COMPENG 2DX3

Microprocessor System Project 2DX3

Instructor: Dr. Haddara, Dr. Doyle, Dr. Athar

Nuha Fahad– L06 – fahadn- 400457365

Date of Submission: April 17th, 2024

As a future member of the engineering profession, the student is responsible for per-  
forming the required work in an honest manner, without plagiarism and cheating.  
Submitting this work with my name and student number is a statement and understand-  
ing that this work is our own and adheres to the Academic Integrity Policy of McMaster  
University and the Code of Conduct of the Professional Engineers of Ontario

Table of Contents

[1 Device Overview 3](#_Toc164290468)

[1.1 Features 3](#_Toc164290469)

[1.2 General Description 4](#_Toc164290470)

[1.3 Block Diagram 4](#_Toc164290471)

[2 Device Characteristics Table 5](#_Toc164290472)

[3 Detailed Description 6](#_Toc164290473)

[3.1 Distance measurements 6](#_Toc164290474)

[4 Application Note, Instructions, Expected Output 8](#_Toc164290475)

[4.1 Instructions 8](#_Toc164290476)

[4.2 Expected Output 9](#_Toc164290477)

[5 Limitations 10](#_Toc164290478)

[6 Circuit Schematic 11](#_Toc164290479)

[7 Programming Logic Flowchart 11](#_Toc164290480)

[Appendix 11](#_Toc164290481)

## 

## 1 Device Overview

### 1.1 Features

The two components that were used in the 2DX3 final project were the microcontroller Texas Instruments MSP432E401Y and the time of flight (ToF) sensor VL531LX. Additionally, a stepper motor, visualization software and other miscellaneous components were used in this implementation. The features for these devices are listed below.

Texas Instruments MSP432E401Y [Source: MSP4302E401Y data Sheet ]:

* Processor – Cortex-M4F Processor with Floating Point Unit (FPR)
* Harvard Architecture
* Bus Speed – 120MHz
* Memory – 1024 KB flash memory
* Operating Voltage – 2.5V to 5.5 V
* SRAM – 256KB
* EEPROM – 6KB
* Analog to Digital – SAR based two 12-bit ADC modules.
* Programing Language – C or Assembly Language using Keil to interface.
* Communication: I2Cs and UART – Serial Communication
* Baud Rate – 115200bps (between microcontroller and laptop)
* PJ1 used as start/stop data collection push button
* Onboard LED PF0 (data measurements) and PF4 (additional measurement status)

VL53L1X ToF sensor [Source: Vl53L1X Data Sheet]:

* Up to 400 cm distance measurement
* Up to 50 Hz ranging frequency
* 3 distance ranges: short range, medium range, and long range
* Operating Voltage: 2.6V to 3.5V
* Communication I2C with (serial communication)

Visualization Software

* Python (for data acquisition and modification)
* Open3D (for data visualization)

Stepper Motor

* Operating voltage 5V
* Steps used 512 steps
* Step angle 11.25 deg

Table 1: Component Costs

|  |  |
| --- | --- |
| Component Name | Component Cost |
| MSP432E401Y MC | $77 |
| VL53L1X ToF Sensor | $33 |
| Stepper Motor | $12 |
|  |  |

### 1.2 General Description

A student designed a system that would collect distance data and in effect map out 3-D model of the space such as a hallway. To achieve this task, the VL531LX ToF Sensor was connected to the MSP432E401Y microcontroller and then mounted on a stepper motor which rotated in intervals. At each interval, distance data was acquired using python, and each measurement was used to derive x,y,z coordinates. After each rotation the mechanism was shifted forward to capture a second frame. After multiple frames were collected the data points were plotted using Open3D to generate a 3-D model of the measured space.

