COMPENG 2DX3
Final Deliverable Report
LIDAR Lite Scanner

Course Instructor: Dr. Haddara/Athar/Doyle

Muhammad Javaid
April 17th, 2024
javaim7
400455837

Table of Contents

Device Overview

Features

Texas Instruments MSP-EXP432E401Y

- Acts as the main interface for transferring data from the TOF sensor to a computer.
- Equipped with a ARM Cortex-M4F 32-bit processor core.
- Operates at a bus Speed of 16 MHz.
- Has 1 MB of flash memory and 256 KB of RAM.
- Uses multiple communication protocols like CAN, USB, I2C, SPI, and UART
- Features GPIO pins and built-in LEDs to improve functionality and provide visual feedback.

VL53L1X Time of Flight Sensor

- Requires a voltage range of 2.6V to 3.5V.
- Capable of measuring distances up to 4 meters.
- Operates at a frequency of 50Hz, ensuring efficient data collection.
- Uses a 940 nm Class 1 invisible laser emitter and a SPAD (single photon avalanche diode) receiving array.
- full field-of-view is 27 degrees.

Communication Protocols

- Uses the I2C protocol for communication between the Time-of-Flight sensor and the microcontroller.
- Utilizes the UART protocol to transmit data from the microcontroller to the PC, operating at a baud rate of 115200 BPS.

28BYJ-48 Stepper Motor and ULN2003 Diver Board

- Operating voltage is 5V.
- Completes a 360-degree rotation in 512 steps, allowing more detailed data for scans.
- Achieves precise movement through a four-phase stepping mechanism.

Visualization Process

- Data measurements are received by the PC from the Microcontroller via UART protocol and stored in a file using a Python script. These measurements are then transformed into xyz coordinates and linked together to create a 3D plot. This is done using the Python Open3D library.

General Description

This device offers a cost-effective alternative to commercial LIDAR for 3D scanning, which is often expensive and bulky. It employs a Time of Flight Sensor integrated with a stepper motor that allows 360-degree distance measurements. The aim of this device is to capture spatial information from the surroundings at every 5.625 degrees and use it to generate a 3D scan.

To configure the device, press the reset button on the microcontroller and then run the python script on your external computer. To initiate the scanning process, press the onboard button (JP1) on the microcontroller. Once the process has started, the motor rotates at an angle of 5.625 deg 64 times to perform a 360 deg scan. At each interval, LED4 flashes to signify that the TOF sensor has taken a measurement. Once a revolution is complete, LED3 blinks once to indicate that the motor is returning to home position.

The TOF sensor determines distances by emitting a light beam from its emitter and timing how long it takes for the light to reflect to its receiver. Once it processes the information, the sensor utilizes the I2C protocol to transmit the distance measurement in mm to the Microcontroller. The Microcontroller then transfers it to a computer via UART at a Baud rate of 115200 Bps for visualization purposes. A Python script converts these distance measurements from polar to Cartesian coordinates and stores them in a file. Finally, these coordinates are linked together using Open3D to produce a 3d mapping of the surrounding area.

