2DX4: Microprocessor Systems Project Final Project

Instructor: Dr.Doyle / Dr. Haddara / Dr. Shirani

Name: Zhaobo Wang

Lab Section: L05

Student Number: 400188525

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 **[Zhaobo Wang, wangz393, 400188525]**

Google Drive Link for Project Demo:

https://drive.google.com/file/d/10tldp9IBMyzrNW5TJBA9nR05SXHt_GoC/view?usp=sharing

Question 1:

https://drive.google.com/file/d/1YSanSzytDwIfBaF1VrVh4HFz0ZVuaenj/view?usp=sharing

Question 2:

https://drive.google.com/file/d/16lqD2Wfj7oysl9pkdNySXYqsyLlEq7l5/view?usp=sharing

Question 3:

https://drive.google.com/file/d/1IsK3ROmnPwAsYhvdzPg0aebZzURyOj1d/view?usp=sharing

Table of Contents

1）**Device Overview**

a. **Features**

b. **General Description**

c. **Block Diagram**

2）**Device Characteristics Table**

3）**Detailed Description**

a. **Distance Measurement**

b. **Visualization**

4) Application Example with Expected Output

5) Limitations

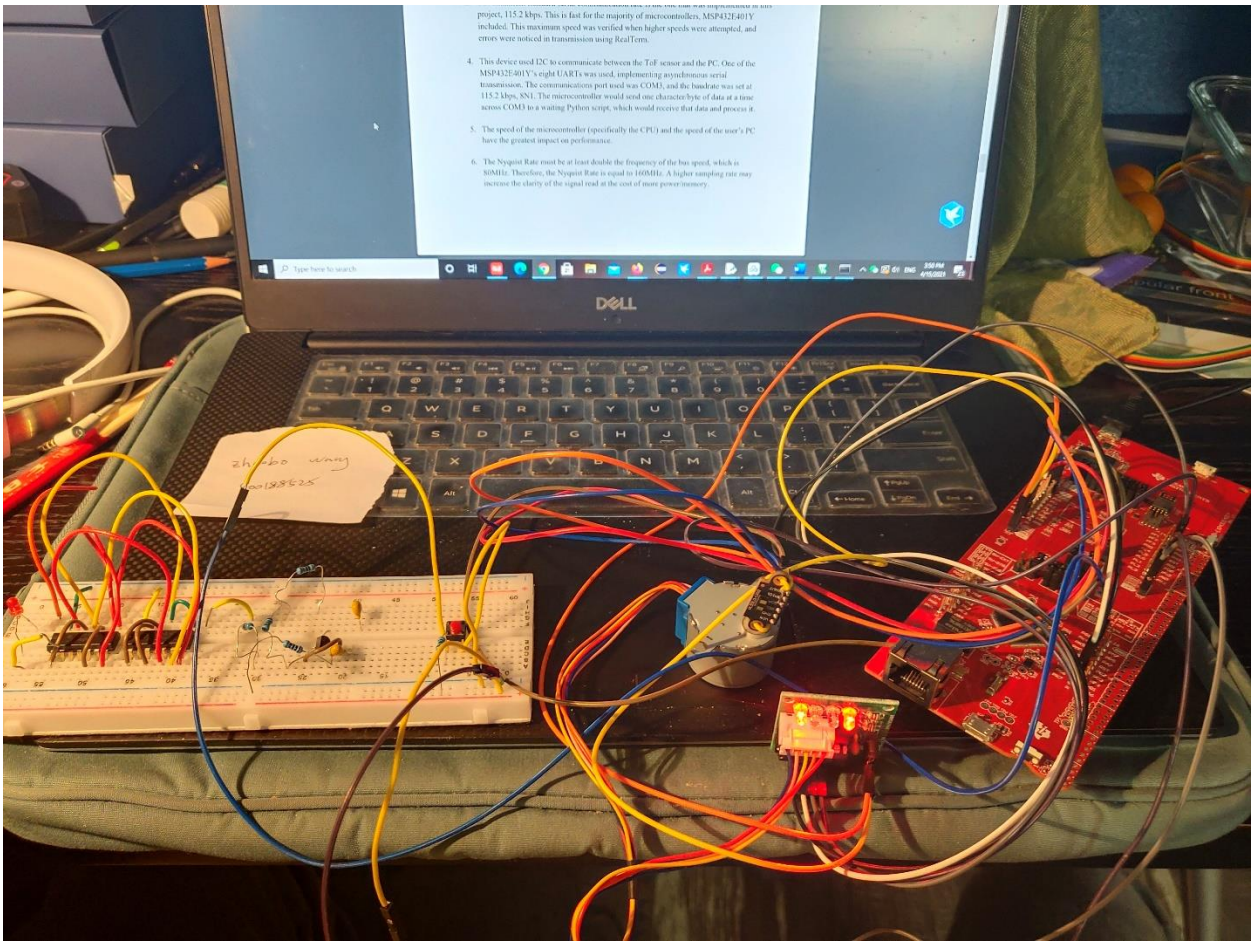
6) Circuit Schematic

7) Programming Logic Flowchart

Device Overview

Features	Details
Lidar System	-360-degree rotation distance measurements -communicates with a PC via USB -3D visualization of the collected data in Python - object exploration and build the volume - orthogonal displacement samples
Texas Instruments MSP432E401Y	- 30MHz bus speed - Arm Cortex-M4F Processor Core -LED for acquisition states
VL53L1X Time-of-Flight Sensor	-Up to 4m range -2.6 - 3.5 v operating voltage - 20mm ranging error and accurate distance measurements - Up to 50 Hz ranging frequency
Data Communication	-I^2C serial communication between the microcontroller MSP432E401Y and ToF sensor VL53L1X -UART serial communication between the PC and microcontroller MSP432E401Y via python -Baud rate of 115200 bps between microcontroller and PC
Visualization	-Supporting on Python 3.8.8 -Create Conda environment for installing open3d, numpy, pyserial -3D visualization of point_cloud data
ULN2003 Stepper Motor Driver	-512 steps per 360 rotation supporting by python. -LED indicators for the step state - 5-12V operating voltage

