2DX3: Microprocessor Systems
Final Project – LIDAR System

Instructors: Drs. Boursalie, Doyle, and Haddara

Aidan Mathew – mathea40 – 400306142 – 2DX3
L07 – Tuesday Afternoon Demo Session

Device Overview

a) Features

1. Low-Cost Lidar System
   - Can effectively map out spaces with sensor range of 4m
   - USB connection allows PC communication to send data
   - Measurements are visualized via Python, 3D plot in matplotlib
   - Motor spin reset to stop sensor wires from getting stuck or broken
2. Texas Instruments MSP432E401Y Microcontroller
   - Cortex M4F Processor – 256KB RAM (static) – 120MHz Clock – 30Mhz Assigned Bus
   - Programmable LEDs to understand microcontroller programming
   - Configurable onboard push buttons to carry out actions
   - Easily programmed in assembly, C, and C++
3. ULN2003 Stepper Motor Driver and 28-48 Stepper Motor
   - Rotating axle allows easy visualization of spins and attaching components
   - 4 LEDs which turn on based on the turning state of the motor
   - Motor has 512 turns to make one full rotation of 360 degrees
   - Voltage range of operation: 5V-12V
4. VL53L1X Time of Flight (ToF) Sensor
   - Infrared light is used to measure distance
   - Maximum measurement range is 4m
   - Voltage range of operation: 2.6V-3.5V
5. Data Communication Methods
   - I2C communication from sensor to the microcontroller
   - UART communication from the microcontroller to the personal computer (115200 bps)
6. Spatial Mapping
   - 3-dimensional plots generated through matplotlib python library
   - Written in Python3 ran through IDLE
   - Can create graphs based on user's specifications
7. Cost
   - AD2 (optional to verify bus speed) ~ 435$
   - COMPENG 2DX Kit (Microcontroller/Stepper Motor/Wires) ~ 189$

b) General Description

The LIDAR spatial mapping system is an intelligent embedded system that makes use of a time-of-flight senor to create a 3-dimensional plot of the surrounding environment. The system uses the 360-degree rotational freedom of the stepper motor to create a 3D map of the surroundings from the POV of the sensor. The entire system must be moved 0.1m for each data collection run. This allows for a depth collection of 1m, allowing 10 rings to be plotted for the surroundings. 64 data points are collected for each run, 640 measurements for 1 map. The

distances are then sent from the sensor to the microcontroller following I2C protocol, where the microcontroller converts the distance to meters.

The microcontroller then sends these measurements to the personal computer using UART communication. Python is used on the PC to compute the distances and create plots based on specific prompts.

The main feature of this system is the microcontroller (MSP432E401Y), as this acts as the 'middleman' permitting the flow of data from the ToF sensor to the personal computer. The micro controller takes care of the sensor requests and the stepper motor rotation. The correct initialization and enabling of ports on the microcontroller are what allows the transmission and receival of distance measurements from the sensor/PC using I2C and UART.

The stepper motor is used to spin the time-of-flight sensor 360 degrees in order to get 64 measurement points for one slice of the map. These measurements are then sent to the microcontroller using I2C.

The computer then receives this data from the microcontroller using UART, where python can access the measurements and store the information into a text file. Using simple trigonometry concepts, the measurements are converted to cartesian from to be plotted in 3-dimensions. Line vectors are then used to connect the slices that are used to create the map of the surrounding environment.