Block Diagram

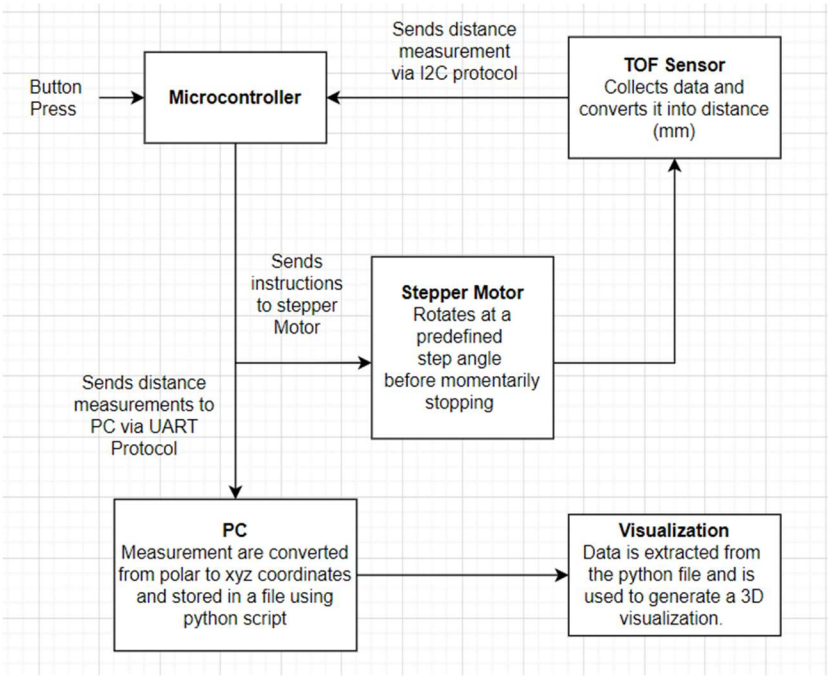

Fig 1: Block Diagram

Device Characteristics Table

MSP-EXP432E401Y		ULN2003 Driver Board		VL53L1X	
Feature	Specifications	Driver Board Pins	Microcontroller pins	Sensor Pins	Microcontroller Pins
Processor	Cortex M4F	V+	5 V	Vin	3.3 V
Clock Speed	16 MHz	V-	GND	GND	GND
Serial Port	COM8	IN1 – IN4	PH0 – PH3	SCL	PB2
LED Configuration	PF0, PF4			SDA	PB3
Baud Rate	115200 Bps				
Onboard Buttons	JP1				

Detailed Description

Distance Measurements

The VL53L1x Time-of-Flight sensor determines distances by emitting infrared light from its emitter, which reflects off nearby objects and returns to the receiver. The sensor calculates the distance by measuring the time delay between the emission and reception of the beam. The distance measurement is modelled after the following formula:

$$Distance = \frac{flight\ time * speed\ of\ light}{2}$$

The TOF sensor is connected to the microcontroller via four pins. The VIN and GND on the sensor connect to the 3.3 volts supply and ground on the microcontroller. The SDA and SCL pins on the sensor are connected to PB2 and PB3 on the microcontroller, respectively, establishing an I2C serial communication pathway. The VL53L1X API library, configures and operates the sensor. The SensorInit function configures the sensor settings, while the StartRanging method initiates the measurement process. The GetDistance function retrieves the calculated distance in mm, and the GetRangeStatus function determines if the data is valid. After capturing the distance, the sensor processes this information through transduction, signal conditioning, and Analog-to-Digital Conversion, which converts the analog measurement into a digital format. This digital data is then transmitted back to the microcontroller via the I2C serial communication protocol, ensuring accurate and efficient data handling.

Furthermore, the sensor is attached to a stepper motor using a 3D printed mount, allowing it to rotate clockwise. The IN1, IN2, IN3 and IN4 pins on the motor connect to the PH0, PH1, PH2, PH3 pins on the microcontroller. The Full Step technique is employed to operate the motor, which involves alternating the polarity of all four electromagnets while ensuring that two adjacent electromagnets are always energized. This method allows for a consistent and controlled rotation of the motor.

To set up the device, the user must press the reset button on the microcontroller. All the onboard LED's will flash to signify that the device is ready to go. After the LED flash, the user must execute the Python script on a computer connected to the microcontroller via a USB-cable. Then the user must press the onboard button (JP1) on the microcontroller to start the scanning procedure. The stepper motor is programmed to rotate at a step angle of 5.625 deg 64 times to perform a 360 deg scan. Furthermore, the algorithm incorporates a brief delay between each step angle rotation to ensure the sensor has enough time to capture data. After completing a revolution, LED3 flashes once to signal that the motor is returning to its starting position.

The step angle can be determined using the following formula:

$$Step\ angle = \frac{360\ deg}{\#\ of\ Measurments}$$

$$Step\ angle = \frac{360\ deg}{64}$$

$$Step\ angle = 5.625\ deg$$

The microcontroller waits for the python script to send an acknowledgement that is tracked using the polling method. Once the micro receives the distance measurements from the TOF sensor via I2C protocol, it transfers the data in bytes to the PC. Since the PC receives the data in bytes, the python script decodes the data into string format for concatenation. After concatenation, the data is converted into an integer format to create a 3d plot.

Data Visualization

The Python Script connects the Microcontroller to the COM8 Port on the computer, allowing serial communication at a Baud rate of 115200 Hz. Once the data is received, it is decoded and converted from polar to xyz coordinates. The following Trigonometric formulas are used to convert the X and Y values:

$$\mathrm{X} = \frac{\text{distance} * \sin(\text{step angle})}{10}$$

$$\mathrm{Y} = \frac{\text{distance} * \cos(\text{step angle})}{10}$$

The Z coordinate specifies the depth for each cycle of measurement and can adjusted depending on the user's needs.