General Description

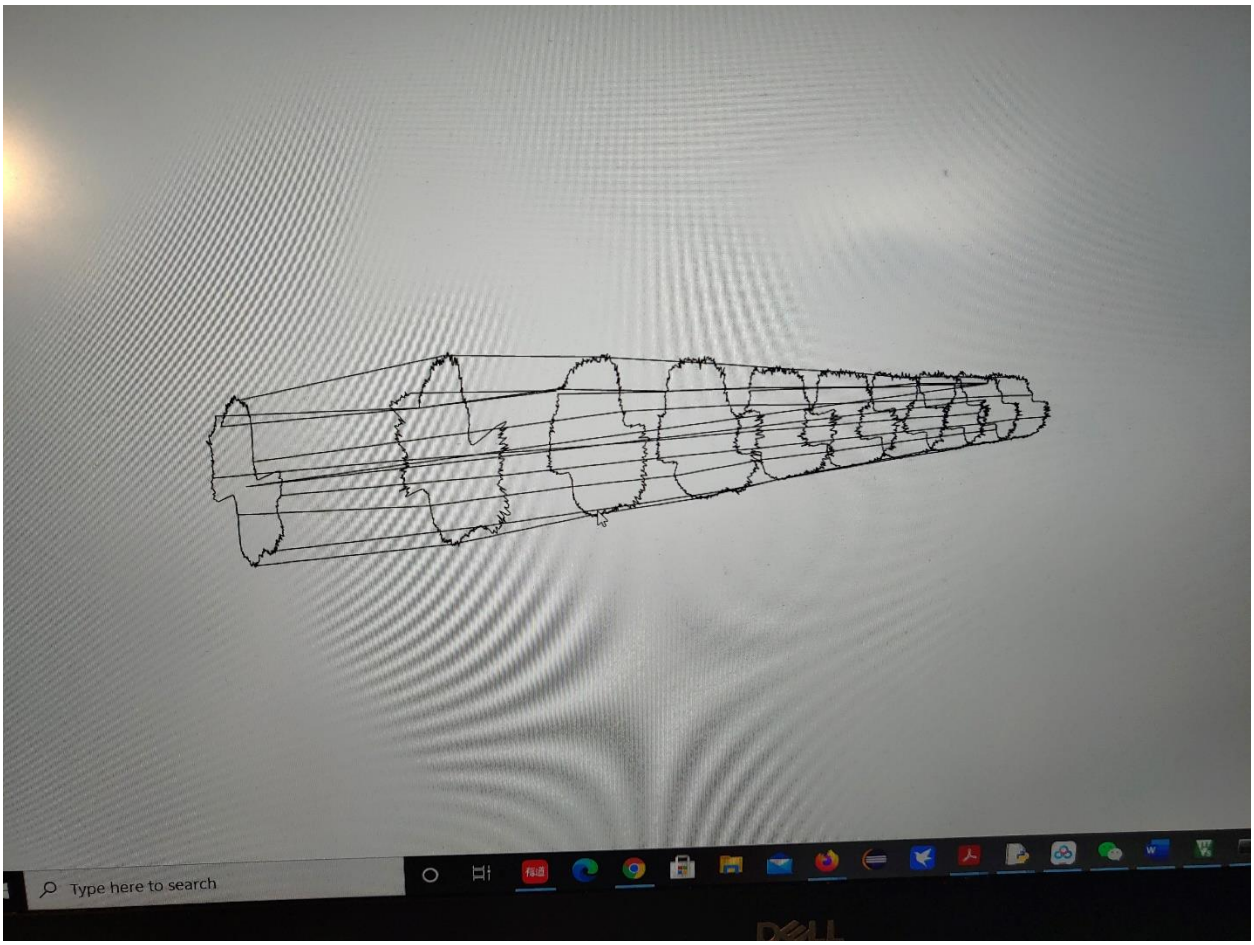


This final project lidar system of the Microprocessor Systems is an embedded spatial measurement system which uses to ToF measurement obtain indoor and outdoor object volume through 3D visualization. This stepper motor is providing 360 degree of rotation which allows the VL53L1X sensor to measure the object in a vertical geometric plane (y-z). The data communication is through serial communication which using both UART and I^2C.

The system consists of a microcontroller, a stepper motor and a VL53L1X sensor. The microcontroller is power supply to the VL53L1X and ULN2003, also controls the entire systems. The microcontroller connected with PC through UART via python. And the microcontroller connected with VL53L1X through I^2C between serial communication. Through the keil project,

the stepper motor will allow ToF sensor to have a 360-degree range of motion and recording the serial data in pyserial, rotating back and rotating forward for 10 times to create the volume.

The VL53L1X ToF sensor emits pulses of infrared laserlight, and at the same time determine the motor angle and distance and x-displacement. With these raw data, using the data based on its configuration settings and sending the data back through I^2C.The data is read and put into a xyz data file. It will visualize by python module open3D via the python code open3d.generate. geometry.



Block Diagram

Device Characteristics Table

MSP432E401Y	**Bus Speed**	**30 MHz**
	Serial Port	**COM4**
	Baud Rate	**115200bps**
	Python Version	**3.8.9**
	Distance Status	**PF0**
	Displacement Status	**PL0**
VL53L1X Pin Map	**VIN** **GND** **SDA** **SCL**	**3.3V** **GND** **PB2** **PB3**
ULN2003	**IN1** **IN2** **IN3** **IN4** **-** **+**	**PH0** **PH1** **PH2** **PH3** **GND** **5V**

Detailed Description

Distance measurement

There will be three components by collecting distance measurement: A transducer, a pre-conditioning circuit, and analog-digital converter (ADC). A transducer is to transmits the non-electrical signal as an electrical signal. ADC converter is the input data as analog and output data as digital. A preconditioning circuit is going to be filtering the electrical signal to be easily convert into digital format. VL53L1X time of flight sensor provide all of data measurement about the surrounding area and it simply provides 360 degree of measurement of distance by using a stepper motor and within single geometric plane. The sensor emits pulses of infrared laser light and measures the time it takes in order to reflect its detectors after reaching the obstacle (objects).