c) Block Diagram

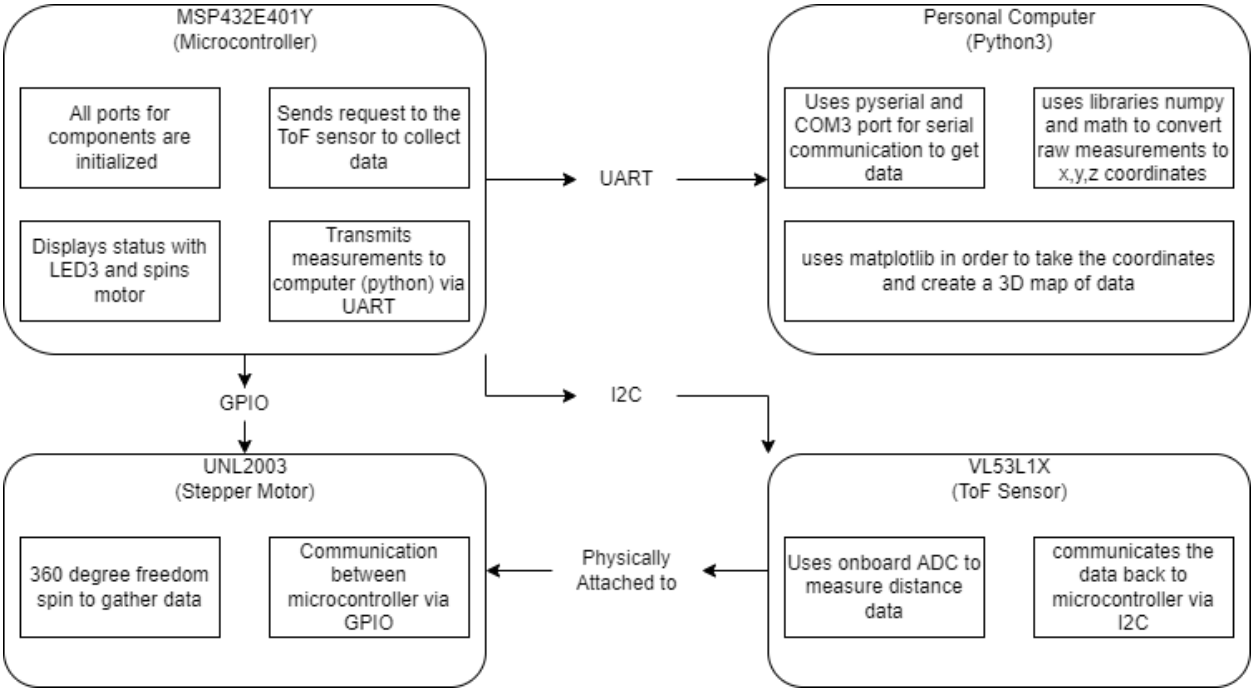

Figure 1: Block Diagram Displaying the components of the system and how they interact.

Device Characteristics Table

Device/System Specification	Characteristics
MSP432E401Y Microcontroller	
Bus Speed	120MHz (microcontroller limit) 30MHz (based on student number)
Serial Port (UART)	COM3 (personal device specific port)
Status LED	PF4 – LED3 (based on student number)
Baud Rate	115200 bps
Data Bits	1 start, 1 stop, and 8 data bits
Software	
Special Libraries	- pyserial - numpy - matplotlib
Python	Python 3.8.7
KEIL	MDK Version 5
VL53L1X (Time of Flight Sensor)	
Pin Map	
Device Pin	Microcontroller Connection
VDD	N/A
VIN	3.3V
GND	GND
SDA	PB3
SCL	PB2
XSHUT	N/A
GPIO1	N/A
28BYJ-48 (Stepper Motor)	
Pin Map	
Device Pin	Microcontroller Pin
IN1	PM0
IN2	PM1
IN3	PM2
IN4	PM3
Positive (+)	3.3V
Negative (-)	GND

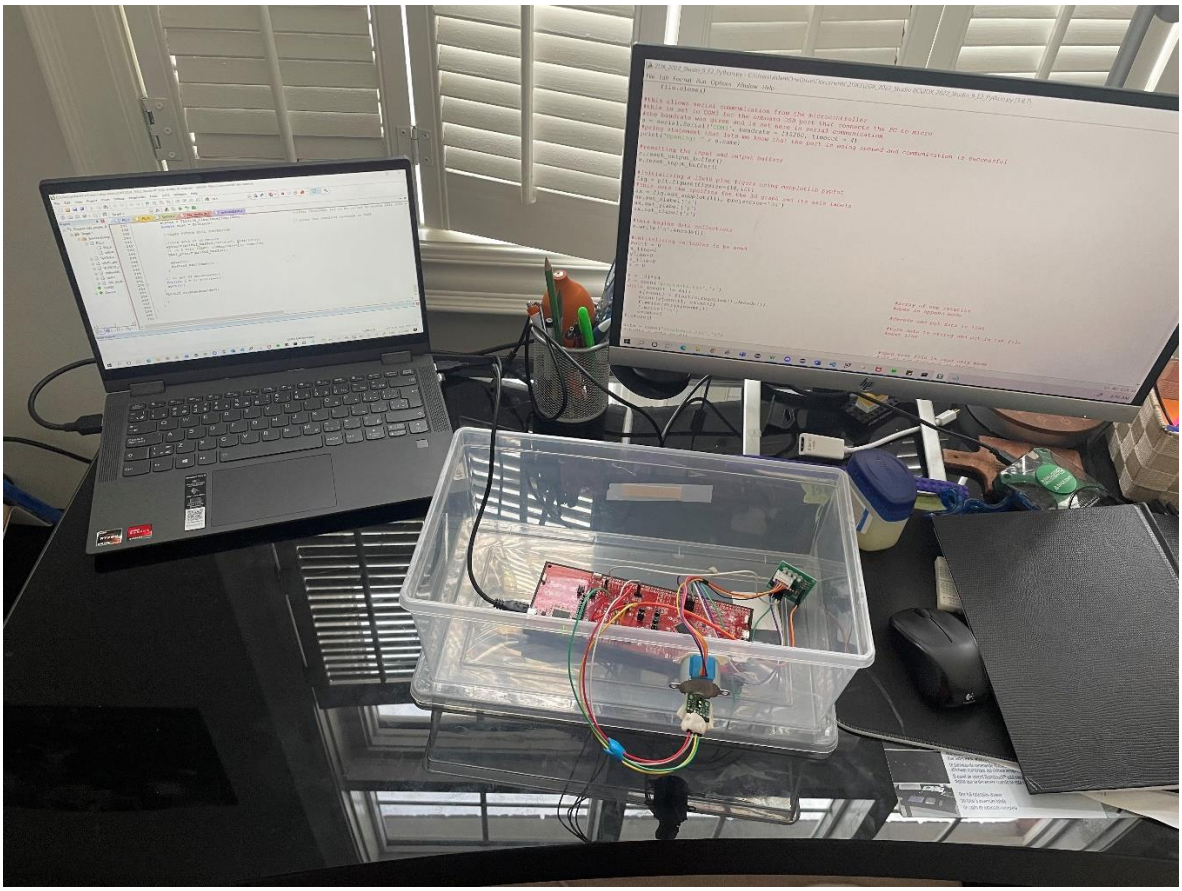

Figure 2: Personal Setup of System connected to PC

2DX_2022_Studio_9_E2_Python.py - C:\Users\aidan\OneDrive\Documents\2DX3\2DX_20...

