Comp Eng 2DX3
Final Deliverable Report

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

Stefan Tosti - L02 - Tostis - 400367761

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
[Stefan Tosti, Tostis, 400367761]

TOSTIS-400367761

Integrated LiDAR Room Scanner

The TOSTIS-400367761 is an integrated LiDAR room scanner that consists of various off the shelf components. Driving the system is the Texas Instrument MSP432E401Y Microcontroller, alongside 2 external push buttons, a mini breadboard, and a Time Of Flight sensor mounted to a stepper motor to capture full, 360° distance data.

Features

- Live data transmission from TOF sensor to Microcontroller via I2C
- Live data transmission from Microcontroller to PC via UART
- Communication at 115200 BPS baud rate
- 60 Hz internal clock operation speed
- Measurement status LED (PN1) and additional status LED (PF4)
- Input voltage supply range from 3.3V to 5V
- 2-push button user interface, to control rotation and data enable
- Fully 3D, automated plot generation via MATLAB

Microcontroller

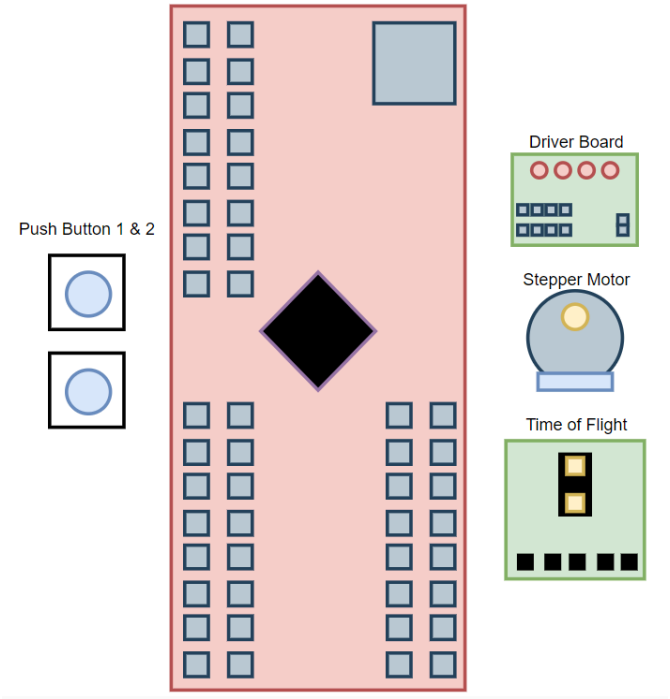
Parameter	Value
Clock Speed	60 MHz
Measurement & Status LED	PN1, PF4
Push Button 1 & 2	PM0, PM1
Serial Port	COM4
Baud Rate	115200 BPs

Stepper Motor

Driver Board Pin	Microcontroller Pin
In1	PH0
In2	PH1
In3	PH1
In4	PH1
V+	3.3V
V-	GND

Time Of Flight Sensor

ToF Pin	Microcontroller Pin
Vin	3.3V
GND	GND
SDA	PB3
SCL	PB2



Device Overview - Part A: Features

The system provides a very simple interface, consisting only of 2 external push buttons, and 2 LEDs that indicate the state(s) of the system

The **Texas Instrument MSP-EXP432E401Y** has the following features…

- Acts as a central hub for communication between the TOF sensor and the Computer that is used to data visualization
- Features a 32-bit Cortex M4F CPU
- 256 Kb SRam, 1024 Kb Flash Memory
- 60 MHz bus speed
- GPIO Pins and On-board LEDs

The **VL53L1X Time of Flight Sensor** has the following features…

- Top end range of 4m
- Operating frequency of 50Hz
- 2.6V - 3.5V operating voltage

The **28BYJ-48 Stepper Motor** has the following features…

- 4 phases per step
- 512 total steps for a full 360 rotation
- 4 On board LEDs indicating current phase
- 5 - 12V supply voltage

