Microprocessor Design Project : COMPENG 2DX3

Final Project Report

Dr. Haddara/Dr. Athar/Dr. Doyle

By: Gurleen Kaur Rahi

rahig

400377038

Device Overview

1.1 Features

- It scans a 3D space to rebuild the space's spatial layout.
- Built around the MSP432E401Y microcontroller and integrated utilising embedded systems.
- Operates at approximately 5 volts at a 50 MHz bus speed.
- Relatively smaller, less expensive parts are used, such as the 75 CAD 3415-POLOLU VL53L1X time-offlight and MOT-28BYJ48 stepper motor.
- Two 12-bit ADC modules of 2Msps
- VREFP: 5 V
- VREFN: 0 V
- Resolution: 12
- possesses a 2-button interface that is almost convenient to plug and use.
- Creates 3D visualization model with intuitive navigation.
- Uses UART at 115200 bps and I2C for serial connection.

1.2 General Description

The Final Project Lidar system is an integrated 3D scanning system capable of capturing distances in numerous 360º planes along an orthogonal axis and processing the recorded data to provide a 3D visualisation of the mapped region. A microcontroller, a stepper motor, and a time-of-flight sensor make up the system. The microcontroller is in charge of controlling all system activities, except 3D visualisation, as well as powering other system components and sending all recorded data via serial communication to an external device.

The installed time-of flight sensor can record distance measurements in a complete vertical plane because to the stepper motor's 360º range of motion. By measuring the amount of time it takes for laser pulses to reflect back to a detector after being emitted, the VL53L1X ToF sensor measures distances to objects. Based on its setup parameters, the sensor uses this timing data to determine the distance and sends it over I2C to the microcontroller.

The SimpleLink MSP-EXP432E401Y LaunchPad Development Kit's microcontroller, model MSP432E401Y, is responsible for controlling both the stepper motor and the time-of-flight sensor. The microcontroller's ARM Cortex-M4F processor is programmed to operate at a bus speed of 30 MHz. It receives 5 volts from the micro-USB cable linked to the PC, which powers it. It has 256 kilobytes of static random-access memory and 1024 kilobytes of flash memory. It

supports the serial communication techniques used in this project, UART and I2C using C++ programming.
The machine launches the provided Python scripts while connected to a PC. The microcontroller sends distance measurements and status signals to the PC via UART. An associated Python script is then used to visualise the data.