File Edit Format Run Options Window Help

```
from cmath import cos, sin
from os import truncate
from turtle import color
import serial
import math
import numpy
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


erase = input("Do you want to delete the data from the previous recordings? P
#if yes, then open to write and delete everything
if(erase == 'Y'):
    file = open('graphdata.txt','w')
    truncate
    file.close()

#this allows serial communication from the microcontroller
#this is set to COM3 for the onboard USB port that connects the PC to micro
#the baudrate was given and is set here in serial communication
s = serial.Serial('COM3', baudrate = 115200, timeout = 4)
#pring statement that lets me know that the port is being opened and communic
print("Opening: " + s.name)

#resetting the input and output buffers
s.reset_output_buffer()
s.reset_input_buffer()

#initializing a 10x10 plot figure using matplotlib pyplot
fig = plt.figure(figsize=(10,10))
#this sets the spcifics for the 3d graph and its axis labels
ax = fig.add_subplot(111, projection='3d')
ax.set_xlabel('z')
ax.set_ylabel('y')
ax.set_zlabel('x')

#this begins data collections
s.write('s'.encode())

#initializing variables to be used
count = 0
x_line=0
yline=0
```

Ln: 11 Col: 117

IDLE Shell 3.8.7

File Edit Shell Debug Options Window Help

```
Python 3.8.7 (tags/v3.8.7:6503f05, Dec 21 2020, 17:59:51) [MSC v.1928 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\aidan\OneDrive\Documents\2DX3\2DX_2022_Studio 8C\2DX_202
2_Studio_9_E2_Python.py
Do you want to delete the data from the previous recordings? Please type 'Y'
for Yes and 'N' for No.Y
Opening: COM3
2.051 1
2.066 2
2.296 3
2.068 4
2.125 5
2.248 6
2.444 7
2.652 8
2.915 9
3.264 10
3.681 11
3.926 12
4.113 13
3.854 14
3.952 15
3.633 16
2.213 17
1.529 18
1.369 19
1.317 20
1.275 21
1.206 22
0.992 23
0.181 24
0.126 25
0.106 26
0.099 27
0.088 28
0.084 29
0.08 30
0.078 31
0.074 32
0.073 33
```

