2DX4: Microprocessor Systems Project Final Project

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

Ruiyan Guo-L09-guor28

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 **[Ruiyan Guo,guor28, 400256752]**

Google Link to Video Demo:
https://drive.google.com/file/d/1w3HK0Qt3bUz2cMHRxB9awobsUf344Dqa/view?usp=sharing

Device Overview

(a). Features

This device is designed as an embedded spatial measurement system that can be used to acquire information from the around environment. The rotatory mechanism together with the Time-of-Flight sensor can provide a 360 degree measurement of distance within a single geometric plane(Y-Z plane) each time, in conjunction with the x-axis displacement, it could generate a complete 3D mapping. All the measured spatial information is stored in the device's memory and can be communicated to the computer for data reconstruction and graphical visualization.

In the market, Commercial Light Detection and Ranging (LIDAR) equipment is extremely expensive and difficult to carry. This device provides people a fast and convenient solution to conducting 3D mapping. And at the same time, it is cheap, approximately 180 CDA.

The device itself could conduct measurements within a single geometric plane(Y-Z Plane). Displacement data (x) is determined manually, it requires the user to push it along the axis to make the device conduct multiple vertical scans to generate a series of points cloud. Then data is transferred into the 3D rendering software (Open3D) to visualize the spatial measurements results. As a result, it can be used to conduct indoor measurements.

Microcontroller: MSP-EXP432E401Y

-Bus Speed:

Modified to be 48MHz (Max 120MHz)

-Memory:

1MB of Flash Memory With 4-Bank Configuration

256KB of SRAM With Single-Cycle Access

-ADC:

Two 12-Bit SAR-Based ADC Modules

Sampling rate up to 2 Million Samples per Second (2 Msps)

-Operating Language:

Programmed in C

Assembly Language(Optional)

Time of Flight Sensor: VL53L1X

-Operating Voltage: 3.3 volts

-Measuring Range: Max 4m

Serial Communication Details:

-Baud Rate: 115200 bps between microcontroller and PC

-Between ToF sensor and Microcontroller: I2C

-Between PC and Microcontroller:UART

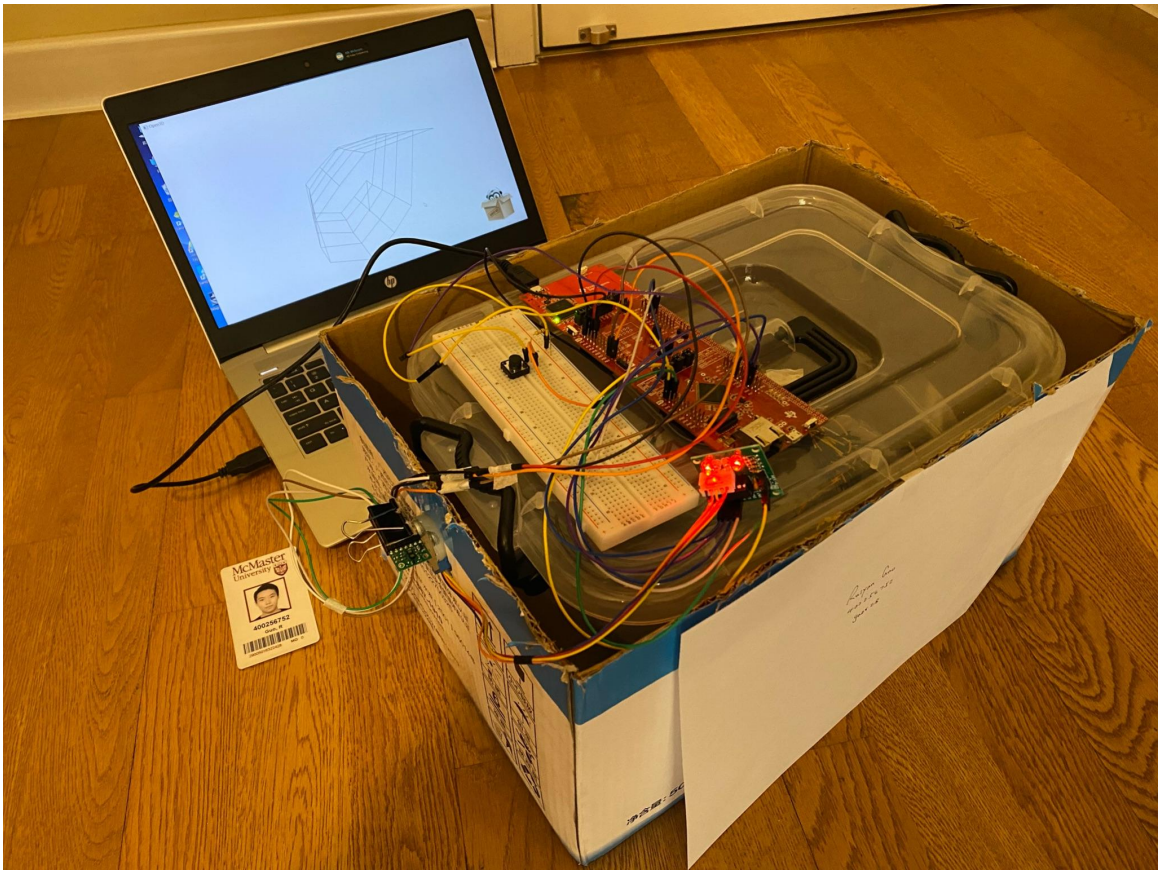
Visualization
-Related code programmed in Python
-Python version: 3.8.8
-Required Open3D to perform 3D mapping

