Device Overview

Features

- Fully integrated compact module
  - 360^{o} field of view via MOT-28BYJ48 stepper motor
  - Measurement status LEDs
  - VL53L1X Time-of-flight sensor with 3 modes of operation
- Fast, accurate distance measurements
  - Measures up to 4m away in long range mode
  - LIDAR lasers travel at light speed, allowing for fast measurements
  - Measurements per rotation can be modified to increase speed
- Low power requirements with powerful performance
  - Onboard ARM Cortex M4 with 1024kB flash memory, 256kB SRAM, and 6kB EEPROM
  - Programmed in C
  - 2 12-bit SAR-based ADC modules with max. sampling rate of 2 million/second
  - 5v DC operating voltage
  - 96 MHz clock frequency
  - Capable of UART communication at 115200 baud rate with '*' character to start measurements
  - Overall cost of $150.00 CAD
  - Interrupt-based data acquisition allows for low-power idle state

General Description

This spatial mapping system features a VL53L1X Time-of-Flight (ToF) laser-ranging module and a 360^{0} pivoting MOT-28BYJ48 stepper motor, providing accurate automated 2D spatial scanning within a compact package. The interrupt-based design ensures that power is saved while the device is idle. When the onboard button is pressed, the system seamlessly wakes up, acquires one set of data, and then returns to sleep mode until the next button press.

The VL53L1X utilizes LIDAR ranging technology to measure distances via harmless, transparent lasers from the VCSEL emitter, and it integrates a leading-edge SPAD array (Single Photon Avalanche Diodes). During boot-up, the sensor's status is displayed by UART. Errors are not handled during runtime; however, the user may choose to implement error handling as required for the application.

The MOT-28BYJ48 stepper motor rotates the ToF sensor. The rotation speed and number of measurements can be configured to adjust the speed and accuracy of scans depending on the user's application.