Ln: 6 Col: 0

Figure 3: PC output IDLE python code

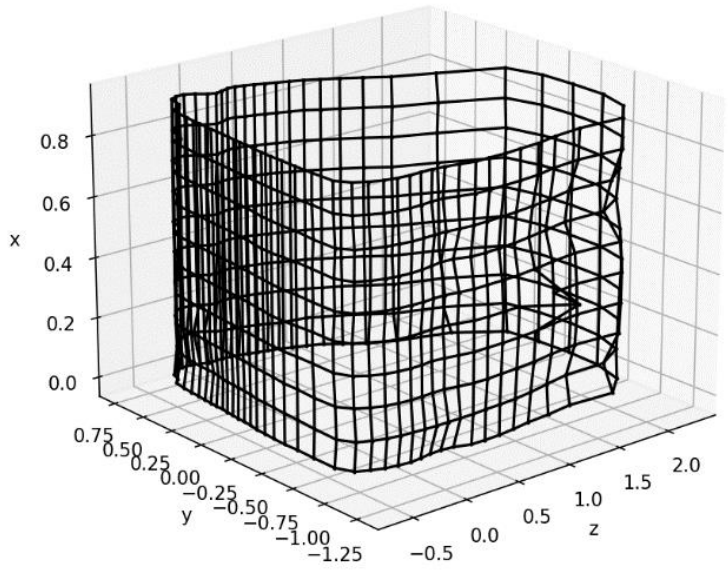

Figure 4: Matplotlib output map after 10 measurements

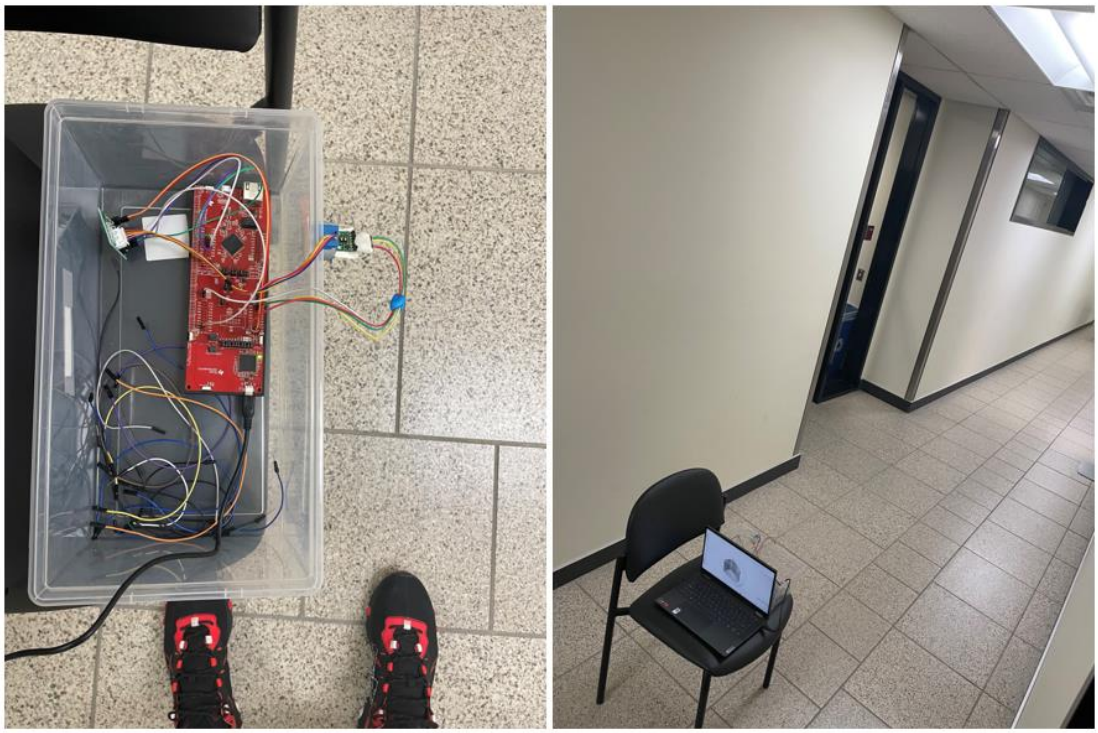

Figure 5: Hallway reconstruction with system and output plot.

Detailed Description

a) Distance Measurement

The LIDAR spatial mapping system measures distance through the ToF sensor (VL53L1X). The sensor is mounted to the stepper motor (28BYJ-48) with double sided tape and putty. This allows the sensor to spin smoothly with a 360-degree range of motion on a fixed plane. The microcontroller interfaces the ToF sensor, controls the stepper motor, monitors inputs from the onboard push button (Port J), and conveying the status through the onboard LED 3 (PF4).

The ToF sensor uses infrared light to measure the distance by measuring the amount of time it takes for the light to hit an object and return to the sensor. This is a simple distance formula where the displacement is the speed of light multiplied by the time divided by 2 to get the one-way distance (d = (speed of light * time) / 2. Using the pre-built ADC process, the sensor produces a measurement in millimetres. This distance measurement can be read by the microcontroller via I2C and is converted to meters for future calculations. Please note that the maximum range for the sensor is set to long mode in KEIL, this means that the highest distance value that can be measured is 4m. This means that the map/plot can only output a maximum of 8m, 4 meters in each direction.

The program for the microcontroller is based on studio 4B and 8C from the COMPENG 2DX3 course. Before flashing and loading the program onto the board, it is important to assign the bus speed and correspondingly change the systick value. The bus speed that was implemented for this project was determined by the student number 400306142, which was a 30MHz bus speed. From here the PSYSDIV value was calculated using the formula:

$$30MHz = \frac{480}{PSYSDIV + 1}$$

Therefore, the PSYSDIV value was set to 15 in PLL.h and the measurement time for systick was set to 30Mhz.

The microcontroller is also responsible for controlling the stepper motor by controlling the speed, direction, and angle. The stepper motor was initialized through Port M using pins PM0-PM3. The motor turned 64 times at an angle of 5.625 degrees to gather 64 data points and spin the full 360 degrees. For this to occur, a spin function was implemented to spin 8 steps which is the 512 total steps of the motor divided by 64 data points. This means that a distance measurement will be sampled every 8 steps of the motor. The delay for each rotation was 10ms. The spin function in the state of data collection will run 64 times through a for loop. Each time that a data point is being collected, the corresponding LED for the project will flash, indicating the status of the system. Upon the completion of 64 measurements, the microcontroller will transmit the data to the personal computer via UART for mathematical analysis in python3. Programming logic such as else and else if statements were used in KEIL to change the direction of the motor as the axle was spun backward to the starting position to prevent the cables of the ToF sensor from tangling up.

In order to determine the beginning and conclusion of data measurements, the onboard button was implemented (push button PJ1). To start gathering data, the python code must be run in IDLE, after a prompt from the code, the push button must be pressed as soon as the first prompt is answered by the user. The if statement logic implements bit-masking as learned through the course and compares the value to a zero to create the active low push button. This configuration allows the user to decide when to begin the data collection process. This is a smart implementation because the x-axis displacement of 0.1m is manual and requires the user to physically move the system 0.1m after each run. This gives the user a great degree of flexibility when moving the system and setting up the PC and device before the next cycle.

As the distance is being measured, the microcontroller uses UART communication (through port COM3) to get the data to the personal computer where pyserial can then access the data. Each distance value is then printed in meters with the measurement number (x out of 64). Using numpy and matplotlib, a 3D render of the environment surrounding the device is plotted to visualize the collected data, this is explained in the visualize section below.

b) Visualization

After extracting the sensor data in meters to the personally computer, numpy was used to convert the distances into cartesian form (through basic trigonometric relationships), and matplotlib was used to create the 3D map of the environment.

The personal device used for this project is a Lenovo 81X2 laptop with Windows 10. The specs of the computer are as follows:

Figure 6: Personal Computer Specifications

Python 3.8.7 was already installed and is not the latest version as python3 has already been used for previous courses. Matplotlib and numpy have already been used so the installation for these libraries were not required. However, pyserial was a new library for the computer and needed installation.

