The system we have designed is a LiDAR scanner capable of taking multiple scans of its surroundings and creating a graphical image of said scans

1. Device Features

- VL53L1X Time of Flight Sensor
  - Up to 400 cm (4m) distance measurement
  - Up to 50 Hz ranging frequency
  - I2C interfacing
  - Input Voltage: 2.6V-5.5V (DC)
- Texas Instruments MSP432E401Y SimpleLink™ Ethernet Microcontroller
  - Arm Cortex-M4F Processor Core
  - 1024KB of Flash Memory, 256KB of SRAM, 6KB EEPROM
  - Input Voltage: 4.75 VDC to 5.25 VDC Via XDS-110 USB Micro-B
  - 120-MHz operation, 150-DMIPS performance
  - 16-bit SIMD vector processing unit
  - Programmable in assembly, c-language
- LiDAR System
  - Preferable for indoor use
  - Ranged LiDAR scanner with visualizing capabilities
  - Visualization capabilities when connected to a PC
- 28BYJ-48 Stepper Motor with ULN2003 Motor Driver
  - Input voltage range between 5.5V and 12V (DC)
  - 512 steps per rotation (accounting for gearing ratio)
- Visualization
  - MATLAB platform and graph
  - Objective C scripting
- Communication
  - I2C communication protocol between ToF sensor and microcontroller
  - UART communication protocol between microcontroller and PC, baud rate of 115,200 bps

2. General Description

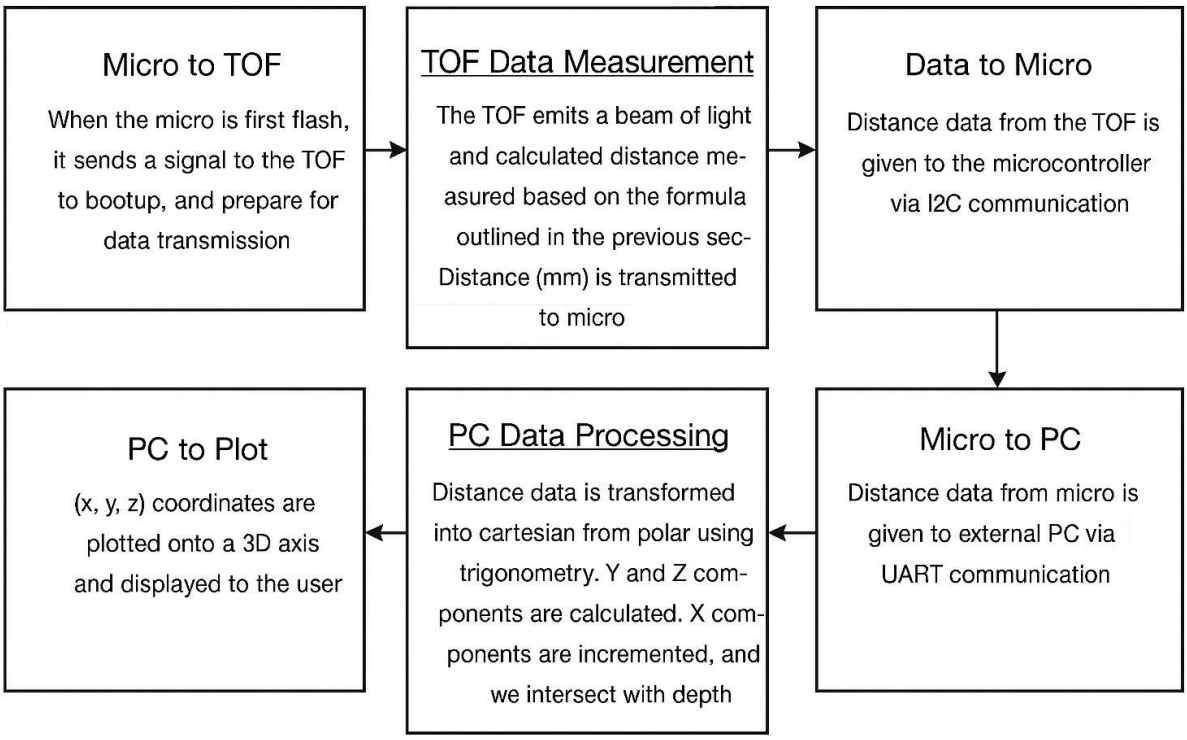
This device is an embedded spatial scanning system, capable of recreating an approximate 3D reconstruction of its surroundings by using its LiDAR system to scan its surroundings. The device operates by using a ToF sensor attached to a stepper motor in a manner akin to LiDAR systems, taking distance measurements across a full 360-degree rotation. When visualizing the entire area as a 3-dimensional grid, with the device resting on the x-axis, the distance measurements taken can be viewed as points on the YZ-plane, and consequently the y-components and z-component can both be isolated from the measurement. The system will be manually moved along the x-axis an appropriate amount between every measurement scan. These measurements, which technically classify as samples for data collection, are collected and visualized in MATLAB such