Block Diagram

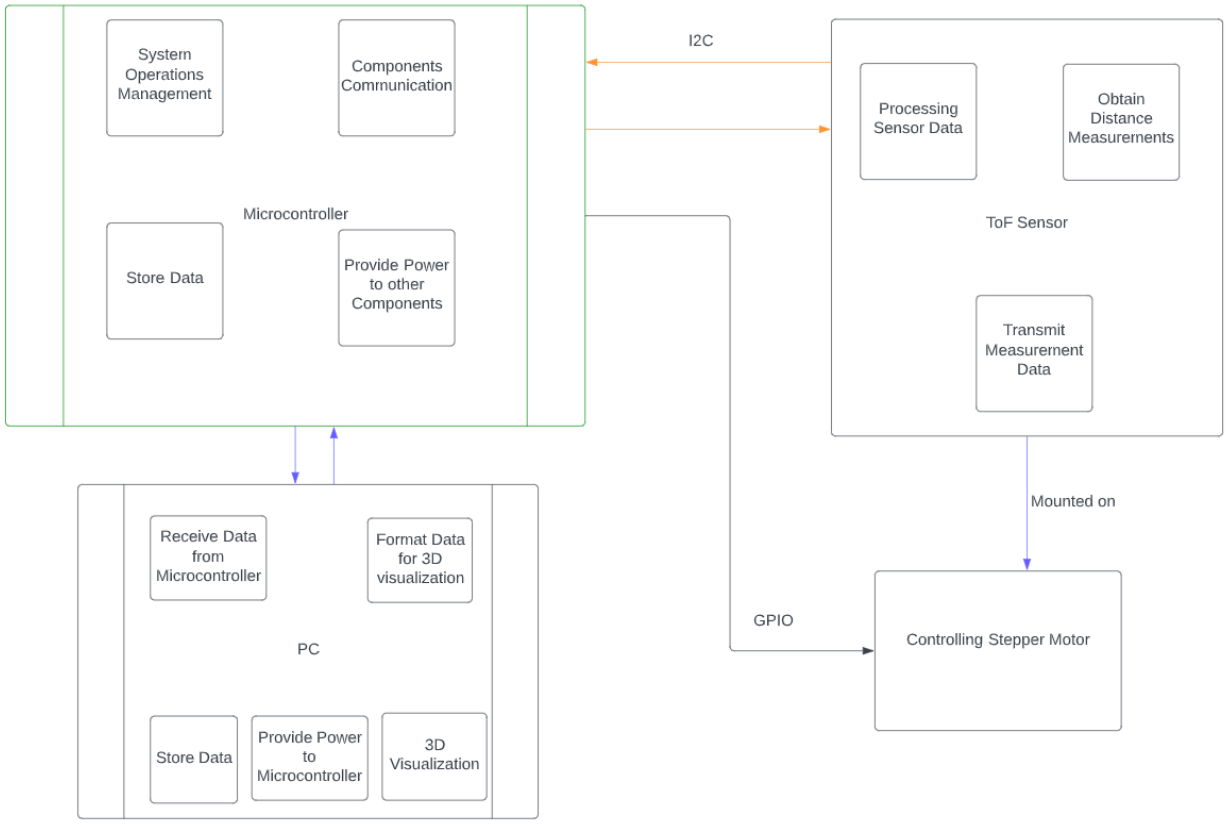

Figure 1: Block diagram

Device Characteristics

Microcontroller		Stepper Motor		ToF Sensor	
Feature	Details	Device Pin	Microcontroller Pin	Device Pin	Microcontroller Pin
Bus Speed	50 MHz	VDD	5 V	VDD	
Serial port	COM4	GND	GND	VIN	3.3 V
Distance	PN0	A	PH0	GND	GND
Displacement	PL1	B	PH1	SDA	PB3
Baud rate	115200	C	PH2	SCL	PB2
		D	PH3	XSHUT	
				GPIO	

Table 1: Device Characteristics

Detailed Description

Distance Measurement

The ToF sensor produces short bursts of infrared laser light, which are reflected off nearby objects. The sensor gauges the amount of time it takes for a nearby item to reflect the emitted infrared laser light pulses onto one of its detectors. Based on its setup settings, the sensor uses the timing data it has collected to determine distance. It then delivers this information to the microcontroller using the I2C serial communication protocol.

There are 4 connections on the ToF sensor. On the microcontroller, VIN and GND are linked to ground and 3.3 volts, respectively. Additionally, SDA and SCL pins are attached to PB2 and PB3 on the microcontroller, respectively, to provide an I2C serial connection in order to communicate with the sensor. The included VL53L1X API library is used to configure and operate the sensor. The given SensorInit function is used to configure the sensor, and the StartRanging method is used to measure the distance. The GetDistance function is then used to get the measured distance.

In addition, the sensor is precisely rotated clockwise between each measurement using a stepper motor. Its IN1, IN2, IN3, and IN4 pins are connected to the microcontroller's PH0, PH1, PH2, and PH3 ports, respectively. Additionally, the microcontroller's plus and minus pins are connected to ground and 5 volts, respectively. The four electromagnets inside the stepper motor are switched around in polarity to produce motion. The shaft between the magnets can revolve in both rotations by switching the polarities in the right order. Full step is a technique for changing polarities in which two adjacent magnets are always energised.

To give the shaft adequate time to rotate between each state change (step), the algorithm pauses for 5 ms. The system will take 32 measurements for each 360-degree set because the resolution is set to 32. The sensor will therefore spin 11.25 degrees after each data gathering thanks to the stepper motor. The "number of measurements" value in formula 1 is used to calculate this value, which determines how many measurements will be in a set.

Step = 360 / number of measurements

The programming language that we used to control the microcontroller is C. Phase-Locked Loop (PLL) is set for the bus speed, SysTick is set for waiting, onboard LEDs are set for visual output, I2C is configured for sensor connection, UART is set for PC communication, and GPIO ports are set for servo controls and button inputs. All necessary functionality is then initialised by the code. The ToF sensor is then initialised and set up. When the sensor is prepared, it communicates the word start over UART and starts an endless loop.

