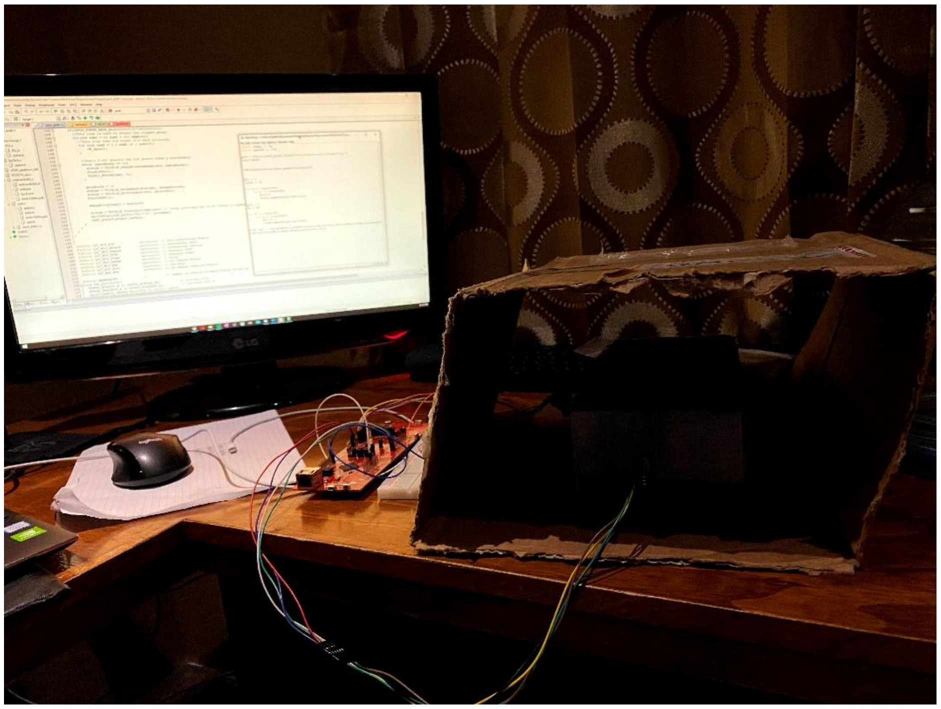
2DX4: Microprocessor Systems Project

Final Project

Instructor: Drs. Doyle, Haddara, and Shirani

Tanbir Badhan - L04 – badhant – 400264047

Setup:



Video Demonstration:

https://drive.google.com/file/d/1Rk2Iy2z9sfsAqIUrow40m-PGt-PkDNqM/view?usp=sharing

Question 1:

https://drive.google.com/file/d/1Uofj-BGRlkVBkwwVTuscKHpDrRo0GKcL/view?usp=sharing

Question 2:

https://drive.google.com/file/d/1aTKz0n-ZN1GIK_n4FvTj9bFaWgNNtrip/view?usp=sharing

Question 3:

https://drive.google.com/file/d/1421siDeaGkxY7bg9uuFtCzlJddn0Z2_P/view?usp=sharing

1. Device Overview:

1.1 Features:

The remote sensing device includes the following features:

- Simple Link MSP 432E401Y microcontroller ($65.53)
  - Bus Speed: 16MHz
  - USB 2.0 Micro A/B connector
  - 1 independent microcontroller reset switch
  - 4 user LEDs
  - Need: This is the motherboard of the device
- 1 user switch
  - Used to start and stop the measurements
- VL53L1X Time-of-Flight Distance Sensor Carrier ($11.95)
  - Communication through I2C
  - Operating Voltage: 3.3V
  - Used to take distance measurement
- MOT-28BYJ48 Stepper Motor ($8.03)
  - Stepper Motor controller
  - Operating Voltage: 5V
  - Used to rotate the time-of-flight sensor 360 degrees
- Open3D graphing
  - Used to visualize the data
- Communication to personal computer through UART and Python
  - Baud Rate: 115200
  - Used to store measurements on the PC