(b). General Description
The system is composed of a microcontroller, a stepper motor, a Time-of Flight sensor and an external push button. The microcontroller is used to control each part of the system and transfer the data between the ToF sensor and the PC. The ToF sensor is used to measure the distance. The rotary motor is working in conjunction with the sensor to provide 360 degree measurements. The external push button is used to control the start/stop of the motor rotation and the sensor measurement, by pressing the button, the microcontroller also starts to transfer data to the connected computer with UART serial communication.

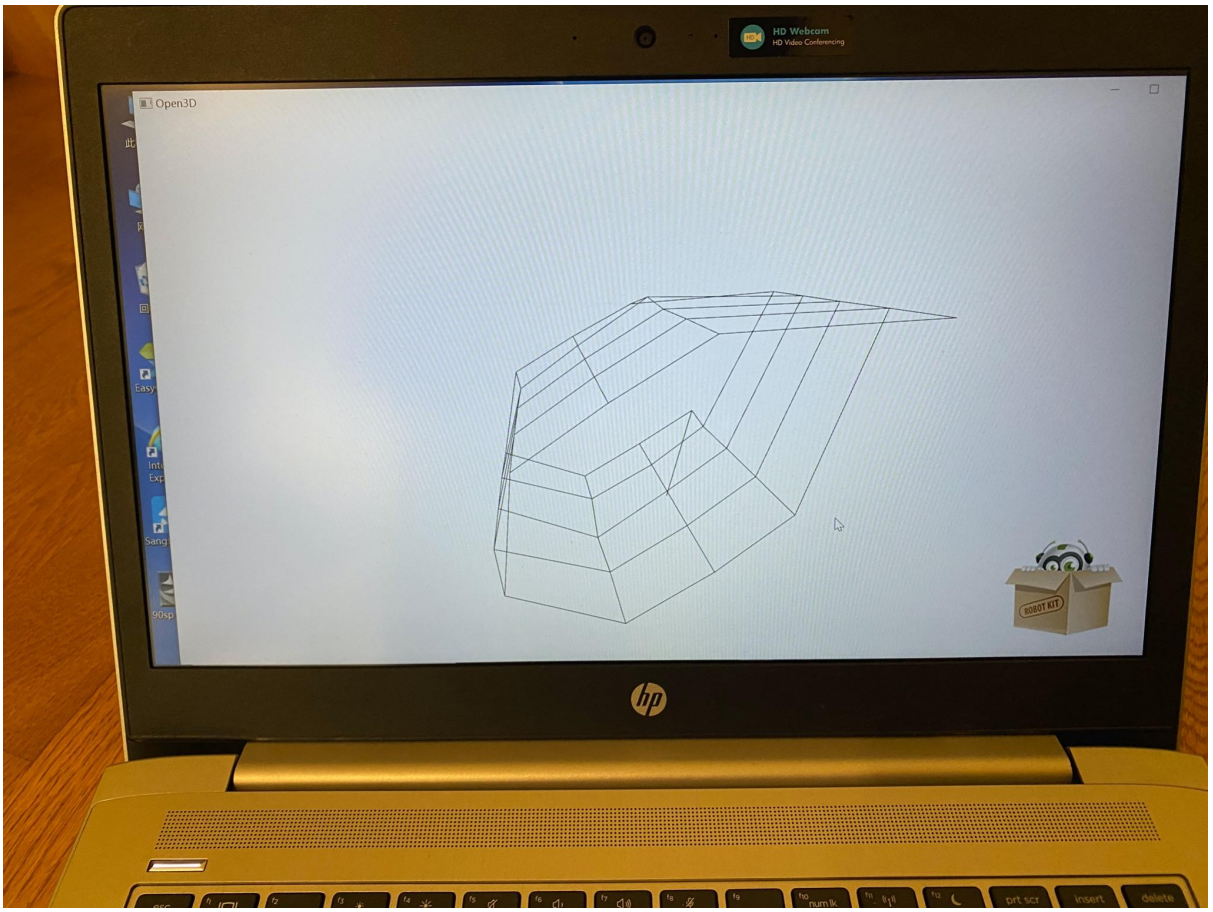