2DX4 course provides VL53L1X time flight sensor API distributed by STMicroelectronics which could be found in the UM2356 user manual. This time-of-flight sensor uses the math equation for that:

*Measurement distance = ½ *time of the emitted lights*speed of the light*

The sensor is using this equation to check if that the final measurement distance is staying in the range or out of the range. It calculates the distance by converting the measured time from analog to digital. It sends the data from the time-of-flight sensor to the microcontroller through the serial communication via I2C.Then the distance data is being outputted through the microcontroller to PC via UART, PC receives the data which opens communication port COM4 at a baud speed of 115200 baud/sec, 8N1.

The breadboard builds the circuit for the interrupt-based button system, which is using for starting and stop the data for the step motor spin function. At the same time, it affects the data capture of VL53L1X sensor. If the button is not being pressed, it will have the value of 0. In this case, it will not capture any data. Otherwise, if the value is 1, it will capture the data. Pressing the button on the breadboard will trigger an interrupt that will change the value of int state. There will be 1 main loop to control manage all of the functions, this main loop will check the value of GPIO PM0 and set state. When this system is in the data capture state, it will begin the data serial communication, the microcontroller and the measurement sensor utilize the I2C protocol, the microcontroller and the PC utilizes the UART and USB port, with these methods, the data is being receiving and writing into a new file named XYZ.

All the raw data sent by the microcontroller to the computer is the x-displacement value and distance value, and the motor angle value. The stepper motor moves one step clockwise and the step counter is adding by 1. One full rotation is comprised of 512 steps. The angle can be determined from the step of the motor:

Angle = 360/512* motor position + 270 degrees

Based on those values, there will be a python file which helps collecting all of data, the x value is staying the same for the distance, the y value is x*cos(angle), the z value is x*sin(angle), all the angle converts from the degree to radians. As the angle values are increasing, y and z components are also changing. XYZ values is stored into XYZ file by formatting them into space. The while loop keeps sending the data, until it hits the break condition. The other python file reads the x,y,z file and in this way, it helps transform the raw sensor data into the format used by Open3D.

Visualization

This visualization run through a DELL XPS 15 (core i7) and running Windows 10 and Python 3.8.8., also create a new conda environment called open3d. In open3d, I install open3d, numpy, pyserial. Open3d used to represent data as point cloud and visualize, Numpy used to represent point cloud as array. The microcontroller would transmit displacement information along with the distance measurements during the data collection, and these would be implemented into the python data_collection.py to adjust the values produced in the xyz file. Reading data from the microcontroller and writing it into a .xyz file gives us a 3D visualization of the data, these data create a point cloud using Open3D library. According to the requirement, it must show for at least 10 scans of measurement. Each plane is 512 data points, given that the step motor will have 512 data points if it wants to complete for one cycle. For the python while loop, one of the conditions to break is that it collects for 512*10 points. The distance data is converted into x,y,z coordinates. The x value is consistent for each point a y-z place and the y and z is determined by the trig function. After creating the point cloud which could be read from the xyz file, a line set must be created for this visualization in the next step. The lines join all consecutive pairs of individual points in a slice, and every 4^{th} point between slices. Once the line set is created for a slice, the previous line set geometry is being replaced and the new line set is adding into this 3D visualization. Once the line set is created, the library visualization function can be used to draw the points and lines present in the line set and visualize them. When all the slices are rendered, the visualization window remains open until we closed by manually.

Application Example

An application of acquiring signal and mapping into using python or other approach:

Global Positioning System (GPS)

The Global Positioning System (GPS) records the precise X,Y,Z location of the scanner. To improve the accuracy most lidar systems use a fixed ground reference station or a Continuous Operating Reference Station (CORS). The data from the ground station or CORS has a known location is used to correct and improve the data collected by the sensor. The GPS data is later post-processed and the precise position of the sensor at approximately every second throughout the flight can be calculated, typically with minimal error (3 to 4 cm). The GPS together with the IMU (see below) allow for the direct georeferencing of the points.

