2DX: Microprocessor Systems Final Project

Instructor: Drs. Boursalie, Doyle, Haddara

Rushi Patel – L04 – pater82- 2DX3 – 400316315- Thursday Afternoon-L04

Device Overview

1. Features

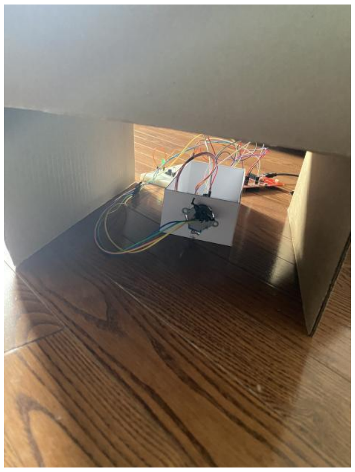


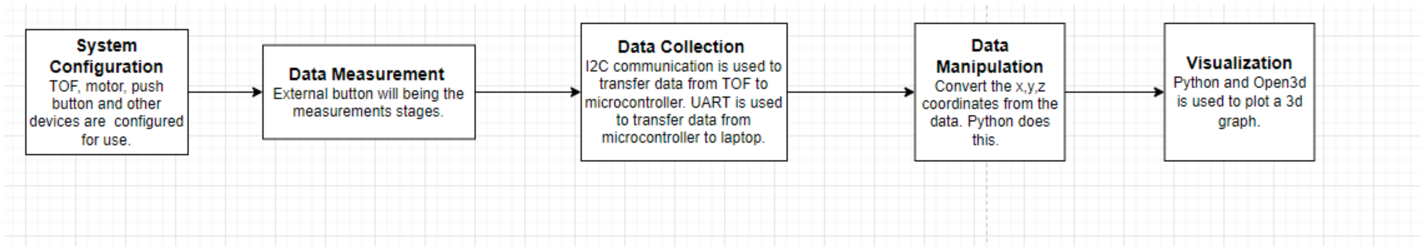
Figure 1

- MSP-EXP432E401Y Microcontroller
    - Reset Switch (start taking measurements)
    - USB Wire
    - 4 Onboard LEDS
    - MHZ
    - Arm Cortex M4F Processor Core
    - $65 cost
- MOT-28BYJ48 Stepper Motor
    - Stepper Motor Controller
    - 5V
    - Use: Rotate the TOF sensor 360 degrees
    - $7 cost
- VL53L1X Time-of-flight Distance Sensor
    - Uses 2.6-3.5 Volts
    - 4mm to 4m range
    - I2C communication
    - $18 cost
- Button
    - Stop taking measurements
- Open3D Graphing Software
    - See the collected data in a visual form
- Computer UART and Python
    - 115200 is the Baud Rate used
    - Communication for computer and microcontroller
    - I2C used for communication with sensor and microcontroller
    - UART communication used between laptop and microcontroller

