3-D Visualization Using Time-of-Flight Sensor Using MSP MCU

By Vinay Gajjar

Special Thanks to: Dr.Bruce, Dr. Haddara, Dr. Hranilovic, Dr. Shirani

1. Device Overview

1.1. Features

This device can be used to measure distances of surrounding planes to recreate a 3D representation of the surrounding space using the MOT-28BYJ48 Stepper Motor with ULN2003 Driver which costs \$7.04 and a VL53L1X time-of-flight(ToF) module which costs \$15.93. The device is controlled using the MSP-EXP432E401Y microcontroller which costs \$65.56 has the following specifications:

- The default Bus speed is 120Mhz, but was used at 60Mhz in this device.
- Operating voltage range is from 2.97-3.63V, Nominal voltage is 3.3V.
- Memory :1024KB of Flash Memory, 256KB SRAM, 6KB EEPROM and internal ROM.
- Two 12-bit SAR-Based ADC modules, that support upto 2 million samples per second.
- The software language used is C/C++ and assembly language.
- It has eight UART each with a baud rate of 115200 bits per second.

The VL53L1X can be used to measure distances of up to 4 meters, and operating voltage range of between 2.6 V to 5.5 V. The communication protocol used is I^{2}C.

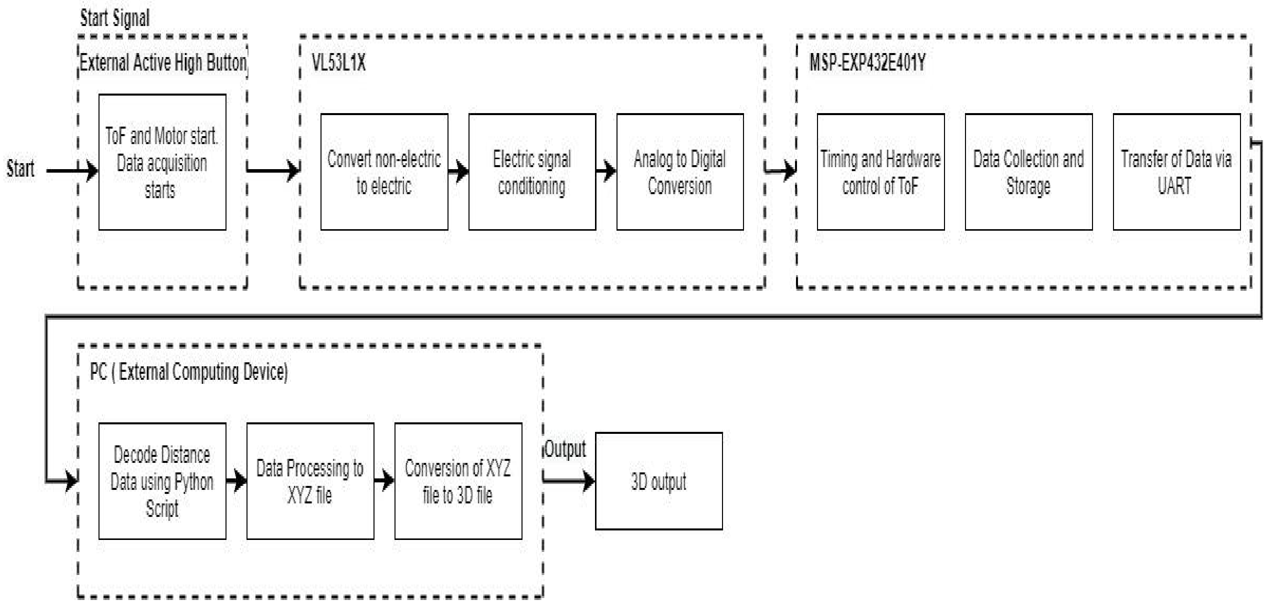
1.2. General descriptions

This is an embedded system which combines measurement devices together with control devices that can be used to get a mapping of the surrounding closed areas within a finite range such as enclosed building spaces, for use as a component within other systems (e.g navigation and layout mapping).

This system consists several parts working together:

- A microcontroller is used to control and communicate with the ToF sensor and Stepper Motor.
- A ToF sensor is used to measure distances(in this case the y-z plane).
- A motor is used to shift the ToF sensor in a single plane to allow for a 360 degree reading.
- An external push button is used with an active high configuration to signal the motor to start spinning and begin the recording of the distance measurements.
- There are Digital I/O onboard LEDs on the microcontroller that are used to provide the status for the ToF sensor as it records data.
- A PC that is used to power up this device, and is where the data is communicated to using UART for further processing to generate a 3D model.