1.2 General Description

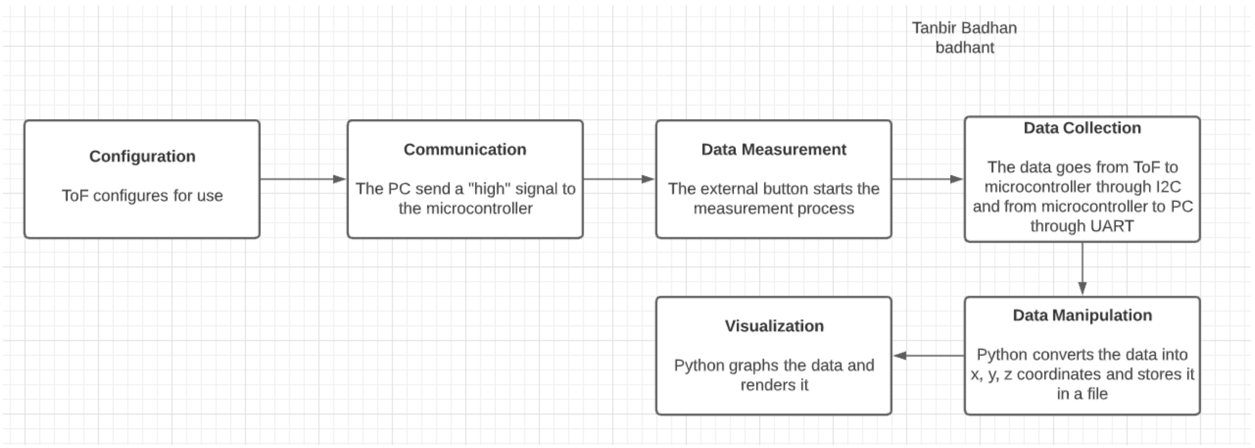
The MSP432E401Y is a 32-bit Arm Cortex-M4F based microcontroller which is used as the motherboard for this device. This microcontroller is responsible for multiple events. To begin with, it configures the time-of-flight sensor and uses the I2C protocol to communicate and collect the distance measurements. During this process the controller also displays the time-of-flight sensor status using the on-board LEDs. Furthermore, it is also responsible for the rotation of the stepper motor, When the push button is pressed, the controller rotates the stepper motor at a particular frequency, so the time-of-flight sensor is able to take a 360 degree measurement. Lastly, the controller communicates with the PC using serial communication via UART. The controller also has a provided reset button, which initiates a system reset of the controller when it is pressed and released.

One push switch is provided for input and controls when the MSP432E401Y microcontroller starts the measurement process. The switch is connected to pin PM0 and uses an input voltage of 3V. When this button is pressed, the motor will start to rotate and the time-of-flight sensor will start to take a measurement until a full 360 degree rotation is made.

MOT-28BYJ48 Stepper Motor is used to rotate time-of-flight sensor 360 degrees. It is connected to Port E and uses a 5V input.

VL53L1X Time-of-Flight is a distance measuring device which uses LIDAR technology to calculate how long a photon of light takes to bounce of an object and reflect back. This component is mounted to the stepper motor and takes a measurement ever 5.625 degrees.

1.3 Block Diagram


Tanbir Badhan
badhant
Configuration
ToF configures for use
Communication
The PC send a "high" signal to the microcontroller
Data Measurement
The external button starts the measurement process
Data Collection
The data goes from ToF to microcontroller through I2C and from microcontroller to PC through UART
Data Manipulation
Python converts the data into x, y, z coordinates and stores it in a file
Visualization
Python graphs the data and renders it

2. Device Characteristics Table

Table 1: Device Characteristic Table

Feature	Description
Button	Pins: PM0 Voltage: 3.3V, GND
Stepper Motor	Pins: IN1 = PE0, IN2 = PE1, IN3 = PE2, IN4 = PE3 Voltages: 5V, GND Clock Speed: 80kHz
Time-of-Flight sensor	Pins: SCL = PB2, SDA = PB3 Voltage:3.3V, GND Clock Speed: Same as Motor
UART	Pins: Serial communication through python Communication Speed: 115200 bps Python, serial, open3d
Microcontroller	Clock Speed: 80kHz Bus Speed: 16MHz

3. Detailed Description

3.1 Distance Measurement

To gather the distance measurements from the time-of-flight sensor, the microcontroller goes through multiple steps. To begin with, when the code is first loaded onto the microcontroller, the user must first have every component must be connected onto the microcontroller according to table 1 above. The reset button must then be pressed to configure the ToF sensor through the I2C protocol. This process can take some time, but the user can ensure the process was done when all the on-board LEDs flash.