When taking the measurement, the ToF sensor emits a laser pulse. It works by calculating the time it takes for the laser to hit the target and reflect back, the speed of the pulse is equal to the speed of light. After the measurement, the sensor will return the distance of the target in millimeters.

The measurement from the ToF sensor is communicated serialy to the PC by the universal asynchronous receiver transmitter (UART). The python code on the PC side reads and decodes the data into an array for further processing.

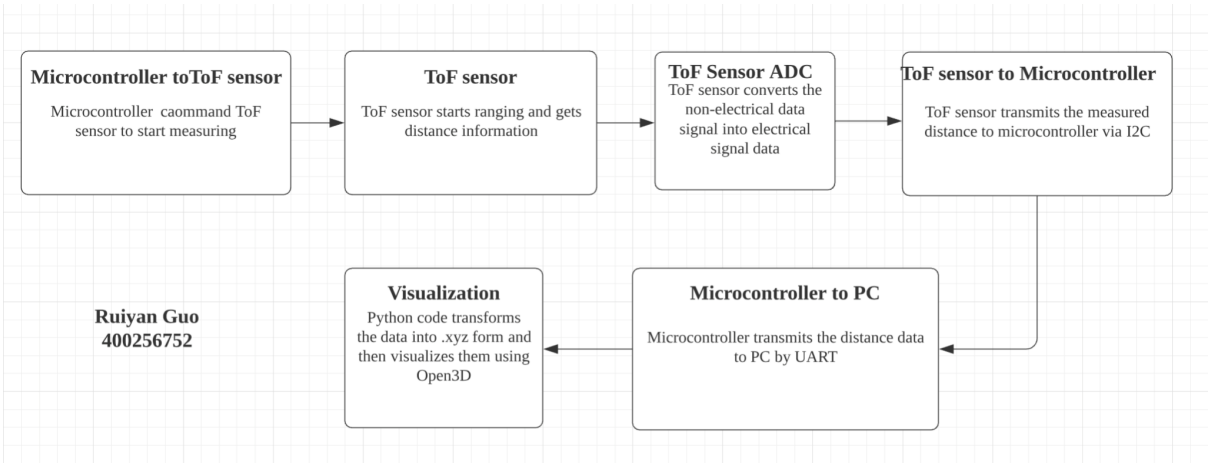
Picture of the entire Device



Picture of PC Display



(c). Block Diagram



Device Characteristic Table

Bus Speed	Modified to be 48MHz (Max 120MHz)
Serial Port	COM3
Baud Rate	115200 bps

