Microprocessor Systems 2DX3
Project Report

Instructor: Dr. Athar/Doyle/Haddara
Siddh Patel

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 **[Siddh Patel]**

1.0 Device Overview

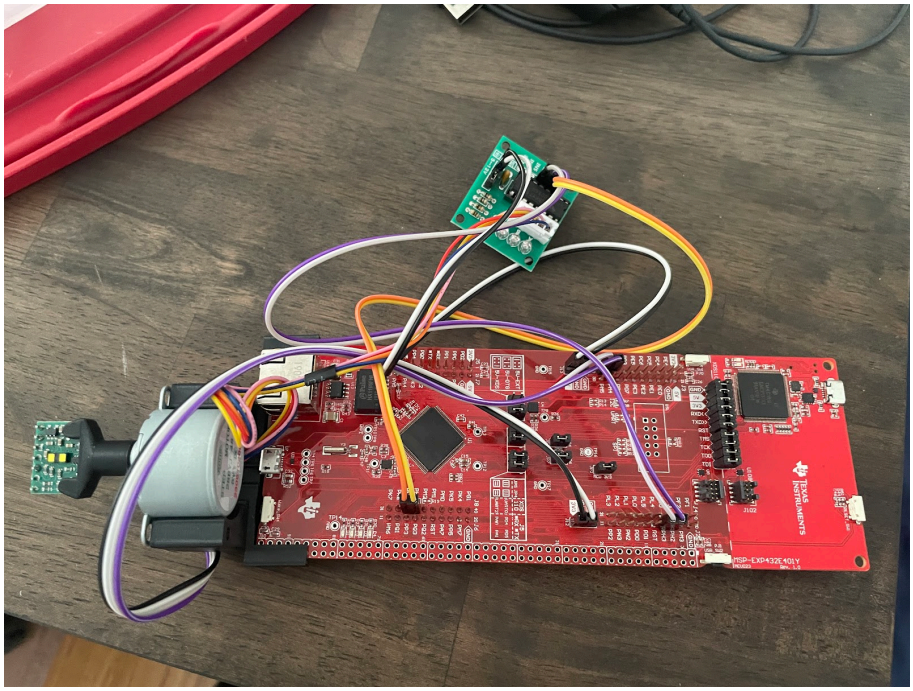

Figure 1: Final Design

1.1 Features

Hardware:

- MSP432E401Y Microcontroller.
    - Bus speed up to 120MHz.
    - Programmable using Assembly and C.
    - 2 Status LEDs
    - 12-bit ADC
- ULN2003 Stepper Motor
    - Operating Voltage: 5V DC
    - Up to 512 samples per revolution (0.703° step angle)
- VL53L1X Time-of-Flight Sensor
    - Operating Voltage: 3.3V DC
    - I2C Communication Protocol to transfer data
    - Up to 400 cm distance measurement

Serial Communication:

- I2C used to transfer data from ToF sensor to MCU
- UART used to send data from MCU to PC via PySerial Library

Mapping Software:

- Uses Python script and Open3D library to map Data from sensor
- Supported on Python 3.9

1.2 General Description

The Lidar system represents an advanced 3D scanning solution, employing a sophisticated system architecture. At its core lies a time-of-flight sensor, capable of precisely measuring distances and providing XYZ coordinates of the scanned area. These coordinates are seamlessly transmitted to the MSP432E401Y microcontroller (MCU) via the I2C protocol. The MCU also controls the rotational motion of the ULN2003 Stepper Motor, precisely advancing it at intervals of 5.625 degrees which can be changed to increase or decrease the accuracy of the map. Mounted atop the stepper motor, the sensor synchronously collects data at each incremental angle, which is then stored within the MCU's memory array. Once all the set of measurements is acquired, the MCU initiates data transfer to a connected PC via UART, facilitated by pySerial communication. Subsequently, using the visualization capabilities of Open3D, the collected data is graphically rendered in three dimensions, offering users a comprehensive and intuitive understanding of the mapped area. This integrated approach ensures seamless data acquisition, processing, and visualization, fostering enhanced insights into complex spatial landscapes.

