Observe, Reason, Act: Spatial Mapping Using Time-Of-Flight

Kevin Le

McMaster University

COMPENG 2DX3

Instructors: Dr. Athar, Dr. Doyle, Dr. Haddara

April 17th, 2023

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 [Kevin Le, 400385350]

1 Device Overview

1.1 Features

Microcontroller
- Texas Instruments MSP432E401Y
- 120-MHz Arm® Cortex®-M4F Processor Core (Clocked down to 20MHz)
- Universal Serial Bus (USB): USB 2.0 OTG
- Eight Universal Asynchronous Receivers/Transmitters (UARTs), each with independently clocked transmitter and receiver (Baud rate used: 115200 bits/s)

Time-Of-Flight Sensor
- VL53L1X
- Up to 400cm distance measurement
- Up to 50 Hz ranging frequency
- I²C interface (up to 400 kHz)
- Operating voltage: 2.6V to 3.5V

Stepper Motor
- 5-wire unipolar stepper motor
- 34.4 mN.m of torque at 15 RPM
- Number of phases: 4
- Operating voltage: 5VDC
- Operating current: 240mA
- Gear ratio: 64:1

Reset Button
- Operation of the stepper motor and ToF sensor stops if is currently active
- The current position becomes the home position

Button 1
- Start/stop button
- Press once to start rotation and scanning
- Press during operation to stop rotation and scanning

Button 2
- Data acquisition button
- Press once to begin data acquisition

1.2 General Description

This device creates a detailed map of a narrow indoor environment such as hallways. The device is intended for use as a component of other systems such as robotics navigation, autonomous drone and layout mapping. The device has one push button that starts data acquisition and one push button that starts and stops the stepper motor rotation and ToF sensor measurement. LEDs show the ready status of the ToF sensor, distance measurement success status and push button status.

The displacement of the sensor must be changed manually by the user and the X-coordinate value must be updated manually. The device automatically measures and calculates the Y-coordinate and Z-coordinate values.

A 3D model is created once the user finishes the measurements, displaying the graphical representation of the measured space.

1.3 Block Diagram

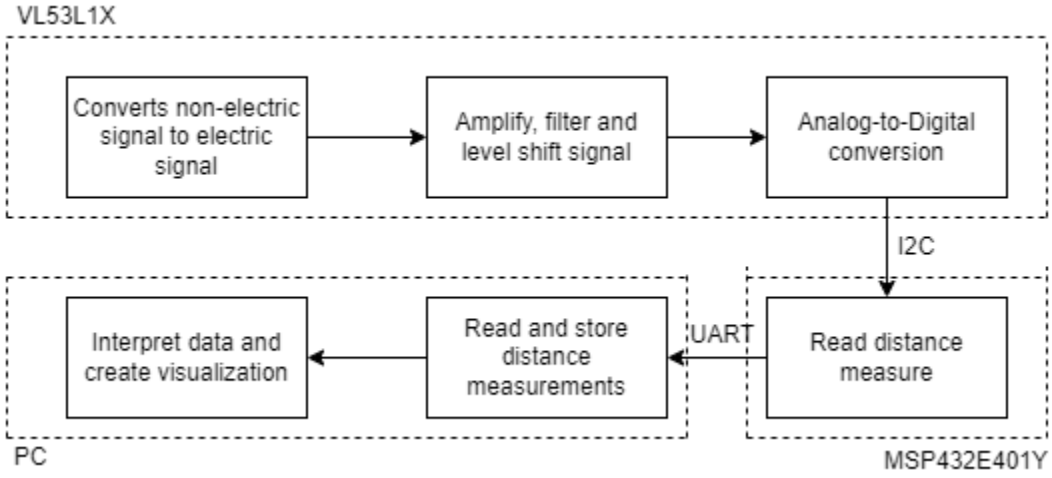

Figure 1-1. Device's block diagram

2 Device Characteristics

Feature	Description
Bus speed	20 MHz
Baud rate	115200 bits/s
Measurement success status	LED 1 (PN1)
ToF data-ready status	LED 2 (PN0)
Stepper motor pins	PH0-PH3
Button 1 pin	PM0
Button 2 pin	PM1
SCL/SDA pins	PB2/PB3
Python Software	Version 3.10
Python Libraries and Packages	Pyserial, Open3d, xlwt, xlrd, numpy, math
Communication port	COM5