Pins:

ULN 2003	**MSP432E401Y**
5v	5v
GND	GND
IN1	PH0
IN2	PH1
IN3	PH2
IN4	PH3

VL53L1X	**MSP432E401Y**
VDD	--
VIN	3v3
GND	GND
SDA	PB3
SCL	PB2
XSHUT	--
GPIO1	--

Detailed Description

(a). Distance Measurement

The VL53L1X ToF sensor uses a laser known as vertical cavity surface emitting laser (VCSEL) which emits a class 1 infrared laser of wavelength 940 nanometer (nm).The sensor works by calculating the time it takes for the laser to hit the target and reflect back. The distance is measured in millimeters (mm). There are three distance modes which could be measured: short with a max distance of 1.3m , medium with a max distance of 2.9m and long with a max distance of 4m.

The ranging measurements are communicated to the microcontroller through the sensor's I2C interface. On the microcontroller, it uses a series of predefined application programming interfaces (API) to both control the sensor and receive data from it.

The microcontroller communicates the measurements data to the PC by universal asynchronous receiver transmitter (UART). It is a serial communication method, the bits are transmitted serially one by one. The data transmitted by UART is in the form of a package. One package will send a data of 8 bits long (1 byte). Inside the microcontroller's code, after a single measurement is transmitted to the PC, it is programmed to transmit a letter 'A' to indicate the end of one transmission. On the PC side, the python code reads and decodes the transmitted data byte by byte. After each read, the decoded number is appended into a string type variable called 'str', until it reads a letter 'A'. After it detects the letter 'A', it means one complete measurement has been read, the code will append content inside the 'str' into an array for further processing, at the same time, the 'str' will be cleared to store the next measurement.

After all the measurements are being appended into the array, the python code will break them into the x, y, z plane with math functions and store them inside a .xyz format's file. As the measurements are taken in the vertical plane, it will be broken into y, z plane (Vertical plane is defined to be 'z', horizontal plane is defined to be 'y'), the displacement x is decided by the user. For the y/z plane conversions, the formula
y = (measurement/1000)*(math.sin(math.pi * angle / 180))
z = (measurement/1000)*(math.cos(math.pi * angle / 180))
are used respectively. It first converts measurement from millimeter to meter; converts the angle where the measurement was taken into radian form, then uses the converted measurement time the sin/cos of the radian. For example, if measurement=1000mm, angle=45deg, then y=1000/1000*sin(pi*45/180)=0.707m, z=1000/1000*cos(pi*45/180)=0.707m. Finally, the converted x, y values as well as the displacement x are written into a .xyz format file for further processing.