General Description

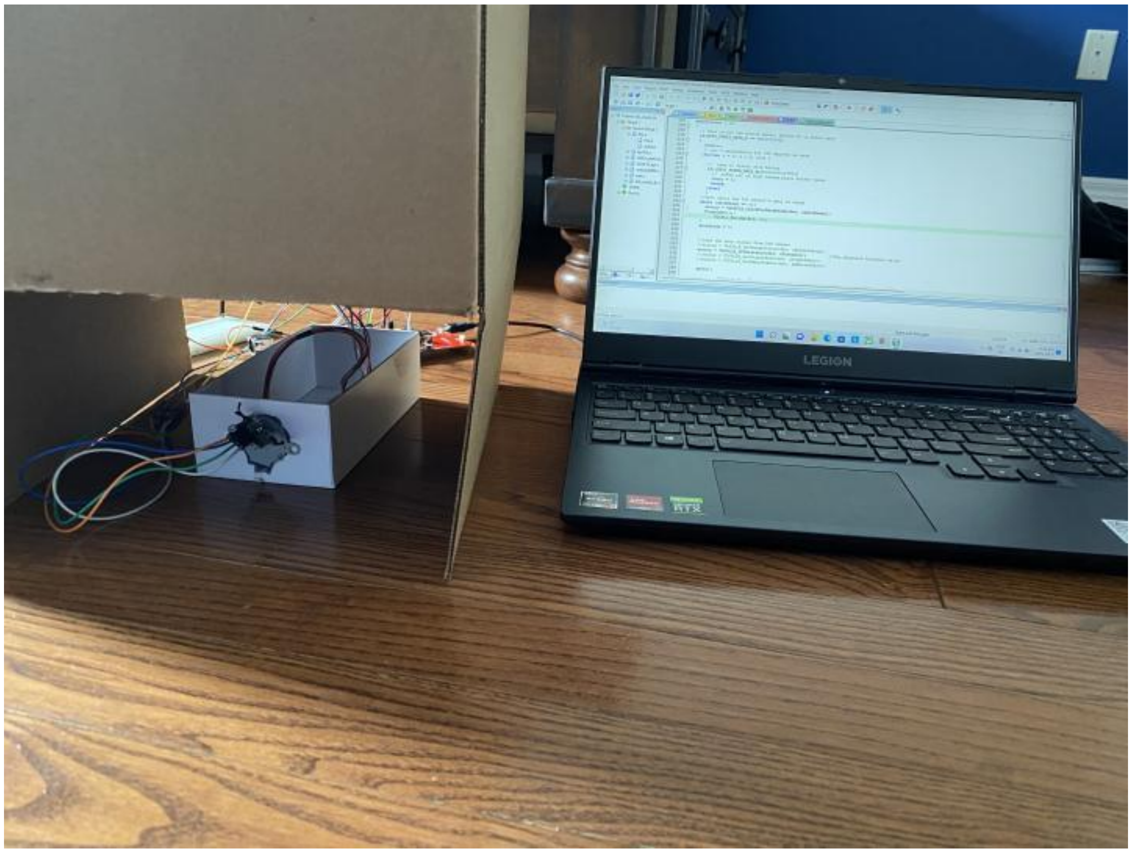
The system for the final project is a 3D scanning system. This is done through obtaining values in terms of distance for 360 degrees. From here the data obtained from the scans is plotted like a graph to visualize (3D). In the design the MSP-EXP432E401Y, the microcontroller is required. It is the motherboard of the device meaning it is the backbone of the circuit. It allows for all the devices to communicate to each other and can execute different functions. In the final project the microcontroller is responsible for configuring the TOF sensor, for example. It also powers up the other parts in the system. For the stepper motor its purpose is to rotate 360 degrees as the sensor is mounted on to it. This then allows the system to get the distance values. Moving on to the sensor the time-of-flight sensor this gets the distance. This is done by releasing a certain type of laser light and waiting for it to be reflected back. The time it takes for the light to reflect back is used for the output of the required information by the sensor. A switch is also used for input and output. It is connected to PM0 on the microcontroller and can control when to stop and start taking measurements. I2C is also used to translate the information to the microcontroller. Additionally, UART is used to enable the microcontroller to communicate to a computer. From here with the use of python the information collected can be viewed.

Block Diagram:


System Configuration
TOF, motor, push button and other devices are configured for use.
Data Measurement
External button will being the measurements stages.
Data Collection
I2C communication is used to transfer data from TOF to microcontroller. UART is used to transfer data from microcontroller to laptop.
Data Manipulation
Convert the x,y,z coordinates from the data. Python does this.
Visualization
Python and Open3d is used to plot a 3d graph.