After the configuration process is done, the python code of data processing should be ran. This will output a high to the microcontroller (this process is called half-duplex communication), this will make the microcontroller start to wait for the external button press to start taking measurements.

After the half-duplex communication between the PC and the microcontroller, the microcontroller is waiting on the external button press. When the button is pressed, the stepper motor will start to rotate for 5.625 degrees and take a measurement and then rotate another 5.625 degrees. This process repeats until a full rotation is made to get the whole surrounding.

This collection of data is then instantly transferred to the PC though serial communication via UART. The python code accepts the distance measurement from the microcontroller and uses it to the x, y, z direction though the following formulas.

$$y = distance*\cos(\theta)$$

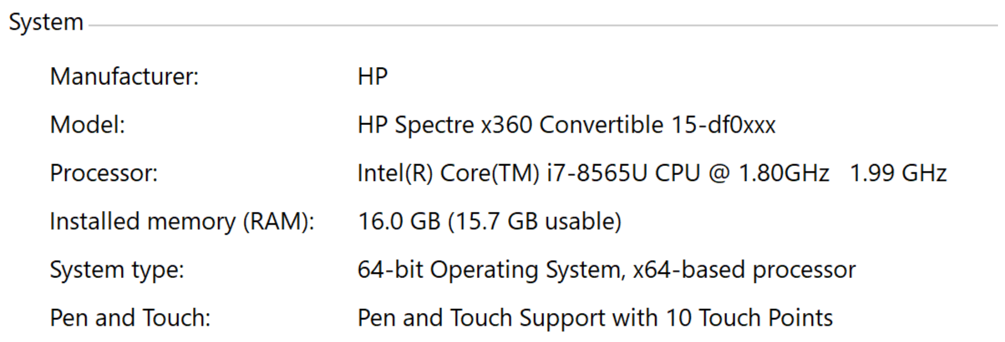
$$z = distance*\sin(\theta).$$

As the flowchart in figure () shows, when a measurement is taken, the program made in python instantly converts the distance measurement into x, y and z coordinates. These x, y and z coordinates are then added into a “measurements.xyz” file and then the program waits for the next measurement (i.e button press).

3.2 Visualization

A windows HP laptop was used to communicate with the microcontroller, it is using windows 10 running python 3.8. The computer specs are shown in the picture below.

Figure 1: Computer Specs



System

Manufacturer:	HP
Model:	HP Spectre x360 Convertible 15-df0xxx
Processor:	Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz
Installed memory (RAM):	16.0 GB (15.7 GB usable)
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	Pen and Touch Support with 10 Touch Points

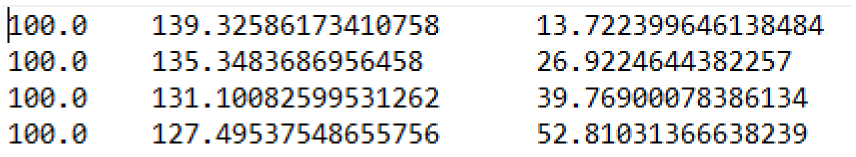
What functions/libraries did you use?

The function/libraries used include import serial, math, open3d and numpy, python file handling.

How is the data stored and processed?

The data goes into serial port COM6 for my specific computer and from the data processing program, the distance measurements are put into a .xyz file. This file contains the x, y and z measurements for each distance measurement taken, this file will include 64 points for each plane and every time the device is moved forward, the program will start to append the next plane into this file as well. Since we take 10 plane measurements, the file will include 640 different coordinates in total. A picture is included below:

Figure 2: XYZ file

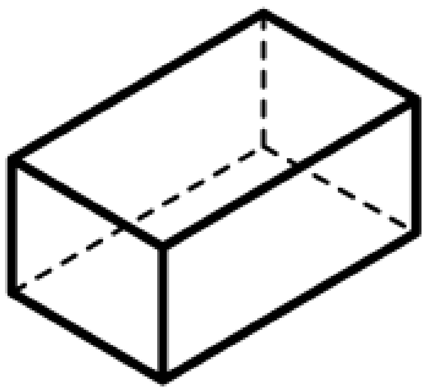


```
100.0    139.32586173410758      13.722399646138484
100.0    135.3483686956458       26.9224644382257
100.0    131.10082599531262      39.76900078386134
100.0    127.49537548655756      52.81031366638239
```

How does it visualize?