1.3. Block diagram


Start Signal
External Active High Button
ToF and Motor start. Data acquisition starts
Start
VL53L1X
Convert non-electric to electric
Electric signal conditioning
Analog to Digital Conversion
MSP-EXP432E401Y
Timing and Hardware control of ToF
Data Collection and Storage
Transfer of Data via UART
PC (External Computing Device)
Decode Distance Data using Python Script
Data Processing to XYZ file
Conversion of XYZ file to 3D file
Output
3D output

The ToF is used as a transducer to detect distance measurements. This signal is then preconditioned to match the ADC design. The processed signal is then transmitted to the ADC and stored in a digital form. This digital data can be further processed to reproduce distance measurements and 3D plots on external systems.

Circuit picture and PC display are shown in **figure 1. and figure 2.**

2. Device Characteristics Table

Feature	**Description**
ToF sensor pins SCL/SDA	PB2/PB3
Stepper motor pins	PH0-PH3
External push button input	PM0
Bus speed	60MHz
Communication protocols	UART and I²C
UART/ I²C, communication speed	115200 bits per second/100Kbps
Communication port	COM4

3. Detailed descriptions

3.1. Distance measurement

The VL53L0X is a tiny lidar system featuring a Class 1 laser. Unlike regular infrared sensors this TOF uses intensity of reflected light to estimate the distance to an object. It works with pulses of infrared laser light that precisely measure the time delay between an emitted pulse and when it is received at the detector.

$$Measured\ Distance\ in\ mm\ =\ 0.5\ *\ Time\ of\ Laser\ Travel\ *\ speed\ of\ Laser(light)$$

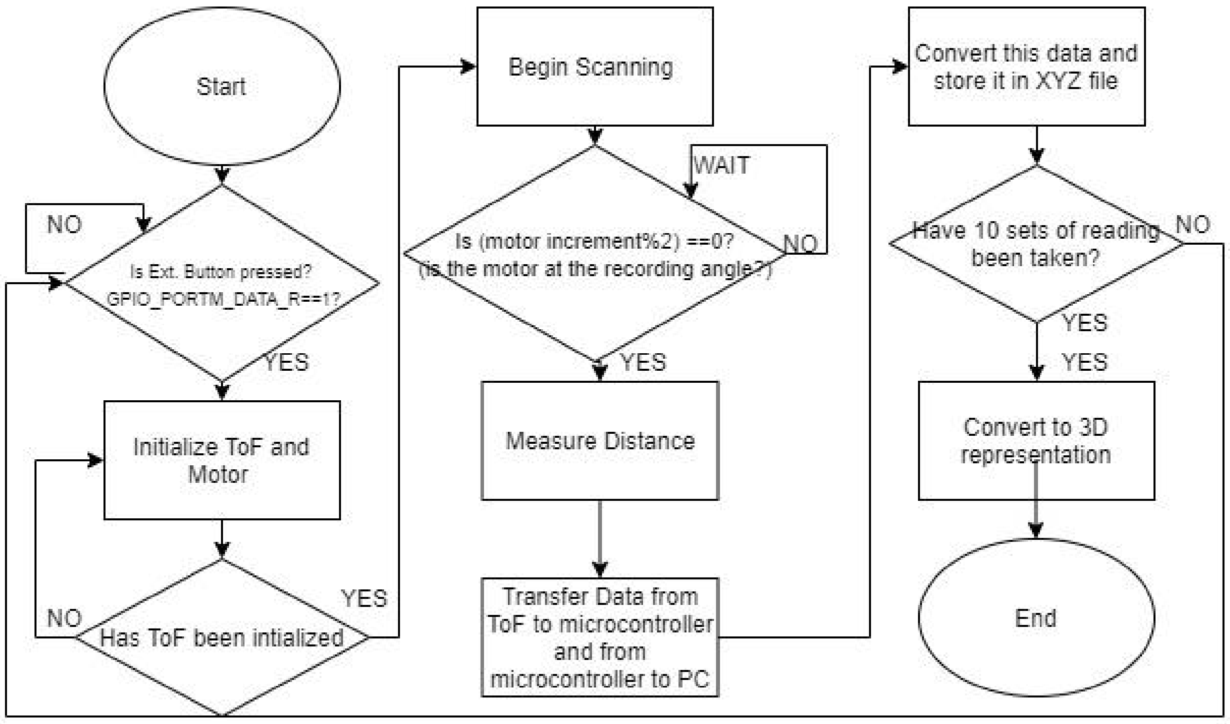
The overall distance measurement program is defined as follows:

- Digital input: An external active high push button is used to set PM0 to a logic high, which signals the ToF to initialize and once this is done, the motor starts rotating and the ToF begins data acquisition.
- Data acquisition: This is done via the ToF which takes in physical input (measurement) and converts this signal into an electrical signal. This electric signal is then automatically preconditioned to match the Analog to Digital design (ADC). This electrical signal goes through the ADC. The ADC module samples the waveform to generate the discrete digital value representations for use by any digital system.
- Data transmission: The data collected from the ToF is transferred to the MSP-EXP432E401Y via serial communication using the I^2C communication protocol at a rate of 100Kbps. This data is then sent to the PC/(external computing device) via serial communication using the UART communication protocol at a baud rate of 115200 bits per second, using the ‘COM4’ as the port of communication.
- Data conversion: The data that received from the MSP-EXP432E401Y is read using a python script (detailed description in section 3.2.) that decodes the data and converts the distance measurements to x,y,z coordinates. The coordinates are calculated using mathematical formulae that use the ToF reading together with the angle of rotation at which this reading was taken. The mathematical formulae are shown below:

  $$X\ coordinate\ =\ increments\ of\ X\ units\ (movement\ of\ entire\ device)$$
  $$Y\ coordinate\ =\ distance\ *\ cos(angle)$$
  $$Z\ coordinate\ =\ distance\ *\ sin(angle)$$

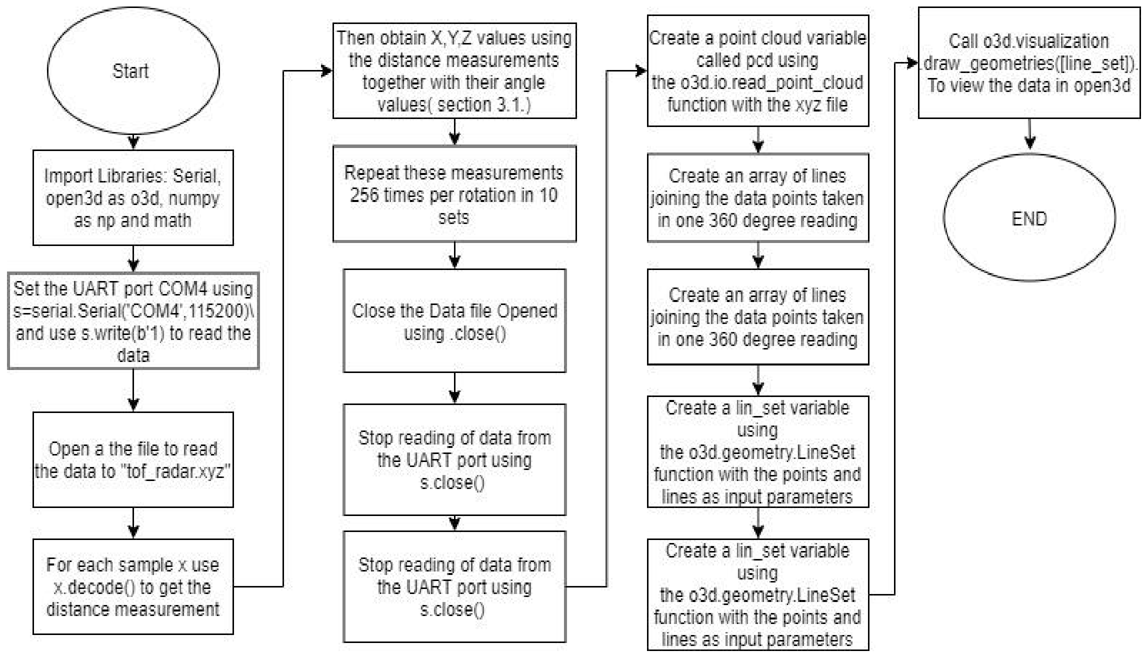
  Example measurement with Distance = 250mm, Angle = 50°:

  $$X\ coordinate\ =\ 0.00\ (first\ reading)$$
  $$Y\ coordinate\ =\ 160.68$$
  $$Z\ coordinate\ =\ 191.51$$


Start
Is Ext. Button pressed? GPIO_PORTM_DATA_R==1?
NO
YES
Initialize ToF and Motor
Has ToF been intialized
NO
YES
Begin Scanning
Is (motor increment%2) ==0? (is the motor at the recording angle?)
WAIT
NO
YES
Measure Distance
Transfer Data from ToF to microcontroller and from microcontroller to PC
Convert this data and store it in XYZ file
Have 10 sets of reading been taken?
NO
YES
YES
Convert to 3D representation
End