The code initially determines whether the button connected to the PH0 is depressed inside the endless loop. If pressed, the data collecting process will start. All of the onboard LEDs flash to signal the commencement, followed by a 40 ms delay. After that, a directive to begin measuring the range is sent to the time-of-flight sensor. The programme checks in with the availability of the sensor data every 5 ms. When the information is ready, it uses I2C to get the distance information from the sensor and temporarily puts it in onboard memory. Following the rotation, the distance information is sent to the PC via UART, as described in the following section. When the transmission is finished, it waits 100 ms before flashing the internal LED attached to the PN1 port. The button PH0 is then once more examined to see if it has been depressed to halt the operation. If it is, the operation is stopped, and the programme loops indefinitely. Depending on the number of measurements listed in the code as resolution, each of the aforementioned steps starting with waiting for data availability is repeated.

Instead, the stepper motor rotates 360º counterclockwise if the button PH1 is pressed.
If the PH1 button is touched once more during this procedure, the operation will end and the programme will go back into the infinite loop. The system uses an active-high setup for both buttons. Each time a button is pressed to end a process, software debounces it. To avoid unintentionally registering repeated inputs, the programme will wait for the button to be released after it has been pushed.

Transmission

The ToF sensor measures distance in the YZ plane and transmits that information to the computer using UART serial connection. The microcontroller is set up so that UART is preferred

over the micro-USB for power and code flashing. UART is set up to use an 8-bit word length, a baudrate of 115200, no parity bits, one stop bit, and FIFOs. The interface between the microcontroller and the personal computer is created using Python collector code. Since the ToF sensor status during startup is also sent by the VL53L1X library internally using UART, the collector must be informed when the actual distance 7 data is about to be transmitted. When the setup code has properly completed, the microcontroller transmits the word "start," and the collector waits until it reads that word before continuing.

The data collecting logic begins to run as soon as the word "start" is received by the collector code over UART. The microcontroller ends each line of data it transmits with a carriage return (‘/r’) and a line feed (‘/n’) character. In this manner, the receiver may quickly determine when all the data for the most recent distance measurement has been transmitted. The collector initially converts each line of input into an integer before dividing each line's received distance information into its Y and Z components in accordance with formulas 2 and 3. Therefore, the value that is then shown is the “distance”. How many times the system will move down the X axis and take a complete set of measurements depends on the "number of sets" setting. How many measurements will be in a set is determined by the "number of measurements" value. The "measurement counter" parameter also maintains track of how many measurements are currently in the set.

Y = (distance/number of sets)*sin((2*pi*measurement counter)/number of measurements)
Z = (distance/number of sets)*cos((2*pi*measurement counter)/number of measurements)

Visualization of Data

The visualizer code reads the XYZ data that the collector code has generated. The programme transforms the XYZ data into a point cloud object using the Open3D library. Even while it is possible to display this data directly as dots in 3D space, the final product is difficult to understand. As a result, a line set is produced by joining each point to its neighbouring points. Both internal connection lines within the set and external connecting lines between the sets are included in this set. A boundary box for each set is also added to the final product and connected to one another in order to enhance the visualisation of the 3D space even further. This establishes a frame of reference for the 3D reconstruction and facilitates comprehension of the outcome. The model can be rotated and zoomed using the user interface designed by the Open3D framework.

Application

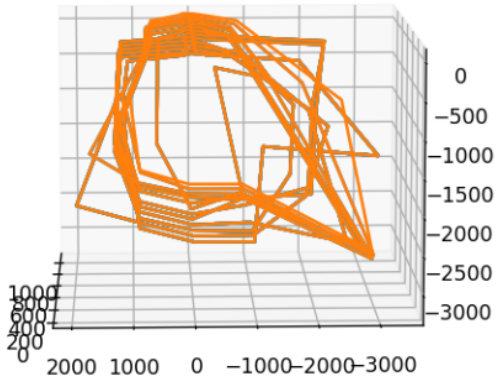

Figure 2: 3D scan of a hallway in JHE

The above figure shows the 3D visualization of JHE hallway. The elongated area on the right side is the elongated area of the hallway as seen in the left side of the second picture below. The code was run 12 times so the scan in the picture above is the scan of the same area that was done 12 times.

Figure 3: Actual images of the hallway in JHE

As shown in the picture below, the device was held in this position while scanning the hallway. This helped us to keep the sensor stable and get the best view of the hallway. The time-of-flight sensor's top is pointed forward. The xaxis is then situated along the sensor's extension. Additionally, the reconstruction's height and breadth are indicated by the Y and Z axes.