Step by step by using integrated lidar system:

1. Connect your PC to the microcontroller by using USB port. It should turn on after you plug it in.
2. Set up the surrounding environment that to be measured with the sensor vertically plane.
3. Open up the Keil project and translate build load the code, do not press the reset button yet
4. Open the python code <data_finalcollect.py> through the pip or conda environment, at the same time, you have to install open3d, pyserial, numpy module by using typing the code through command window <pip install open3d><pip install pyserial><pip install numpy>
    - For this python file, you have to change the COM port which is suitable for your computer, my computer has the port number COM4 which is for serial communication. If you do not change it, you might not receive any data showing on the screen.

5. Press the button on the microcontroller and at the same time, press the button on the breadboard. GPIO PM0 will receive this digital signal which is going into sending to microcontroller. The microcontroller will start to work and the UNL2003 LED is blinking, after waiting a few milliseconds, the stepper motor begin to spin and rotate.
6. The sensor measures data and by sending the data back to microcontroller via I2C. And at the same time the receiving data is being sent to PC via UART, so the command python shell start to show serial number each by each.
7. For each 512 points, the stepper motor is rotating into one completed cycle and it will counter clockwise rotating back to the 0 degree. And then rotate again for one completed cycle. It will scan 10 times in total, rotating back and forward for 10 times.
8. After 10 times, the stepper motor is going to stop and the pyserial is going to stop receive more data. All collecting_data is written into a file called XYZ file. For the next python code that we need to use is 3d.py
9. Once data collection has been completed, you will want to create a 3D visualization of the recorded data. To achieve this, the 3d.py python file is used, it will read the file XYZ and using the point_cloud and line set method to geometry the final graph.

Define the XYZ axis:

The coordinates are defined by x,y and z coordinates. The positive x direction is pointing towards the user, the positive y direction is pointing up, the positive z direction is pointing to the right. Each set of measurements is taken along the y-z plane, with each displacement being in the x direction.

Limitations

1) The transmitting data has maximum standard serial communication rate in this project which is 115200 baud/sec. The maximum speed was verified when higher speeds were attempted, checking the port setting for the XDS110 UART Port.

2)The IMU module has the maximum speed limits of 400 kHz, the serial communication between microcontroller and measurement sensor uses I2C have a 100000 baud/sec. The microcontroller and IMU module communicate through I2C which depends on the clock speed set.

3) Maximum quantization is equal to resolution, which is mentioned in the 2DX4 lecture note, $V_{FS}/2^m$, ToF has 8 bits, MSP has 12 bits, IMU has 16 bits

