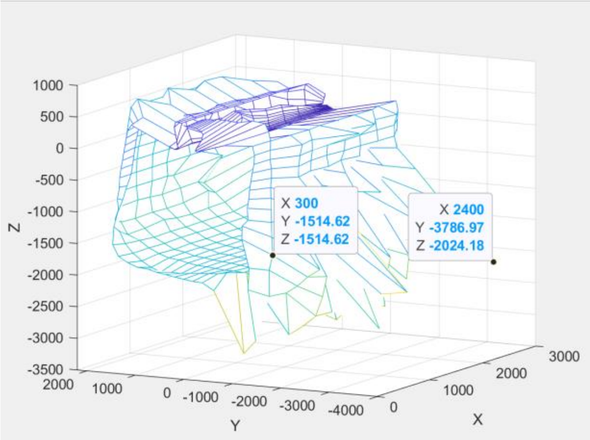
Application

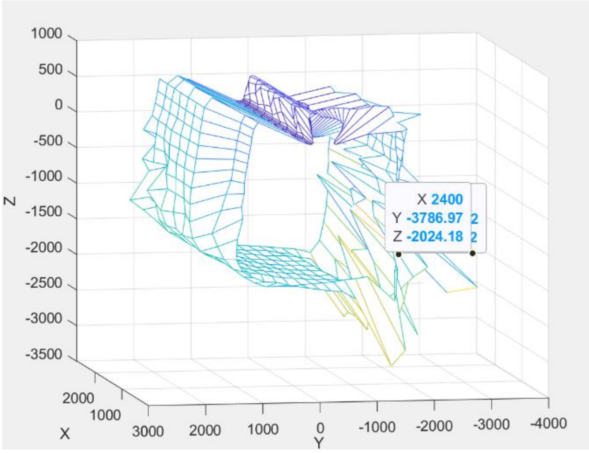
This system was tested in the I'th scanning location in John Hodgins Engineering building.





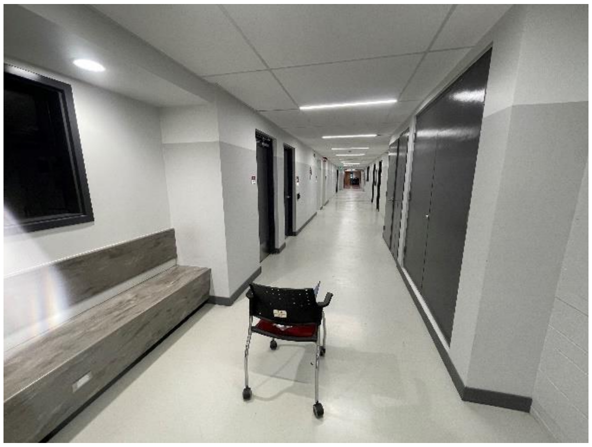
The output of the scan at 10 steps and 64 scans is the following:

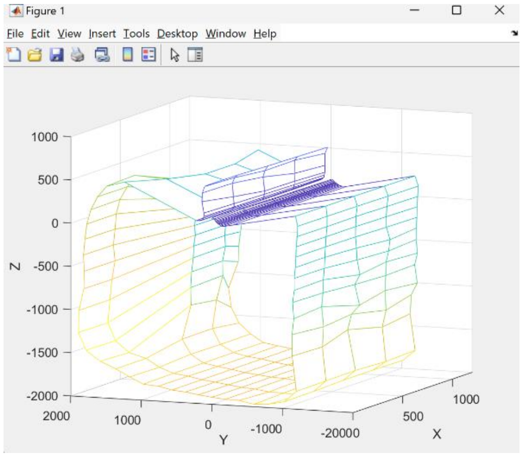




It should be noted that the y axis refers to the sides of the room and the z axis refers to the top and bottom of the room. The x axis is the direction that the step is taken in.

Clearly, there is a discrepancy between the scan and the real structure. This can be attributed to the glass doors, interfering with the ToF sensor results. When the scan was taken further down in the hallway where there are no windows, the accuracy of scans increase:

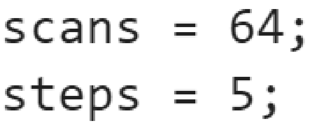




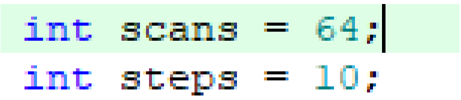
Instructions

To take a scan, the following steps should be taken:

1. Setup the system using the pins that were specified earlier and attach the ToF sensor to the motor.
2. Input the number of steps and scans that are going to be taken into Keil and MATLAB, where steps are the physical steps that are being taken, 300mm at a time and scans are the number of slices that will be taken.