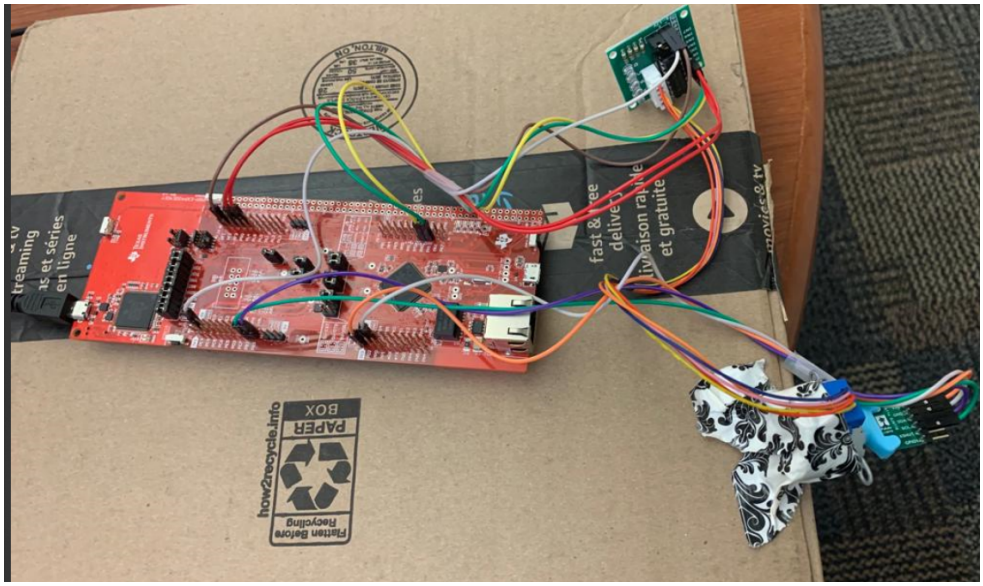

Figure 4: Design set-up

Steps to Set-up the device

1. Installing Python 3.10.10 on PC.
2. Install pyserial, open3d, and numpy as well.
3. Connect the Microcontroller to the PC and press the reset button.
4. Edit the serial port to add the address that is assigned by your computer in the form of COMK where K is a number.
5. Run the Keil code and press the reset button on the Microcontroller.
6. Then, run the Python code and wait for the “Start” command to show up on the screen.
7. Once it showed up, press the two buttons on the side of the microcontroller.
8. This will start to spin the motor and the ToF sensor that is attached on it.
9. Make sure that the sensor is able to access the surroundings because if the sensor is covered, it won’t be able to record the distance values.
10. Move the system a step forward (approx 20 cm) after every revolution.
11. Once the sensor has spinned the number of times that is mentioned in the code, it will stop spinning.
12. Observe the 3D visualization that will be created and automatically displayed on the screen by the system.

Limitations

1. The Open3D library that was used in the Python Collector code is not compatible on other Python versions except Python 3.10.10.

2. Since Python float values are stored as 64-bit double-precision numbers, the highest value that can be calculated given a measurement is roughly $1.76*10^{308}$.
3. The only serial communication protocol that can interface the ToF sensor is I2C, which has approximately 395 kHz of bandwidth restriction.
4. The maximum range of the ToF sensor at a maximum frequency of 50 Hz is 4 metres.
5. The stepper motor takes a lot of time to spin which is the slowest part of this project. However, we can adjust the bus speed according to the lowest significant digit of our student number.
6. The standard maximum Baud Rate of UART is 115200 bps. This is used by the serial communication with the PC.

