Microprocessors Systems [COMPENG 2DX3] Final Report

Instructor: Dr. Haddara/Doyle/Athar

Zayeed Ghori – L06 – ghoriz1- 400398943

3D Scanner

Features

- Configurable Bus/Clock Speed (default: 30 MHz)

- Operating Voltage (2.5 – 5V)

- 2 12-Bit SAR-Based ADC modules

- 16 Digital Comparators

- C/C++ Compiler

- ASM compiler

- 8 UARTS each with independent transmitter and receiver

- 10 I2Cs with high-speed support

- 115200 Baud rate (bps)

- Cost of approximately 45-70 CAD

VL53L1X ToF Sensor:

- 2.6-5.5 V Operating Voltage Range

- Single Power Supply (2v8)

- Emitter: 940 nm invisible laser

- SPAD receiving array with integrated lens

- Maximum of 4000 mm distance measurement

- Maximum of 50 Hz ranging frequency

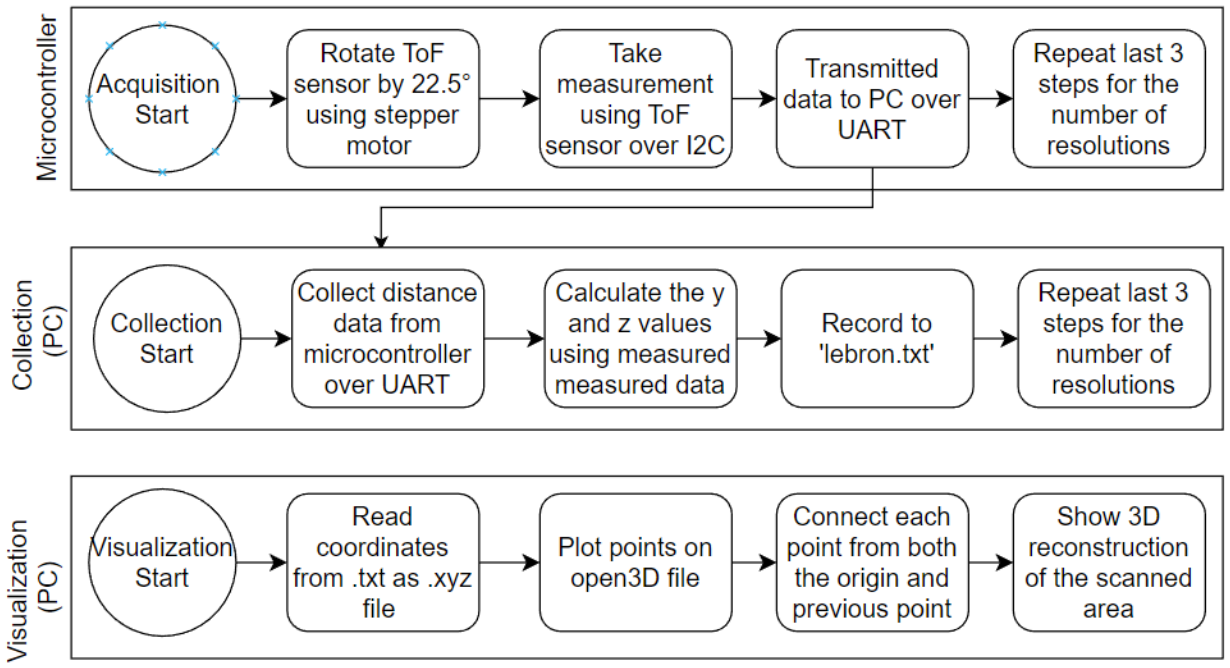
- I2C Interface for up to 400 kHz

- Cost of 6-25 CAD

General Description

This device uses a LiDar sensor to scan a 3D diagram activated by a push button. Using the Texas Instrument MSP432E401Y microcontroller, which is programmed using C, activates a stepper motor to turn it in increments of 22.5 degrees, and stops briefly to capture readings in the y-z axis using I2C to communicate distance readings from the VL53L1X ToF Sensor to the microcontroller. The analog reading from ToF sensor is converted internally using a 16-bit ADC conversion to a digital time signal, then using the speed of light converts this into a distance, which is then transmitted to the microcontroller over I2C. User movement is then required to complete multiple scans in the x direction to complete a diagram. The microcontroller sends a UART signal through USB to the operating computer. The computer Python program takes the UART distances and assumes that the first UART transmission is at 0 degrees and each successive transmission is at an additional 22.5 degree increments. Using simple trigonometry, the distance value is converted into y and z measurements.