### 1.3 Block Diagram

A diagram of steps and steps

Description automatically generated

Figure 1: 3D scanner block diagram

## 2 Device Characteristics Table

|  |  |  |
| --- | --- | --- |
| ***Device Characteristic*** | ***Used Value*** | ***Purpose*** |
| Bus Speed | 30MHz |  |
| Digital I/O LED Status | PF0 (LED D3) ,PF4 (LED D3) | * D3 used for measurements status * D1 used for additional status |
| Operating Voltage | 3.3 V |  |
| Digital I/O push button | PJ1 |  |
| Baud Rate | 115200 bps |  |
| Microcontroller Pin Connections | PH0-PH3, PB2-PB3, 3.3V, GND, 5V | * Port H used for stepper motor * Port B used for ToF sensor * 3.3 V used for ToF * 5 V used for stepper motor |
| Stepper Motor Driver Pin Connections | IN1-IN4, + and - |  |
| Python Software Version | Python 3. |  |
| Python Libraries and Packages | Open3D |  |
| Communication Port for UART | COM 3 | * This varies across devices, and should be updated in the python file |

External Applications

|  |  |
| --- | --- |
| ***Application*** | ***Purpose*** |
| Keil | C and Assembly used to program microcontroller |
| Visual Studio Code | Used to run python code |

## 3 Detailed Description

### 3.1 Distance measurements

To collect distance data the ToF sensor uses LIDAR to measure the distance between itself and an object. This is accomplished by measuring the time it takes for an emitted pulse of light to reach an object and then be received/reflected back to the sensor. The distance traveled by the photon is then calculated using the speed of light. The formula is shown below:

Division by two is important in this case to account for twice the distance travelled (on the way there and back), to get the accurate distance travelled we need to eliminate the distance back to the sensor. Moreover, the speed of light is fixed in the above equation as .

The process to receive the complete coordinate data can be broken into a multistage process. First data is acquired by the ToF sensor. In this step, I2C protocol is used for the ToF sensor to communicate and transmit measured data to the microcontroller. I2C protocol requires two communication lines, to accommodate bidirectional data transfer. The serial clock line keeps track of the clock, and the serial data line transmits data to and from the ToF sensor. This stage also incorporates the stepper motor which rotates 360 degrees, with a total of 32 steps. At each step, the ToF sensor measures distance data, for a total of 32 measured values in a single rotation.

Next, the data is transmitted to the PC for manipulation. Using UART and ‘COM3’ as the communication port (which varies across devices), the data is sent to the PC at a baud rate of 115200 bps. Using this asynchronous data transfer method, the data is transmitted after each acquired measurement. To save time and increase efficiency, only the distance measurement is sent to the PC.

Using the python code, this measurement, along with a calculated angle, XYZ coordinates are generated. The X coordinate in this case is manually determined. It is set to 0 for the first frame and is incremented by 2000 mm for every subsequent frame. The Y Z coordinates are determined using equation 1 and 2, where .

Y = distance ∗ cos (1)

Z = distance ∗ sin (2)

X = X + 2000 (mm) (3)

It should be noted that the coordinates can be described as X being the depth value, Y being the vertical value and Z being the horizontal value.

Furthermore, this data is then written to a file in .xyz format using file I/O in python.

After one frame of data is collected, the mechanism is moved forward to capture a second frame. After the desired number of frames have been collected, the complete data can be visualized.

A screenshot of a computer program

Description automatically generated

Figure 2: Image of the data collection code

3.2 Visualization

After data points have been collected and stored in the correct file the data can be plotted using open3D. To run the visualization code, users will need a PC and software, that can effectively run python code. The visualization for this project was completed using Open 3D software and the python libraries numpy and o3d. From section 3.1, distance data was collected and xyz coordinates were generated and written to a text file named “datacapture.xyz”.

During the visualization process, this file is re opened and the points are extracted and stored as a point cloud representing spatial data points. Next, the number of frames is calculated by dividing the number of points by 32, since there are 32 pints per frame. This is a common practice in LIDAR scanning when completing rotational scans where multiple frames are generated for a 3-D scan. For each frame, the each point is connected to the next, forming a continuous loop. This step allows for visualizing the structure within each frame, making the geometric figures more visible. Finally, the corresponding points in consecutive frames are connected. This is essential for visualizing changes in the frames as the depth(X coordinate) increases. Lastly, these connections are compiled into a LineSet, a type of geometric representation in Open3D that consists of points and lines connecting these points. The LineSet is then rendered using Open3D’s visualization tools, producing an interactive 3D model that can be rotated, zoomed, and explored in detail.