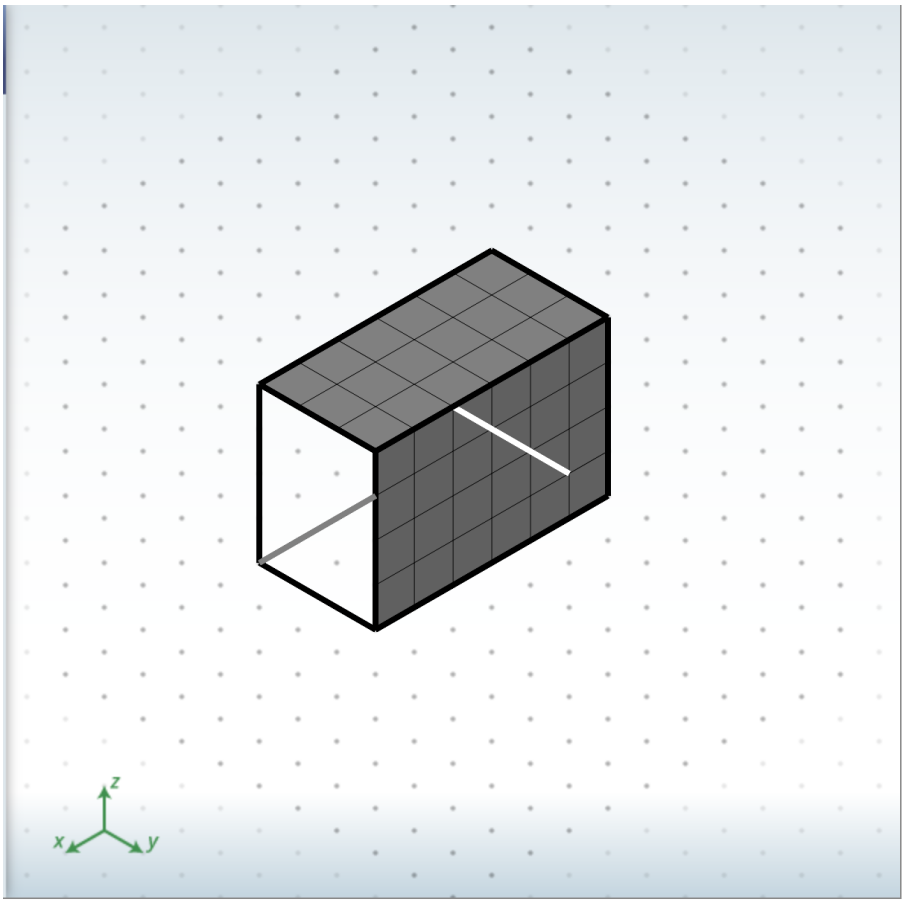
(b). Visualization

The visualization is performed on a HP ZHAN 66 Pro G1 laptop running Windows 10 Family Version 20H2, OS Build 19042.928. The laptop features a Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99GHz.

The visualization is performed by Python 3.8.8 with pyserial, Open3D, Numpy installed, and requires open source Python libraries.

The python code opens the .xyz file that was generated in the Distance Measurement part mentioned above and transforma all the coordinates into a point cloud with the function **read_point_cloud()** from the Open3D. Then the code proceeds to connect lines between vertices. Firstly, it will connect the vertices within the same plane. The code introduces a variable to mark the offset within the plane and an array to store the individual line, then store the lines into the array with function **lines.append()**. Secondly, the code will connect the vertices between planes. It will introduce another variable to mark the offset between planes, and then store the lines into the array with function **lines.append()**. Finally, the code creates a lineset with the function **LineSet()** and visualizes the connected points and lines with function **visualization.draw_geometries()**.

Rough Dimension Sketch



Application Example with Expected Output

The sensor was programmed to take 8 measurements per rotation(measure every 45 degrees). The python code is initially to read 10 scans in total.

Corresponding to the plane that the sensor is scanning, the y-axis is defined to be the axis perpendicular to the sensor chip when it is not rotating, z-axis is the axis perpendicular to the y-axis. x-axis is parallel to the path that you are moving the device.

In order to use this device, users need to follow the instructions below. Install the necessary software and connect the hardware correctly. And users need a PC that runs Windows 10, since this device has only been tested on this platform.