The Open3d library was used to visualize the data put in the .xyz file. A point cloud was first created with the data in the measurements file and was then printed on the screen. This plotted every single coordinate onto a graph which can be seen in figure 4 and 5. This was done by looking at the code and functions provided to use in lecture 9. This point cloud helps the user visualize the different axis but does not fully show the shape. To get a full render of the environment all the point in the planes were first connected using a nested for loop and then all of the planes were then connected to get a full shape which can be seen in figure 6 and 7. These nested for loops append to the lines array, which is coded in the program, all the connects of the lines are appended in this array and this is what helps us visualize the final image. Methods like read_point_cloud, append, line_set, and visualization.draw_geometries were all used from the open3d library to achieve this image.

Figure 3: Space Recreated



4. Application Example

Acquiring Signal and Mapping:

1. Run the "ReadingFile.py" code after the microcontroller is done configuring the sensor. When the serial port is opened, the program will say "Opening: COM#".
2. After the serial port is opened, press the external button and in the python console, the measurements will start to appear in this format (Distance   x-cord   y-cord   z-cord).
3. Once the sampling is done, move the device forward and press the external button to take the next set of measurements.
4. Repeat steps 2-3 for 10 samples.
5. The measurements should now be in the "measurements.xyz" file
6. Open and run "Open3d.py" file and it will visualize the data according.

Device Setup

1. Connect the circuit according to figure ().
2. Connect the microcontroller to the PC via USB
3. Mount the ToF sensor on top of the stepper motor.
4. Load the keil code onto the microcontroller.
5. Press the reset button on the controller and wait for the lights to stop flashing.
6. Run the python code provided ("ReadingFile.py").
7. Hit the external button and wait for the stepper motor to finish rotating.
8. Move the device forward and repeat step 7 for 10 samples.

Figure 4: Open3d Plot1

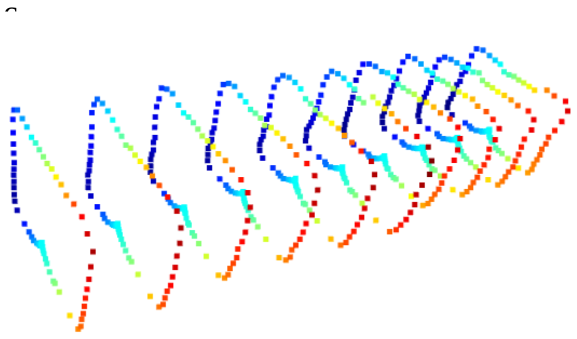


Figure 5: Open3d Plot 2

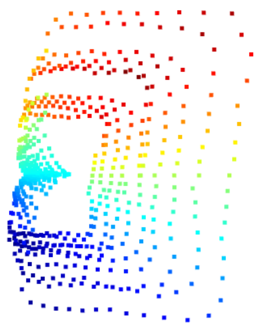


Figure 5: Open3d Render 1

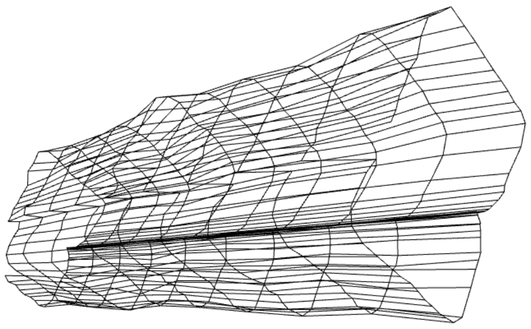
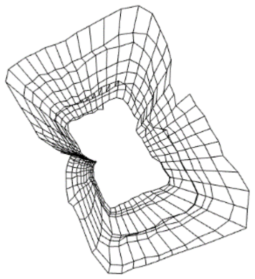
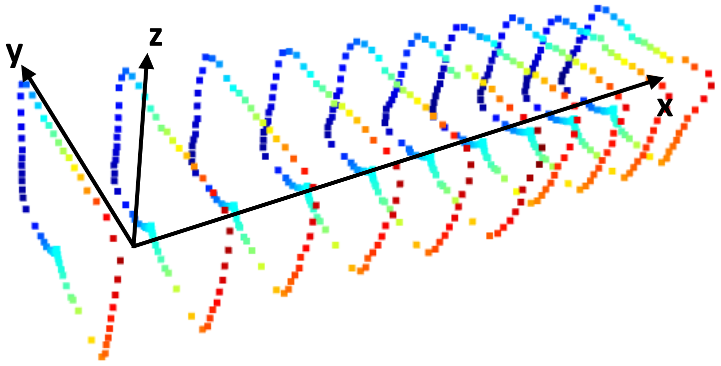


Figure 6: Open3d Render 2



Planes:

Figure 7: Planes



This figure above visibly describes all the axis planes. The x plane is the representation of the device moving forward and the y z planes are vertical slices which the sensor records.