ToF res.= 5/2^8

MSP res.=5/2^12

IMU res.=5/2^16

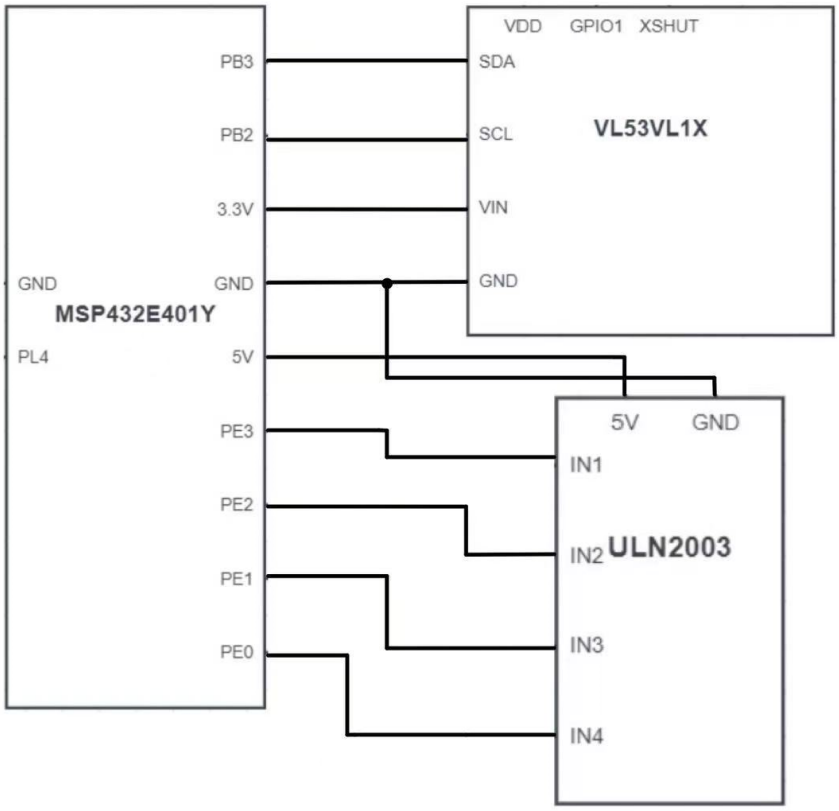
4) Floating – Point Unit (FPU) could be used single-precision 32-bit math calculation. It could also perform the decimal data formats and the real data formats. Because FPU is only 32 bits wide, if the bits are more than 32 bits, (i.e. 64bits), it would require more than 1 instruction to let FPU sperate into 32 bits.

5) A sufficient sample-rate is therefore anything larger than 2B samples per second. For a given sample rate fs, perfect reconstruction is guaranteed possible for a bandlimit $B < fs/2$. This is called Nyquist rate theorem. It states that rate must be at least double the frequency of the bus speed,

which is 30MHz, so the Nyquist is equal 60 MHz. As the sample rate goes high, it might require more memory and power supply.

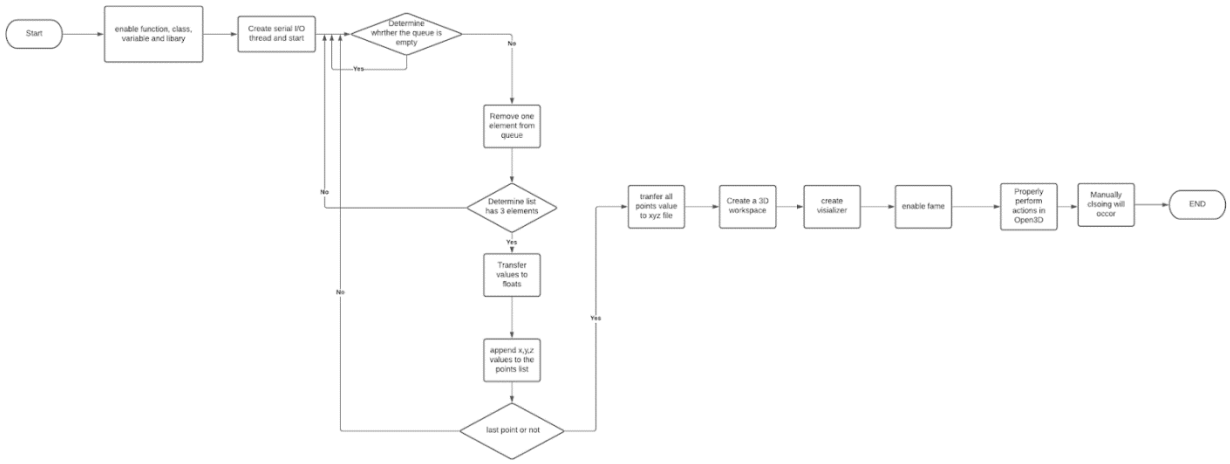
6) The speed limitation is the ranging of ToF sensor. This is because the VL53L1X sensor has a timing budget and after adding the minimum delays, the time consuming is longer than any other process. In the test, trying to make any of these delays less than their minimum values would cause the ToF sensor to not report any data and cause the entire system to stop, which is expected as the sensor is not designed to operate under those conditions.

Circuit Schematic



Programming Flow Chart

Microcontroller



Python