Instructions:
Steps:

1. Install the python from https://www.python.org/downloads/, search for version 3.8.8 and download the installer. When opening the installer, remember to select Add Python to PATH and hit download. After python has been installed on the PC, open the Command Prompt, type 'pip install pyserial' wait for the installation and type 'pip install open3d'. After that, type 'python -m serial.tools.list_ports –v', remember the port name that has the word UART next to it (port name example: COM3). Then, type in the next 2 statements in sequence 'import serial', 's=serial.Serial('COM3', 115200)', notice that: COM3 should be substituted by the port name you remembered earlier. Right now, your software has been set up successfully.
2. Go to your computer's search bar, search the word 'IDLE', if there is a match up called IDLE, then open it. Inside the opened python shell, click file and open the python file with the name '3-AllinOne_guor28'. On the Line11 and Line48, you can change "pointscloud.xyz" according to how you want to define the file name, but DO NOT change the .xyz to anything else. On Line 12, you may need to change the "COM3" to the port name you remembered from step 1.

   If you want to modify the value for x-displacement, go to Line37 and change 0.5 to any value you want.

   If you want to modify the numbers of scans you want to perform, then go to Line35. The code is right now programmed to take 10 scans, as a result there are in total 9 sets of OR conditions inside the if statement. If you want to take more scans, you may need to add more OR conditions in it, following the same pattern. The numbers of OR conditions you need to add is equal to (the number of scan you want to take - 9), and the number you need to add for the 'counter' is equal to (the number of scan you want to take*8+1).
3. Connect the microcontroller to the USB port on your PC

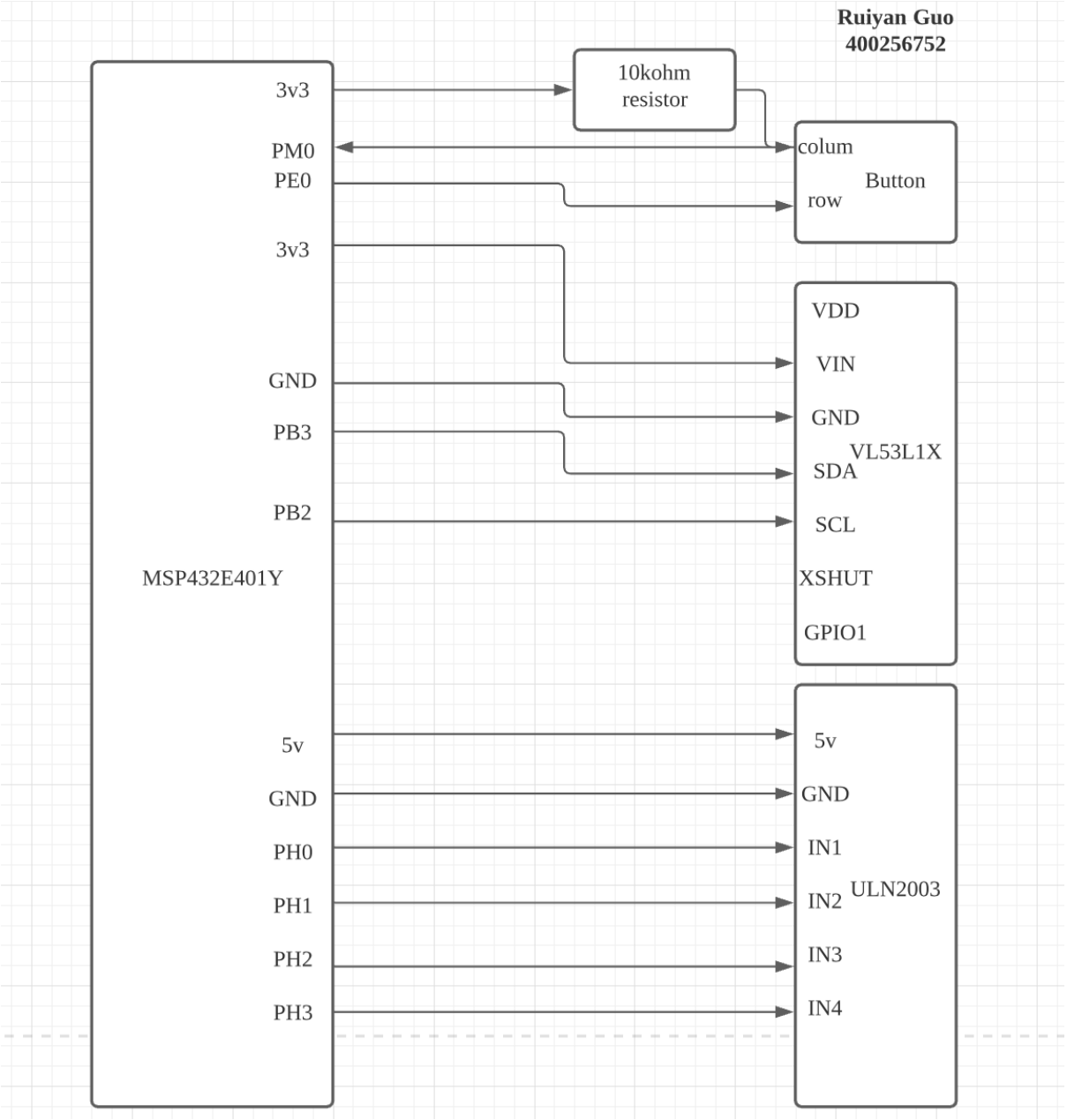
4. Then, you need to run ‘3-AllinOne_guor28’, wait for the python shell to print ‘start’.
5. When the word start is printed, you can hit the button that is on the breadboard to conduct the first scan. After the motor stops rotation, you can hit the button to conduct the next scan ……After the number of scans that you have modified in the ‘3-AllinOne_guor28’ is reached, the 3D plot that generated by Open3D will pop out on the screen of your PC automatically. Inside the python shell, you will see all the measured distances have been printed out. Also, the corresponding .xyz file would have been generated in the same place that you place the file ‘3-AllinOne_guor28’.
6. If you want to take the measurement again, you run the ‘3-AllinOne_guor28’ file, wait for the ‘start’ to be printed, and then hit the button that is located on the breadboard like before.