Flow chart of distance measurement program

3.2. Visualization


Start
Import Libraries: Serial, open3d as o3d, numpy as np and math
Set the UART port COM4 using s=serial.Serial('COM4',115200)\ and use s.write(b'1) to read the data
Open a the file to read the data to "tof_radar.xyz"
For each sample x use x.decode() to get the distance measurement
Then obtain X,Y,Z values using the distance measurements together with their angle values(section 3.1.)
Repeat these measurements 256 times per rotation in 10 sets
Close the Data file Opened using .close()
Stop reading of data from the UART port using s.close()
Stop reading of data from the UART port using s.close()
Create a point cloud variable called pcd using the o3d.io.read_point_cloud function with the xyz file
Create an array of lines joining the data points taken in one 360 degree reading
Create an array of lines joining the data points taken in one 360 degree reading
Create a lin_set variable using the o3d.geometry.LineSet function with the points and lines as input parameters
Create a lin_set variable using the o3d.geometry.LineSet function with the points and lines as input parameters
Call o3d.visualization draw_geometries([line_set]). To view the data in open3d
END

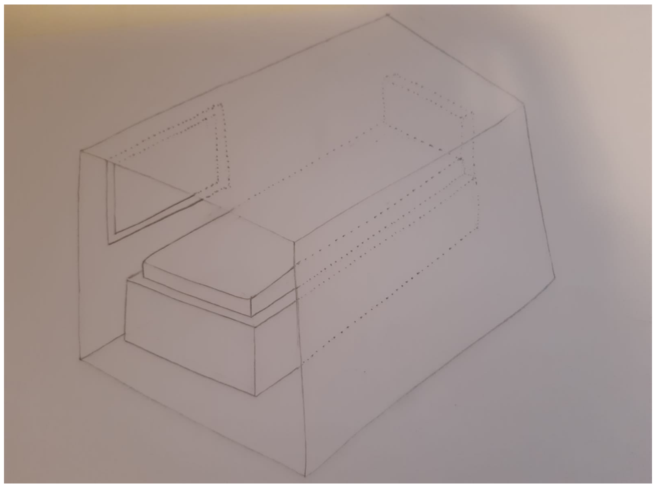
Flow chart of visualization program

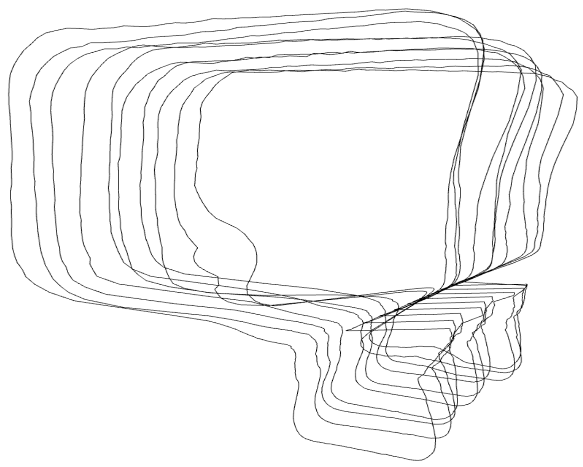
Device Specifications
The computer system used was a MacBook pro 2019 model, running bootcamp with an operating system of Windows 10 and Python Version 3.8.8.
Some of the specifications of the computer are listed below:

- Core is 8th Generation2.4 GHz Intel Core i processor (I5-8279U).
- Memory includes,128 MB of eDRAM embedded on the processor die, and a 6 MB shared level 3 cache. 8 GB of onboard 2133 MHz LPDDR3 SDRAM, 256 GB or 512 GB of flash storage.
- Integrated Intel Iris Plus Graphics 655 graphics processor.
- Two Thunderbolt 3 ports (USB C).

The overall data visualization program is as follows:

- Data acquisition: This data is received via UART from the microcontroller, this is then run with Python (3.8.8) using a script to process the data.
- Libraries used : serial(pyserial), open3D, numpy and math libraries were used.
- Data Storage and Processing (Functions used): To read the data the serial libraries are used and the *serial.Serial('port',baud rate)* function is used to set the input port for UART communication. This data is converted to a plane of coordinates (X Y Z) and stored using the file i/o functions in Python. For open3D the function *o3d.read_point_cloud* to read in data from the xyz file and used together with *03d.geometry.LineSet* to create the line set for visualization.
- Data Visualization: The processed data from the ToF can be visualized using Python and Open3D. The xyz data is read into open3d as a cloud of data. This data is processed using Lineset and points on the cloud are connected. The planes are also connected this way. Then this can be visualized directly open3D as a 3D representation of the recorded data.