System Block Diagram

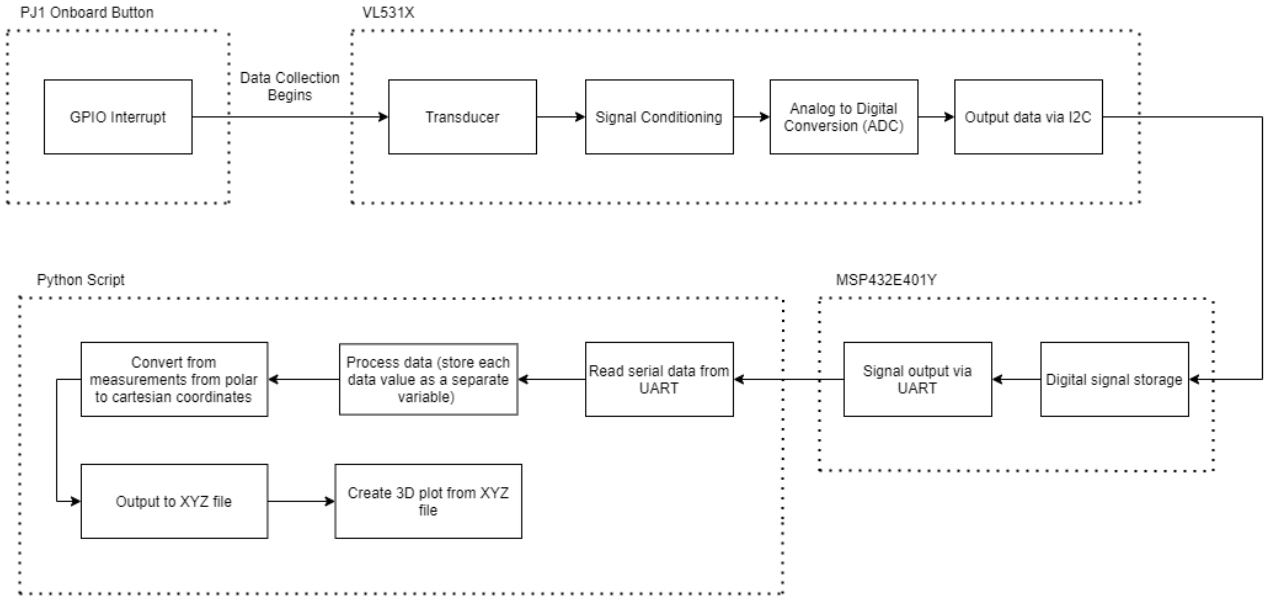

Figure 1: System Block Diagram

Physical Device and PC Output

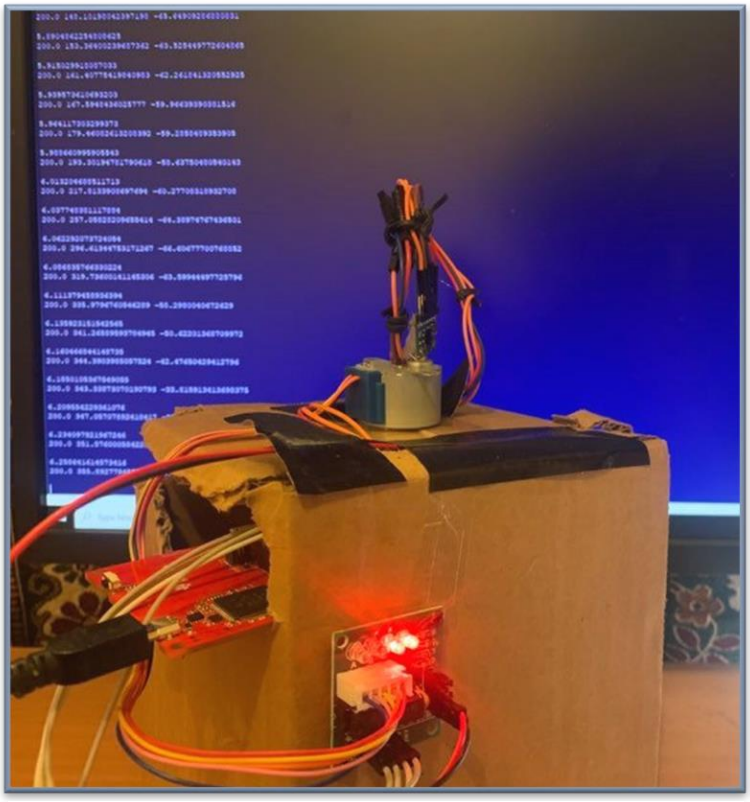

Figure 2: Device taking measurements

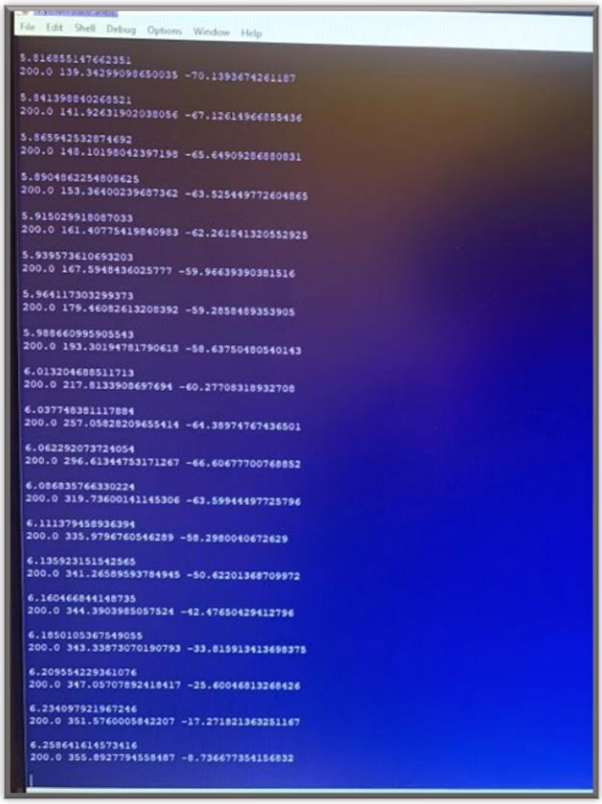

Figure 3: Measurement output during operation

Device Characteristic Tables

VL53L1X Pins

Device Pin	Signal Description	MSP432E401Y Pin
VDD	Regulated output voltage	-
VIN	Positive bias voltage	3.3V
GND	Ground	GND
SDA	I^2C Serial Data Line	PB3
SCL	I^2C Serial Clock Line	PB2
XSHUT	Shutdown Pin	-
GPIO1	Interrupt Output	-