A screenshot of a computer program

Description automatically generated

Figure 3: Image of visualization code

## 4 Application Note, Instructions, Expected Output

### 4.1 Instructions

The following steps can be followed to run the application:

1. Download and install a version of Python between 3.7-3.10, this is important to allow open3D to work.
2. Next, install pySerial and open 3D, this can be completed by entering the following lines in command prompt.

*pip install pySerial*

*pip install open3d*

1. Connect the microcontroller to the PC, mount the ToF sensor the stepper motor. An implementation of the mechanism can be seen in figure 7 and 8. Esure the wiring is completed according to the schematic in figure 7.This is important to ensure the functionality of the code.
2. Adjust the COMS variable in the “data\_collection.py” python file to match the used devices communication port.



1. Open the Keil Uvision project and load the code onto the microcontroller.
2. Run the “data\_collection.py” file
3. Hit the reset button on the microcontroller.
4. Read the terminal output, once prompted to start data collection press the onboard button “PJ1” on the microcontroller. The terminal will start to display distance data as well as the frame number.
5. Once a complete frame has been collected (32 steps). The stepper motor will rewind to its original location and the terminal will prompt the user to collect data again for a second frame.
6. Move the mechanism 2000mm forward to account for the change in depth and to capture a second frame.
7. Press PJ1 again and capture a second frame (32 points).
8. Repeat steps 8-10 for subsequent rotations. Users can capture as many frames as required.
9. Once satisfied with a number of frames. Close the terminal and open the “open3d\_visualization.py” file.
10. Run the code in the “open3d\_visualization.py” file. A popup will come up displaying a 3-dimensional sketch of the scanned space.

### 4.2 Expected Output

A sample space was scanned. This was the hallway on the second floor of JHE at McMaster University. The image of the space can be seen in figure 4. Using the steps outlined in section 4.1, a scan was generated for this space, which can be seen in figure 5 and figure 6.

A hallway with a door open

Description automatically generated

Figure 4: Image of hallway scanned as a reference

A black and white drawing of a plane

Description automatically generated

Figure 5: Scanned output when scanning the hallway in figure 4

A black and white drawing of a curved object

Description automatically generated

Figure 6: A view of the scanned space from an alternate angle

Additional Notes

Upon comparison, it is clear that there are small anomalies in the scan which are more visible in figure 6. This is possible due to the presence of reflective surfaces. These surfaces interfere with the measured distance. The way ToF sensors work by emitting a light signal and measuring the time it takes for the signal to reflect off an object and return to the sensor. However, when the emitted signal encounters a reflective surface, such as a mirror-like material or highly polished surface, it can bounce off at an angle that does not return to the sensor directly. This phenomenon, can cause the sensor to receive a weaker or distorted signal, leading to inaccuracies in distance measurement. Additionally, multiple reflections may occur when the signal bounces off multiple surfaces before reaching the sensor, further complicating the measurement process. This is a possible explanation for distortions that can be seen in the scan.

## 5 Limitations

Microcontrollers have limited precision in floating-point arithmetic due to hardware constraints, often leading to rounding errors and reduced accuracy, particularly with extreme values. Moreover, executing trigonometric functions like sine, cosine, and tangent on microcontrollers with limited processing power can significantly increase execution time, impacting real-time performance in time-sensitive applications. To counteract this issue, in this project, the data is communicated through UART to the PC. Then, the Python math library is used to complete the trigonometric functions in order to derived coordinate values. This avoids type casting and retains an accurate measured value.

The maximum quantization error for each tof module can be calculated, The measured data from the ToF sensor is a 16 bit integer:

Short Range Distance Mode (1360 mm)

0.0208 mm/bit

Medium Range Distance Mode (2900 mm)

0.0443 mm/bit

Long Range Distance Mode (3600 mm)

The communication method used between the ToF sensor, and the microcontroller was I2C serial communication protocol to transfer data.

## 6 Circuit Schematic

A diagram of a computer

Description automatically generated

Figure 7: Circuit Schematic

## 7 Programming Logic Flowchart

## Appendix

A cardboard box with wires and a black label on it

Description automatically generated

Figure 8: 3-D space scanner can be mounted to a box using double sided tape

A box with wires and wires

Description automatically generated

Figure 9: ToF mounted onto stepper motor using Lego, elastics and double sided tape