Table 2-1. Device Characteristics Table

3 Detailed Description

3.1 Distance Measurement

The VL53L1X ToF sensor is the core component for the distance measurement of this device. The ToF sensor emits a 940 nm invisible laser that hits surfaces and reflects back to the receiver of the device. The travel time divided by two, multiplied by the speed of light equals the distance from the measured surface to the sensor.

Data captured from the ToF sensor goes through a process to convert non-electric signals to digital signals. First, the transducer of the device converts the non-electric signals to electric signals. Then, the signal undergoes signal conditioning that amplifies, filters and level shifts the signal to prepare for Analog-to-Digital conversion (ADC). Next, the modified electric signal is converted to a digital form that can be interpreted by the MSP432E401Y microcontroller. Data from the ToF sensor is transmitted to the microcontroller using the I2C protocol, a serial communication protocol that requires synchronization.

The microcontroller communicates with the PC through UART, an asynchronous, serial communication protocol at a baud rate of 115200 bits/s. Data sent from the microcontroller is received character by character. This means distance measurements are received digit by digit and the PC needs a way to know when a distance measure finishes. The microcontroller is configured to send a '!' character at the end of every distance measurement and the detection of the '!' character ends the appending of digits to the string of digits.

The device has 2 push buttons, both in active-high configuration with a 1k ohm pull-down resistor.

Button 1 when pressed, sends a logic-1 to port PM0. When the device is powered on, PM0 is checked, through polling, for a logic-1 signal. When button 1 is pressed and PM0 receives a logic-1 signal, the stepper motor begins to rotate and the ToF sensor begins to collect data. As the device operates, a polling process checks PM0 every step (512 steps per rotation) for a logic-1. Pressing button 1 sends a logic-1 to PM0 and the operation stops.

Button 2 when pressed, sends a logic-1 to port PM1. When the device is powered on, PM1 is checked through polling for a logic-1 signal. When PM1 receives a logic-1 from button 2, the microcontroller sends the character '@' to the PC through UART. The program *distancetoexcel.py* checks for the character '@' through UART and moves on only when it has received '@' sent from the microcontroller.

After both button 1 and button 2 are pressed, the stepper motor begins rotation and the ToF sensor measures distance every 1/512th of a full rotation or 0.703 degrees.

After one full rotation and 512 measurements, the variable *count* in the *distancetoexcel.py* program equals 512 and the condition "if count == 512" becomes true. The user is asked to input the X-coordinate of the next plane of measurement or the character 's' to stop the program. Entering an X-coordinate resets the variable *count* to 0 and waits for push button 1 to be pressed. The infinite while loop that checks for UART data and writes to *dataset.xls* continues. Entering 's' ends the *distancetoexcel.py* program and data in *dataset.xls* will not be modified unless *distancetoexcel.py* is run again.

3.2 Visualization

The Python module *excelto3d.py* handles the data in the *dataset.xls* file to generate a 3D model of the data points.

The program starts by creating a new xyz file named *points.xyz* and saves it. For every point that needs to be added, *points.xyz* is opened in append mode and the new point can be added. This is safer and prevents data loss compared to writing all points at once and saving at the end.

Row 1 and column C onwards contain the X-coordinate for that column. Row 2 to row 513 of each column contains 512 measurements used to identify the Y and Z-coordinates. Every measurement in a column corresponds to an angle from 0 to 360 degrees. Utilizing principles from trigonometry, the Y and Z-coordinates can be calculated with the equation in Figure 3-1 below.

```
y = distance * (math.cos(angle)) / 1000
z = distance * (math.sin(angle)) / 1000
```

Figure 3-1. Equations used to calculate Y and Z coordinates of every point

Note that the angle is in degrees (0 to 360) and must be converted to radians (0 to 2*pi) before performing the calculation for *y* and *z*.