The 2 **Off Board push buttons** have the following features…

- 3.3V supply voltage
- Push button 1 enables the motor to preform a full 360 degree rotation, taking data at every 11.25 degrees. PN1 (LED1) will blink every 11.25 degrees
- Push button 2 enables the data capture of the motor, allowing it to actually capture plot data. PF4 (LED3) will light up when data capture is enabled

Serial communication has the following features…

- I2C protocol used for communication between TOF sensor and board
- UART protocol used for communication between board and PC
- 115200 BPS baud rate is used for UART communication

3D visualization has the following features…

- MATLAB is used for serial communication between board and PC
- MATLAB is used for data reading, point plotting, and 3D reconstruction

Device Overview - Part B: General Description

This embedded device provides the user with a low-cost, easy to operate method for 3D visualization. The on board ToF sensor captures data along the yz plane every 11.25 degrees. Each time a data measurement is taken, the onboard LED1 (PN1) will flash, indicating that a distance measurement is being read, and the motor will momentarily pause. In order to commence a full rotation, off board button 1 should be pressed. To enable the system, and allow it to take distance measurements, off board button 2 should be pressed. Pressing this button will turn on LED3 (PF4) on the board, indicating that data capture is enabled. To prevent wires from tangling, each time the sensor has finished a full rotation in one direction, the subsequent rotation will be in the opposite direction.

Each time the sensor takes a reading, the data is sent to the board via I2C communication, and then sent to the external PC via UART communication. This data is then read by MATLAB, and saved into the program's local variables.

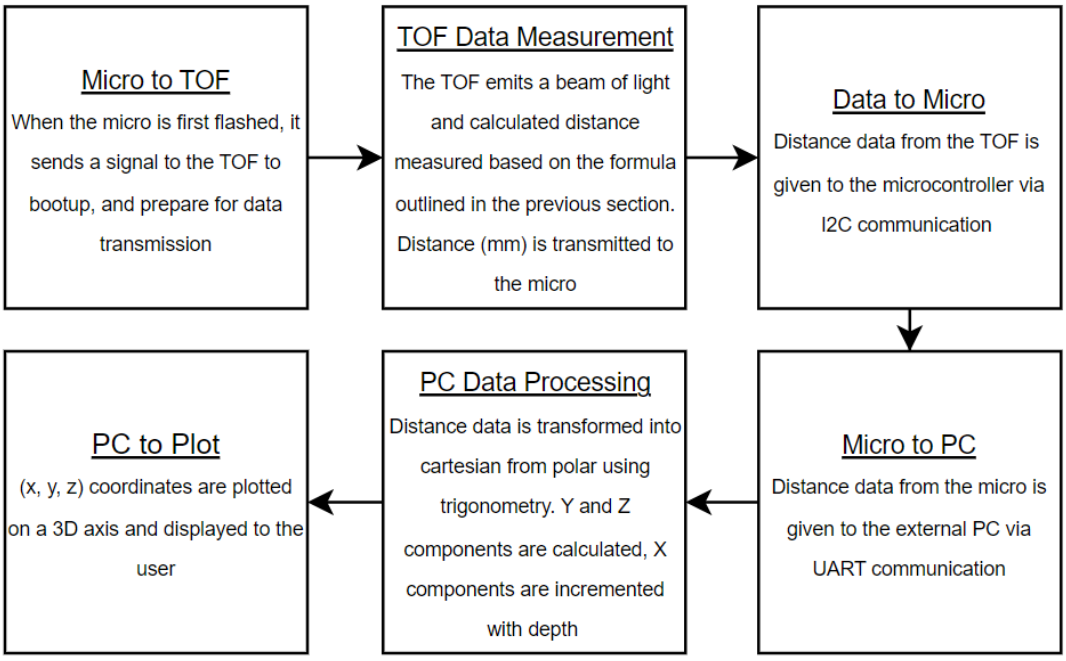
The brains of the operation is the MSP-EXP432E401Y microcontroller. Amongst other things, the microcontroller keeps the timing of all of the operations, and dictates a bus speed of 60 MHz.