Block Diagram


Microcontroller
Acquisition Start
Rotate ToF sensor by 22.5° using stepper motor
Take measurement using ToF sensor over I2C
Transmitted data to PC over UART
Repeat last 3 steps for the number of resolutions
Collection (PC)
Collection Start
Collect distance data from microcontroller over UART
Calculate the y and z values using measured measured data
Record to 'lebron.txt'
Repeat last 3 steps for the number of resolutions
Visualization (PC)
Visualization Start
Read coordinates from .txt as .xyz file
Plot points on open3D file
Connect each point from both the origin and previous point
Show 3D reconstruction of the scanned area

Device Characteristics

Table 1: System Characteristics

Characteristic	**Value/Correspondence**
Time of Flight Sensor Ports	Vin (3.3 V), GND, SCL (Port B2), SDA (Port B3)
ToF Range	40mm to 4000mm
Push Button	Port M0 (Configured with Active HIGH Polling), 3.3 or 5V, and GND with Pull down resistor (~300 Ohms)
Serial Communication Speed	115200 Baud rate (bps)
Default Serial Port	COM9
Bus Speed	30 MHz
Stepper Motor	Port H0:H3, Vin (5V DC), GND
Special Python Libraries	math, serial, numpy as np, open3d as o3d
Required uC header files	PLL, SysTick, tm4c1294ncpdt, vl53l1x_api, uart,

Detailed Description

Distance Measurement

Once the circuit is set up according to the diagram in the Circuit Schematic section, the system is ready to start its work. The distance measurement process starts by initializing and configuring all necessary ports, such as Port H for sending signals to the stepper motor, Port N for the LED indicator light, Port B for I2C, and Port M for the push button. The main part of the distance measurement process, which includes the first three sections of the Programming Logic Flowchart, can now be initiated.

The Vl53L1X time-of-flight sensor is used to collect spatial data by being placed on a motor shaft that completes four cycles, two clockwise and two anti-clockwise, with a pause after turning 22.5 degrees and completing 16 steps per cycle. The bus speed is set to 30MHz by default (due to the 2nd least significant digit of the student number on the title page). The Systick_wait() function is used to introduce a time delay after each step. Using flight-sense technology, the time-of-flight sensor measures the time between emission and reception of reflected infrared light rays in the 16x16 SPAD detector from nearby objects. Once a HIGH input is activated using the push button in PM0, a 940 nanometer class 1 laser is used in the sensor's LIDAR system to emit a pulse of photons and record the time of travel until the photons bounce back. The distance is calculated using the formula:

Distance = Speed of light (299792458 m/s in vacuum) * 10 ^3 mm/m * (time of travel)/2

Ex:

For an object 1000mm away:

*Distance = 299792458 m/s * 10 ^ 3 mm/m * (6.6712819 * 10^-9 s)/2 = ~1000mm*

The duration of the photons' round trip to the nearest object is measured to obtain the displacement between the two objects. This value is multiplied by 10^-3 because the sensor outputs the distance measurement in millimeters to the microcontroller via I2C. Once the distance measurement is obtained from the ToF sensor, it is saved to an integer variable and sent to the PC via UART through the USB port. An onboard LED (D3) is flashed to indicate that a distance measurement has been taken. The PC receives and stores the data using a Python program. The microcontroller then rotates the stepper motor 32 steps or 22.5 degrees using the full step method and signals the sensor to obtain a new distance measurement. After 16 distance measurements have been taken and transmitted to the PC, the motor rotates in the opposite direction to avoid wire entanglement. The system then waits for the user to move forward the specified increment, which should be in the positive x direction, and repeat the process by pressing the push button. This process can be repeated until the desired number of cross sections is obtained.

Visualization

The data visualization process is performed by the deliverable.py Python program on the PC. Initially, necessary libraries such as serial, math, open3d (as o3d), and numpy (as np) are imported using pip. The program creates a variable "s" representing the serial portm, "COM9" by default, with a baud rate of 115200 bps, similar to that of the microcontroller, and opens the port while clearing any existing data. A “.txt” file is opened for writing; if the file with the name “lebron.txt” already exists, it will be overwritten, and if not, it will be created. Variables to keep track of the number of steps the motor has taken and the initial forward displacement in the x direction are initialized. The program waits for the user to enter the number of full scans they plan to take. After obtaining an integer value from the user, the system is ready to receive data from the microcontroller and signals it by sending “start” through the open serial port. The program then reads and decodes the data from the microcontroller while discarding any newline or return characters.