ULN2003 Motor Driver Pins

Device Pin	Signal Description	MSP432E401Y Pin
V+	Positive bias voltage	5v
V-	Negative bias voltage	GND
IN1	Digital input	PK0
IN2	Digital input	PK1
IN3	Digital input	PK2
IN4	Digital input	PK3
5-Socket Port	Connection to 28BYJ-48 motor	-

MSP432E401Y Parameters

Parameter	Description	Value
Bus Speed	Bus clock frequency	96 MHz
Baud Rate	Serial communication rate	115200

Detailed Description

MSP432E401Y Program

The process of creating a 3D spatial scan can be broken down into three main categories: data acquisition, processing, and visualization. The system is implemented as half-duplex. In order to start taking measurements, the microcontroller sends the python script a '*' character using UART serial communication, which tells the script to start recording data being sent by UART.

In order to acquire data (measure distances) the VL53L1X Time-of-Flight sensor is used. This module contains an onboard microcontroller with dedicated RAM, ROM, and non-volatile memory, as well as a SPAD array, advanced ranging core, and VSCEL driver in order to measure distances.

Data acquisition with any sensor begins with transduction, which is the conversion of physical measurements (in this case distance) into electrical signals. The ToF sensor measures distances by outputting a laser and measuring the time taken for the laser to return to the sensor. The time is then halved, as the laser will have travelled twice the distance being measured. The ToF sensor assumes that the beam travels at the speed of light in a vacuum, however depending on the medium light will travel at slightly slower speeds. The distance is given by the formula $D = \frac{Ct}{2}$. The angle that the laser is emitted is equal to the angle of impact at the sensor, which, by symmetry, results in the angle having no impact on the distance measurement.

The ToF module also handles signal conditioning. The transduced signal is amplified to fit within the 5v range of the sensor, and then it is passed to the onboard analog to digital converter (ADC). The successive approximation register method (SAR) is used to convert the distance to a digital voltage. This voltage is then passed to the MSP432E401Y microcontroller as a digital input. Using the VL53L1X API, the input is converted to a distance measurement in millimeters by calling a predefined function. This process is repeated multiple times as the stepper motor rotates in the YZ plane. The angle is determined by taking the current step and dividing by the total number of steps, then multiplying that fraction by the total angle per rotation. The formula for the angle is $A = \frac{i}{512} * 360^0$, where i is the current step.

The distance and angle are printed to a temporary buffer in the format "[distance], [angle]". The flowchart in figure 4 illustrates the flow of the microcontroller program. The

program does not terminate, and instead it returns to low power mode after each measurement cycle.