2. Device Characteristics:

Specification		
MSP432E401Y	Serial Port	COM3
	Baud Rate	115200 BPS
	Bus Speed	48 MHz
	Distance	PN1
	Libraries	Pyserial Open3d
Button	PM0 Pin 3.3V	
Stepper Motor (MOT-28BYJ48)	Driver	Pins
	IN1	PH0
	IN2	PH1
	IN3	PH2
	IN4	PH3
	+	5V
	-	GROUND
TOF Sensor	Time of flight sensor components	Microcontroller Pins
	Vin	3.3 Volts
	Ground	-
	SCL	PB2
	SDA	PB3



Picture of setup including device

3. Detailed Description

Distance Measurement:

The distance measurements are obtained from the help of the TOF sensor. For the values the time-of-flight sensor measures everything in a 360-degree manner. As mentioned, previously this is done by mounting the sensor on to the stepper motor in which the system makes the stepper motor rotate 360 degrees. In this case the motor would get measurements every 45 degrees. The sensor obtains its values by releasing infrared light and once this light reflects off an object, the time it takes for the light to reach is used to obtain the necessary values. From here the data is sent to the microcontroller by using I2C communication. The equation for the sensor calculation is $Distance = \frac{(time)(C)}{2}$.

Looking at the base parameters given in VL53LI CORE API these base values are used to initialize the sensor. All the functions used are also available in this API. Looking at the flowchart, which is figure 6, this shows how the data is collected and then turned into a 3d graph. For the inter-integrated circuit, the microcontroller has to initialize this during the start phase so this can allow the sensor to be able to communicate with the microcontroller.

The button used to start the sensor to get data and the external push button that can be used to stop the system from getting data. Once the button on the microcontroller is pressed there should be onboard LEDs that start to flash and then eventually stop. After this the code in the python file will run for the data processing part. After the word "Enter" shows up the user presses the PJ1 button which will start the system to collect data. From here the data gets sent to the laptop using UART. The distance measurements values will be processed by the laptop from the python file which will write into the .xyz file. This is done by using pyserial.

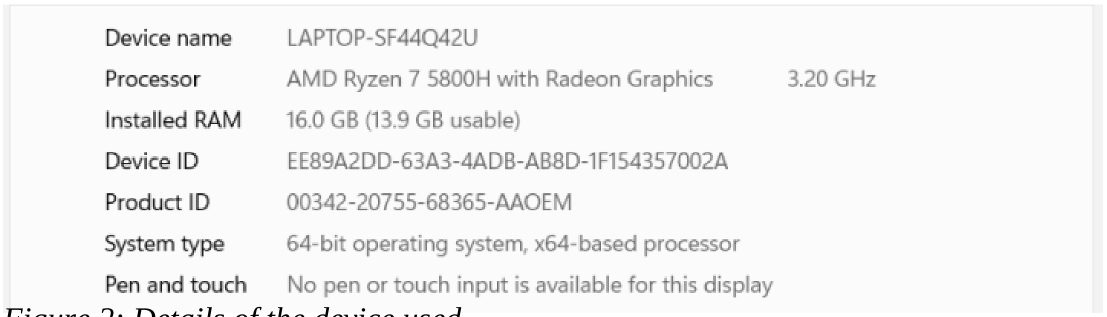
The x component of the data starts of at 100 and increases by this value each time. For the y and z components these values are calculated by using the cos function for y, and sine function for z. For the motor the angle that it turns is determined by knowing the number of steps and multiplying that by pi/4. We do pi/4 as we want the data for every 45 degrees. This can also be seen by the flowchart which is *figure 6*.

$$Y = distance\cos(theta)$$

$$Z = distance\sin(theta)$$

Visualization

The computer used was a Lenovo Legion 5 laptop. It is a window 11 laptop with a 64 bit operating system. The ram available is 16GB running on 3200 Mhz , NVIDIA GEFORCE RTX 3070 and processer is AMD Ryzen 7 5800 H. More details on the laptop can be seen in *figure 2*.