Limitation

(1). Summarize any limitations of the microcontroller floating point capability and use of trigonometric functions.
The MSP432E401Y has a Floating-Point Unit (FPU) that can support single-precision (32-bit) operations and calculations such as addition, subtraction, multiplication and division. The FPU can also perform conversions between fixed-point floating-point data formats, and floating-point constant instructions. Trigonometric functions from the math.h library can be applied on float variables. Because the FPU is only 32-bits wide, 64-bit operations will take more than 1 instruction to complete since the FPU will separate it into two 32-bit words.
(2) Calculate your maximum quantization error for each of the ToF module.
Maximum quantization error for ToF module is equal to 4000mm/(2^16) = 6.10*10^(-2) mm.
Maximum quantization error for IMU module is equal to 32g/(2^16) = 32*9.8/(2^16) = 4.79*10^(-3) m/s^2
(3) What is the maximum standard serial communication rate you can implement with the PC. How did you verify?
The maximum standard serial communication rate that can be implemented with the PC is 128000 bits per second. It can be verified by checking the port settings for the XDS110 UART Port in the PC's device manager.
(4) What were the communication method(s) and speed used between the microcontroller and the ToF modules?
The microcontroller and the ToF module uses the I2C protocol with the clock speed configured to 100KHz for a transfer rate of 100kbps.
(5) Reviewing the entire system, which element is the primary limitation on speed? How did you test this?
The ranging on ToF is the one that primarily limits the speed. The minimum time that allows 4m to be reached in the Long distance mode is 140ms. So in total there is almost 280 ms delay plus the time that the sensor calculates the time of flight. 280ms is a delay that is