To ensure that only numerical data is written to the coordinate file, the program verifies the data using the predefined ".isdigit()" method and a while loop until integer values are read. The angle of rotation of the motor is calculated by dividing the total number of steps taken by the number of steps required for a full 360-degree rotation and converting it to radians using “math.radians(degrees)”. The vertical y-coordinate and horizontal z-coordinate are then calculated by multiplying the transmitted distance measurement by the sine and cosine of the angle, respectively. The received data is converted to 2D coordinates using the formulas:

y = (distance obtained) * cos (angle of rotation in radians)

z = (distance obtained) * sin (angle of rotation in radians)

To capture the third coordinate, one step is taken while receiving data (which is assumed to be 16cm), resulting in z. The final 3D model is created by plotting the results in the xyz plane. The data processing is shown in the following flowchart through the microcontroller.

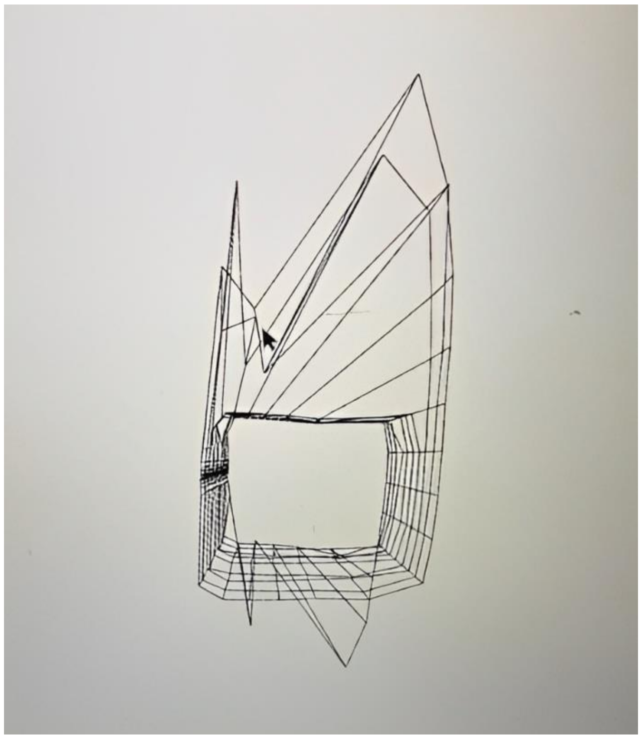
Application Example

The device can be used to map corridors or rooms with dimensions of a maximum of 4m (4000mm) away from the sensor or can be used with smaller objects with a minimum of 4cm (40mm) or distance to the sensor.

The example provided is of a hallway.

Hallway:

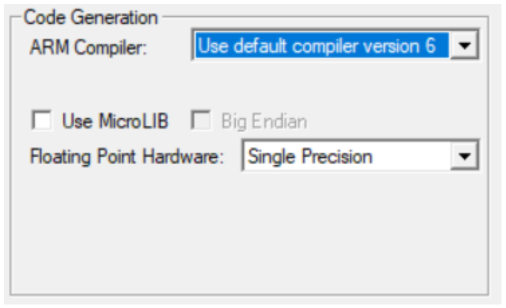
Example Expected Output:



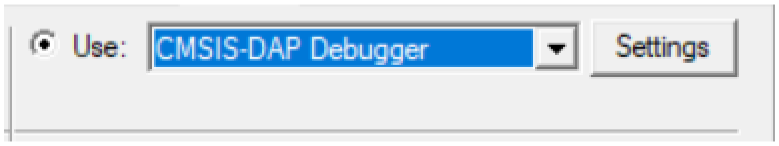
User Guide

1. Connect the MSP432E401Y microcontroller, VL531X ToF sensor, stepper motor and push button as outlined in the circuit schematic, taking special care that all the pins are secured to the correct wires.
2. Ensure that the motor is mounted to a flat vertical wall and that the ToF sensor is securely placed in its mount and then put onto the stepper motor shaft.
3. Connect the USB A to Micro-USB wire to the side of the microcontroller without the large ethernet (networking) connector (As shown in the device setup)
4. Go to Device Manager (on Windows) and search for Serial Connections. Take note of the COM# of the connection that says UART,
5. Install Python 3.6 to 3.10.
    a. Make sure that python is added to PATH.
6. Open Command Prompt (Windows):
    a. Type in “python --version” and press Enter to verify Step 4 is completed correctly.
    b. Type “pip install pyserial”, press Enter.
    c. Type “pip install open3d”, press Enter.
7. Open Python IDLE
    a. Open “deliverable2.py”
    b. Change “COM9” in line 17 to the COM# found in step 3 and save.
    c. Keep open for Step 8.
8. Install Keil uVision 5 and open it.
    a. Open “2dx_studio_8c”

b. Click on the options for Target 1 and ensure that the ARM compiler is set to default:


Code Generation
ARM Compiler: Use default compiler version 6
Use MicroLIB
Big Endian
Floating Point Hardware: Single Precision

c. Go to the Debug tab in the Target 1 Options and ensure that the CMSIS Debugger is used:

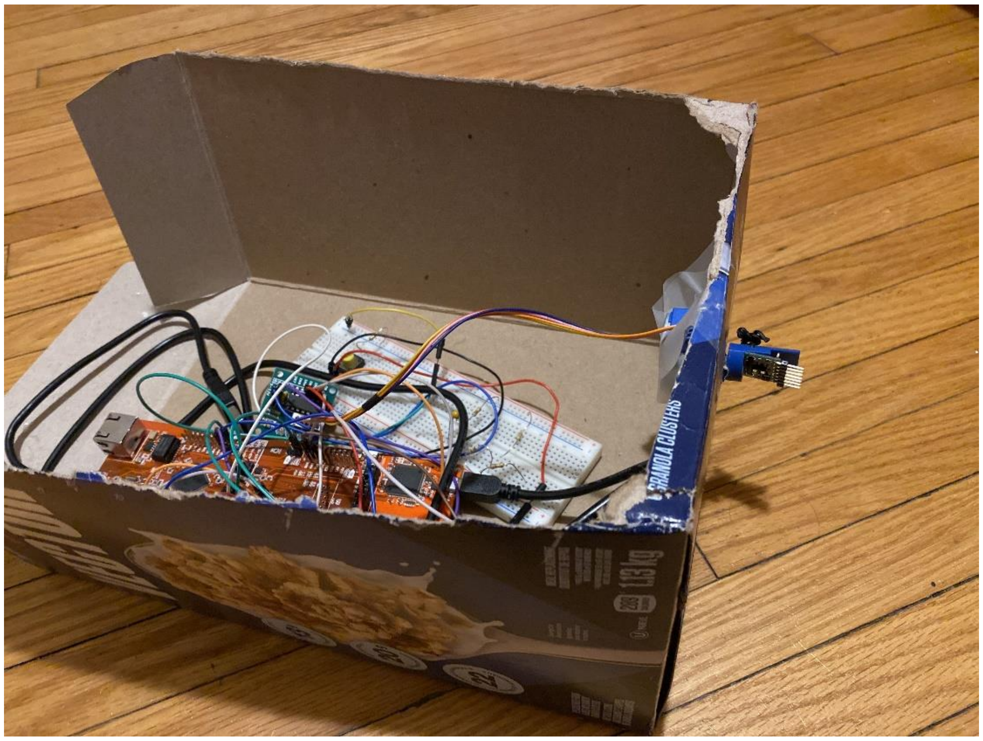

Use: CMSIS-DAP Debugger

d. Exit the Target 1 Options menu.

e. Click on Translate , then Build , then Load and ensure there are no Errors that prevent the microcontroller from being flashed.

9. Run “deliverable2.py” in Python IDLE
   a. Press Enter when prompted.
10. Move to the location you would like to scan.
11. Press the reset button on the microcontroller.
12. Press the push button to start the y-z scan.
   a. The motor will do a full 360 degree rotation, stopping ever 22.5 degrees to take measurements, ensure that the ToF wires don’t get tangled.
13. Once the first scan is complete, take one step either forward or backward (ie. in the x-direction).
14. Repeat steps 11-13 (continuing to move forward if done so in the previous step or backwards, not changing the direction of movement) until 3 cycles have completed.
15. Once the process is complete, a graph of dots will appear in a new Window.
16. Close that Window to view the 3D model.