1.3 Block Diagram

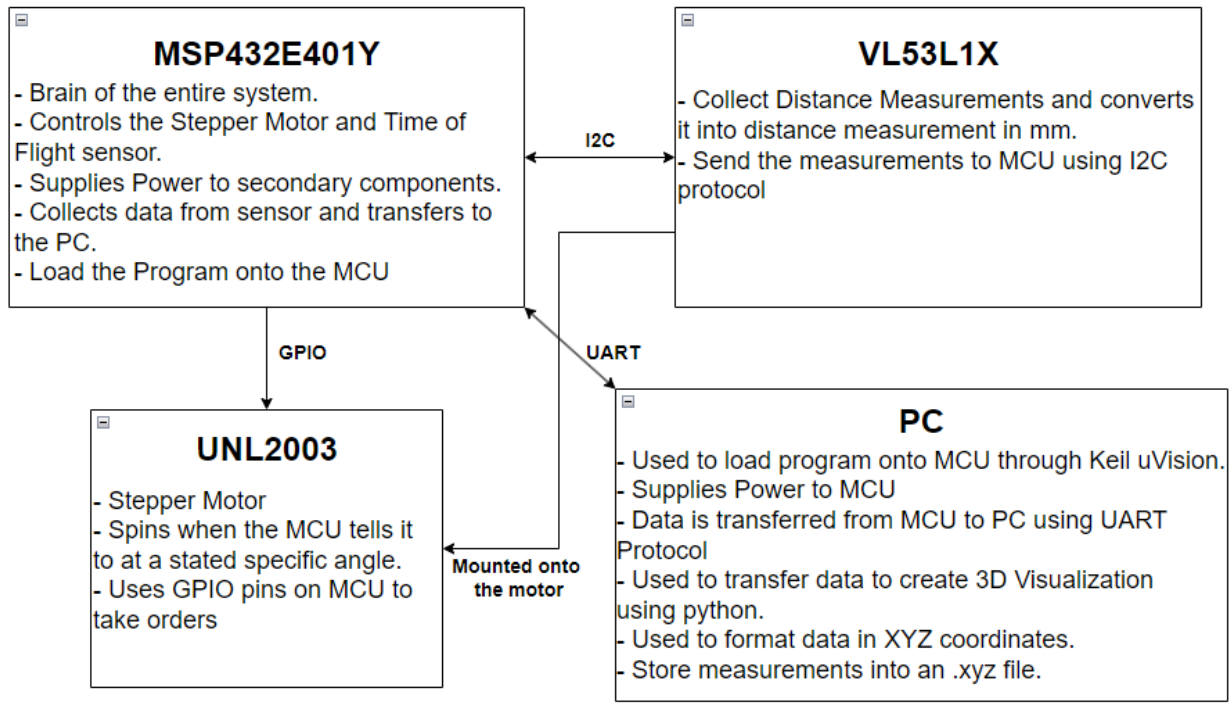

Figure 2: Block Diagram of the entire system.

2.0 Device Characteristics Table

Table 1: Power Requirements for all components.

Components	Power Requirements
MSP432E401Y	+5V, Connected to PC
VL53L1X Time of Flight Sensor	+2.8V – +5V
ULN2003 Stepper Motor	+5V

Table 2: Parameters used in this specific test.

Parameters	Values
Bus Speed / Clock Speed	96 MHz
I2C Address	0x29
UART Baud Rate	115200 bps

Table 3: Pin Names and Description

Pin Name	Signal Type	Port Description
GPIO [H3:H0]	PWM	Output to Stepper Motor
GPIO [J1]	INPUT	On Board button to start Data Collection Process
GPIO [B2]	SCL	I2C Clock
GPIO [B3]	SDA	I2C Data
GPIO[N1:N0]	Output	Onboard LED

3.0 Detailed Description

3.1 Distance Measurement

In this specific system, the VL5L1X ToF sensor is employed for data collection. Emitting a 940nm laser, invisible and harmless to humans, the sensor gauges distance by measuring the time taken for the laser to rebound from objects to the sensor. Subsequently, the equation $d = \frac{\Delta t * c}{2}$ is utilized to derive a distance value, where c represents the speed of light. Securely positioned on the stepper motor, the sensor can gather data while the motor is in motion, resulting in a 2D scan of the surroundings at a fixed z coordinate. To provide real-time feedback on data collection, the onboard Led D2 is programmed to blink when the sensor acquires data.