Isometric view of capture location

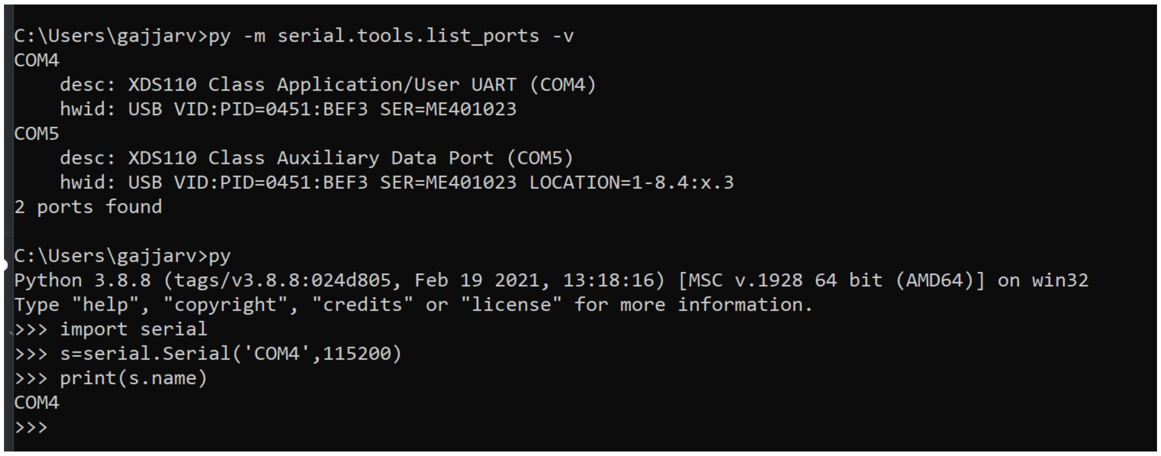

3D output from program

4. Application Example

To use this device the following steps can be followed:

1. To set up the programming software to use with the computer, download and install Python version 3.8.8.
2. Now open the Command prompt or similar on your operating system and then type in the the following commands:
   pip install serial
   pip install pyserial
   pip install open3d
   pip install numpy
3. Once these libraries are installed, then in the prompt type in python/py to enter the python terminal or exit() to exit out of python.
4. To set up the communication port between the computer and the microcontroller via UART the following steps can be performed (Make sure the microcontroller is connected via USB before testing this out):

   - To check which ports your computer can access type in the following command *python -m serial.tools.list_ports –v or py -m serial.tools.list_ports -v.* This will display all the ports available for access.
   - Type in py/python to access python and then type the commands as shown below to test whether the port for UART is working.

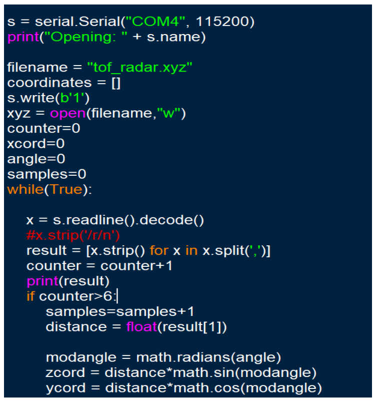


```
C:\Users\gajjarv>py -m serial.tools.list_ports -v
COM4
    desc: XDS110 Class Application/User UART (COM4)
    hwid: USB VID:PID=0451:BEF3 SER=ME401023
COM5
    desc: XDS110 Class Auxiliary Data Port (COM5)
    hwid: USB VID:PID=0451:BEF3 SER=ME401023 LOCATION=1-8.4:x.3
2 ports found

C:\Users\gajjarv>py
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import serial
>>> s=serial.Serial('COM4',115200)
>>> print(s.name)
COM4
>>>
```

   - Set a variable s = serial.Serial('COM4',115200) which represents an object in serial UART with port COM4 and a baud rate of 115200 bits per second (depends on microcontroller). COM4, can be changed based on which ports your computer has available. The print function ensures that the port has been correctly set.