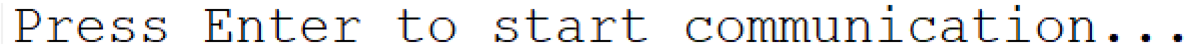


```
scans = 64;
steps = 5;
```



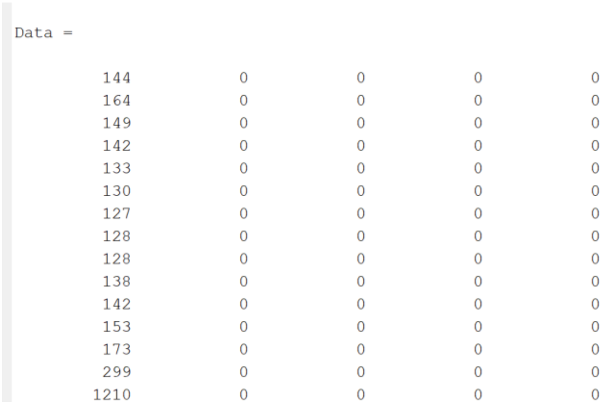
```
int scans = 64;
int steps = 10;
```

3. Build and load the Keil code onto the microcontroller and press the reset button.
4. Run the MATLAB code and press enter when ready to begin the MATLAB communication.



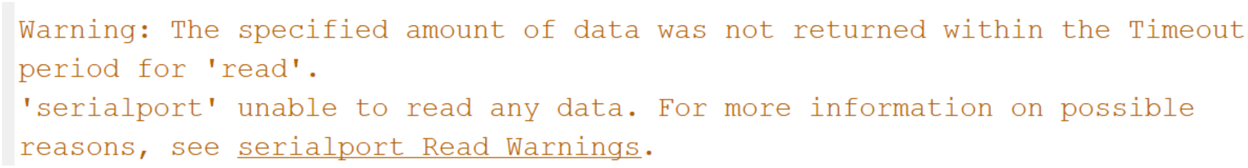
```
Press Enter to start communication...
```

5. Press PJ0 to start a scan. After the scan is complete, the following data should be shown on MATLAB.



```
Data =

         144           0           0           0           0
         164           0           0           0           0
         149           0           0           0           0
         142           0           0           0           0
         133           0           0           0           0
         130           0           0           0           0
         127           0           0           0           0
         128           0           0           0           0
         128           0           0           0           0
         138           0           0           0           0
         142           0           0           0           0
         153           0           0           0           0
         173           0           0           0           0
         299           0           0           0           0
        1210           0           0           0           0
```

6. Once the following warning appears, the system is ready to scan. Press PJ0 to start the scanning process again.



```
Warning: The specified amount of data was not returned within the Timeout
period for 'read'.
'serialport' unable to read any data. For more information on possible
reasons, see serialport Read Warnings.
```