The python code that creates the 3D plot starts with a prompt to the user that asks if the previous data should be cleared. In the case where the data needs to be cleared, then the respective text file that has the data is opened, read, and cleared. This is beneficial as the user has the freedom to start again or continue with the existing data from previous collections.

```
10
11 eraseData = input("Do you want to delete the data from the previous recordings? Please type 'Y' for Yes and 'N' for No.")
12 #if yes, then open to write and delete everything
13 if(eraseData == 'Y'):
14     file = open('graphdata.txt','w')
15     truncate
16     file.close()
```

Figure 7: First user prompt for data collection

The pyserial library is implemented to collect the data points from the UART USB port COM3 at a baud rate of 115200bps. A while loop reads all the measurements and then added to a .txt file to store data and read from later.

A for loop is used to determine the slice number using programming logic by taking the modulus of the counter variable that is measuring the amount of turns of the stepper motor. For example, if it is the 3rd slice, the counter value will be 64*3 = 192. So, if the counter modulus 64 is zero and the counter is larger or equal to 64 then it will plot the nth slice. The assignment of the 'y' and 'z' coordinated are determined through trigonometric relationships. Each distance measurement is multiplied by either the sine or cosine component of the angle. Python functions in radians so the angle was manually converted to radians, hence the value of 0.09817477 present throughout the code. After each data point has a cartesian equivalent, the point was plotted. To see each slice of the map, a line connects the points on each axis through a while loop that uses the same modulus arithmetic as described above. It will draw a line for each slice if the counter modulus 64 is zero and the counter is larger than or equal to 64.

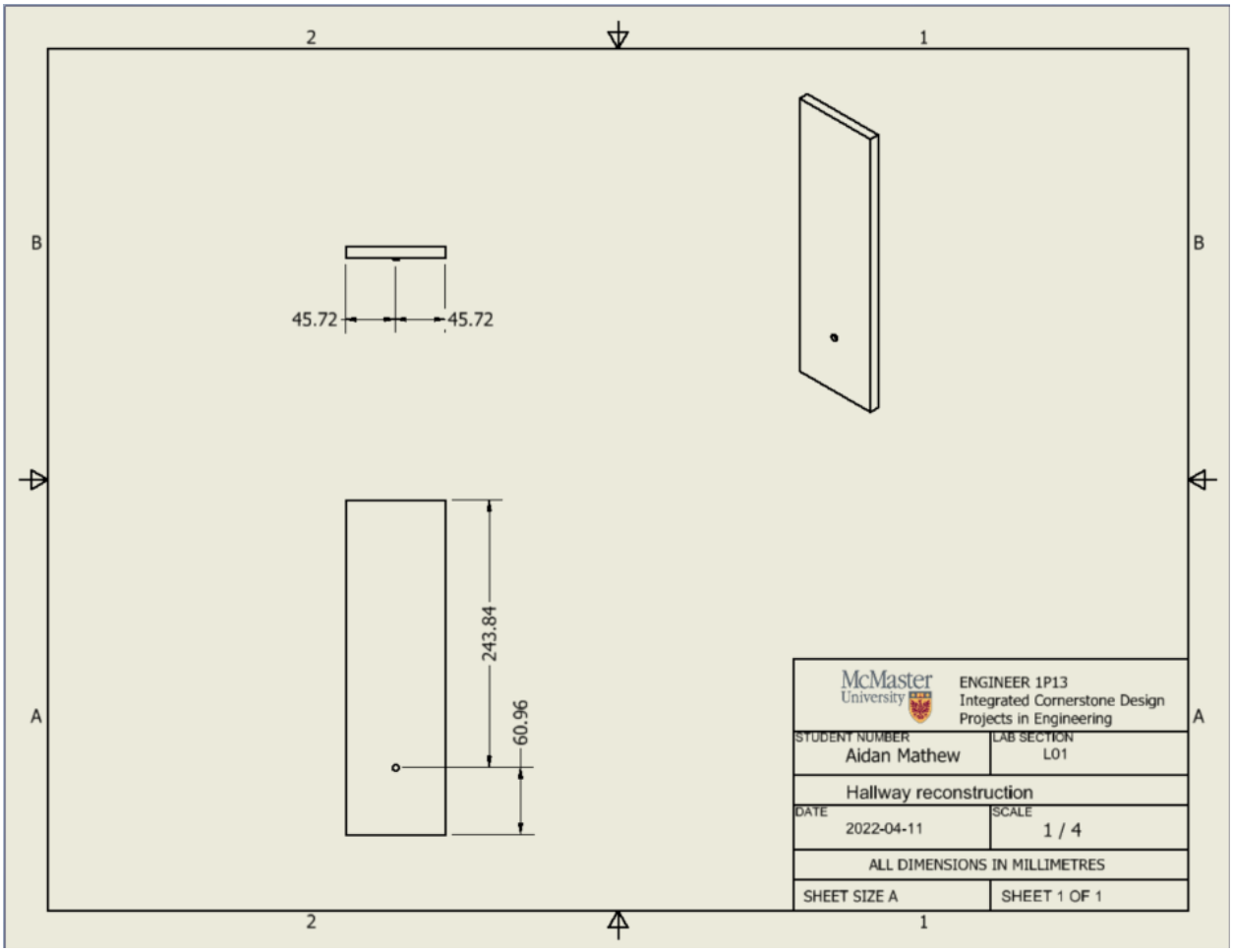

Figure 8: Isometric and Multiview sketch that depicts the hallway that was used to gather data.