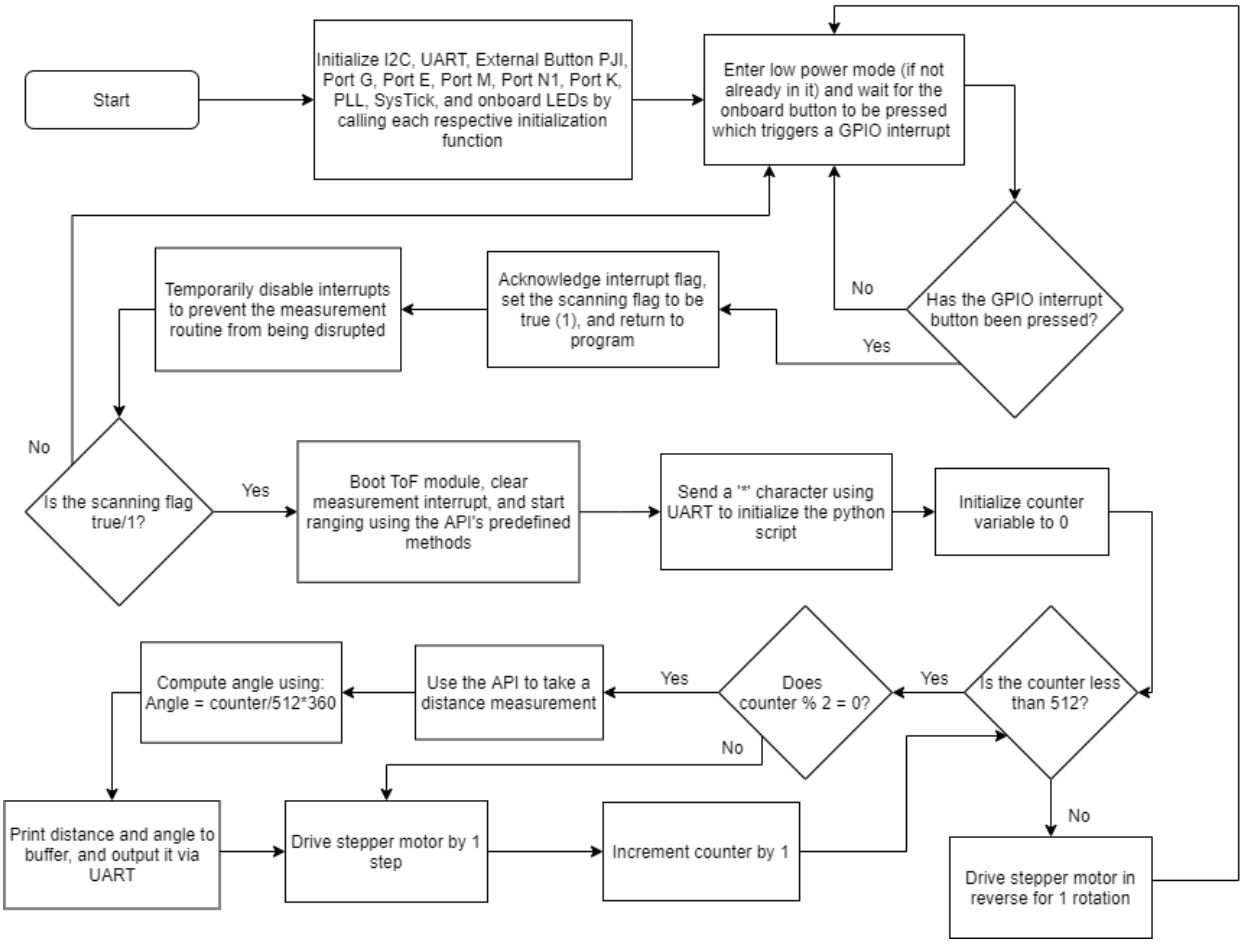

Figure 4: Microcontroller program flowchart

Python Program

The Python program is run on a 64-bit computer running Windows 10 pro. It features an AMD Ryzen 7 3700X 8-core processor with 16 GB of ram. The Python program uses python 3.8.0, and requires the use of the PySerial library alongside Open3D, NumPy, Pyrender, Trimesh, and Math.

The Python program begins by opening a serial line at COM6 with a baud rate of 115200. The data file “tof_radar.xyz” is opened for writing, and if it does not already exist then it is created. Since the microcontroller outputted the ‘*’ starting character before taking measurements, the Python script actively monitors the UART data line for data packages. After the microcontroller sends the data buffer using UART, the script actively reads and stores the data packet as a string. Python’s split method is used to separate the comma-separated packet into a list containing the integer distance float angle. These values are then stored in separate variables. Python’s math library is used to convert the angle from

to radians, which is required because the math library uses radians in sine and cosine functions.