7. Once the final scan has been completed, the program will output a mesh of the data.

Limitations

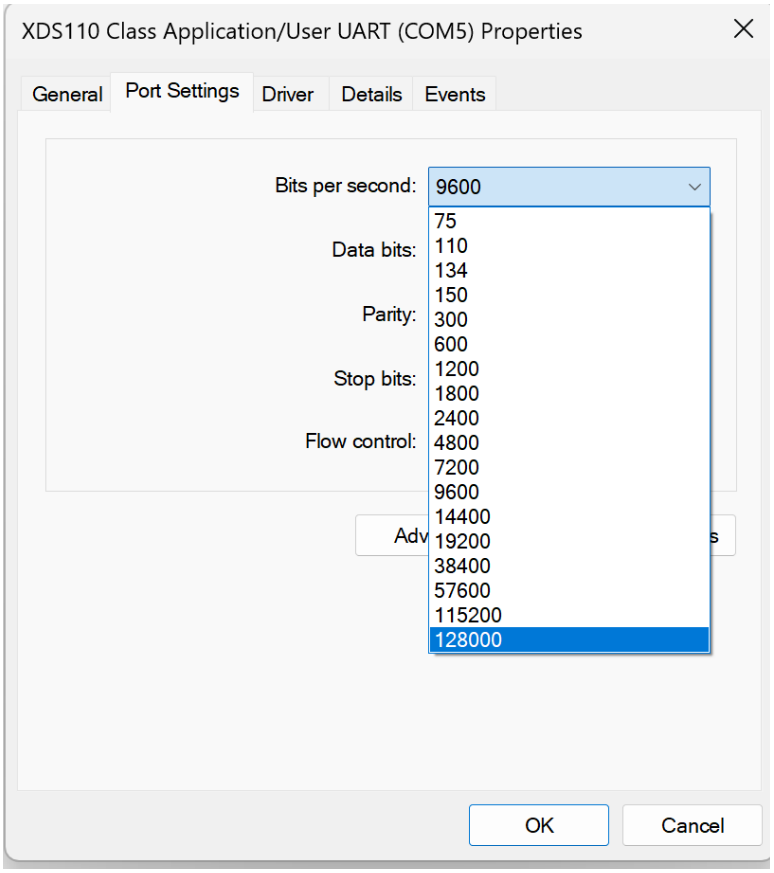
The two primary limitations of the current implementation of the project is the FPU capability as well as the way the data is being converted into y and z coordinates. Firstly, ,the floating point unit of the microcontroller does not have infinite precision, leading to inaccuracies when calculating decimal values. Over a long process, this can lead to a large deviation from the true desired values. In addition to this, the trigonometric functions that were used such as sine and cosine can lead to values with large decimal places. When the data is transferred to MATLAB and the trigonometric functions are applied, the value of the data deviates further from the true values.

The maximum quantization error of the ToF sensor is modelled by:

$$\text{Quantization Error} = \frac{V_{FS}}{2^n}$$

Where V_{FS} is the full scale voltage and n is the number of bits. Due to the fact that we know the maximum quantization error happens at half the value of the least significant bit, the maximum quantization error can be defined as 0.5mm as the least significant bit of the system is 1mm.

The maximum standard serial communication rate that can be achieved by the system and the PC according to the device settings on the PC is 128000bps. This is found in the device menu of the desired port:



XDS110 Class Application/User UART (COM5) Properties

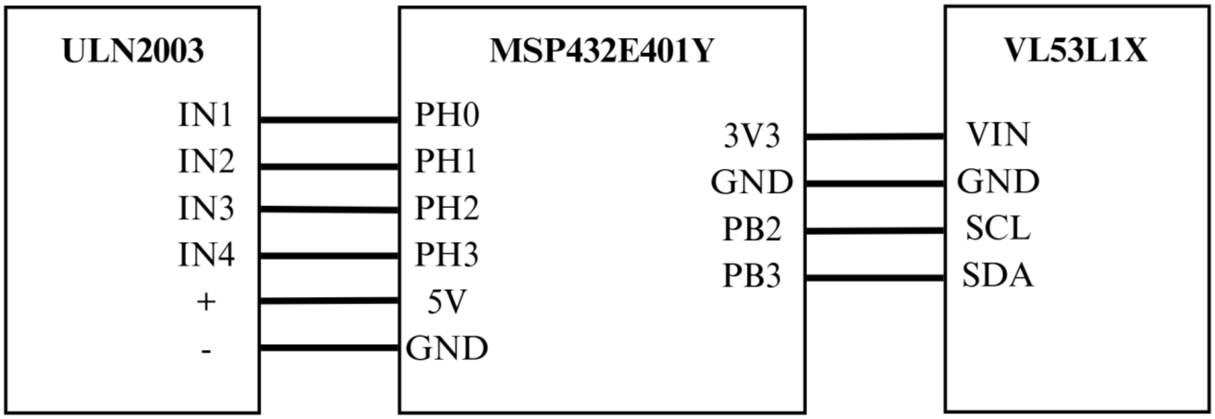
Bits per second: 9600

In addition to finding it in the device menu, if a higher baud rate is tested, we can see that the microcontroller does not communicate information to MATLAB or any other serial monitor.