The ToF sensor determines distance measurements by emitting a beam of light, and then measuring the time it takes for that beam to bounce back. This is done using the formula

$$Measured\ Distance\ = \frac{Photon\ Travel\ Time}{2} \times Speed\ of\ Light$$

The incoming data is manipulated directly in MATLAB. The data is first converted from polar form into cartesian form using trigonometry. Then, the Y and Z components are extracted from these calculations, while the X coordinates are manually incremented to simulate a depth. This data is then plotted on a 3D cartesian axis, and displayed to the user.

Device Overview - Part C: Block Diagram



Device Characteristics

Microcontroller		*Stepper Driver Board*		*Time of Flight*	
Clock Speed	60 MHz	**In1**	PH0	**Vin**	3.3V
Measurement and Status LED	PN1, PF4	**In2**	PH1	**GND**	GND
Push Button 1 & 2	PM0, PM1	**In3**	PH2	**SDA**	PB3
Serial Port	COM4	**In4**	PH3	**SCL**	PB2
Baud Rate	115200 BPs	**V+**	3.3V		
		V-	GND		

Detailed Description - Part A: Distance Measurements

The VL53L1x Time-of-Flight sensor operates by emitting a beam of infrared light using its "emitter". This beam of light reflects off an object in close proximity and returns to the "receiver". By measuring the time delay between the emission and reception of the beam, the sensor is able to determine the distance between itself and the object. The equation used to obtain this distance measurement is $D = \frac{in\ flight\ time}{2} \times speed\ of\ light$. Following this measurement, the sensor performs a variety of operations, including transduction, conditioning, and Analog-to-Digital Conversion, resulting in a digital representation of the analog distance measurement. This information is then transmitted to the microcontroller via I2C serial communication.

To activate the ToF sensor and initiate ranging, the ToF sensor boot function is invoked in the Keil C Program. Although the sensor is now active and primed to take distance readings, this process does not occur immediately. Instead, the microcontroller waits for a trigger from its peripherals through a polling method. This occurs with the use of 2 external push buttons. The first push button, PM0, is a stepper motor control button which is used to rotate the stepper motor a full 360 degrees. The second external button, on PM1, toggles a variable called scanEnable, as well as an additional status LED that lets the user know that the system is ready to take data measurements.

Inside the main block of my C code, we control the distance measurements being taken. We first check if the motor has rotated 11.25 degrees, by checking the total number of steps with the modulo of 16 (This is because a full rotation is 512 steps or 360 degrees. This gives us a value of 0.703125 degrees per step. Thus, there are 11.25 / 0.703125 = **16 steps per 11.25 degrees).** The program checks if the motor has rotated 11.25 degrees, flashes a measurement LED, and sets a scan variable which tells the program to take a distance measurement. Within Keil, the data that is transmitted is RangeStatus, Distance, Step, Depth and SpadNum. RangeStatus, Distance, and SpadNum are all measurements that are retrieved from the TOFs API functions. Depth and Step are 2 variables that are controlled within the program. The step variable is incremented by one every time the motor moves a single step. Thus, when the step is any multiple of 16, we know the motor has moved 11.25 degrees. This gives us an effective way to determine the angle of the motor. The Depth variable is incremented by 1 every time the motor completes a full rotation, thus allowing us to simulate the depth of the hallway that we scan. These 4 variables are then transmitted to the board via I2C, and then to the PC via UART.

Detailed Description - Part B: Visualization

Once the various pieces of data have been transmitted to the computer, MATLAB is responsible for the rest of the processing. First, MATLAB connects to the COM4 port on the computer, and opens it for serial communication. For every piece of data that is read into matlab, we have to do some processing before it can get plotted. First, we parse the data into a more readable format. The data is put into a matrix of the form *[check_bit, distance, angle, depth, spadNum]*. In this matrix, you will notice that there exists a value for angle, but from Kiel, I only mentioned that we pass a 'step' parameter. This angle measurement was determined by multiplying the number of steps at the given location by 11.25/16, which represents the "degree per step" conversion factor. From here, we can convert the the distance and angle measurements into cartesian using our knowledge of trigonometry; $x = distance \times sin(angle)$ and $y = distance \times cos(angle)$. Once we know these x and y coordinates, I treated this x coordinate like a z coordinate, and plotted them along the yz plane. The depth parameter was then used as the x variable, as it gets incremented for each full rotation, and mimics the 3D aspect of the scan.