As seen by the figure above, the device was placed on a chair (represented by the circle) that was relatively low to the ground. In hindsight, this indicates the irregular shape of the graph. If this were to be done again, try to keep the sensor in the middle of the room.

Application Example with Expected Output

1. Configure the microcontroller to the personal computer by connecting the microcontroller via USB to the first UBS port on the righthand side of the Lenovo 81X2. The ON status LED should be on, displaying successful pairing with the personal computer.
2. Open the KEIL uVision5 project and ensure that the systick and psysdiv values are correct based on the student number. Also view the main 2dx_studio_8c.c file and ensure all port definitions and main functions are declared properly with no errors.
3. Transfer the project onto the microcontroller by pressing translate, build, and load (keep in mind the order matters). If this step is successful, KEIL will tell you in the output terminal that the program load is successful. After a successful load, make sure to press the reset push button on the microcontroller to remove any other project that could previously be on the device.
4. Now that the microcontroller has been initialized and configured correctly, it is ready to gather data. Open IDLE (python3.8.7) and open the python file in the final project folder and run the code.
5. The user will not be prompted to either keep the previous data collection or delete the previous data. Input either ‘Y’ or ‘N’ depending on the stage of the data collection process.
6. Now the system is ready to measure its surroundings. When the device is in the correct location, press the onboard push button located on the side of the board. The data collection process will begin, and the sensor will gather a distance measurement every 8 motor steps which correspond to 5.625 degrees. For each collection of data, the corresponding LED PF4 for LED 3 will flash, indicating to the user that data is being collected.
7. The program will then prompt another input from the user, asking if the collected data should be plotted. To quickly map everything at the end, the user can say ‘N’ for no, and then wait to generate the plot at the end of 10 runs. In the case where the user would simply just like to see what one slice looks like, they can press ‘Y’ to see the mapped area.
8. Repeat steps 4-8 10 times, moving the device 0.1m forward after each slice, this is required since the program is hardcoded and assumes that after each slice, the device is displaced 0.1m forward in the x-direction.
9. After repeating the previous steps for the required number of slices, the final graph for 1m with 10*0.1m slices can be generated for viewing.

One key concept that should be understood is the depiction of each axis on matplotlib. The sensor in this intelligent microsystem only could measure in one plane as it collects a distance measurement which can be broken up into 2 components, sine, and cos. In this case, the 2 components were chosen to be the ‘y’ and ‘z’ axes. The x-axis was hardcoded due to the nature of the device and is fixed from 0m to 1m incrementing by 0.1m for each slice. The x-axis represents the user physically moving the system forward 0.1m after each scan. Refer to figure 9 to better understand the output map.

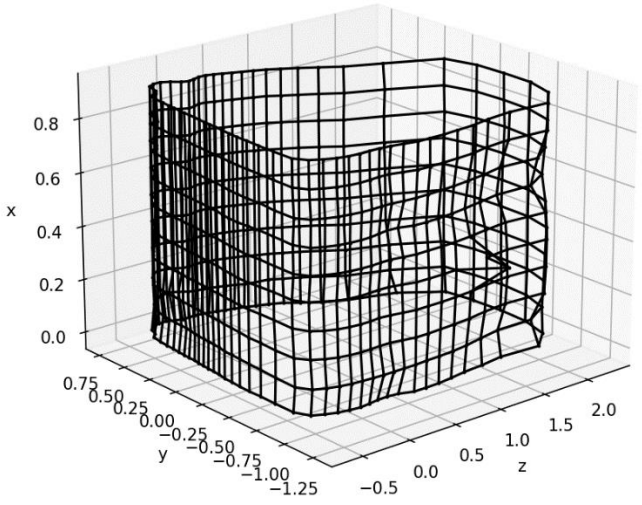

Figure 9: Final hallway reconstruction complete with 10 slices, 640 data points.

Brief Users Guide

This users guide will provide all the necessary steps that are required to configure this intelligent microsystem. These steps are not fixed and can change based on the users operating system and personal computer.

* Please note that these steps were specific to the Lenovo 81X2 that is mentioned in figure 6 *

1. KEIL Installation
    a. Download KEIL as it is used to program on the microcontroller. Ensure to download MDK Version 5 (keil.com). Follow the installation procedure and choose the MSP432E401 device when prompted by the Pack Installer. Proceed to drop down the "device specific" section to download and install the required pack for the board.
2. Python Installation
    a. Download Python 3.8.7 or newer from Download Python | Python.org. In the case that the code does not run, please use the version mentioned, however python3 is usually friendly and should not create any problems. During installation ensure that python3 is being added to the computer path for successful usage.
    b. After python has been installed it is required to download the required libraries for the project. Run the following commands in the command prompt individually to download the specific libraries.
        i. pip install pyserial
        ii. python -m pip install -U numpy
        iii. python -m pip install matplotlib

3. UART
    a. Now to enable serial communication between the board and the personal computer, identify the name of the physical USB port that connects the microcontroller to the PC. This can be done by doing to device manager, open the ports tab, and identify the port number that matches with XDS110 USER UART. It will have a value COMX where X is an integer.
    b. After noting down the communication port number ensure to go to python code and change what is currently set to “COM3” to the respective number. This is a vital step as it allows the communication between the microcontroller and the python files.

If all the initialization steps are complete, then the system should produce a map assuming all components are functional. Refer to figure 9 for a complete map.