Device name	LAPTOP-SF44Q42U	
Processor	AMD Ryzen 7 5800H with Radeon Graphics	3.20 GHz
Installed RAM	16.0 GB (13.9 GB usable)	
Device ID	EE89A2DD-63A3-4ADB-AB8D-1F154357002A	
Product ID	00342-20755-68365-AAOEM	
System type	64-bit operating system, x64-based processor	
Pen and touch	No pen or touch input is available for this display	

Figure 2: Details of the device used

The libraries used for the project were open 3d, import serial, import NumPy, import math and open3d. Python 3.9.12 was used to run the python file. The values first go into COM 3, this port can be different for others. From here the appropriate values are sent to the .xyz file, which stores the x, y, and z measurements. The total amount of measurements will depend on how many scans done.

To store the data and process it the data first goes into the respective serial port. In this case for the device used the port was COM3. From here the distance measurements are transferred into the xyz file. In this file there are three variables. First one is the x displacement which is manually measured. The other two are for the y and z measurement values. Every time the system is moved the values are filled up for the next plane and sent to the same file. After the data points are all connected to see a 3d representation. This can be seen in *figure 4*.

To visualise the data open3d library is used. This function lets the data be read and processed so it can be ready to be visualized. After a point cloud would be created where the measurements values are outputted and from here all the coordinates produced are used for the 3d graph. The point cloud will include all the points and for the lines it will put 2 slices to connect everything. This can also be seen in *figure* 3.

4/5. Application and User Guide

To fully run this program on your own there few steps that must be done.

1. Download python from the internet. Python 3.7-3.9 is the recommend versions. For this project the version used was 3.9.12.

2. Connect the microcontroller to the computer. This can be done by using the USB cable provided.

2. Open device manager and under ports check the UART number.

3. Go to the command window and type in "python -mserial.tools.list_ports -v" to see the port your own computer can use. In this case port 3 is used. If the user trying this has a different port they must used this has to be changed in the program. Baud rate must be 115200 bps.

4. Connect all the necessary components to each other into the correct ports. For example, build the correct things onto the breadboard, connect PM0 from the microcontroller to the stop button etc.

5. Set up the time-of-flight sensor onto the stepper motor. Also have a hallway. The hallway can be recreated from the picture inserted into the document earlier (*figure 1*).

6. Open the ".py" file. This should be done once the microcontroller is done initializing the sensor. In Keil it is important to build, load and run the code.

7. When the python file is open press "run module". Now the message "Opening: COM3. Press Enter to start communication" will be seen. From here press the rest button on the microcontroller. Now wait for the leds that light up to stop. If they do not stop flashing, then there is a good chance the wires being connected to the sensor from the microcontroller have been put in incorrectly.

8. Press enter into the python window and press the PJ1 button. This will make the system gather the x, y, and z coordinates. The positive y-axis is the direction the sensor faces from the start, and the positive z axis will rotate 90 degrees clockwise. This is in reference to the y-axis.

9. The program will take 8 different measurements which can be seen on the python window. Also, the motor must have rotated 360 degrees. Make sure the wires do not over tangle as this can cause some faulty connections. From here move the system forward to get the next set of displacement values.

10. Steps 2 and 3 can be repeated as many times as the user prefers. If there is an error in the python code when trying to get the other samples the user will not be pressing the PJ1 button fast enough. To have more time to do this the number for this can be changed in the python code.

11. The program can be stopped obtaining data by pressing the stop button on the breadboard.

12. A pop up window of "Open3d" will show up. Press exit on it to see another pop up window which shows the full 3d graph.

Limitations

1. There are no limitations for this project when it comes to floating point capability and the use of trig functions. Since there is the use of python in this case, python's math library can be used. This means any limitations mathematical wise in this case would not exist. The microcontroller does have the FPU feature also which can do basic mathematical operations also, however it is not as precise .