Circuit Schematic

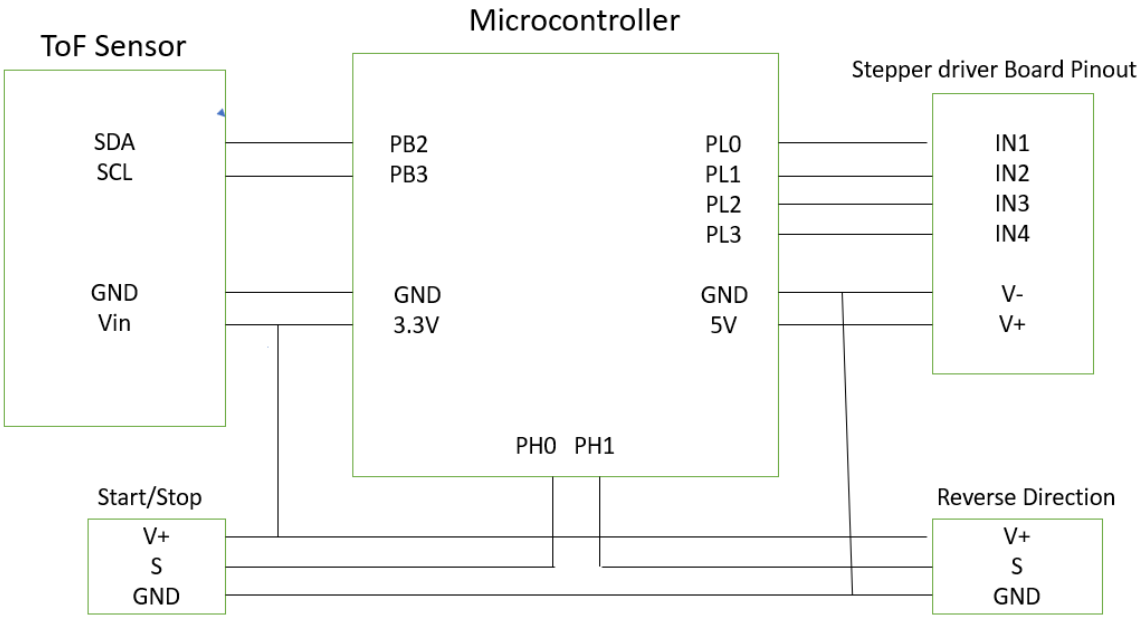

Figure 5: Circuit Schematic

Programming Logic Flowcharts

Microcontroller Code

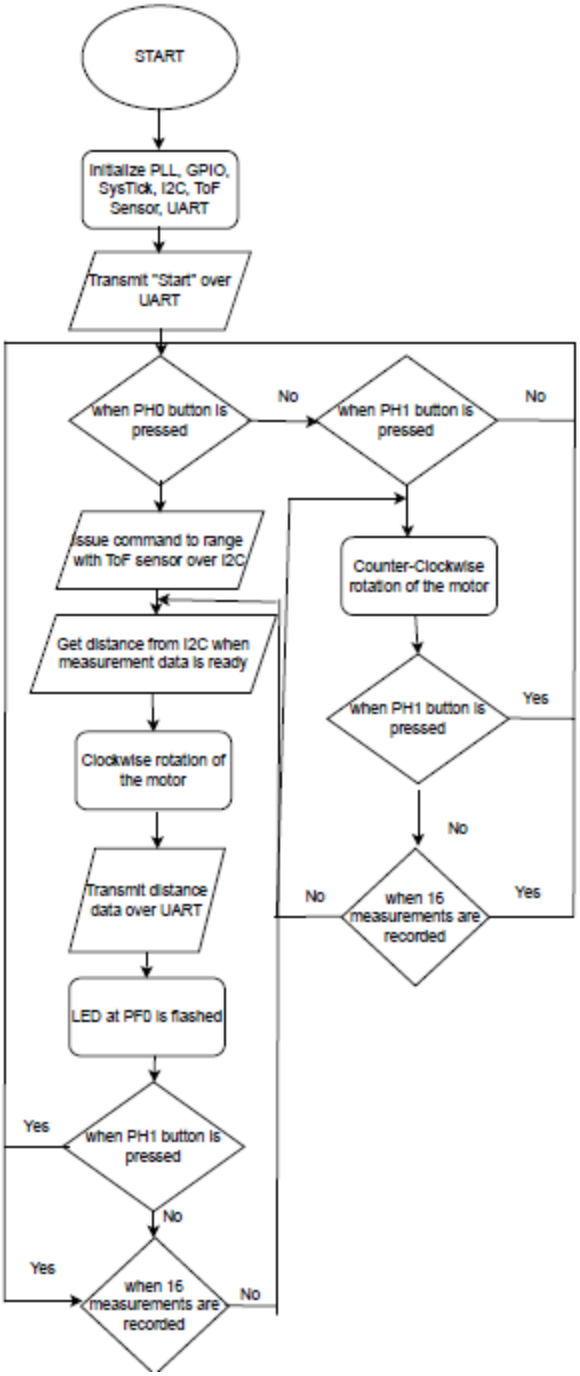

Figure 6: Flowchart for Microcontroller code

PC - Code Collector

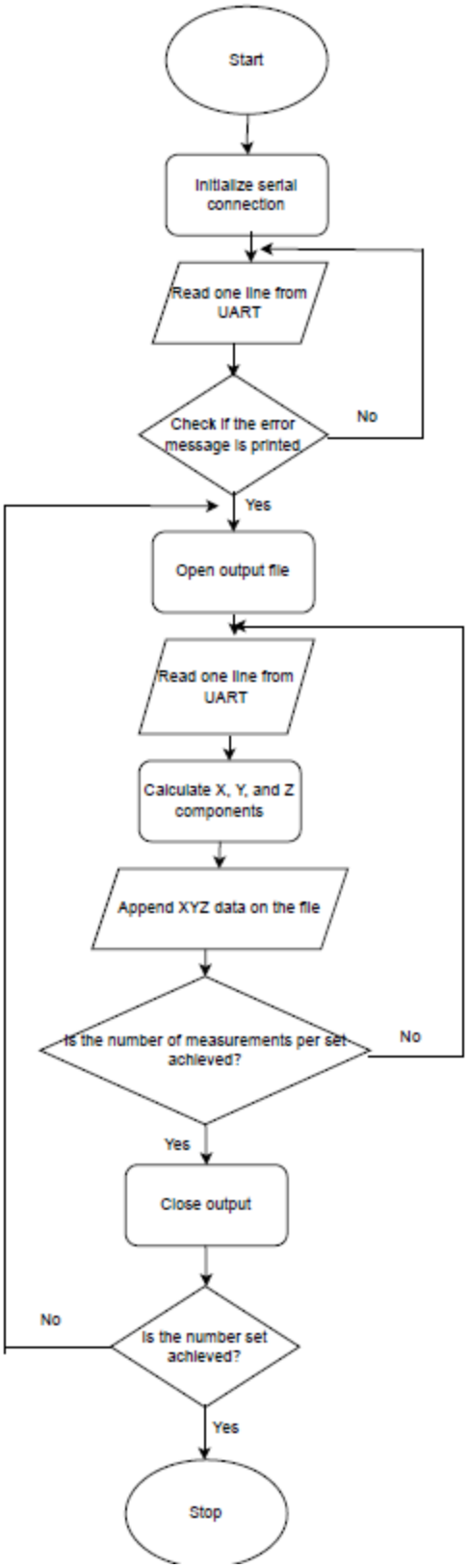

Figure 7: Flowchart for Microcontroller code collector

PC Code - Visualizer

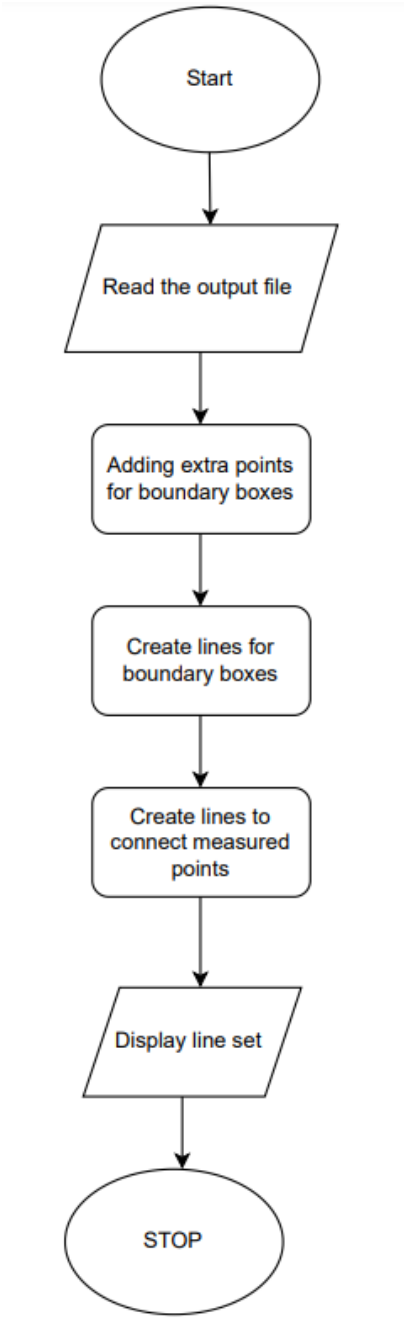

Figure 8: Flowchart for code visualizer

References

[1] *Datasheet - McMaster University*. [Online]. Available: https://avenue.cllmcmaster.ca/d2l/le/content/512368/viewContent/3979008/View. [Accessed: 17-Apr-2023].

[2] *Technical Reference Manual - McMaster University*. [Online]. Available: https://avenue.cllmcmaster.ca/d2l/le/content/512368/viewContent/3979010/View. [Accessed: 17-Apr-2023].