The distance and angle form a point in polar coordinates, however in order to plot the points in 3D space, they must be converted to cartesian coordinates. The following formulas are used to convert the points:

$$Y = Distance * \cos(angle) \ , \ \ Z = Distance * \sin(angle)$$

The X component initially starts at 0 for the first measurement. The X, Y, and Z components are concatenated to a space-separated string in order to meet the criteria of the XYZ file format. This data string is then printed to the screen for user verification, and also written to the xyz data file. The X component is then incremented by 200mm (the delta value) after the rotation to account for the user taking a step in the X axis after each successful 'slice' measurement.

This entire process occurs 10 times using count variable and a while loop. Then, the file and serial lines are both closed. Using the Open3D library, a point cloud structure is generated from the xyz data file. The structure is iterated through, and each point is connected to the previous and next point. Finally, a line set structure is created using a method from the Open3D library on the point cloud. This line set is then visualized using another method, which produces a 3D image of the spatial scan. The flowchart in figure 5 shows the Python program flow.

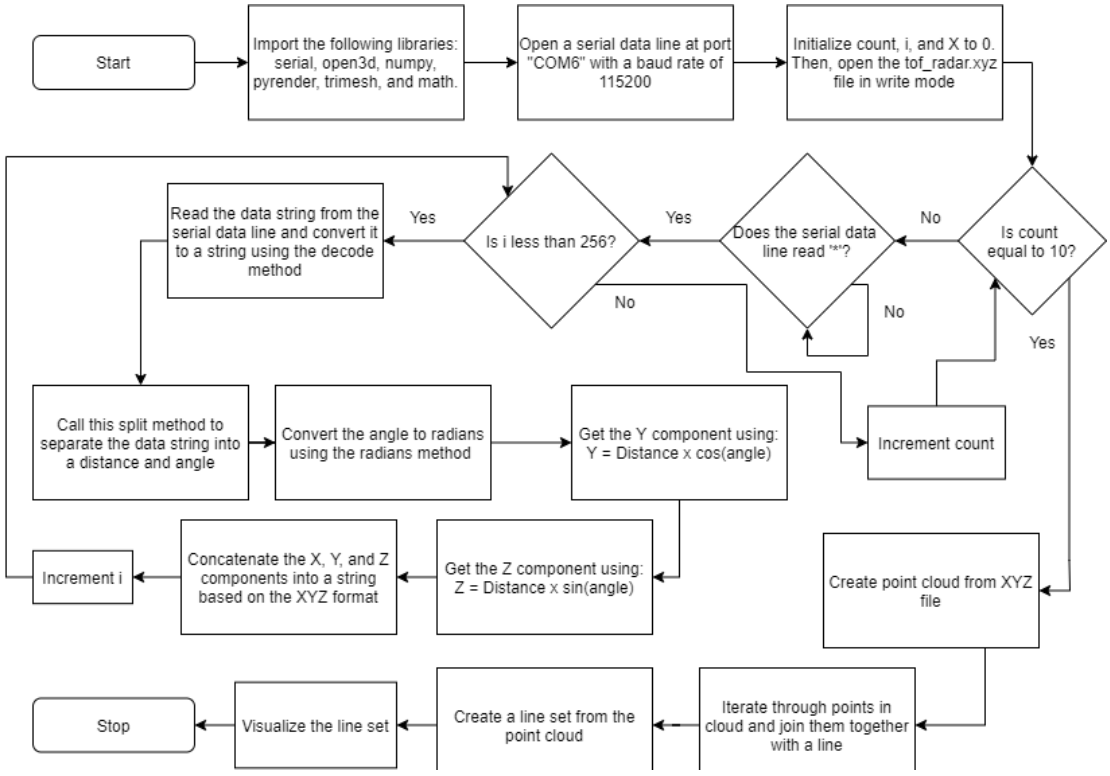

Figure 5: Python program flowchart

A multiview drawing of the room that was scanned is shown in figure 6, alongside the actual scan in figure 7.