The MCU is programmed to reverse direction once a complete revolution of data collection is achieved, preventing wire entanglement with the ToF sensor. Thus, Led D1 signals when the sensor is in the process of data collection and when it returns to its home position for another round. The MCU program incorporates a global variable to specify the desired number of data sets to be collected. This entire process initiates when the Python code is running, and the user activates the PJ1 button on the MCU.

As the sensor exclusively collects 2D data, users must physically move approximately 20cm forward to gather another set of data at varying z coordinates. They repeat this movement after each round until the entire area is scanned. Upon completion, the MCU utilizes UART protocol and pySerial to transfer data to the PC, where a Python program employs the following functions to calculate precise X and Y coordinates:

$$x = Distance * cos(\Theta)$$
$$y = Distance * sin(\Theta)$$

These XYZ coordinates are then stored in an .xyz file for subsequent visualization of the room.

3.2 Visualization

Following data collection, the pySerial library facilitates the seamless transfer of data from the MCU to the PC utilizing the UART protocol. Subsequently, the acquired data undergoes processing using the above equations to derive precise XYZ coordinates. These coordinates are then organized and stored into a designated [data.xyz] file for future reference and analysis.

To translate the data into a visually interpretable format, a python script equipped with NumPy and Open3D libraries is employed. This script plots the data.xyz file onto a 3D graph. Using the same python script the points are connected using solid lines forming closed loops to create a comprehensive 3D visualization, offering users an immersive representation of the scanned area. Additionally, the visualization allows for enhanced analysis and exploration of

spatial data, providing valuable insights into the scanned environment's characteristics and structure.

4.0 Application Instructions

Using the Lidar System required a certain setup process. The user will require a windows laptop to ensure all required softwares is able to run smoothly. The following are the steps to download required softwares.

1. Download Python version 3.9 using the following link: Python Release Python 3.9.0 | Python.org
2. Download Keil IDE for programming MSP432E401Y microcontroller using the following link: Keil MDK (arm.com)
3. Open python and install required Libraries by typing the command in the terminal:
    a. “pip install pyserial”. To transfer data from MCU to PC.
    b. “pip install open3d”. To convert a .xyz file to 3D Visualization.
4. Download the attached source code for the Lidar System.

The following are the steps to setup Lidar System.

1. Open the attached source code in Keil uVision.
    a. Navigate to the PLL.h file. On line 29 you can change the constant to desired bus speed.
    b. Navigate to SysTick.c. On line 72 modify the constant to your selected bus speed in section 1a. Delay constant can be calculated using the following equation: Bus Speed * 10m.
    c. Navigate to main.c. Modify TOTAL_Z_STEPS variable to the total number of revolutions of data you want to collect.
    d. Navigate to main.c. Modify the SAMPLE variable to the amount of times you want to collect data per revolution.
2. Open the attached python script.
    a. Change the COM Port to your specific laptop. This can be found in Devices and Manager.
    b. Modify the TOTAL_Z_STEPS variable in the python script as well.
    c. Modify the SAMPLE variable based on your Keil program.
3. Now connect MCU to Laptop and wire the given circuit.
4. Load the main code using KeiluVision.
5. Wait until the LED on MCU flashed twice. Run the Python script.
6. When you are ready to collect data. Press PJ1 on MCU
7. The motor will start spinning and collect data, once a full revolution is completed the motor will spin in the opposite direction. At this time the user will move forward 20cm and the sensor will automatically spin again and start collecting data. Repeat this until you have scanned the entire area.

8. After completing all rotations, the python script will automatically create the 3D Visualization file.

Here is an example of expected output.