Application Example

Lidar scanning devices use laser light to measure distances and create 3D maps of objects and environments, making it a powerful remote sensing technology with a wide range of applications.

One of the primary uses of lidar is in mapping and surveying, where it can create high-resolution digital elevation models that are essential for urban planning, flood modeling, and resource management. Lidar is also increasingly utilized in the development of autonomous vehicles, where it provides real-time data on the vehicle's surroundings to aid in obstacle detection and avoidance.

Additionally, lidar scanning has significant applications in archaeology, forestry, and other fields. In archaeology, lidar can produce detailed maps of archaeological sites, uncovering hidden features and structures that may not be visible from the surface. In forestry, lidar can help to create accurate and detailed maps of forests, which can be used to monitor and manage resources, and identify potential hazards like wildfires and landslides.

Instructions

This instruction and setup process assumes that the user has all of the necessary software downloaded and configured for their specific system. This includes the installation of Kiel Development Environment, MATLAB Development Environment, and that the TI-EXP432E401Y microcontroller settings have been configured and properly setup in Kiel.

1.) The first step for the user is to plug their microcontroller into their computer, and determine their COM port. To do this, plug the microcontroller into your computer, then hit the windows key on the keyboard, type in ‘Device Manager’ and hit enter. Once in the device manager menu, scroll down to ‘Ports (COM & LPT)’ and then expand the arrow. In the drop down menu, take not of the value of # in the statement ‘XDS110 Class Application/User UART (COM #)’
2.) Open the matlab script, double click on the ‘ports’ variable in the Workspace variable editor. Then, change the value of X in the statement ‘s = serialport (X), 115200’ so that X corresponds to the index of the COM port that you determined in step 1. Now, serial communication should be all set up
3.) Configure the circuit by following the pin table found in the **Device Characteristics** portion of the manual
4.) Open the Kiel project. Click ‘Translate’ → ‘Build’ → ‘Load’
5.) Flash the project onto the board by clicking the reset button
6.) To enable data transmission, the second off board button should be pressed. Data transmission is enabled when LED 3 is illuminated

7.) In the MATLAB script, change the ‘depth’ variable to the number of depths that you would like to simulate
8.) Begin running the matlab script
9.) Use the first button to rotate the stepper motor, and repeat until the number of depths has been fully achieved
10.) Once finished, the finalized plot should appear on our computer with a 3D remodeling of the scanned area.

Expected Output

The expected output of the device can be visualized below by comparing my assigned location for scanning, to the 3D representation. My assigned scan location was location B. Below is a picture of the hallway in real life, and a direct comparison with the reproduced scan by the finished product.

As can be seen by the finalized scan, the system is able to capture the details of the scanned locations. Figure 1 shows that the general shape of the hallways was captured. The hallway begins wide, narrows out, and then widens out again near the end. Figure 2 shows how some of the finer details were captured in the scan. On the bottom left of the image, it can be seen how the recycling bins that were in the hallway were also captured in the final scan of the hallway. Figure 3 and 4 can be used to compare the general contours of the scan to the real life locations.