5. The next steps involve setting up the actual device. The device consists of four main components. The microcontroller, the ToF sensor, the stepper motor and the external push button.
6. To start the process ensure that the microcontroller is powered on and the code to run the program(from keil) has been loaded and flashed onto the mcu.
7. Ensure that all the connections for all the devices have the right pinout and connected to the right supplies.
8. Make sure that the python script(talked more in detail in section 3.2.) is running before you run the device
9. Push the external button to turn on the system, and this should cause the motor to start rotating 360 degrees.
10. To ensure the ToF is taking readings look at the onboard led D4 that flashes each time the ToF takes a reading.After 1 rotation the device comes to a stop. To start the motor and scanning process again press the external button once more. This step will be repeated 10 times in total for 10 readings to be obtained.
11. The data collection process happens through a Python script via UART communication between the microcontroller and the computer.
12. The pseudo code for what the code should do is a follows:
    - Import the serial, numpy ,math and open3d libraries.
    - Open the port for UART communication using s = serial.Serial("COM4", 115200) and using s.write(b'1') to read from the MCU.
    - Take this data and use the serial decode function to convert the data to a string.
    - Then perform file i/o to extract the distance measurement from the incoming data.
    - To convert the data from distance to X Y Z coordinates using the sin and cos math functions together with the distance. Eg. Y = distance*cos(angle),Z = distance*sin(angle)
    - Store this data in the format X Y Z in a file (shown to the right). Since The motor rotates in the YZ plane the X component is manually incremented in 10cm increments.

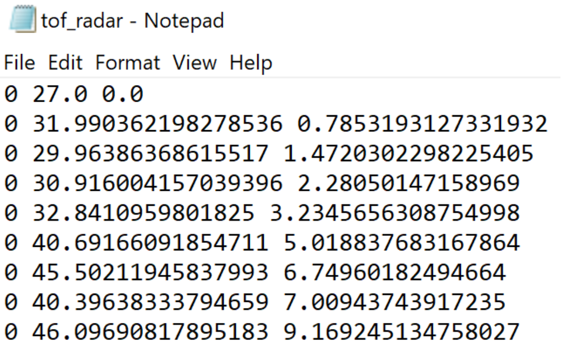


```
s = serial.Serial("COM4", 115200)
print("Opening: " + s.name)

filename = "tof_radar.xyz"
coordinates = []
s.write(b'1')
xyz = open(filename,"w")
counter=0
xcord=0
angle=0
samples=0
while(True):

    x = s.readline().decode()
    #x.strip('/r/n')
    result = [x.strip() for x in x.split(',')]
    counter = counter+1
    print(result)
    if counter>6:
        samples=samples+1
        distance = float(result[1])

        modangle = math.radians(angle)
        zcord = distance*math.sin(modangle)
        ycord = distance*math.cos(modangle)
```



tof_radar - Notepad

File Edit Format View Help

```
0 27.0 0.0
0 31.990362198278536 0.7853193127331932
0 29.96386368615517 1.4720302298225405
0 30.916004157039396 2.28050147158969
0 32.8410959801825 3.2345656308754998
0 40.69166091854711 5.018837683167864
0 45.50211945837993 6.74960182494664
0 40.39638333794659 7.00943743917235
0 46.09690817895183 9.169245134758027
```

13. To visualize the data any 3D visualization tool can be used with the X Y Z file. In this program the open3D library was used with point cloud visualization and lineset functions.

5. Limitations

Within the overall system there are a few areas that could not reach maximum potential for the best performance of this device due to device constraints. These areas are discussed below and could be mitigated by ensuring future upgrades with the individual components.

1. The microcontroller has a floating point unit that performs certain single precision mathematical functions. This is limited to 32 bits which is about 7.22 decimal places, and this value cannot be exceeded and hence the distance measurements are limited to a precision set by this value. The in build trigonometric functions were not used within this code, however the math library can be used to perform calculations. It uses double precision parameters, while the FPU can only handle 32-bit operations, so processing time would overall be longer.
2. The maximum quantization for the ToF module is given by:

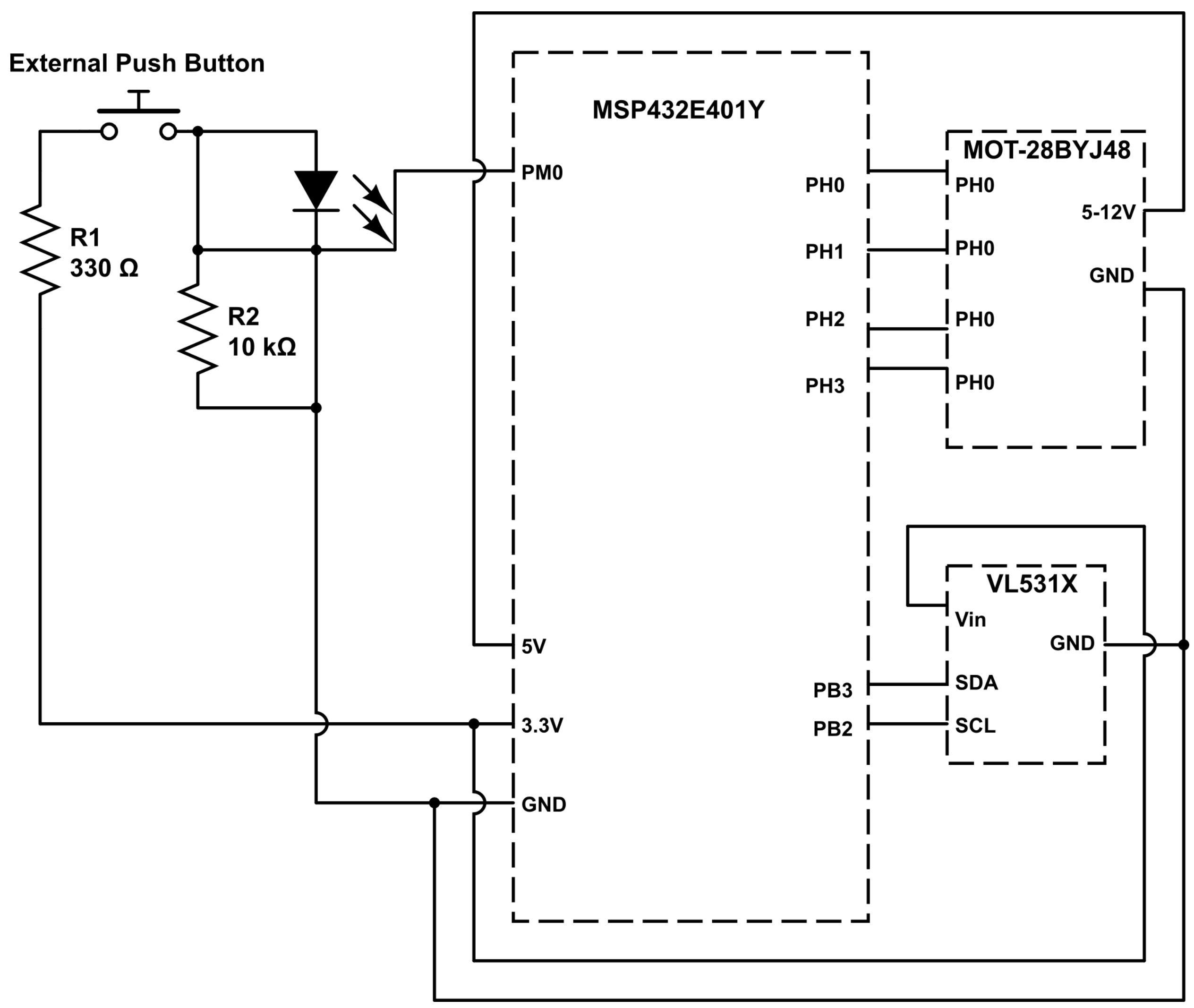
   $$Max\ quantization\ error\ = Maximum\ Distance/2^{m}$$

   Where the maximum distance is the max range for the ToF and m is the number of bits,

   $$Max\ quantization\ error\ = 4\ meters/2^{16}\ =\ 6.10\ *\ 10^{-5}m$$

3. The maximum standard serial communication rate with the PC is 115200 bits per second. This was tested by increasing the baud rate upto a value when there was a transmission error in the data. Since the input data is coming in faster than can be processed by UART.
4. The method of communication is serial communication using the communication protocol I²C with a speed of 100Kbps.
5. Reviewing the entire system, the element that is the primary limitation of the speed is the ToF sensor (the motor has a speed limit but it is way higher than that of the ToF). The ToF is limited to a maximum reading rate of 50Hz, which means that the more the readings that were taken the slower the motor would rotate. This was tested by setting the ToF at maximum sampling rate and increasing the motor speed slowly to almost maximum, despite this the time to scan a full 360 degrees remained almost constant. This meant that the speed was dependent on the ToF in this system.

6. Circuit Schematic


External Push Button
R1
330 Ω
R2
10 kΩ
MSP432E401Y
PM0
5V
3.3V
GND
PH0
PH1
PH2
PH3
PB3
PB2
MOT-28BYJ48
PH0
PH0
PH0
PH0
5-12V
GND
VL531X
Vin
GND
SDA
SCL

Appendix

Figure 1.