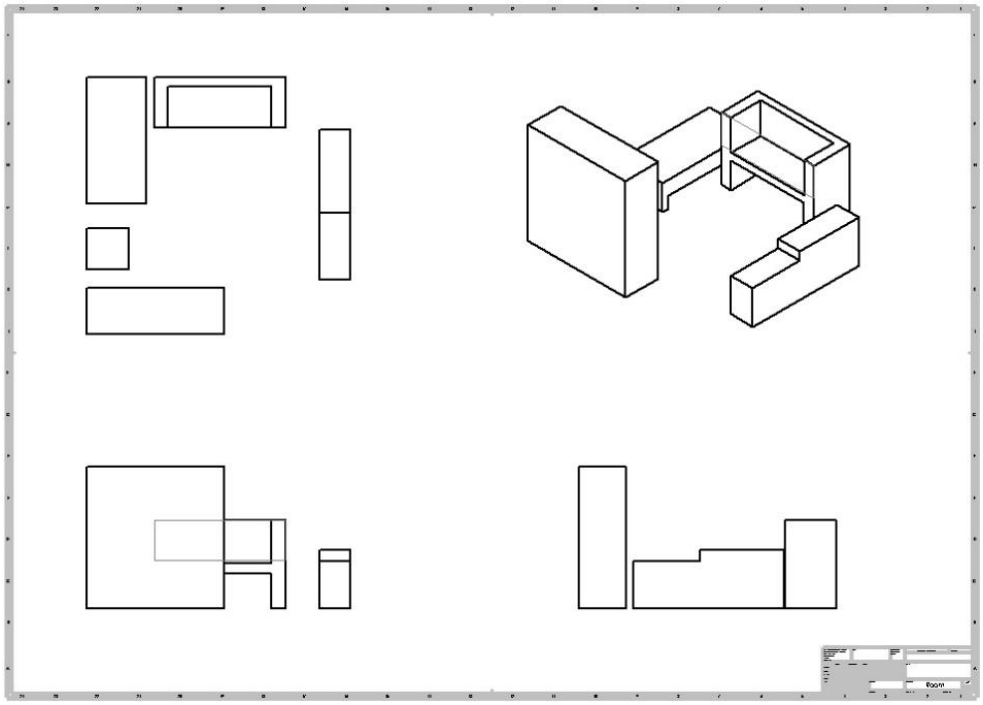

Figure 6: Room multiview (walls excluded)