Limitations

1. FPU
    a. The microcontroller employs a 32-bit floating-point unit which is capable of basic arithmetic. The trig functions that were invoked in python use a double point format which is 64 bits, therefore requiring more work for the FPU, limiting the microcontroller.
2. Maximum Quantization Error
    a. The max quantization error is defined by = $\frac{full\ scale\ distance}{2^{number\ of\ bits}}$. The number of bits = 16. Therefore, the error = $\frac{4m}{2^{16}} = 6.1 * 10^{-5}m$
3. Maximum serial communication rate (Unique to PC)
    a. For the specific device mentioned in figure 6, the maximum serial communication rate is 128Kbps.
4. I2C Limitations
    a. The means of data transmission between the microcontroller and the sensor follows I2C. The speed for this transmission is 100kbps, which is determined by the clock speed for the respective ports (100KHz).
    b. I2C also needs more space to function since the resistors take up valuable space.
    c. Slower communication speed as pull up resistors are used instead of push-pull resistors.
5. Stepper Motor / Sensor Limitations
    a. Both the stepper motor and the ToF sensos have set limits for the maximum speed. In the case of the stepper, if the delay is small, then the motor will not spin, but the system will gather data from a fixed point. If the sensor readings are delayed, then the entire device ceases to function.

Circuit Schematic

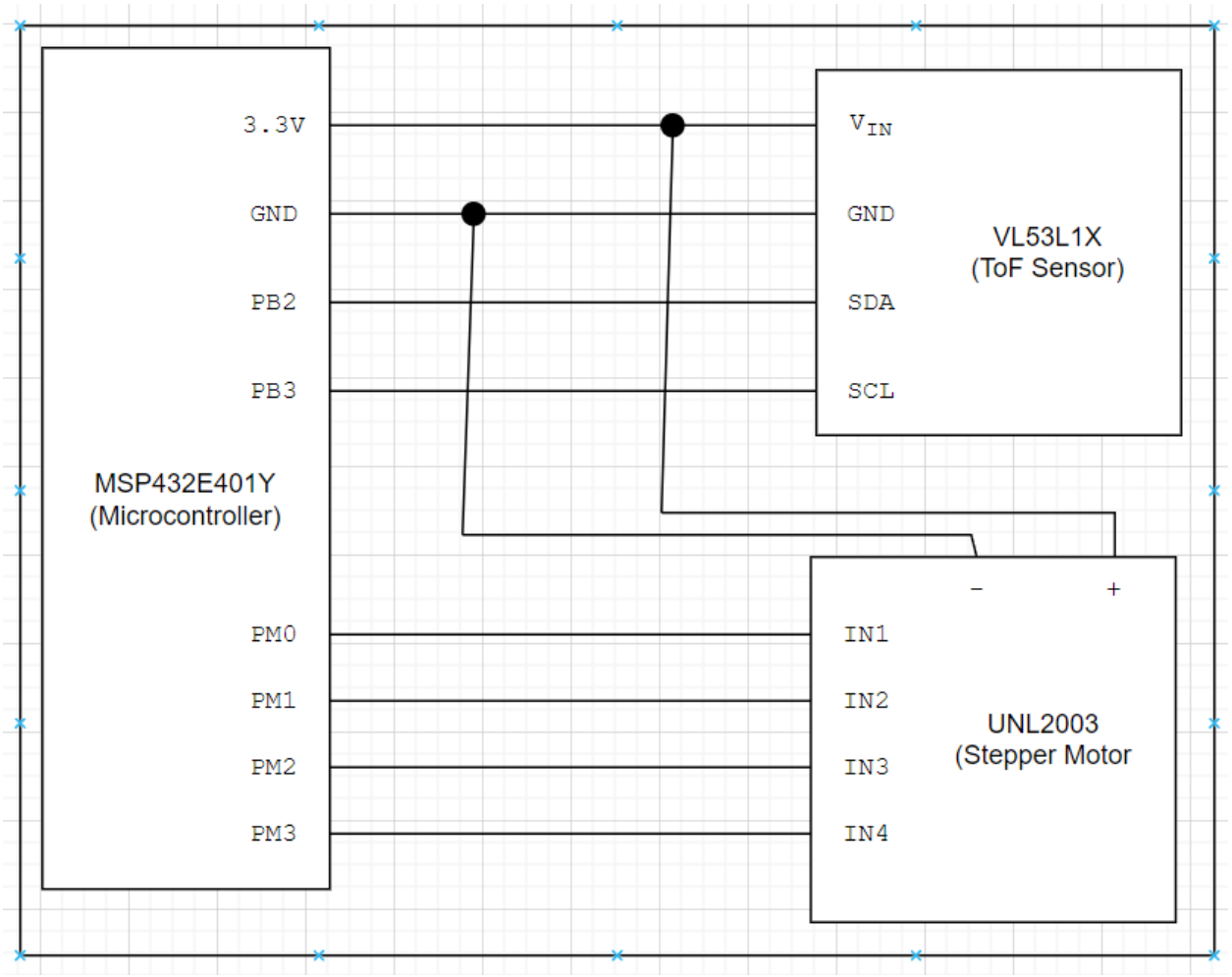

Figure 10: Circuit Schematic depicting pin connections.

