COMP ENG 2DX3: Microprocessor Systems Project
Deliverable 2 Report

Instructor: Dr. Yaser M. Haddara, Dr. Shahrukh Athar, Dr. Thomas Doyle

David Cosentino

Device Overview

(a) Features

- MSP432E401Y SimpleLink™ Ethernet Microcontroller
  - 120-MHz Arm® Cortex®-M4F Processor Core with Floating-Point Unit (FPU)
  - Bus Speed: 80 MHz (Default 120 MHz)
  - 256KB of SRAM With Single-Cycle Access
  - +5V Operating Voltage
  - Two 12-Bit SAR-Based ADC Modules each supports up to 2 Msps
  - Eight Universal Asynchronous Receivers/Transmitters (UART)
  - Ten Inter-Integrated Circuit (I^2C) Modules with High-Speed Mode Support
  - Baud Rate: 115200 bps
  - Flash Memory: 1024 KB
  - EEPROM: 6 KB
  - Cost: $73.40 CAD
- VL53L0X Time-of-Flight (ToF) distance sensor
  - Up to 400 cm distance measurement
  - Up to 50 Hz ranging frequency
  - Typical full field-of-view (FoV): 27 °
  - Operating Voltage: +3.3 V
  - Cost: $25.30 CAD
  - Serial Communication: I^2C
- Miscellaneous
  - 4-Pin momentary push button
  - 28BYJ-48 Stepper Motor
  - ULN2003AN Driver Board

(b) General Description

This device is an embedded spatial measurement system that uses a stepper motor with the VL43L1X ToF sensor mounted on top to collect the distance data. Based on the data collected, this device will create a 3D model of the scanned area. The distance measurements are collected by measuring the time it takes for a photon released from a sensor to reach back after hitting a surface using a predefined formula. The ToF sensor then conducts the required analog to digital conversion and then transmits the resulting data to the microcontroller serially via the I2C protocol. The data is then serially transmitted to a PC serial port by the UART protocol. The data is then reconstructed into an image with a Python program. In order to reconstruct an effective 3D model, multiple measurements must be taken. To operate this device, connect the microcontroller to a PC then place the box on the ground or an elevated surface such as a chair. Press the reset button on the microcontroller and begin the Python program. The program will prompt the user to enter the number of scans they want to do and the total distance in cm that they will be measuring. Once this data is collected, the program will ask the user to push the button to begin measurement. The stepper motor will then begin turning for one 360° rotation, pausing every 11.25° to allow for the ToF to take a measurement. After this rotation, the motor will reset itself by rotating 360° in the opposite direction, returning to the home position. The user must then physically displace the box push the button again once the python code prompts to collect another set of data. Once the max distance has been reached, the program will terminate and generate the 3D plot of the room

(c) Block Diagram

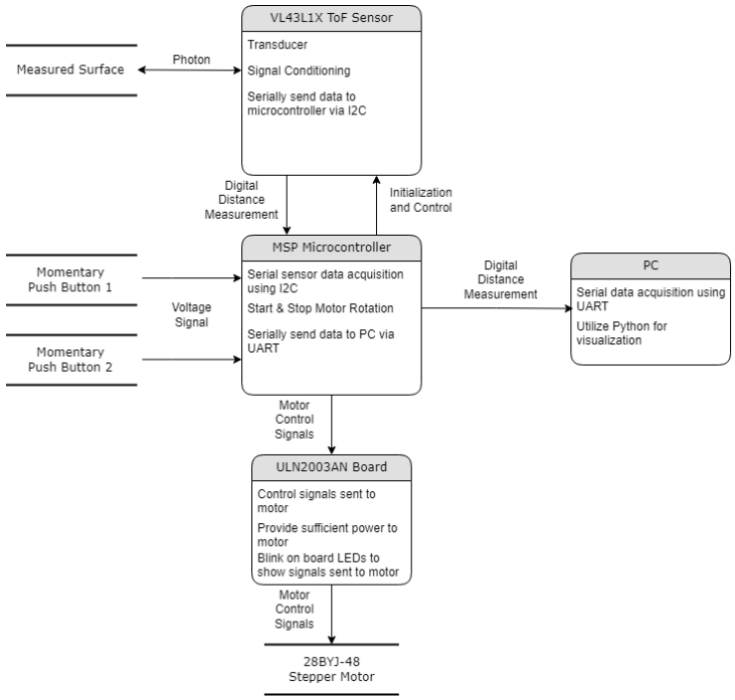

Figure 1: Data Flow Graph of the Device

Device Characteristics Table

Property	Value
Max Measurement Distance in The Dark	360 cm
Max Measurement Distance in Ambient Light	73 cm
Distance Measurements	1 sample per 11.25 degrees
Baud Rate	115 200 bps
Bus Speed	80 MHz