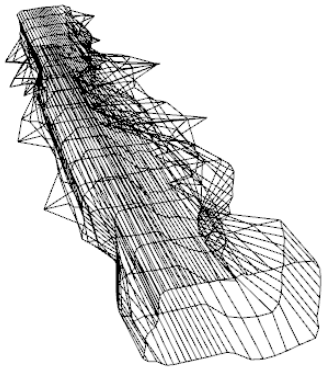

Figure 3: 3D Visualization Example.

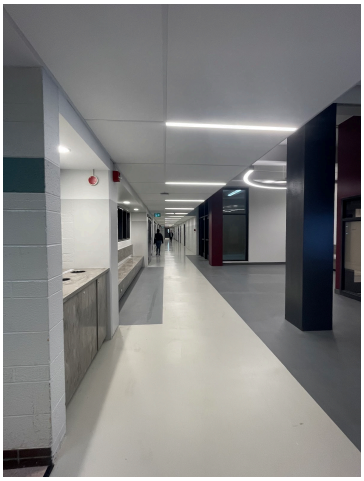

Figure 4: Actual Picture of the location.

5.0 Limitations

The MSP432E401Y microcontroller boasts a 32-bit Floating Point Unit (FPU) optimized for single-precision (C float) data-processing operations, offering advantages such as faster execution times compared to double precision. However, this architecture introduces inherent limitations, notably in precision and support for certain IEEE 754-2008 standard operations like calculating Remainder and rounding float to integer-valued float. Additionally, the Cortex-M4F floating-point instruction set may not encompass all trigonometric functions typically found in the C math library, necessitating careful consideration when utilizing such functions. Therefore, while the MSP432E401Y microcontroller offers robust floating-point capability and trigonometric function support, users must navigate these limitations to ensure optimal performance and precision in their embedded applications.

The following equation was used to calculate maximum quantization error for the ToF module.

$$Max\ Quantization\ Error\ = \frac{4000\ mm}{2^{12}} = 0.977\ mm$$

The highest achievable standard serial communication rate on the PC is 128000 bits per second. This can be confirmed by connecting the MCU to the PC and accessing the Device Manager on the PC. Locate the category titled "Ports (COM & LTP)" and expand it. Within this category, select the UART port, where the maximum baud rate can be observed under "Port Settings." Expand the bits per second option to view the maximum baud rate setting.