2. Quantization error = $V_{FS}/2^m$

$$Mircocontorller\ Error - \frac{V_{FS}}{2^m} = \frac{3.3}{2^8} = 0.013\ V.$$

$$Time\ of\ flight\ sensor\ - \frac{V_{FS}}{2^m} = \frac{3.3}{2^{12}} = 0.00080\ V.$$

3. The maximum standard serial communication rate is 128000 bits. In order to see this information, go under the device manager option. For the project the speed used was 115200 bps. To verify the speed if the speed was increased then there would be errors seen on the RealTerm application.

4. I2C serial communication is used between the microcontroller and TOF module. In this project the microcontroller is the leader, and the time-of-flight sensor acts as a follower. For the communication speed referring back to the datasheet it can be seen the speed is 100 kbps.

5. After reviewing the entire system the element that is limiting the speed the most is stepper motor. It moves to slow compared to the other parts used in the system. For example, the sensor it outputs the values almost instantly. However, at the start the stepper motor has a slight delay in starting and takes time to do a 360 rotation. This affects the speed for the whole system in general. To test this when the clock speed was decreased the stepper motor moves faster relative to the system. Looking at the values the sensor outputs the accuracy does not change. So, since the sensor has to wait for the stepper motor for all the scans done (x8 for each scan), the stepper motor is the main device that imposes the limitation for the speed.

Pictures (Circuit Schematics, Flowchart)

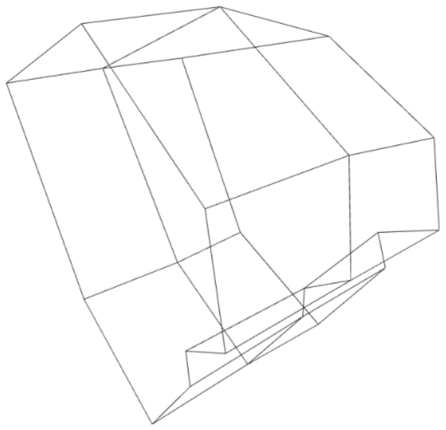
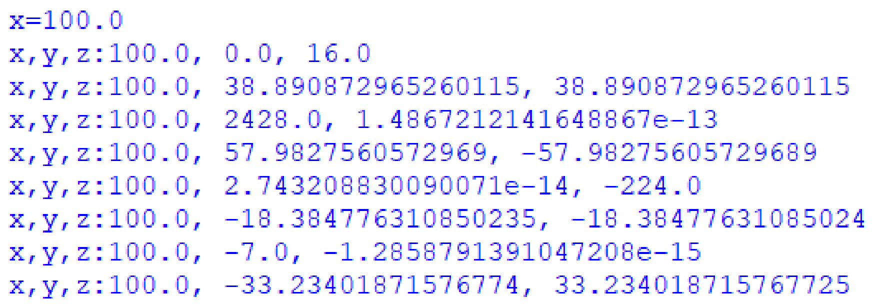


Figure 3: 3D graph reconstruction from 3d



```
x=100.0
x,y,z:100.0, 0.0, 16.0
x,y,z:100.0, 38.890872965260115, 38.890872965260115
x,y,z:100.0, 2428.0, 1.4867212141648867e-13
x,y,z:100.0, 57.9827560572969, -57.98275605729689
x,y,z:100.0, 2.7432088300900071e-14, -224.0
x,y,z:100.0, -18.384776310850235, -18.38477631085024
x,y,z:100.0, -7.0, -1.2858791391047208e-15
x,y,z:100.0, -33.23401871576774, 33.234018715767725
```