Detailed Description

(a) Distance Measurement

The VL43L1X ToF sensor measures distance using Light Detection and Ranging (LIDAR). The device first releases a photon at a wavelength of 940 nm from its emitter. The pulse then bounces off a nearby surface and returns to a detector on the sensor. The time from the photon being emitted to reaching the detector is then divided by 2 to get the time it takes to travel in one direction from the sensor to the surface. This time measurement is then multiplied by the speed of light (~300 000 000 m/s) to get the distance between the surface and the sensor. This can be summed up by this formula and an example:

$$Distance = Photon\ travel\ time/2 \times speed\ of\ light$$

$$Distance = 6.6667\ ns/2 \times 300000000\ m/s \approx 1000\ mm$$

This distance is then returned as a measurement in millimeters. For this project, the default long distance mode was capable of measuring surfaces from 360 cm in the dark and 73 cm in strong ambient light. The sensor supports the I2C serial communication protocol, which was used to communicate the data collected to the microcontroller. A set of measurements would consist of a 360° rotation, with the motor pausing every 11.25° to stop, initiate, and collect a measurement from the sensor. 32 samples would be collected and communicated serially to the microcontroller, which would then use the UART protocol to communicate the data to a PC.

A Python program will open and read the serial port collecting data (e.g., COM3). The program keeps track of the current angle of the sensor and uses this angle to break up the measurement into y and z components. This is done by multiplying the distance measurement by cosine to get the y coordinate and sine to get the z component. The x component is also incremented by a constant amount after each set of measurements for one plane is collected. This incrementation was reflected in the physical world by moving the device by 20 cm after each set of measurements. This allows us to collect data for the x, y, and z coordinates so that our program can recreate an xyz plane. The data is saved to a .XYZ file where each line of data corresponds to a measurement of a set. For example, a measurement from the second set of measurements could be:

$$x = 200\ mm$$

$$y = 200 \cos 90\ mm$$

$$z = 200 \sin 90\ mm$$

(b) Visualization

Visualization was achieved by utilizing Open3D and numpy in Python. Open3D is a Python module that allows for 3D visualization by providing several functions and features that a programmer can take advantage of. Open3D can take the data from our .XYZ file, convert it into a point cloud, and display it to the user. A point cloud exists in a 3D space and is all the recorded coordinates mapped within that space. Once this point cloud was created, lines were used to join the points together to reconstruct the 3D space. This was done by using a nested loop structure. The inner loop would increment each point and store it into an array, repeating for each measurement in a set. The outer loop would repeat this loop for each set of data collected. The numpy library was then used to convert this array into a format suitable for Open3D to utilize. Open3D was then able to construct the 3D space that was mapped out using our device. The function o3d.io.read_point_cloud() was used to read the .xyz file that contains our data and construct a corresponding point cloud. The function open3d.visualization.draw_geometries() was used to display this point cloud and reconstructed 3D space to the user. The function open3d.geometry.LineSet() was used to connect the points collected with lines. The function open3d.utility.Vector3dVector() and open3d.utility.Vector2iVector() was used to convert a numpy array into Open3D format. The function np.asarray() was used to convert our Python arrays into numpy arrays.

Instructions, and Expected Output.

For this device, the x-direction will be the direction of displacement of the device. The y and z direction will be collected from each scan by the device forming the vertical distance slices. This

device can be used to measure difficult to reach spaces, especially if mounted on a remote-control device that can move. The following is an explanation of how to utilize the device with an example:

1. Connect device from USB port on PC to microUSB port on microcontroller. You must connect the microUSB end of wire to the port that is opposite of the ethernet port on the microcontroller.
2. If your device does not already have the operation code on it, use Keil to translate, build, and flash instructions to microcontroller.
3. Hit the reset button on microcontroller.
4. Run the "2DX Data Collection" Python file. Ensure the proper COM port is open (Default is COM3)
5. Enter the total length of the room you are trying to scan.
6. Hit enter in the command line of the python file to begin collecting data.
7. Press the button to begin measurement.
8. Once a measurement is complete, displace the device 40 cm in one direction, then repeat steps 6 & 7 until enough data is collected.
9. Once complete, run the "2DX Graphing" Python file to reconstruct the area that was measured. This program will output both the corresponding point cloud and the reconstructed area.