To create a visual with lines connecting all points to form a mesh, the array *mesh* is created. It stores arrays of 2 integers. These integers represent points in the *points.xyz* file. If only one rotation is measured, 0 would represent the first point and 511 would represent the last point. By connecting adjacent points (0 & 1, 1 & 2, …, 511 &512) as well as points with the same angle in adjacent planes (0 & 512, 512 & 1024, 1024 & 1535), an accurate 3D representation of the measured environment is created.

4 Application Example

4.1 Example

1. Place the device on a mobile, elevated surface such as a chair or light desk, in the center of the hallway being mapped
2. Load the C program in Keil uVision5 to the microcontroller
3. Press the reset button
4. Run the *distancetoexcel.py* module
5. Press button 2 to begin data acquisition
6. Press button 1 to begin the stepper motor and ToF sensor operation
7. After a full rotation, the Python console will ask for user input. Enter ‘2000’ and move the device forward 2m (2000mm)
8. Press button 1 to begin the stepper motor and ToF sensor operation once again
9. After a full rotation, the Python console will ask for user input once again. Enter ‘4000’ and move the device forward 2m.
10. Press button 1 to begin the stepper motor and ToF sensor operation
11. After a full rotation, the Python console will ask for user input. Enter ‘s’ to stop the program. All measured data are stored in the Excel file *dataset.xls*.
12. Run the *excelto3d.py* module
13. A 3D visual of the hallway will be generated

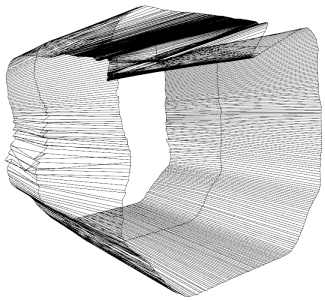

Figure 4-1. 3D visualization generated from the above example

Figure 4-2. Photograph of the mapped-out hallway

4.2 User Guide

1. Refer to figure 6-1, connect the microcontroller, ToF sensor, stepper motor and push buttons to their corresponding connections
2. Mount the stepper motor to a stable container such as a tissue box
3. Mount the ToF sensor to the stepper motor using cardboard and hot glue
4. Place the microcontroller in the tissue box and keep the breadboard with the push buttons on the outside
5. Identify the UART port number of the PC and change the default value of 'COM5' to the corresponding port number if needed.

```
ser = serial.Serial(
    port='COM5',\
    baudrate=115200,\
    parity=serial.PARITY_NONE,\
    stopbits=serial.STOPBITS_ONE,\
    bytesize=serial.EIGHTBITS,\
        timeout=0)
```

Figure 4-3. Section of code to be changed in step 5, found in distancetoexcel.py file

5 Limitations

1. **Can the device operate in all lighting conditions?**

No. The device performs best in the dark with a maximum distance of 360cm. Under strong ambient light, the maximum distance reduces to 73cm. Targets with high reflectance (White 88%) also have a higher maximum distance compared to targets with low reflectance (Grey 17%).

2. **What is the accuracy for each distance measurement?**

In dark conditions, the VL53L1X ToF sensor has a ranging error of ± 20mm. Ranging error takes into account both accuracy and repeatability errors. The minimum ranging distance is 4cm and under this minimum distance, a target will be detected but the measurement will not be accurate.