Figure 2: Prototype



Limitations:

Our device's floating-point unit is limited to single precision mathematical operations. However, the Python code will use double precision calculations, which may result in some inaccuracies when converting single precision measurement values to double precision. Additionally, when calculating trigonometric values using Python functions, the degrees of rotation are converted to radians by multiplying values by pi and dividing by 180.

The maximum quantization error is calculated as Vfs / 2^m, where Vfs is 3.5 and m is 12, resulting in a value of 0.000854.

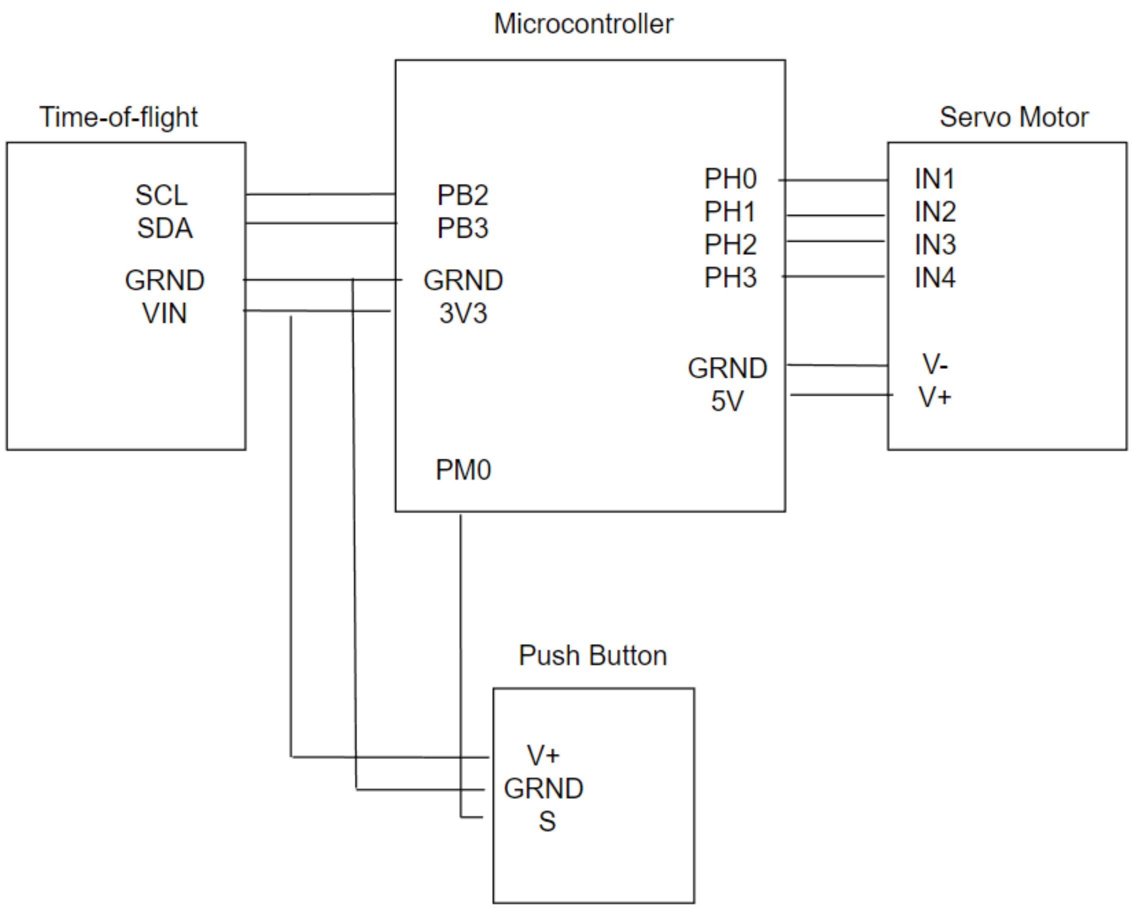
To determine the maximum serial communication rate between the device and the computer, I navigated to the COM4 port in the device manager and checked the properties, which showed that the maximum rate is 128K bits/s.

The communication between the sensor and the microcontroller is accomplished through I2C serial communication at a speed of 400K bits/s and the address of 0X52.