Programming Logic Flowcharts

a) Microcontroller Flowchart

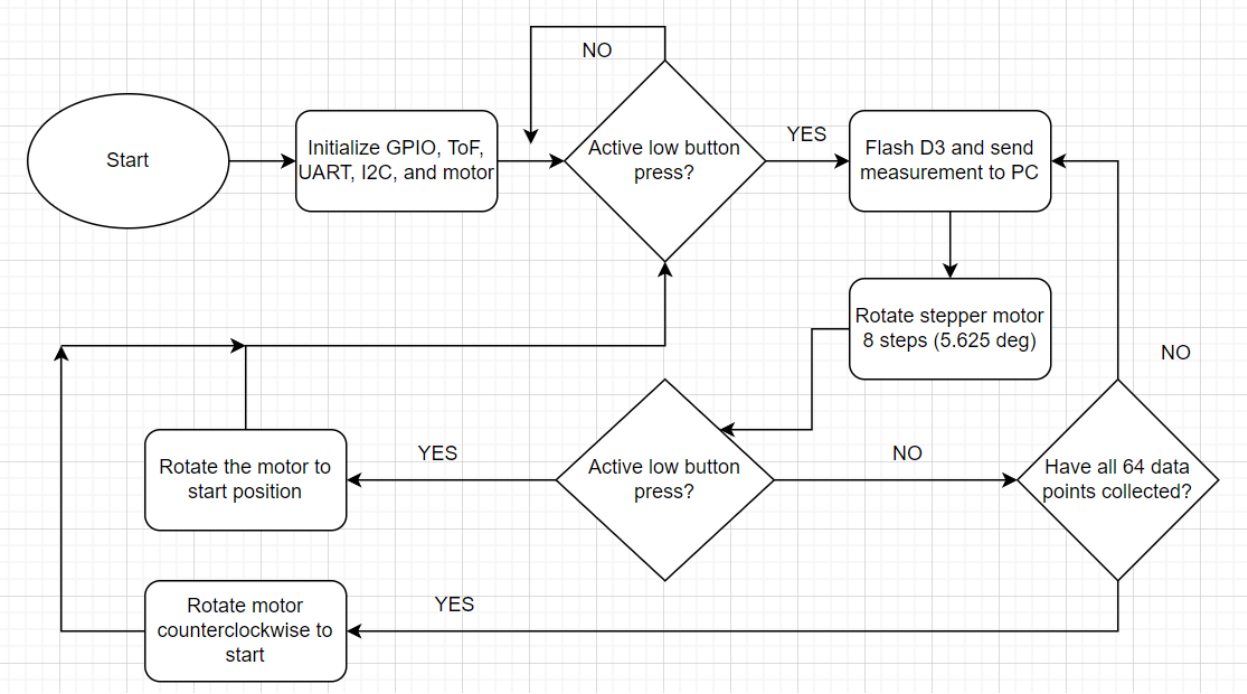

Figure 11: Microcontroller Flowchart

b) Python Code Flowchart

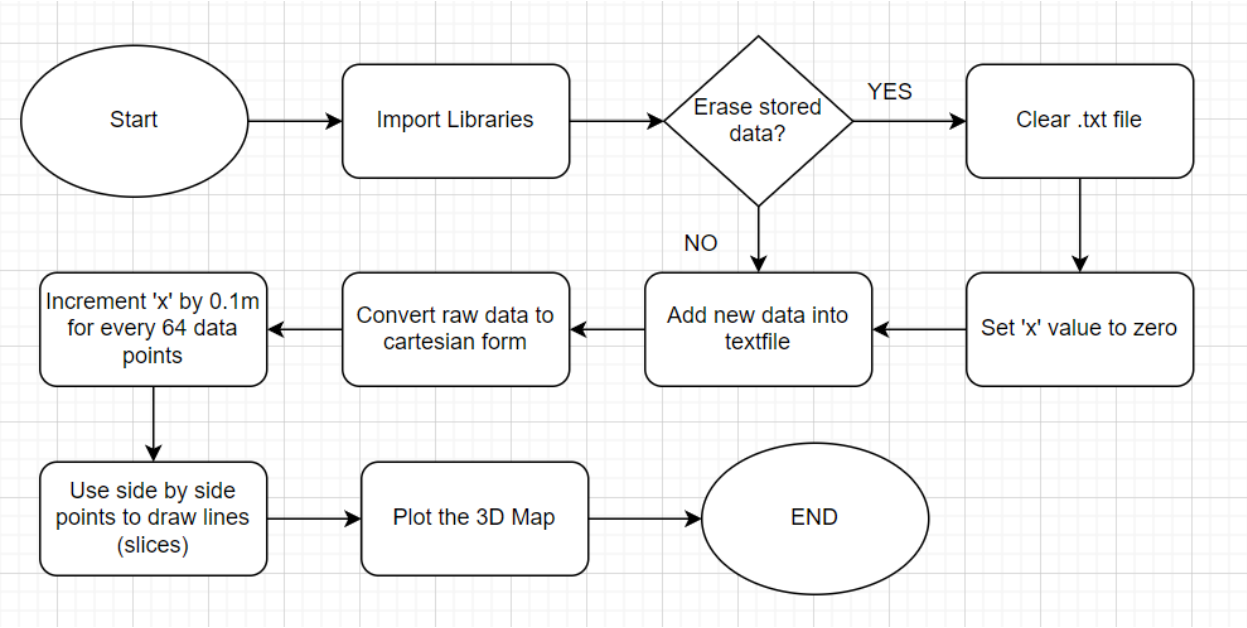

Figure 12: Flowchart for python code