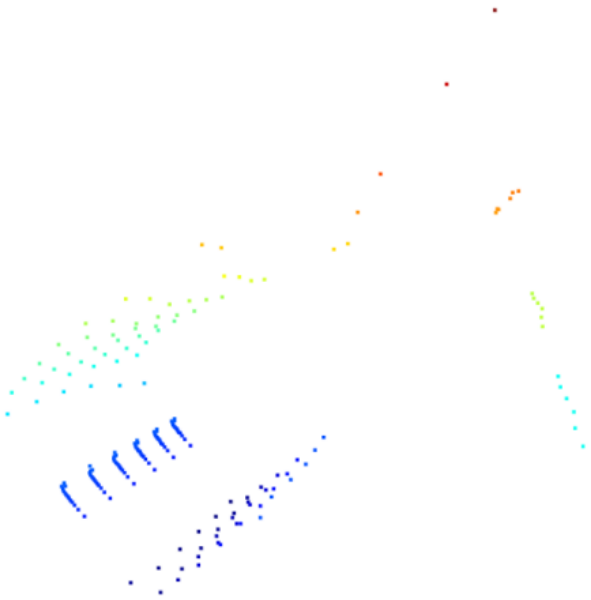

Figure 2: Point Cloud generated after scanning

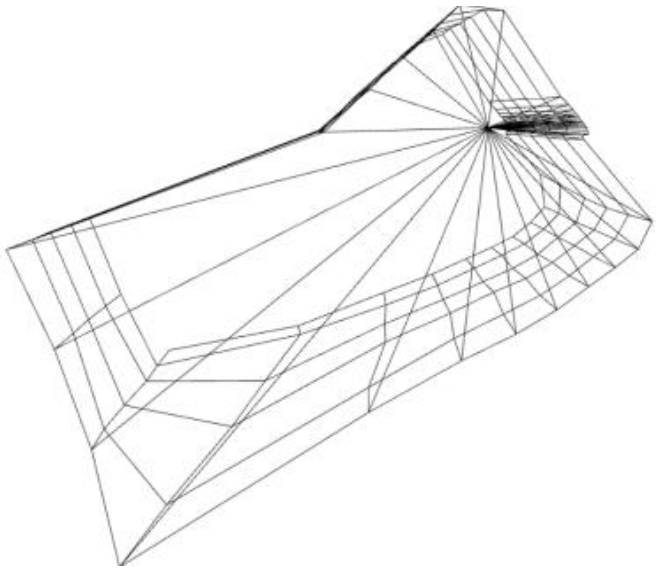

Figure 3: Line Set generated from the point cloud. This is the final room rendering.

Limitations

Microcontroller Floating Point Capability and Use of Trigonometric Functions

Our microcontroller only supports single precision floating point where the data occupies 32 bits. To contrast this, double precision floating point allows us to store floating point numbers with 64 bits. It is possible to implement trigonometric functions on the microcontroller by importing the math.h library. However, this would use extra space on the microcontroller and, in combination with the floating-point limitations, will cause data to be lost. This is not a problem for our purposes as data was stored as an integer and sent to a PC. The Python program on the PC was able to conduct the necessary trigonometric functions without any loss of data

Maximum Quantization Error of ToF Module

Max quantization error is the maximum possible error present when digitizing values. First, assume the full-scale voltage, VFS, is 3.3 V and the number of bits is 16. The maximum quantization error of the ToF module can be calculated to be:

$Max\ Quantization\ Error = V_{fs}/2^n = 3.3\ V/\ 2^{16} \approx 5.04 \times 10^{-5}$

Maximum Standard Serial Communication Rate that can be Implemented with PC VS Speed Implemented and Verified

The maximum standard serial communication rate that can be implemented with the PC is 128000 bps. This can be confirmed by using an AD2 to view the serial port while using different speeds. The speed that was implemented for this project is 115200 bps and verified using an AD2.

Communication Methods and Speed used between Microcontroller and ToF Module

The communication method used between the microcontroller and ToF module was the I2C serial protocol. In this protocol, there are 8 data bits, a start bit, stop bit, an address, read/write bit, and acknowledgement bits. The speed that was used was 115200 bps.

Primary Limitation on System Speed

The primary limitation on system speed is the stepper motor and ToF sensor. This can be determined as it has a max sampling rate, so the stepper motor must pause for enough time to allow the sensor to collect its data. This can be evaluated by increasing stepper motor speed, the result will be inaccurate data. This problem can be solved by utilizing multiple ToF sensors. For example, using 2 ToF sensors will cut the duration of measurement by half.