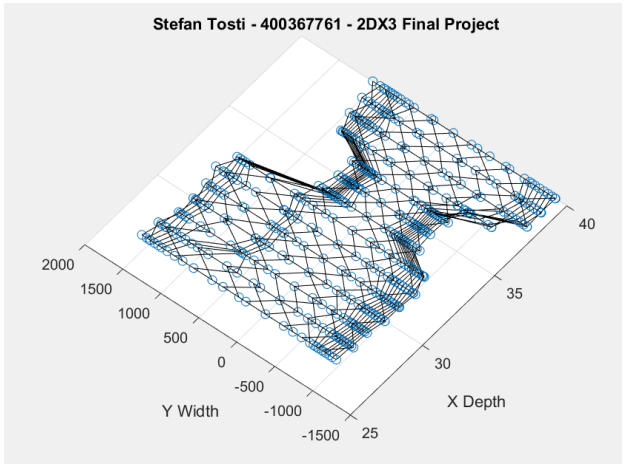


Figure 1 - Birds Eye view (General Shape)

Figure 3 - Real Life hallway (Start)

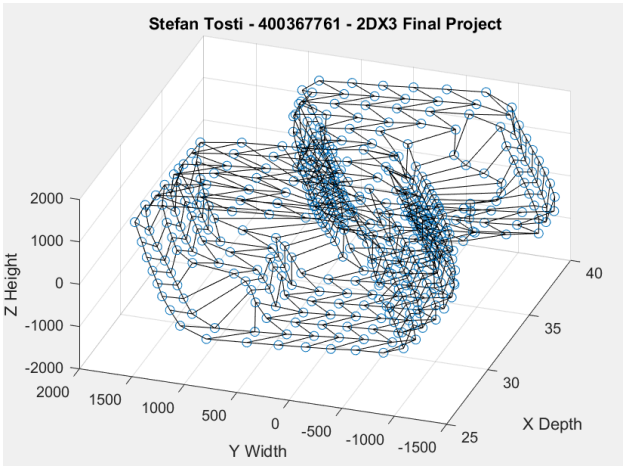


Figure 2 - Side view (Recycling bin Contours Limitations)

Figure 4 - Real Life hallway (End)

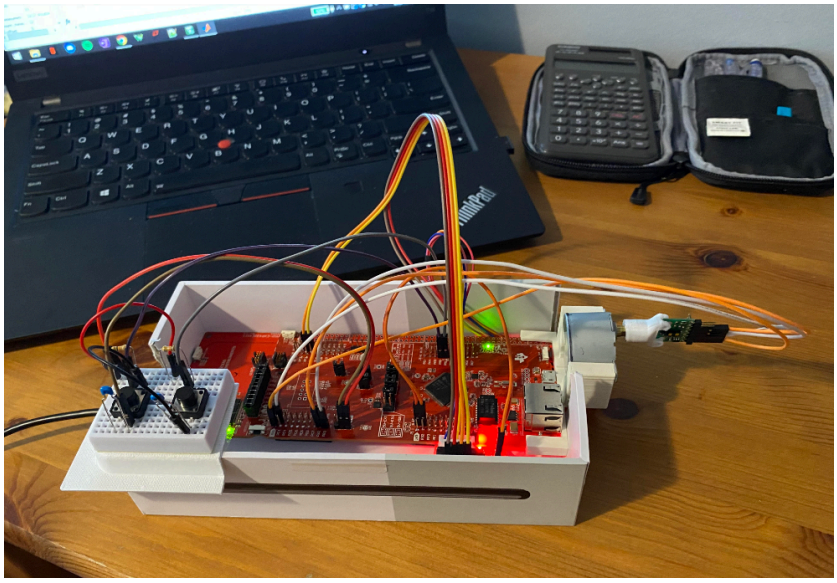


Figure 5 - Physical Mechanism, fully assembled

The above figure depicts the fully assembled scanning mechanism that was used to achieve the above plots.