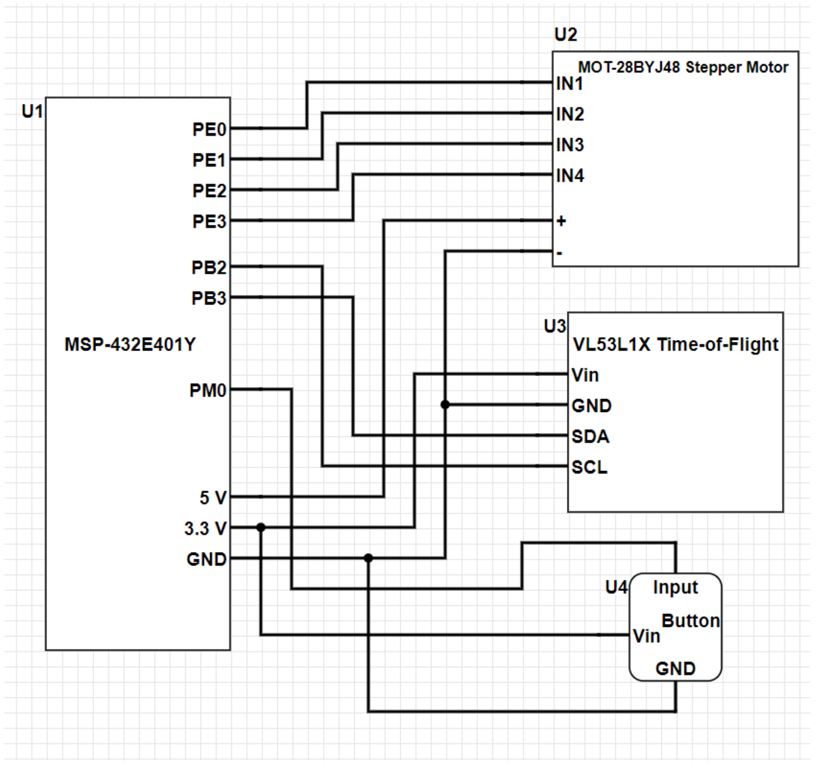
5. Limitations

1. A limitation of the microcontroller includes its rounding as the time-of-flight sensor rounds to the nearest whole number. This rounding causes the measurements to be less accurate since they are not exact. The microcontroller is seen to transfer a whole number for every distance measurement limiting the precision of the device. Another design limit is that the Time-of-flight sensors wired connections moves the sensor while taking the measurements which causes them to be less accurate, again hurting the precision of the design.
2. Quantization error = $V_{FS}/2^m$. Since the ToF sensor uses 8 bits to communicate, the maximum quantization error will be $3.3/2^8 = 0.0129$ V

3. The UART maximum baud rate was found though the data sheet and it is 5 Mbps, however the microcontroller is limited to 115200 bps listen in the data sheet. Therefore, the maximum baud rate which can be used for our design is 115200 bps.
4. The ToF sensor used I2C protocol to communicate with the microcontroller, for our case, the microcontroller acts as the master and the ToF sensor acts at the slave. Through the datasheet of the ToF sensor, it can be seen that it takes measurements at a speed of 100kbps
5. The primary limitation of speed would have to be the stepper motor since it operates very slow. When looking at the entire design, it can be seen that the time-of-flight sensor measurements are instant and not noticeable at all times wise. Communication is also fast and not noticeable. However, it can clearly be seen that the stepper motor takes a noticeable amount of time to do a full 360-degree rotation which hinders the speed at which this device functions. This was tested by altering the bus and clock speed. It was seen that when the bus and clock speeds are decreased, the stepper motor starts to move quicker and does not decrease the accuracy of the distance measurements. This clearly shows the affect that the stepper motor has on this design. With the bus and clock speed I am given for this project, it really hinders the speed of the design.