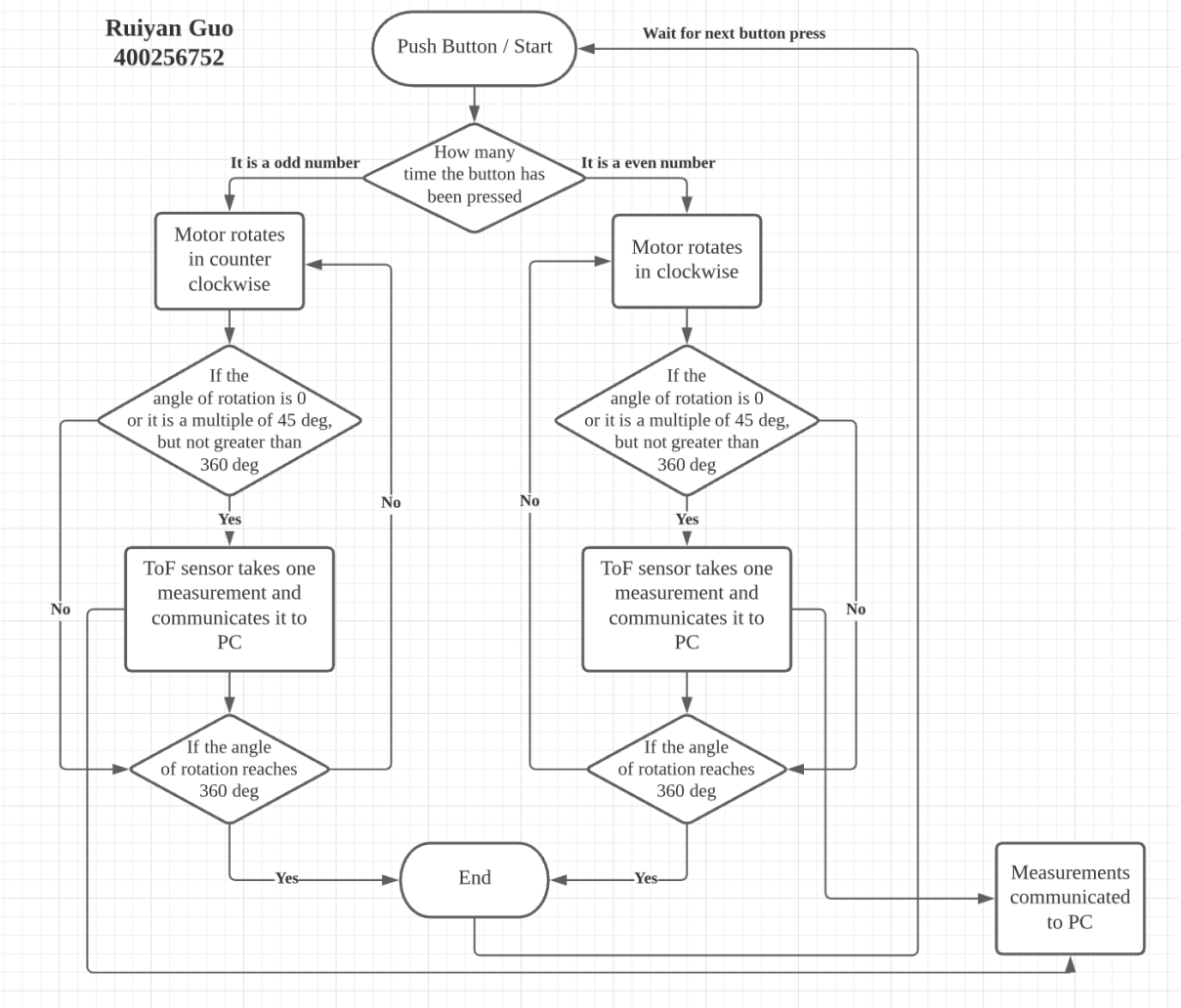
longer than any other components' delay in the entire system. It is verified, because when taking the measurement, the working of the ToF sensor makes the motor not rotate smoothly.

Circuit Schematic



Programming Logic Flowcharts

(1). Microcontroller Flowchart



(2) Python Code Flowchart