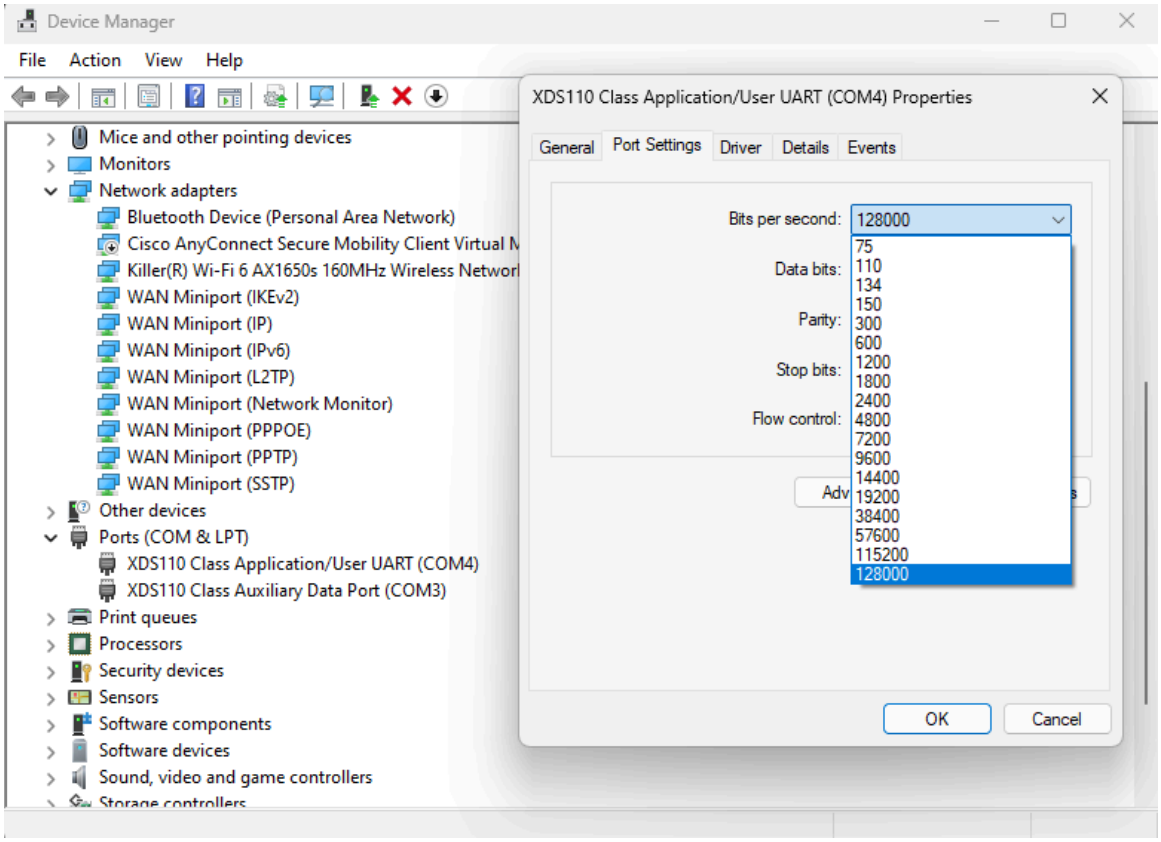

Figure 5: Where to find the max baud rate.

The I2C serial communication protocol was used to transfer data from ToF to microcontroller along the SDA data line. The timer period written into I2C_MTPR_TPR is 0x3B(59). The system clock speed is 96MHz. The SCL_HP is 4, and SCL_LP is 6. According to this formula, the SCL line period is 12500ns, and the frequency is 80kbps.

SCL_PERIOD = 2*(1+TRP)*(SCL_LP+SCL_HP)*CLK_PRD = 2*60*10*(1/96M)

In our system, the Time-of-Flight (ToF) sensor emerges as the primary bottleneck for speed optimization. The fundamental challenge lies in the sensor's ranging process. This inherently limits the system's speed. Even though we can adjust the delay times for motor rotation, the sensor's lengthy ranging procedure requires approximately 140 ms to detect distances of just 400 mm. Since we are not able to change the delay for the ToF ranging process, it is the primary element that affects the speed of the overall design. This process was tested using leds to measure how long the ranging process takes by turning an led on right before starting the process and turning it off after the process finishes.

To achieve my assigned clock speed of 96MHz, I used the following equation to calculate the PYSCHDIV value to be inputted in PLL.h.

$$System\ Clock\ Speed\ = \frac{480MHz}{PYSCHDIV + 1}$$

$$96MHz = \frac{480MHz}{PYSCHDIV + 1}$$

$$PYSCHDIV = 4$$

The delay value in SysTick.c was also changed based on the assigned bus speed. The delay was calculated using the following equation. 96MHz * 10ms = 960000.

6.0 Circuit Schematics

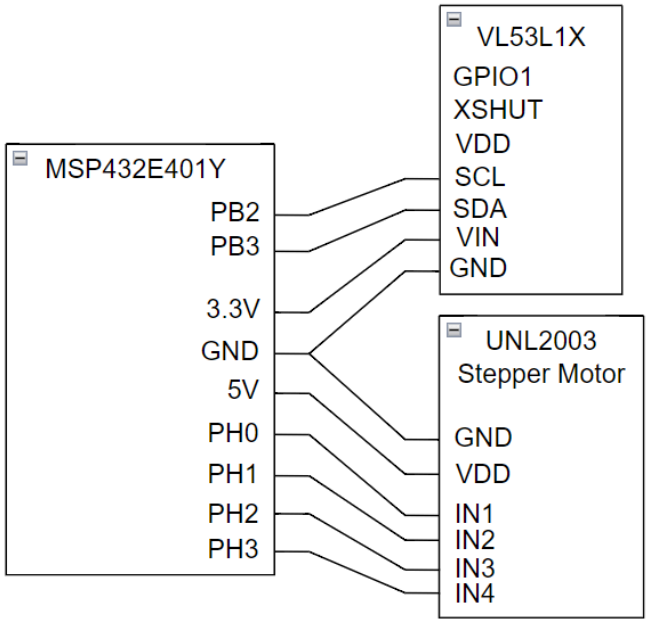

Figure 6: Schematic of how to wire up the system.