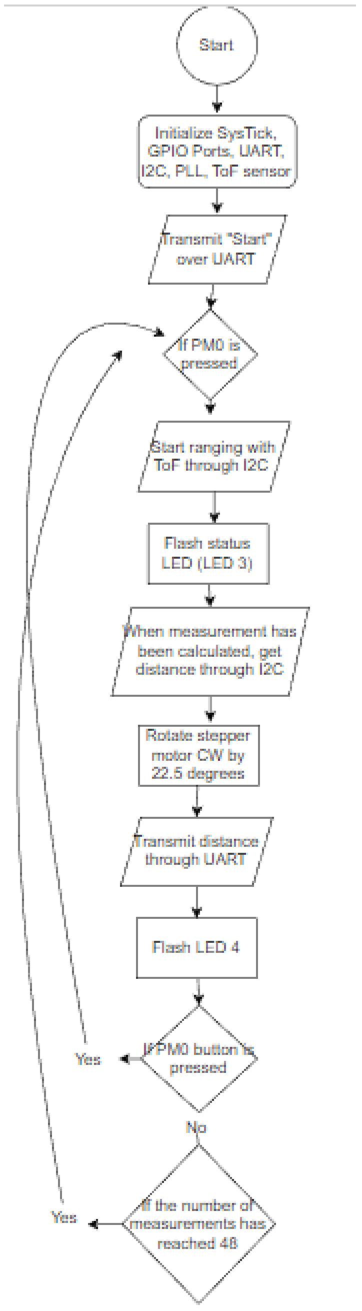
The primary limiting factor for speed was the stepper motor, which required a 0.01s delay to operate properly at a frequency of 100Hz. This delay is obtained using the formula period = 1/f = 1/100s = 0.01s.

The ToF sensor has a range of 40mm to 4000mm limiting the features that can be scanned within a radius of 40mm to 4000mm.

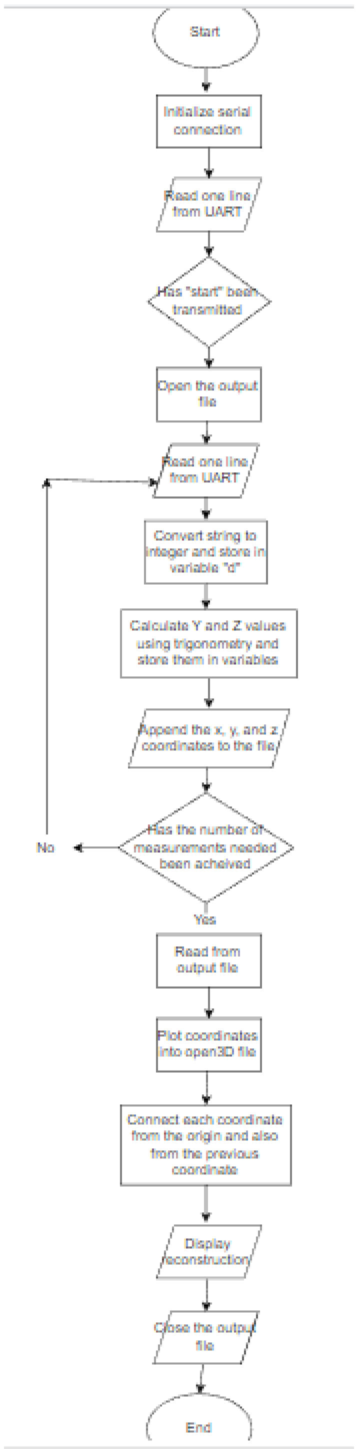
Schematic



Microcontroller Code:


Start
Initialize SysTick, GPIO Ports, UART, I2C, PLL, ToF sensor
Transmit "Start" over UART
If PM0 is pressed
Start ranging with ToF through I2C
Flash status LED (LED 3)
When measurement has been calculated, get distance through I2C
Rotate stepper motor CW by 22.5 degrees
Transmit distance through UART
Flash LED 4
If PM0 button is pressed
Yes
No
If the number of measurements has reached 48
Yes

Python Code:


Start
Initialize serial connection
Read one line from UART
Has "start" been transmitted
Open the output file
Read one line from UART
Convert string to integer and store in variable "d"
Calculate Y and Z values using trigonometry and store them in variables
Append the x, y, and z coordinates to the file
Has the number of measurements needed been acheived
No
Yes
Read from output file
Plot coordinates into open3D file
Connect each coordinate from the origin and also from the previous coordinate
Display reconstruction
Close the output file
End