As previously explored, the data plotting process requires the use of trigonometric functions, such as sin and cosine. These functions often result in very long, repeating decimal values. To deal with this, the TI-EXP432E401Y incorporates 32×32-bit single precision registers. These are located within the Arithmetic Logic Unit (ALU) and make up the Floating Point Unit (FPU). This allows the system itself support single-precision floating point operations

The maximum quantization error of a system can be calculated with $Max\ Quantization\ Error\ = \frac{Max\ Reading}{2^{number\ of\ ADC\ bits}}$. We can use this formula to determine the max quantization of our system. The maximum reading from the ToF sensor is 3m, and our micro reads the its values in a 16 bit format, thus we can conclude that

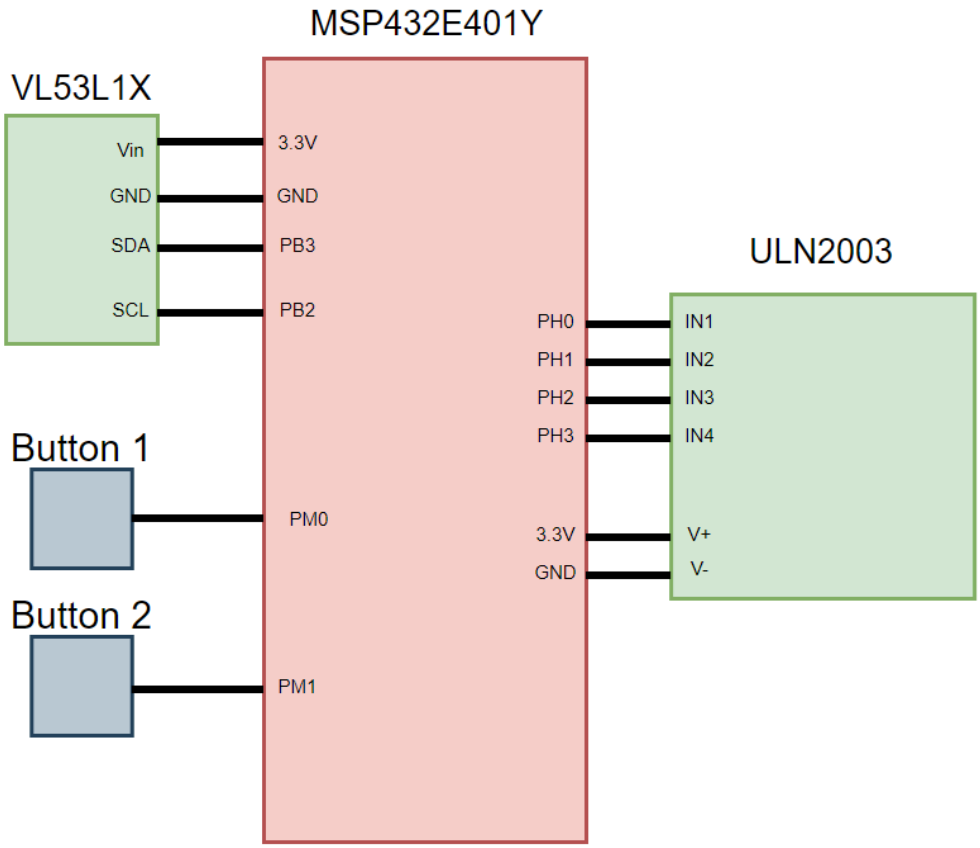
$Max\ Quantization\ Error\ = \frac{3000mm}{2^{16}} = 0.04577$

The maximum speed that my PC can implement is 128 000 bps. This was verified by going into device manager, then ports, then checking the BPS information relating to the XDS110 Class Application/User UART port. The speed that was implemented in my project was 115200 bps.

The communication protocol used to interface the ToF to the microcontroller was the I2C protocol. The ToF sensor has a maximum transmission speed of 50Hz

There were two primary factors that affected the speed: the time required to initialize the ToF sensor and begin ranging, and the speed of the motor's rotation. The speed of the motor was the most significant factor influencing system performance, as it was determined by the time delays between the four phases necessary for each rotation. Even if the ToF sensor took longer to initialize, the motor would rotate more frequently, resulting in longer operation times. To mitigate the impact of this factor, I conducted tests on the motor's operation by gradually decreasing the time delays between phases until the motor could no longer rotate. The shortest delay allowed between phases was approximately 2 ms.

Circuit Schematic



Programming Logic Flow