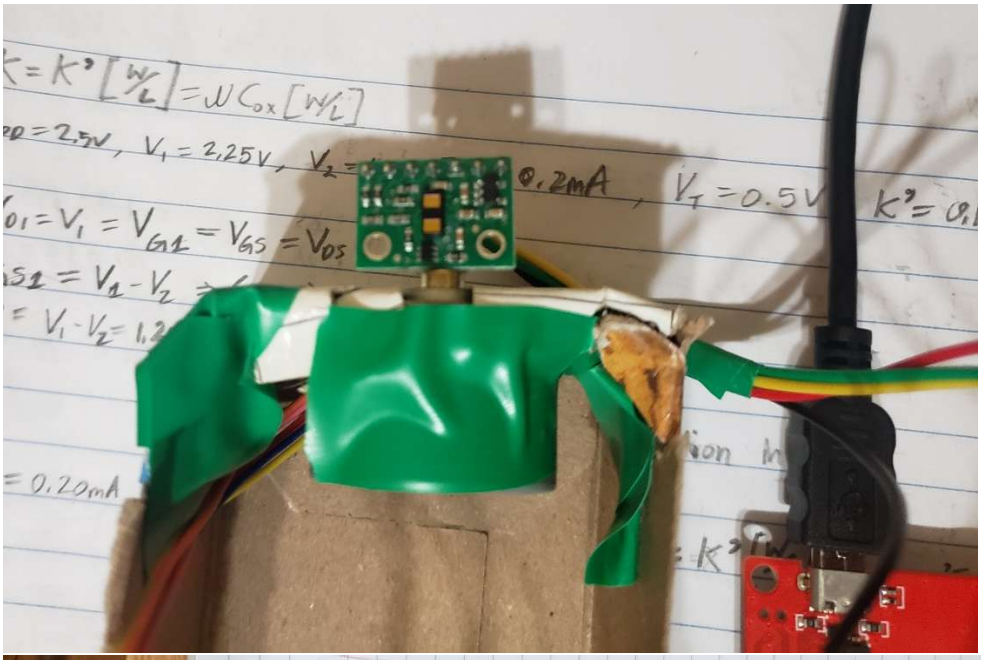
that the spaces between measurement coordinates are graphed and connected to one another, creating a 3D-model of the surrounding area detected by the system.

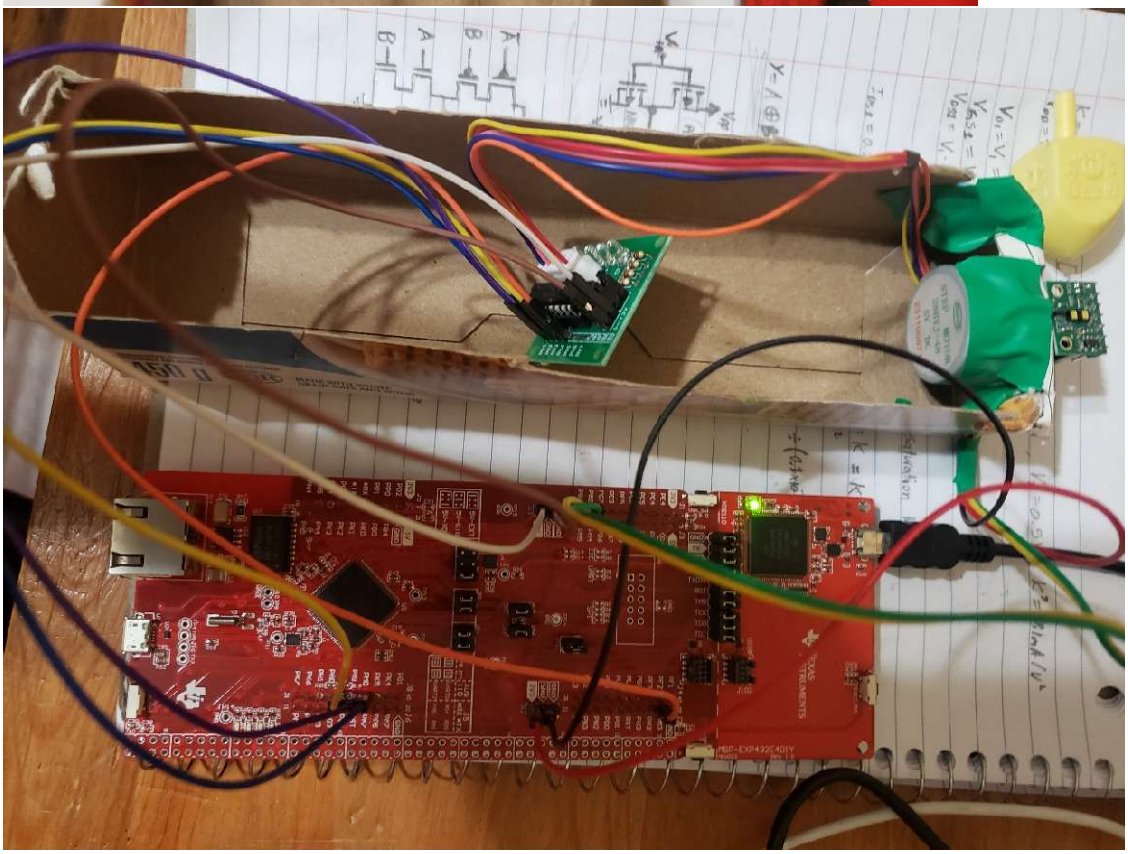
The system is composed of a MSP432E401Y microcontroller to control the device, a VL53L1X Time of Flight sensor module to take distance measurements and convert them to digital data, a MOT-28BYJ48 stepper motor connected to a ULN2003 driver to allow the system to rotate up to 360-degrees, two pushbuttons that can either be outsourced and attached externally or attached directly to the microcontroller (built-in push buttons), and two indicator LEDs that can either be attached externally or directly (built in LED) to the microcontroller. The microcontroller is programmed using C-language to manage control loop routines and communications between the ToF sensor and the PC. MATLAB is used on the PC to visualize data on a 3-dimensional graph.