6. Circuit Schematic

Figure 8: Circuit schematic



7. Programming Logic Flowcharts

Figure 9: Microcontroller Flowchart

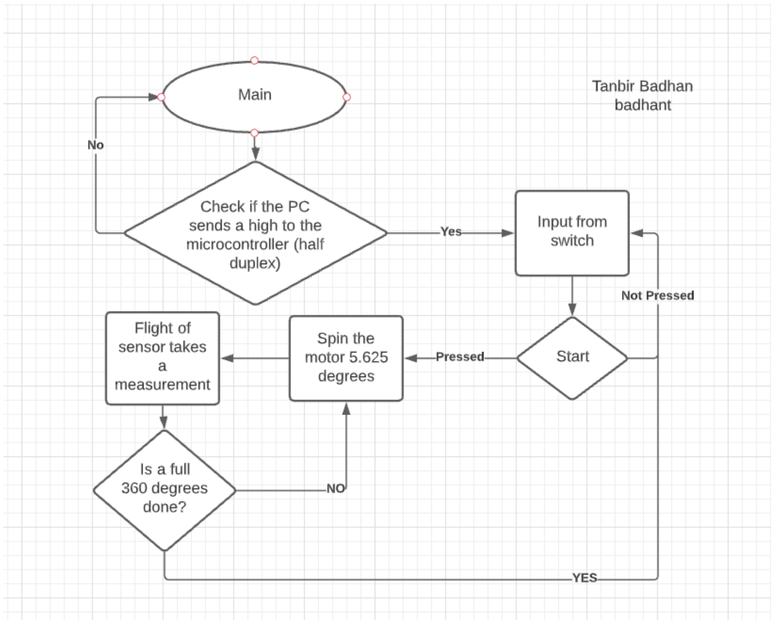
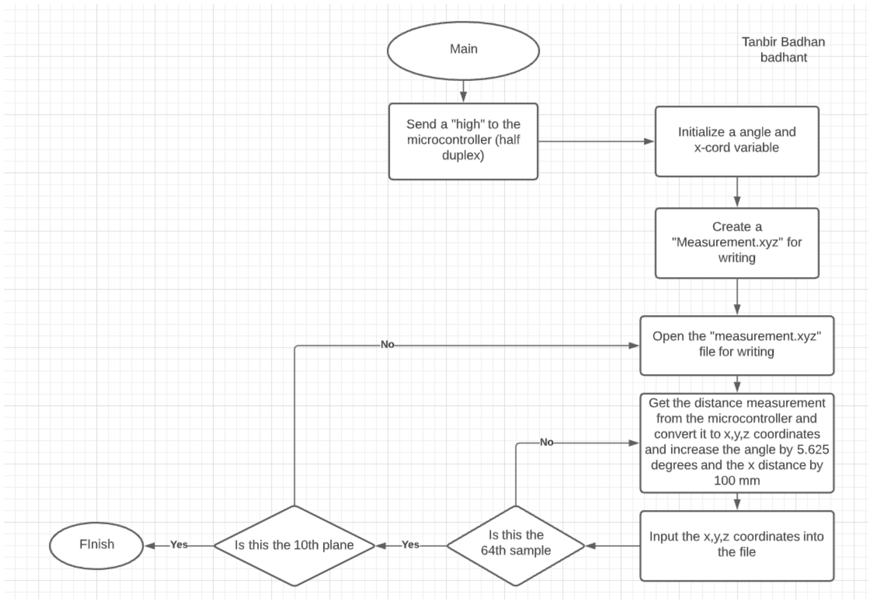

Main
Tanbir Badhan
badhant
No
Check if the PC sends a high to the microcontroller (half duplex)
Yes
Input from switch
Not Pressed
Flight of sensor takes a measurement
Spin the motor 5.625 degrees
Pressed
Start
Is a full 360 degrees done?
NO
YES

Figure 10: Python Flowchart


Main
Tanbir Badhan
badhant
Send a "high" to the microcontroller (half duplex)
Initialize a angle and x-cord variable
Create a "Measurement.xyz" for writing
Open the "measurement.xyz" file for writing
No
Get the distance measurement from the microcontroller and convert it to x,y,z coordinates and increase the angle by 5.625 degrees and the x distance by 100 mm
No
FInish
Yes
Is this the 10th plane
Yes
Is this the 64th sample
Input the x,y,z coordinates into the file