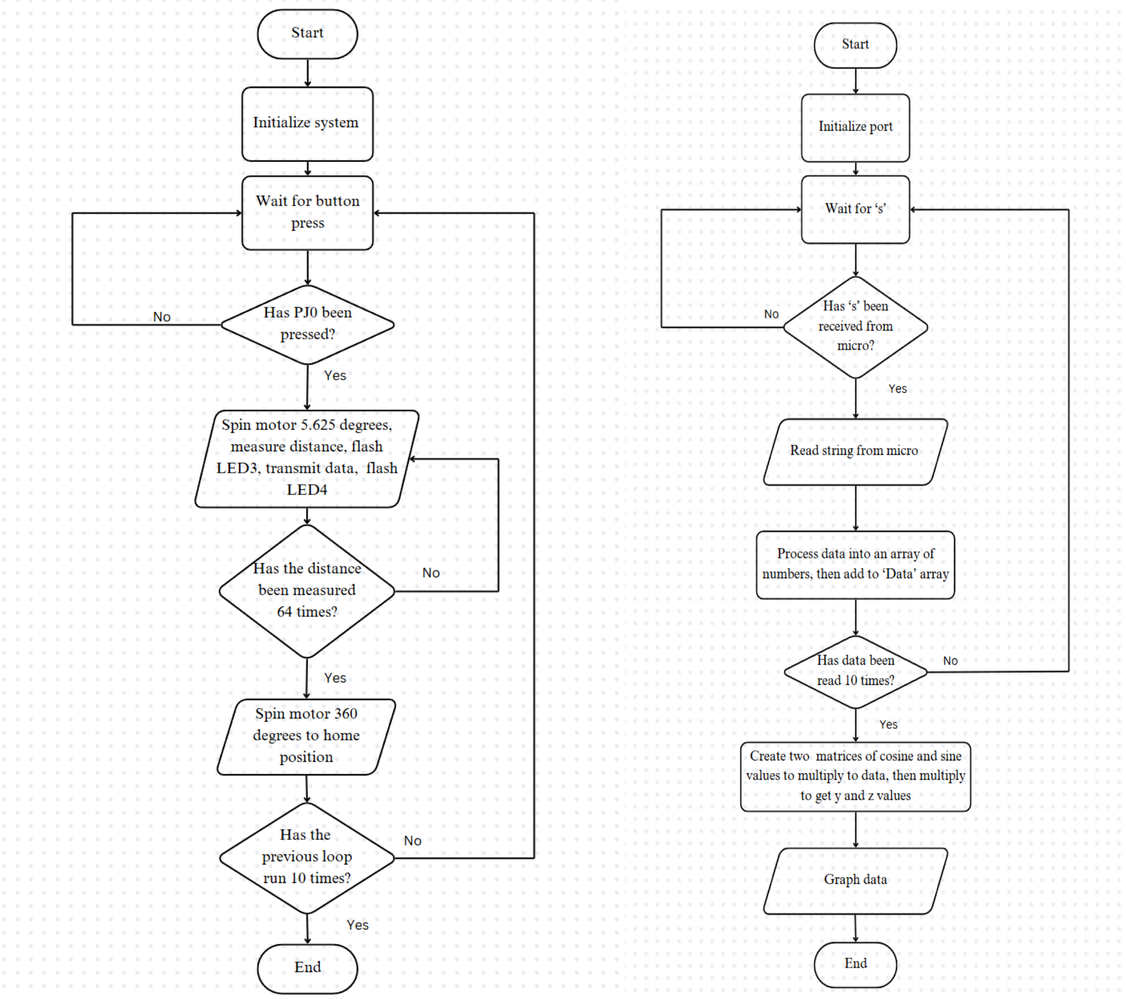
The communication method between the microcontroller and the ToF sensor is I2C. In the current implantation, it is configured to run at 100kps. From the data sheet it is understood that it can be running at a much faster speed, at approximately 400kHz.

Finally, the stepper motor and the ToF sensor are both bottlenecks of the entire system as they slow down the scanning process. Firstly, the ToF sensor is only able to sample at a rate of 50Hz, which is significantly slower than that of the rate of the rest of the system and how it is currently configured. The stepper motor is a bottleneck as the sensor is only able to take a measurement once the stepper motor has turned the appropriate amount. This can be tested on the system by seeing how quickly the stepper motor is able to turn in comparison to how quickly the data can transmit the data and process it. The greatest bottleneck of this particular system is the rate at which the motor can spin.

Circuit Schematic



Programming Logic Flowchart (Keil on left, MATLAB on right)



Start
Initialize system
Wait for button press
Has PJ0 been pressed?
No
Yes
Spin motor 5.625 degrees, measure distance, flash LED3, transmit data, flash LED4
Has the distance been measured 64 times?
No
Yes
Spin motor 360 degrees to home position
Has the previous loop run 10 times?
No
Yes
End
Start
Initialize port
Wait for 's'
Has 's' been received from micro?
No
Yes
Read string from micro
Process data into an array of numbers, then add to 'Data' array
Has data been read 10 times?
No
Yes
Create two matrices of cosine and sine values to multiply to data, then multiply to get y and z values
Graph data
End