The VL53L1X module calculates the distance measured by the built in ToF sensor, which measures distance using a pulse of infrared light to measure a photon's time of flight. The sensor angle can be adjusted by rotating the stepper motor the desired number of steps (though the software allows users to provide a desired angle as opposed to the specific required step size). The measurement data is transmitted from the VL53L1X to the microcontroller via I2C protocols. This measurement data is then transferred from the microcontroller to the user's PC via UART communication protocols, to then be visualized using MATLAB by plotting each sample coordinate on a cartesian plane.

3. Block Diagram







4. Device Characteristics

System Feature	Characteristic

UART Baud rate	115,200 bps
Microcontroller Bus Speed	24 MHz
Microcontroller Power	5V DC USB A (PC connection)

VL53L1X Feature	Microcontroller
VDD	--
VIN	5V DC
GND	GND
SDA	PB3
SDC	PB2
XSHUT	--
GPIO1	--

LED Configuration	Routine
PN0	Measurement status is present
PN1	Data has been transmitted via I2C

Button Configuration	Routine
PJ0	Begin data acquisition cycle
PJ1	Initialize data transfer with PC (UART)

ULN2003	Microcontroller
+	+5V DC
-	GND
IN1	PM0
IN2	PM1
IN3	PM2
IN4	PM3

5. Detailed Description

5.1) Distance Measurement

The system operates using a VL53L1X ToF module to acquire distance measurements. This module comes as an integrated package, including the transducer from the ToF sensor, conditioning IC's, a built in ADC, and an API to communicate using I2C. The ToF sensor sends out a 940 nm pulse of infrared light and uses the time between the initial pulse and the reflected signal to determine the relative distance between the sensor and the nearest surface in front of it. The time between the initial infrared pulse and the sensor receiving the reflected pulse of light in considered the photon time of flight, as it is the total time that the emitted photon travels for. Since the speed of light is a universal constant, the photon time of flight can be divided by 2 to get the photon time of flight in one direction (since the measured time is both the pulse and rebound), then multiplied by the

speed of light constant to determine the distance travelled by the emitted photon. The data is then conditioned and converted from analog to digital data by the module's ICs and an integrated ADC (no data on process publicly available). This digital data is transmitted to the microcontroller through I2C using the UM2510 API, which is also integrated in the VL53L1X module. The module also communicates a confidence variable to specify how accurate a measurement status is likely to be. The VL53L1X will send a "0" if there is no detectable error, between "1" and "2" if an error possibly occurred, and between "4" and "7" if an error occurred with certainty. Once the error has been verified to be no error (0) by means of software, the distance measurement is recorded in millimeters and saved in its corresponding position within a 2D-array for transmission later.

The ToF sensor is attached to the 28BYJ48 stepper motor through a non-conductive adhesive applied on both the sensor and the motor shaft. The stepper motor then precisely alters the angular position of the output shaft for each ToF measurement. Each measurement starts at a position facing straight upwards, or 90-degrees to the horizontal. The ToF sensor takes a distance measurement and stores the value in a 2D array, dataArray. These arrays are then sent via I2C to the microcontroller. The resolution of the final 3D reconstruction is proportional to the number of measurement points by the ToF module taken at a given x-coordinate. Therefore, to increase recreated resolution, we can reduce the number of steps taken by the stepper motor for each measurement, which in turn increases the time and number of measurements taken for each reading, which can be adjusted by a specific variable, STEP_SIZE, which defines the number of steps taken by the stepper motor. Additionally, the rotate function allows you to adjust the number of degrees rotated by the stepper motor, which in turn alters the number of steps taken by the stepper motor. The current resolution of the device is configured to take approximately 500 measurements (0.72 degrees for a full rotation), but this can be adjusted accordingly.

When the microcontroller initializes relevant programs, all other systems such as the I2C and UART are also initialized and primed as needed. To start and stop data acquisition, a polling-based system is used via the microcontroller integrated push buttons, which are connected to PJ0 and PJ1. While constantly checking for the state of PJ0 and PJ1, if the state of the system, spin_toggle, is set to 0 when not collecting data, and set to 1 when collecting data. When a button is pressed, an interrupt is triggered that toggles the spin_toggle variable, which in turn causes the program to enter its data acquisition state.