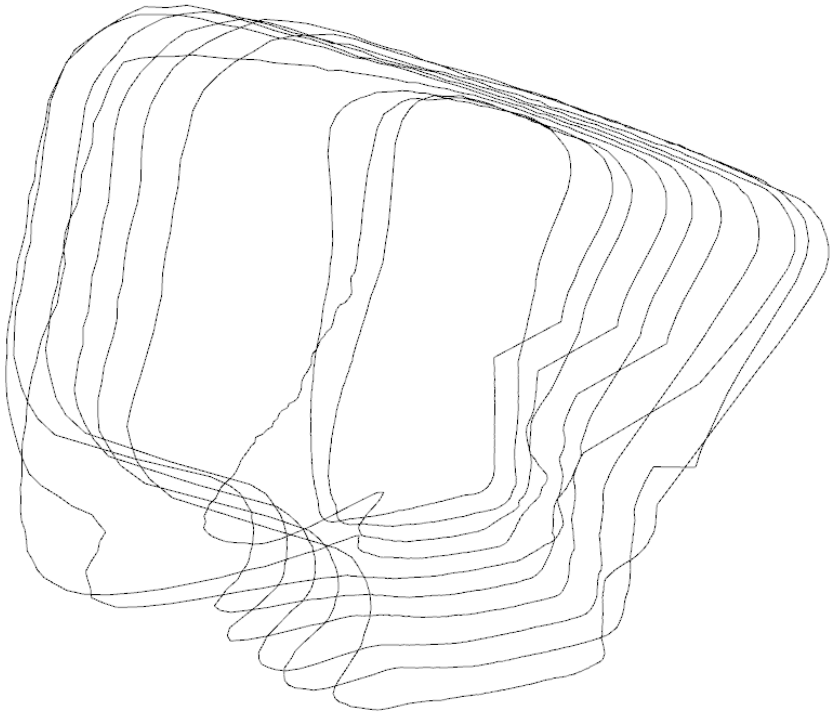

Figure 7: Room 3D scan

Application Example

In order to use this device to acquire sensory data, process it, and display the 3D spatial map on your computer, the following steps are required. Please note that these instructions are for Windows 10 operating systems, however a similar process will be used for MacOS and Linux operating systems.

1. Download and install the XDS emulation software for your operating system from (https://softwaredl.ti.com/ccs/esd/documents/xdsdebugprobes/emu_xds_software_package_download.html). This software allows debugging and communication with the microcontroller using UART.
2. Install Python 3.8.0 (https://www.python.org/downloads/release/python-380/) by following the instructions on the website.
3. On windows, open the command prompt and navigate to the python install folder. From there, navigate to scripts. Now enter “pip install open3d” and “pip install pyserial” in order to download the required libraries for this program to run.
4. Right click on FinalProject.py and open with IDLE 3.8. The COM port is set to COM6 by default. This must be changed to the port that the device is plugged into, which depends on the USB port being used. In order to determine the correct port for your computer, open command prompt and enter “python -m serial.tools.list_ports -v” with the device plugged into your desired USB port. The baud rate is 115200 by default. Do not modify this value.
5. In order to modify the software to suit your measurements, change the value of x inside the for loop (x += [value]) to configure the manual displacement between measurements in millimeters. The number of samples per slice can be configured by changing the range of the for loop, and the total number of slices can be changed by modifying the while loop condition.
6. Run the FinalProject.py script and press the PJ1 onboard button on the MSP432E401Y microcontroller. The measurement data should appear on the screen in real time.
7. Once the sensor completes one rotation and begins to rotate in counter-clockwise back to the starting position, move it forward in the X-direction by the distance specified by the x value in the python script (200mm by default).
8. After all rotations are finished (given by the condition of count), return to the PC and a 3D model file will have opened. This file contains your 3D spatial scan.

Limitations

1) The Cortex-M4 has limited accuracy regarding floating-point values. The microcontroller has a floating-point unit (FPU) onboard which has 32 registers capable of 32-bit precision floating point data processing. The FPU can perform all basic arithmetic operations on float variables however it uses double precision arguments. Given that the FPU is 32-bit, double precision (64-bit) operations take longer to complete since the FPU must split the operation into 2 32-bit operations. This issue is important when considering whether data should be processed on the microcontroller or in Python. It was decided to use the python script to compute trigonometric functions, as my PC has a much more powerful processor with more accurate floating point capabilities.

2) The maximum quantization error for the VL53L1X ToF sensor is:

$$Max\ Quantization\ Error = \frac{MaxDistance}{2^B}$$

Where max distance is the maximum distance that the ToF sensor is able to measure accurately (4m), and B is the number of bits used to store the measurement (16 bits).

$$Max\ Quantization\ Error = \frac{4000mm}{2^{16}} = 0.061mm$$

3) The maximum standard serial communication rate that can be implemented is 921600 Bps (bits per second). In the high-speed mode, the serial communication rate maxes out at 1 843 400 Bps. To test this rate, the IBRD and FBRD registers were changed for UART. During testing, it was found that this rate resulted in data loss as data was being outputted to the UART line faster than it was being read. This mean that it was not a viable rate to use. Thus, it was empirically determined that the maximum communication rate that can be implemented is 115200 Bps. At this rate, no data loss was observed when reading the UART data from RealTerm.

4) In order to communicate between the microcontroller and ToF sensor, the I^2C protocol is used. The communication speed is 100 KBps by default.