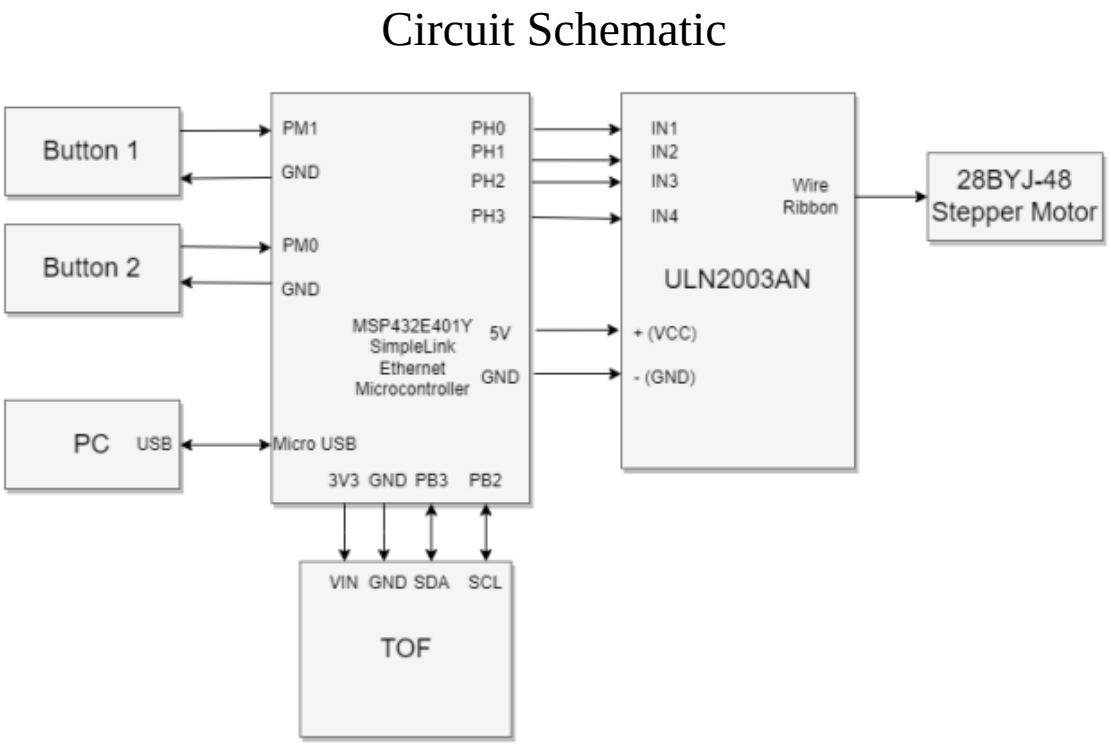

Figure 4: Final Circuit Schematic for prototype

Programming Logic Flowchart(s)

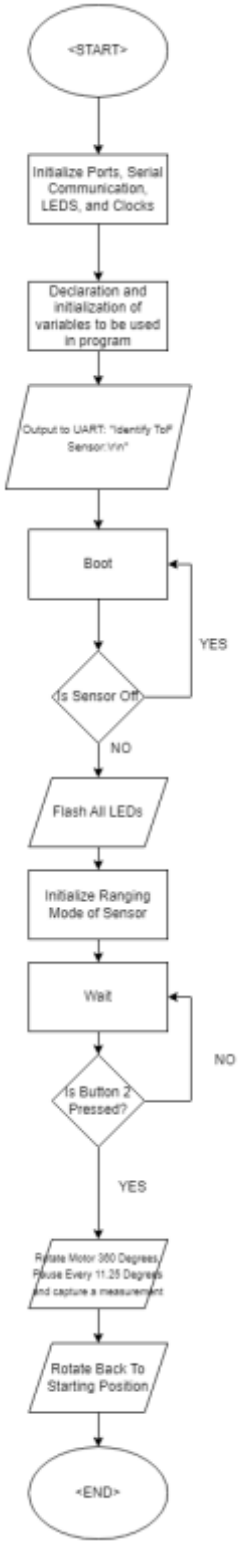

Figure 5: Keil Code Flowchart

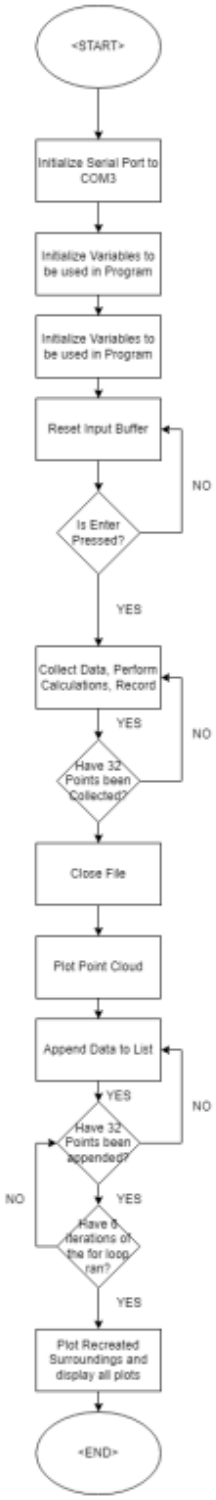

Figure 6: Python Code Flowchart