7.0 Programming Logic Flowchart

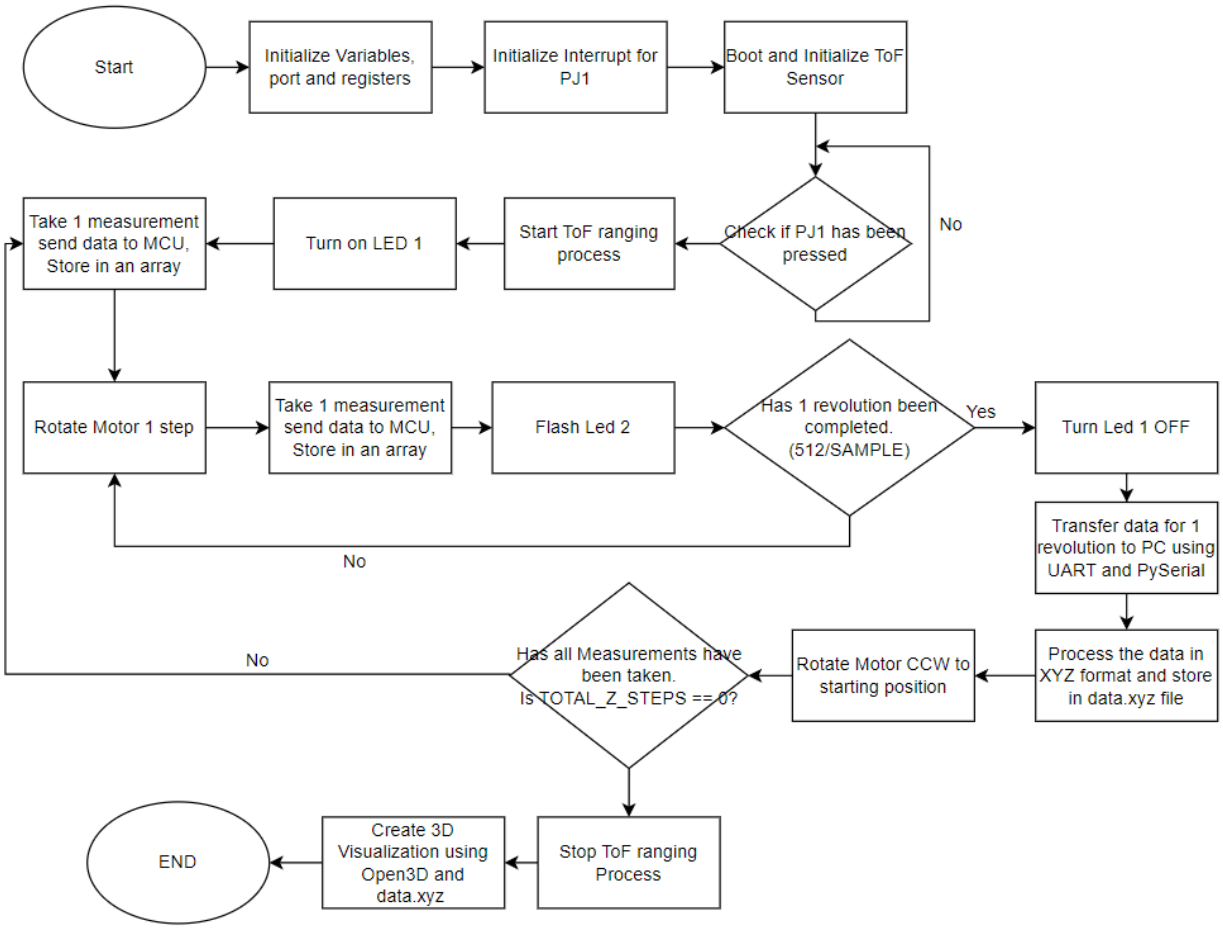

Figure 7: Overall program flowchart for Kiel and Python Script.