5) Overall, in this entire system the component that primarily limits the speed of measurements is the Time of Flight sensor itself. The stepper motor is able to rotate at approximately 1 rotation per second at the maximum speed, however the time of flight sensor causes a delay during each measurement. In order to test this, I ran the stepper motor without the Time of Flight sensor taking measurements, and as expected the entire rotation took approximately 1 second. Then, I adjusted the number of measurements made by the ToF sensor per rotation, and I found a direct relationship between the number of measurements made per rotation and the total rotation time. This empirical relationship indicates that the ToF sensor is the limiting component of the rotation speed.

Circuit Schematic

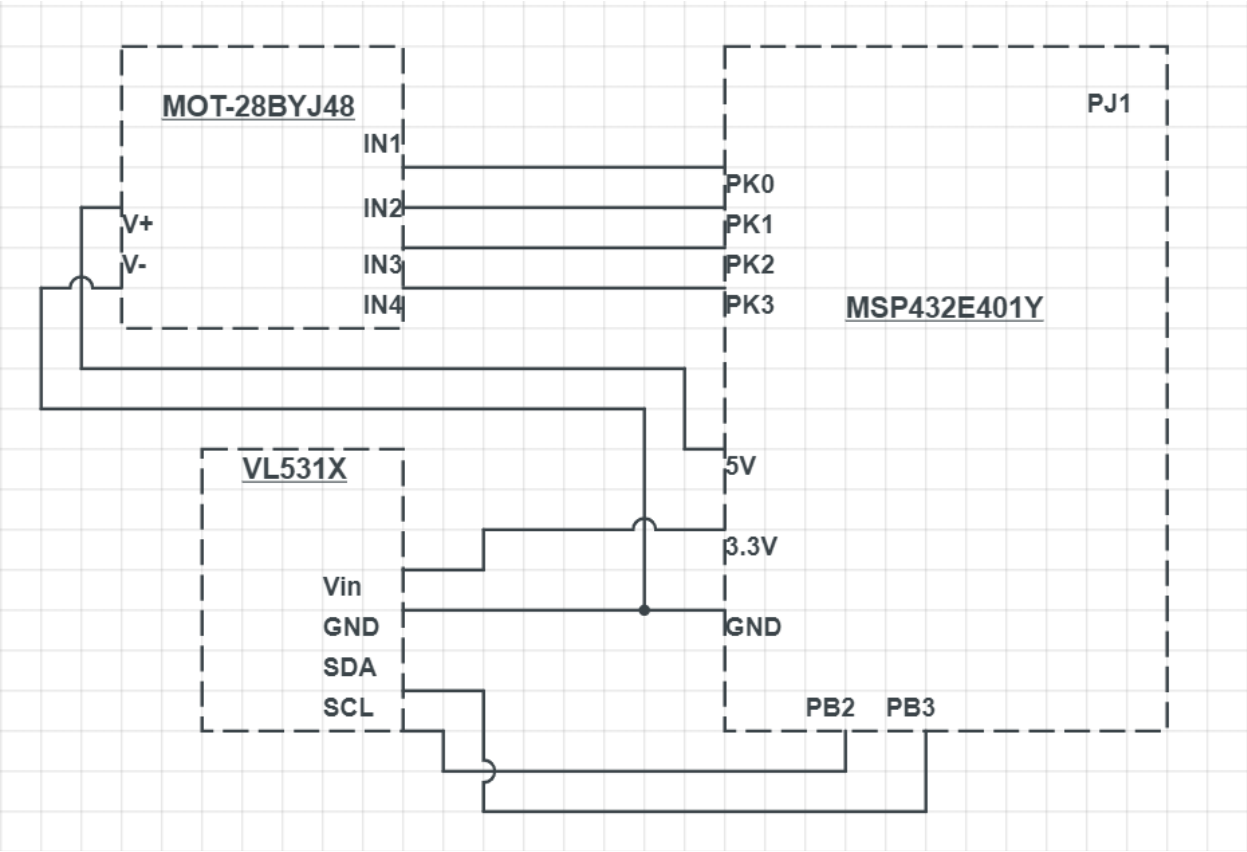

Figure 8: Device circuit schematic