When the system enters its data acquisition state, the program enters a waiting state until the sensor state, status, returns as available (as a 1). Once the sensor is ready, the data measurement is retrieved, then transmitted from to the microcontroller via the built in I2C communication. This acquisition occurs over a full rotation of the stepper motor. Once finished the acquisition stage, this data is transmitted to the users PC via UART to a MATLAB script that processes the data into a 3D construction. Once a full rotation has been performed by the stepper motor, the ToF module ends its ranging mode (setting its status to 0) and toggles the spin_tiggle to enter a static waiting stage, until the user presses PJ0 (button) again.

5.2) Visualization

The MATLAB script collects data from the connected UART COM port, and runs the data processing loop for however many levels are specified. The level act as layers to the 3D reconstruction, meaning that if you wanted a higher resolution are wanted to scan a longer area, you can increase the number of layers taken in the visualization. The number of layers also dictates the number of x-coordinates the final recreation will have, as well as the number of times the system will need to be manually moved along the x-axis. When graphing the measurement points, the x-coordinates are mapped according to the number of layers taken by the MATLAB program. The y-coordinate and z-coordinates are both calculated by multiplying the measurement value by sin() and cos() (respectively) of the angle travelled by the motor with each measurement (d*sin(angle) for y and d*cos(angle) for z). These x,y,z coordinates are mapped on a cartesian plane in MATLAB (though the x and z coordinates are swapped to match MATLAB formatting). Once all layers have been plotted out on a cartesian plane, the MATLAB script creates a set of vertical and horizontal lines to connect the entire plot into a reconstruction. First, the layer points are connected to each other to create a cross section of the area, then each cross-section layer is connected to each other to create a general reconstruction of the entire area. Once the reconstruction is completed and displayed, the MATLAB script stops reading data from the UART COM port, clears the variable holding any remaining data, and ends by displaying the 3D reconstruction for users to view.

The entire process is designed such that minor adjustments to code can be made to produce scans of different resolutions and lengths, which can prove useful when measuring different room types and sizes.

6. Application Example with Expected Output

6.1) Application Examples

The following system acts as a LiDAR scanning system, and is capable of creating 3D maps of its surroundings, making it considerable effective in tasks requiring some remote sensing technology and data acquisition of nearby object distances. For instance, the system can be used for mapping and surveying applications, as it can create high resolution images and models of an area, which is especially useful for urban planning, natural disaster preparation, and mapping areas only accessible through some remote-control vehicle, such as within caves or deep in the ocean on an ROV.

Additionally, a LiDAR scanning system such as this one can be used at higher speeds with autonomous systems to allow them the ability to dynamically adjust to their surroundings. For example, autonomous vehicles use LiDAR scanning systems to actively monitor the vehicles surroundings and aid in obstacle detection and avoidance while actively driving/travelling.

6.2) Instructions

The following steps are used to configure the device to take a 3d scan of its environment. This instruction set also assumes the user has alread

1) Plug in microcontroller to PC, open up MATLAB script, and open device manager to determine which port the microcontroller is connected to. In device manager, open the Ports list and use the COM Port labelled as USB UART in the MATLAB script where it specifies to enter the COM port. Also ensure that the baud rate is set up to 115,200 as that is the baud rate configured by the microcontroller.

2) Configure the circuit between the system and the microcontroller according to the figure 5 below displaying pin numbers

3) Open the Keil UVision project, and ensure the angle size is configured to whatever you may desire (if the default is not adequate). Once done, proceed to translate and build the code

4) Load the code onto the microcontroller, press the reset button to ready the microcontroller to run the program

5) On MATLAB, adjust and values, such as depth and number of levels needed in the 3D recreation, according to whatever you may desire

6) On MATLAB, run the main code, making any relevant file paths requested by MATLAB. The script will ask you to press the enter button on your PC. This is to verify that the UART protocols work. Press enter on your keyboard and press the reset button on your microcontroller. Once a confirmation message is returned, press enter again to begin data acquisition and immediately execute step 6 below as soon as possible.

7) To enable the data acquisition stage, press the second on-board button of the microcontroller, PJ1. Once LED 3 flashes, the system has began collecting data, and should be spinning. In MATLAB, the terminal should also be displaying the distances measured by the ToF sensor as it rotates

8) Once a rotation is finished and the sensor has returned to its starting position, move the system forward manually, according to the number of layers in the program and the total distance being measured (displacement = total distance of area/# of layers). The x-coordinates are the total displacement of the system with each level, while the y-coordinate and z-coordinate are used as vertical distance slices.

9) Repeat steps 6,7, and 8 for however many levels you've set in the MATLAB code.

10) Once finished, a 3D recreation of the area should be displayed. You may save this graph as a MATLAB workspace for later reference.

6.3) Expected Output

The expected output can be seen below, in which the device creates a visualization of the surrounding areas pictured below. I myself was assigned location "i", which can be seen pictured below. Beside my assigned location is the 3D scan produced by the system.