3. **What is the limitation on the bus speed of the microcontroller?**

The default settings of this device set the bus frequency of the microcontroller to 20 MHz however the bus frequency can be modified by modifying *PSYSDIV* in the *PLL.h* file found in the *Project_Keil_Files* folder. To increase the bus frequency to 120 MHz, change PSYSDIV to 3.

```
// The #define statement PSYSDIV initializes
// the PLL to the desired frequency.
#define PSYSDIV 23
// bus frequency is 480MHz/(PSYSDIV+1) = 480MHz/(3+1) = 120 MHz
// IMPORTANT: See Step 6) of PLL_Init().  If you change something, ch
// IMPORTANT: You can use the 10-bit PSYSDIV value to generate an ext
```

Figure 5-1. Default value of PSYSDIV for this device. Sets bus frequency to 20 MHz.

PSYSDIV	SysClk (Hz)
3	120,000,000
4	96,000,000
5	80,000,000
7	60,000,000
9	48,000,000
15	30,000,000
19	24,000,000
29	16,000,000
39	12,000,000
79	6,000,000

Figure 5-2. Table guide for changing the value of PSYSDIV

6 Circuit Schematic

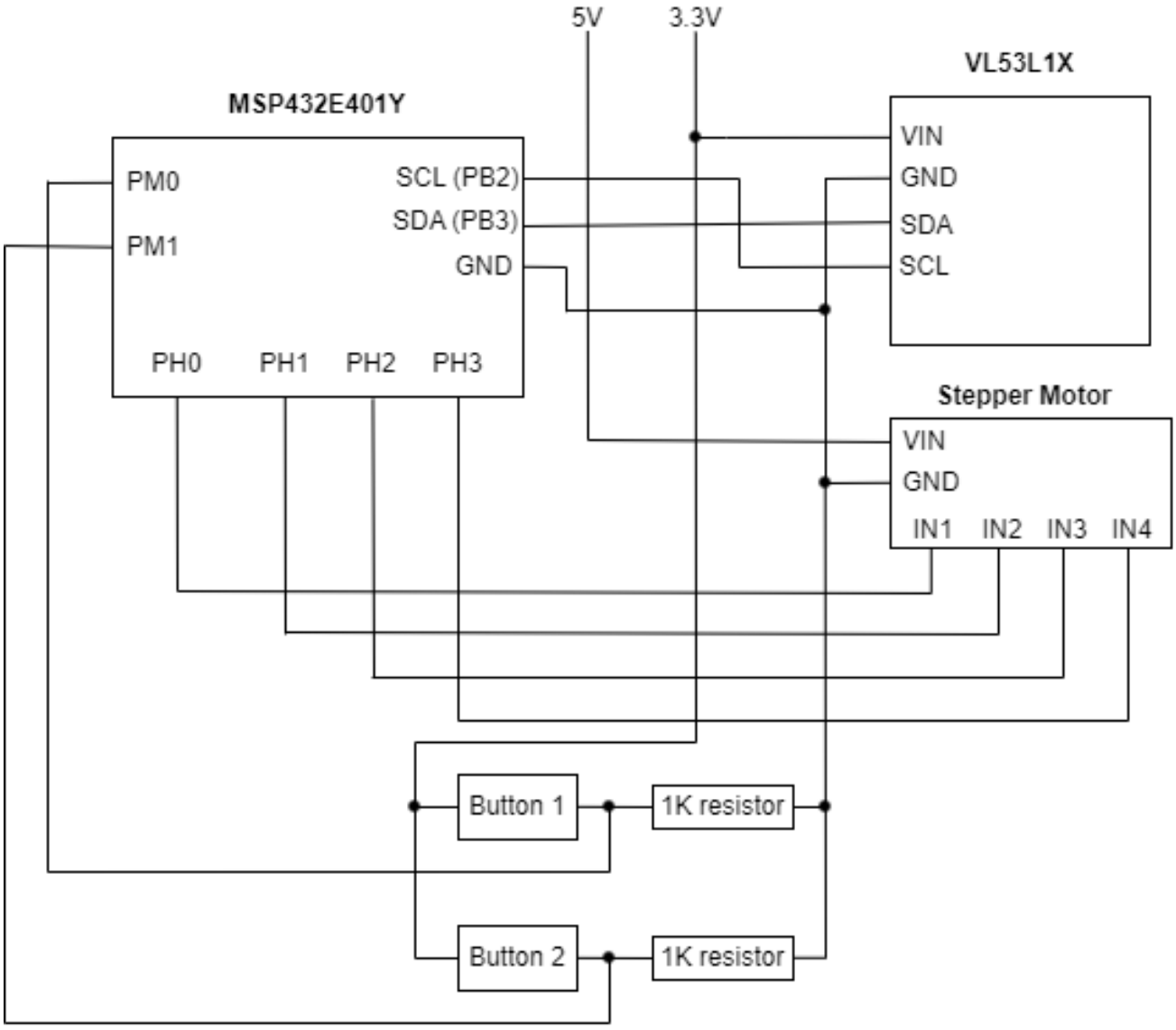

Figure 6-1. Circuit Schematic

7 Programming Logic Flowcharts

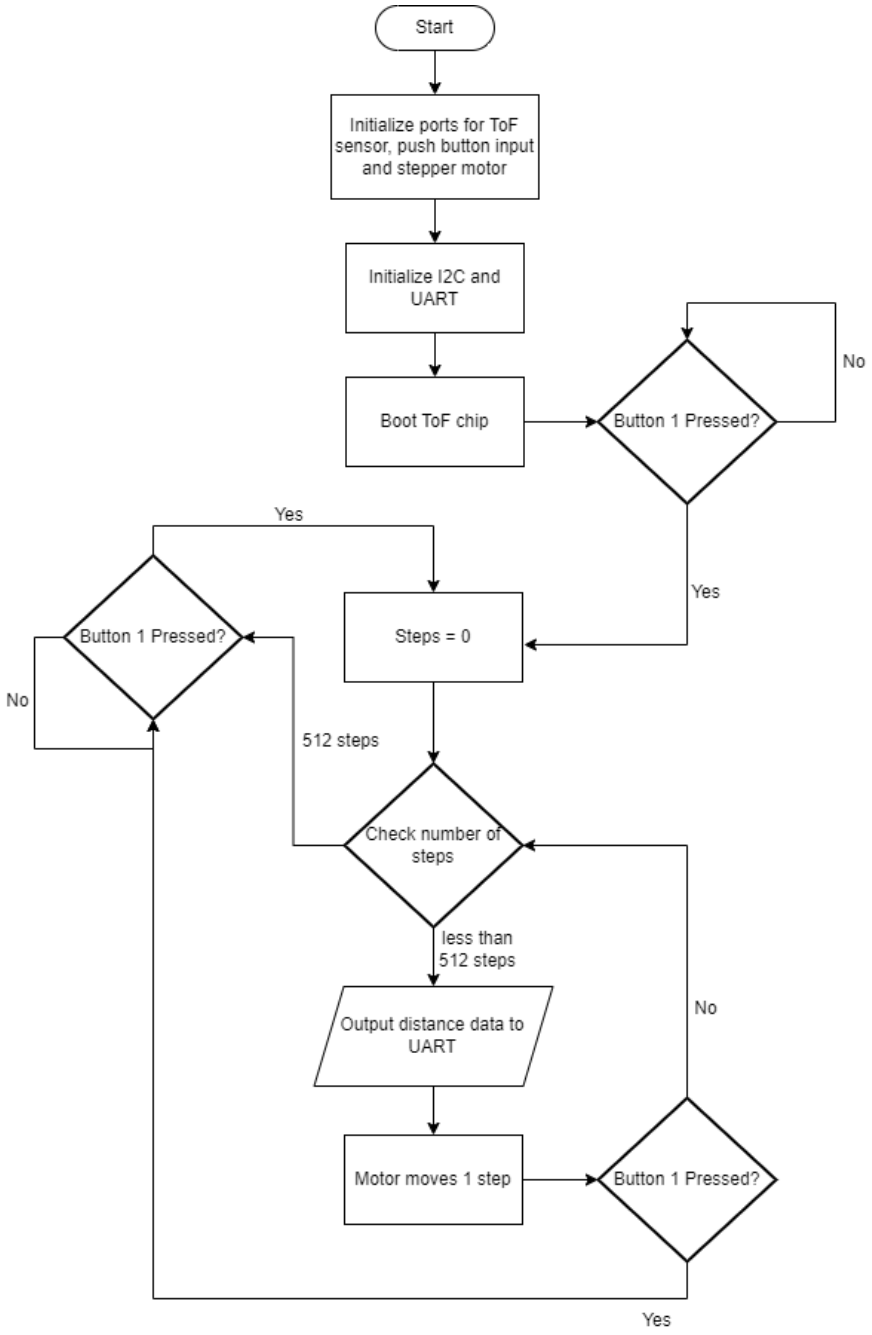

Figure 7-1. Flowchart of microcontroller program

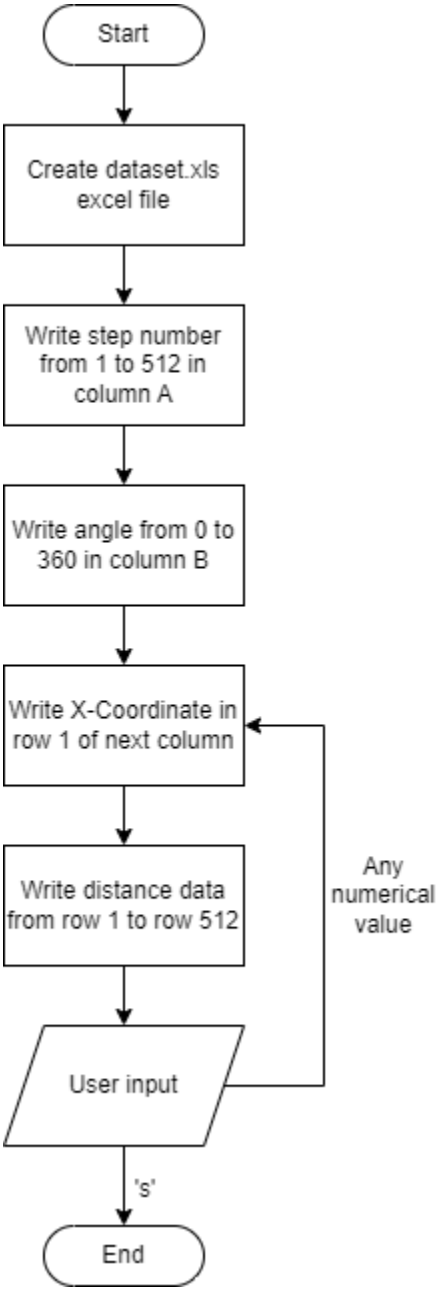

Figure 7-2. Flowchart of distancetoexcel.py program

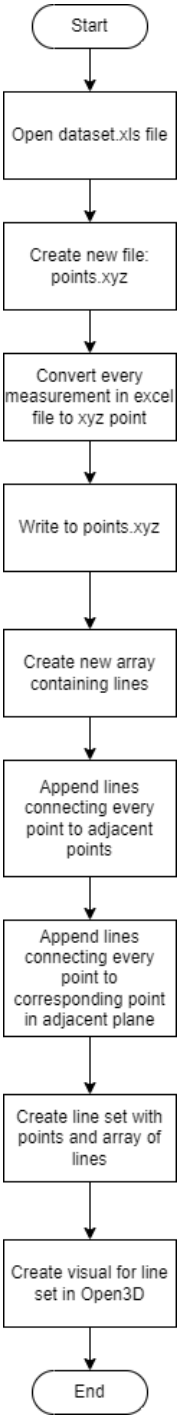

Figure 7-3. Flow chart of excelto3d.py program