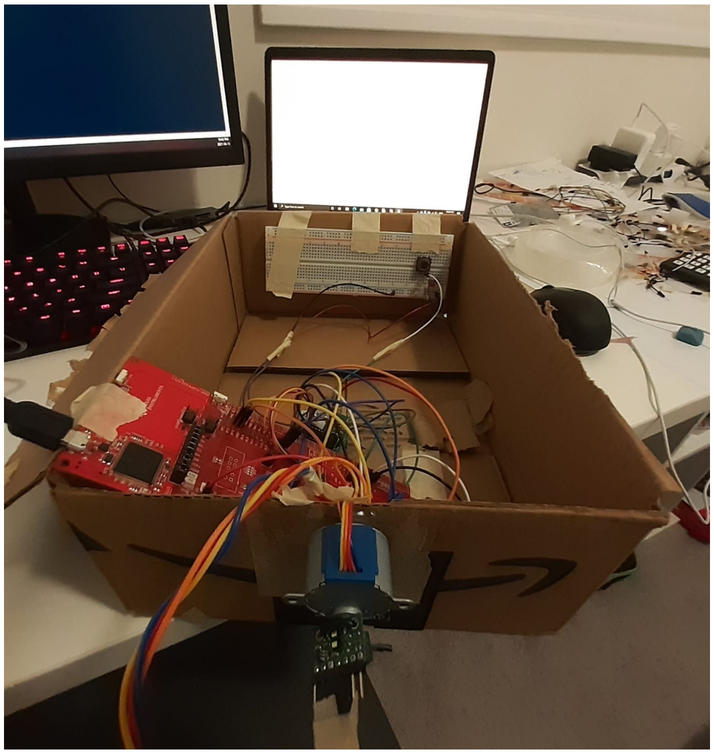
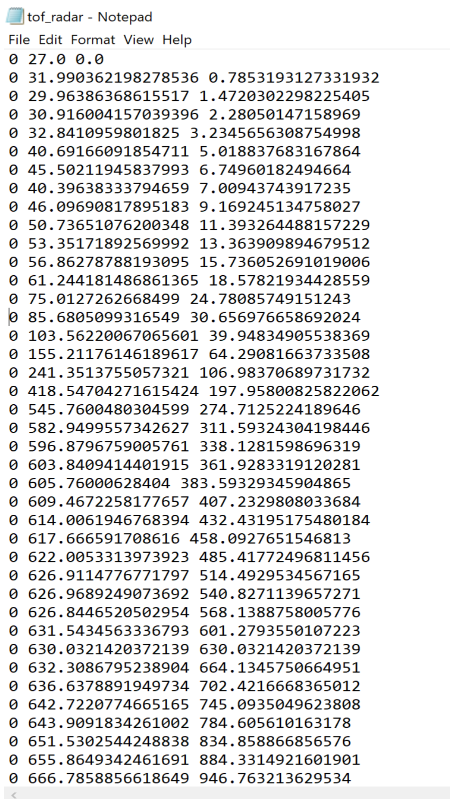


Figure 2.



```
0 27.0 0.0
0 31.990362198278536 0.7853193127331932
0 29.96386368615517 1.4720302298225405
0 30.916004157039396 2.28050147158969
0 32.8410959801825 3.2345656308754998
0 40.69166091854711 5.018837683167864
0 45.50211945837993 6.74960182494664
0 40.39638333794659 7.00943743917235
0 46.09690817895183 9.169245134758027
0 50.73651076200348 11.393264488157229
0 53.35171892569992 13.363909894679512
0 56.86278788193095 15.736052691019006
0 61.244181486861365 18.57821934428559
0 75.0127262668499 24.78085749151243
0 85.6805099316549 30.656976658692024
0 103.56220067065601 39.94834905538369
0 155.21176146189617 64.29081663733508
0 241.3513755057321 106.98370689731732
0 418.54704271615424 197.95800825822062
0 545.7600480304599 274.7125224189646
0 582.9499557342627 311.59324304198446
0 596.8796759005761 338.1281598696319
0 603.8409414401915 361.9283319120281
0 605.76000628404 383.59329345904865
0 609.4672258177657 407.2329808033684
0 614.0061946768394 432.43195175480184
0 617.666591708616 458.0927651546813
0 622.0053313973923 485.41772496811456
0 626.9114776771797 514.4929534567165
0 626.9689249073692 540.8271139657271
0 626.8446520502954 568.1388758005776
0 631.5434563336793 601.2793550107223
0 630.0321420372139 630.0321420372139
0 632.3086795238904 664.1345750664951
0 636.6378891949734 702.4216668365012
0 642.7220774665165 745.0935049623808
0 643.9091834261002 784.605610163178
0 651.5302544248838 834.858866856576
0 655.8649342461691 884.3314921601901
0 666.7858856618649 946.763213629534
```