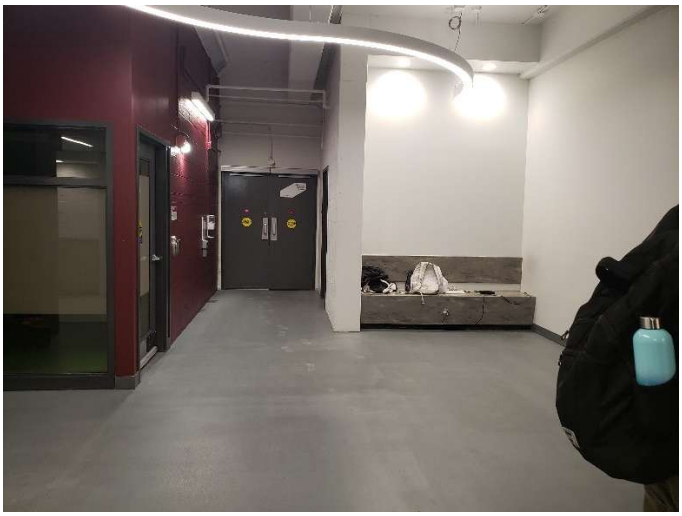


Figure 1: Image of Hallway

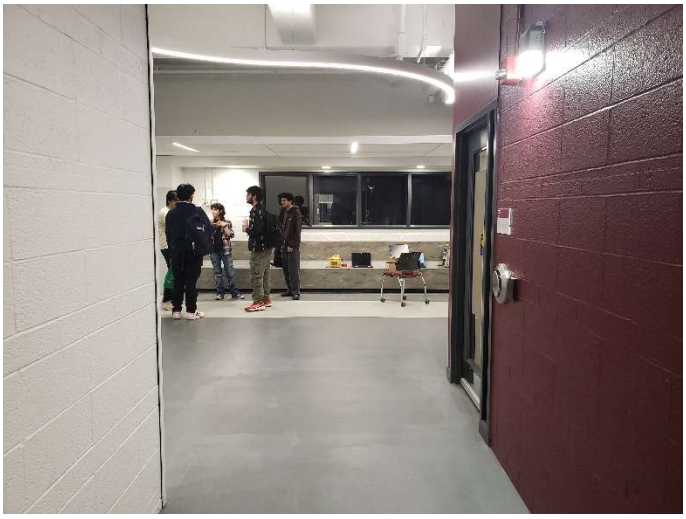


Figure 2: Different angle of same hallway