Figure 4: XYZ file output

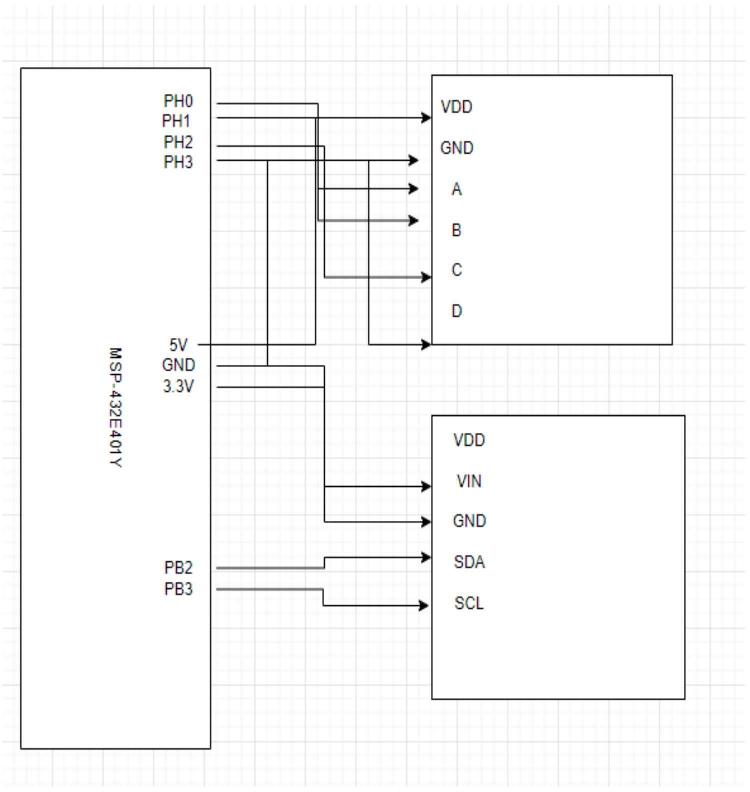


Figure 5: Circuit Schematic

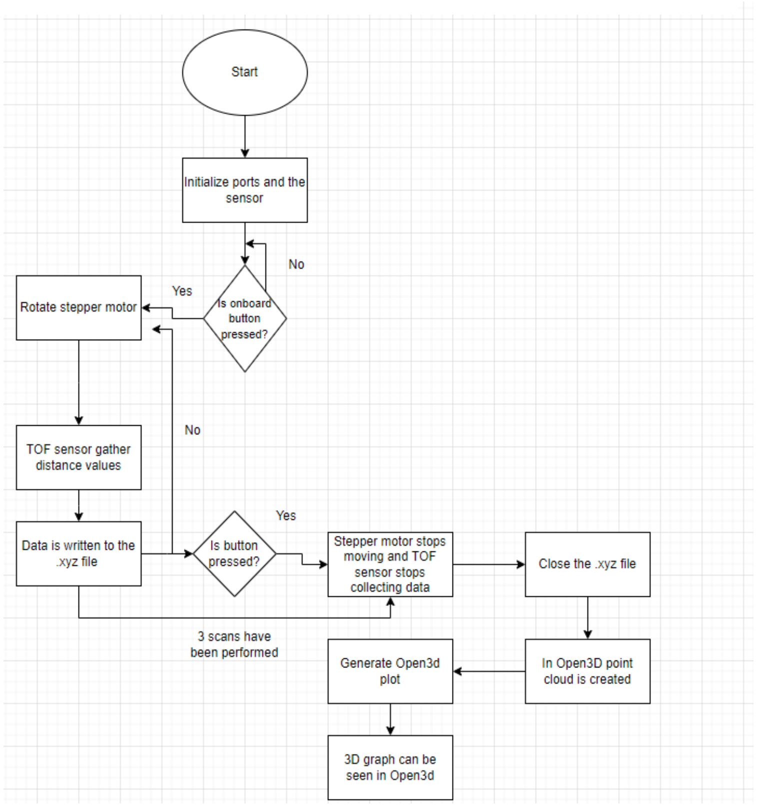

Start
Initialize ports and the sensor
No
Yes
Is onboard button pressed?
Rotate stepper motor
No
TOF sensor gather distance values
Data is written to the .xyz file
Is button pressed?
Yes
Stepper motor stops moving and TOF sensor stops collecting data
Close the .xyz file
3 scans have been performed
Generate Open3d plot
In Open3D point cloud is created
3D graph can be seen in Open3d

Figure 6: Flowchart of System