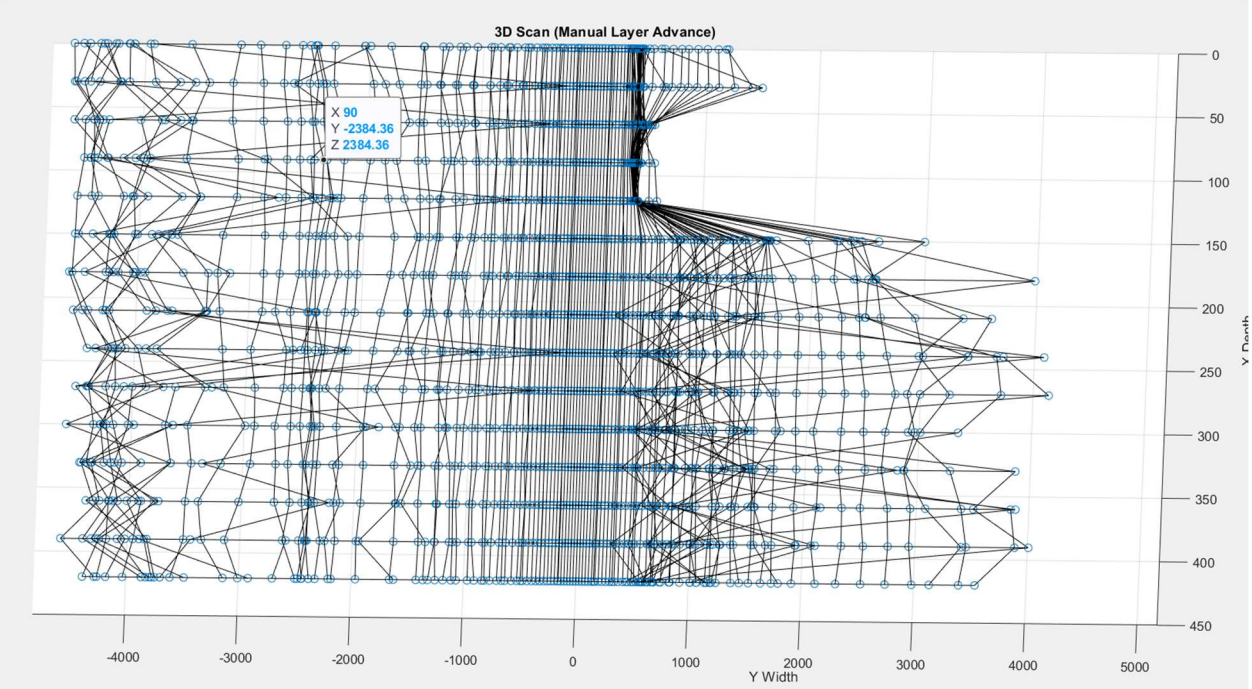


Figure 3: Top view of scan done in hallway

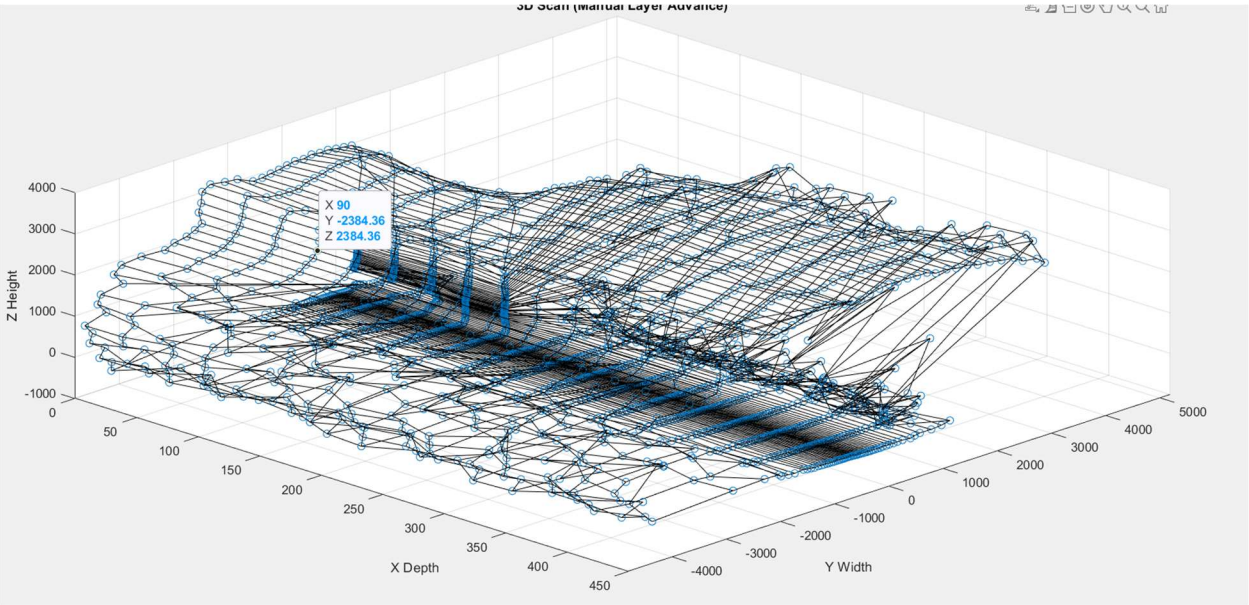
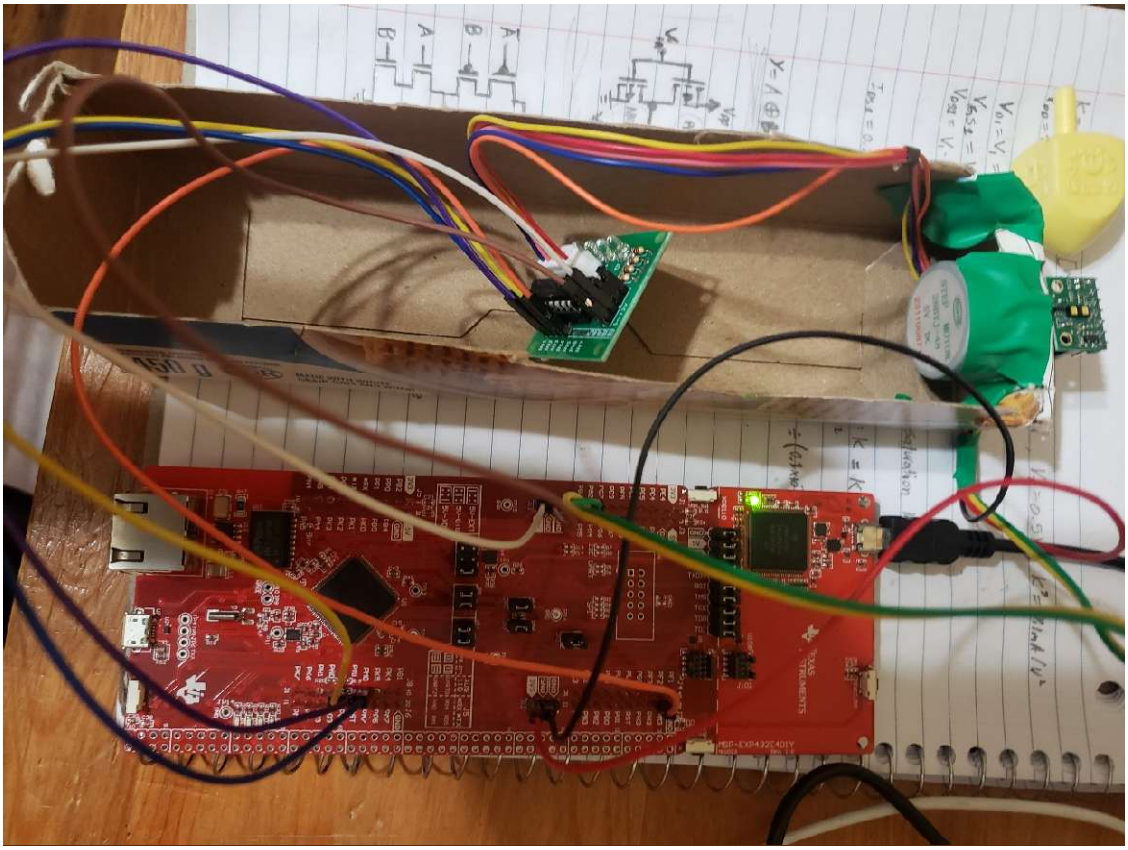


Figure 4: Different angle of hallway scan to give perspective of depth

As can be seen by the finalized scan, the system was able to accurately capture the details of the scanned locations. Figure 3 and 4 illustrate that the general shape of the hallways was captured. The hallway began as a narrow hall, and then widens out towards the end, close to the entrance toa nearby lecture hall. Figure 1 and 2 show the actual locations, and the details can be overlayed the scans to see the exact spots scanned. On the bottom left of figure 4, it can be seen how the

benches that were in the hallway were also captured in the final scan as a small bump towards the end of the wall. Figure 3 and figure 4 can be used to compare the general 3D recreation to the actual hallway in figure 1 and figure 2.



The following is the device used to produce these exams

7. Limitations

The sensor measures up to 4m (4000 mm)

Max Quantization Error = Max Readin/2^(number of bits in ADC) = (4000 mm)/2^(16)

Max Quantization Error = 0.06103515625

The maximum speed capable of most PCs (mine included) is 128,000 bps, which was verified by entering the device properties of my UART port and checking the maximum speed capable of the PC, which was 128,000 bps. The baud rate used for this device was 115,200 bps.

The ToF sensor transmits data at a maximum speed of 50 Hz, which is done via I2C communication protocols. However, I2C is capable of supporting upwards of 100kbps.

The microcontroller is limited in the number of floating-point units it can process, storing each value in 32 bits of memory, 23 of which are available to represent the fractional part of the number. Although it is not completely floating-point capable, floating-point approximation errors

within the microprocessor itself are not present in this arrangement. This is because the device firmware makes use of only 8-bit and 16-bit integer variables in its data processing, meaning all floating-point calculations, including trig functions, are performed directly within MATLAB. Thus, any rounding errors are due to the visualization software in the outside computer. Since the x, y, and z coordinates are handled as floating-point numbers, there is a built-in rounding that is a source of error. When integer-based distance measures are converted into spatial coordinates, an approximation of π is used by MATLAB, along with the cosine and sine functions. These approximations individually make minor errors in the final 3D image reconstruction.

The stepper motor also vibrates a considerable amount, introducing considerable noise into the system. With the entire system, including the SCL and SDA pins lying on the motor shaft, the vibration more than likely introduces some unwanted noise to the device, altering transmitted values.

Additionally, the system costs upwards of $275 to implement, including the kit provided to us, which includes the microcontroller, stepper motor, and ToF sensor, as well as an additional minor cost due to the device housing, though I did not pay for this (considering it was fished out from my recycling bin for free).

Logic Flowchart

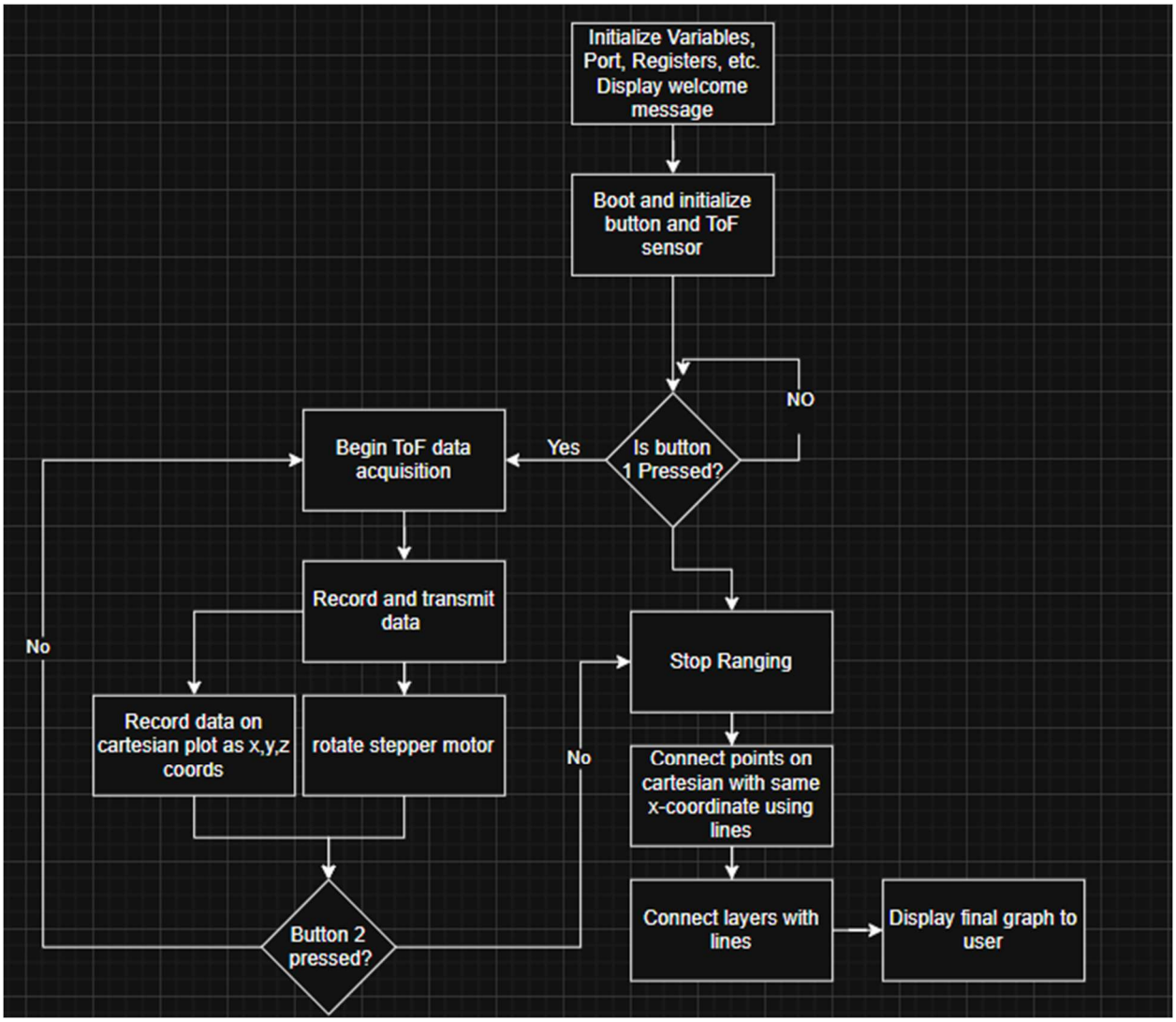


Figure 5:Logic Diagram used by system to process and display info

Circuit Schematic

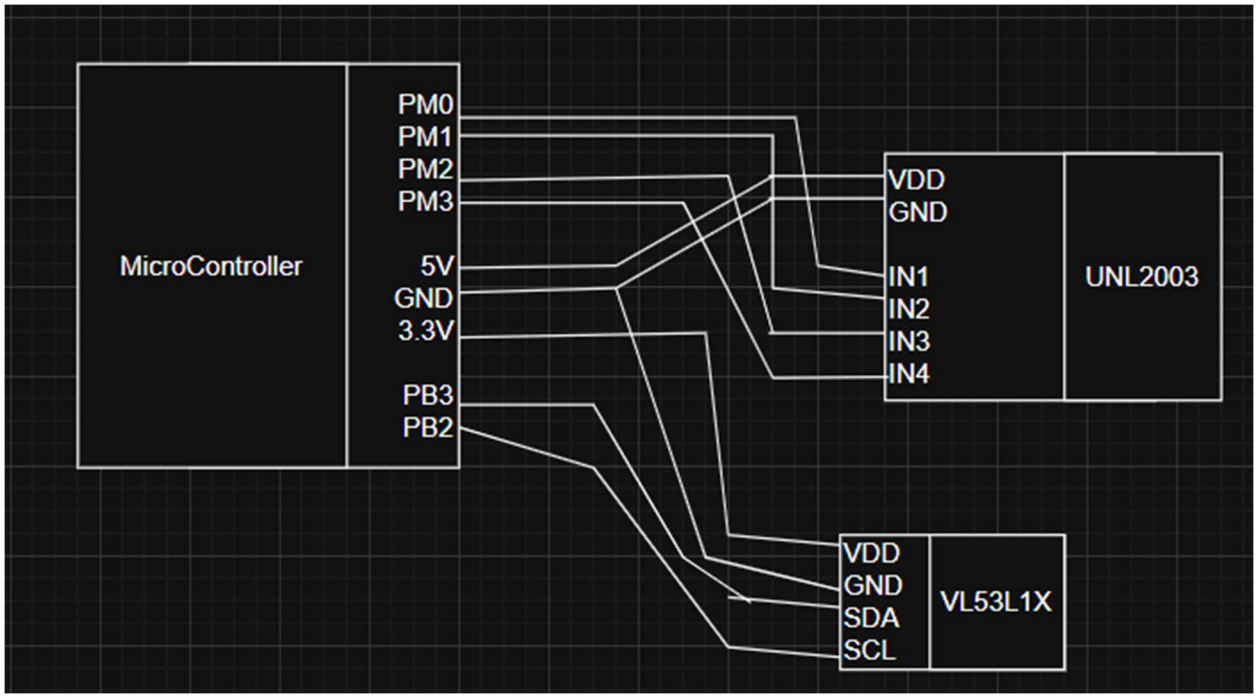


Figure 6: Circuit schematic of system