After all the data points are stored in an array, they are linked together to form layers. Each of these layers are then connected to one another using Open3D to form a 3-Dimensional model. The coordinates are stored in a text file and the 3D scan is stored as an xyz file which can be accessed using another python script.

Application Example

Application Note

This device is engineered to create accurate 3D representations of surroundings. It employs advanced lidar technology, which uses infrared light to determine distances, making it highly effective at detecting objects. Such detection capabilities are vital for autonomous vehicles as they navigate intricate environments by swiftly identifying and manoeuvring around obstacles. At the heart of this technology is the integration of sophisticated components, including the Texas Instruments MSP432E401Y microcontroller and the VL53L1X Time-of-Flight sensor. The system's precision and flexibility are further enhanced by the addition of a 28BYJ-48 stepper motor paired with a ULN2003 driver board. This combination not only improves the spatial accuracy of the measurements but also allows the device to adapt to a variety of environmental conditions, thereby broadening its application scope in robotics and automated systems.

Instructions

Before using the LIDAR system, some software and Python libraries need to be configured.

1. Download and install Python 3.8 on your device. Do not use Python 3.10 or later releases as they are not compatible with the Open3D library.
2. Launch the command prompt as an administrator to install the required Python libraries, including Open3D and pySerial. Enter “pip install pyserial” and await a confirmation message verifying that all packages have been successfully installed. Similarly, for installing the Open3D library, type “pip install open3d”.
3. Install Keil uVision on your device.

Once all the required installations are complete, you can start configuring the device.

1. Go to the main.c file and adjust the NUM_MEASUREMENTS and NUM_SAMPLES variable. NUM_MEASUREMENTS variable stores the amount of 360 deg scans and the NUM_SAMPLES variable stores the number of times the sensor will capture a reading during one revolution.
2. Go to the Python Code to change the variables as well. They can be found at the start of the code. Make sure to match the variable values in both the Keil and Python Script.
3. Connect the microcontroller to your computer and use the Keil software to upload the code. In Keil, click on 'Translate', then 'Build', followed by ‘Load’. After the code is loaded, press the reset button located next to the USB port on the microcontroller.
4. Open the python script. You can open it with an IDE of your choice.
5. Set the COM port in the code to match the one your computer uses. To find out what Port your computer uses, paste the following in command prompt ‘python -m serial.tools.list_ports’.
6. You can adjust the depth by changing the variable ‘layer’ on line 37 before it appends the value to the array.
7. Run the Python Script.
8. Press the onboard button (JP1) to begin the scanning process.
9. Once the scanning process ends, a 3D scan will pop up on your screen.

Expected Output

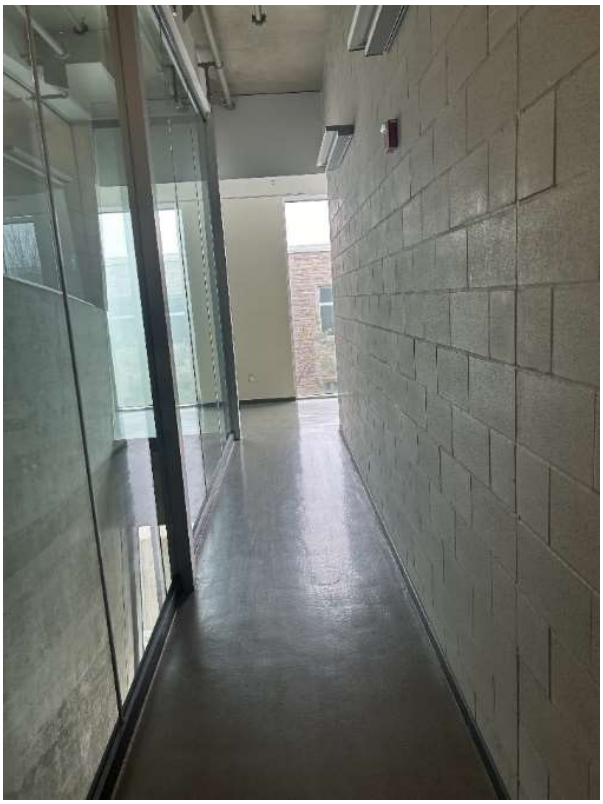

Fig 2: Starting Position POV

Fig 3: End of Hallway Side View

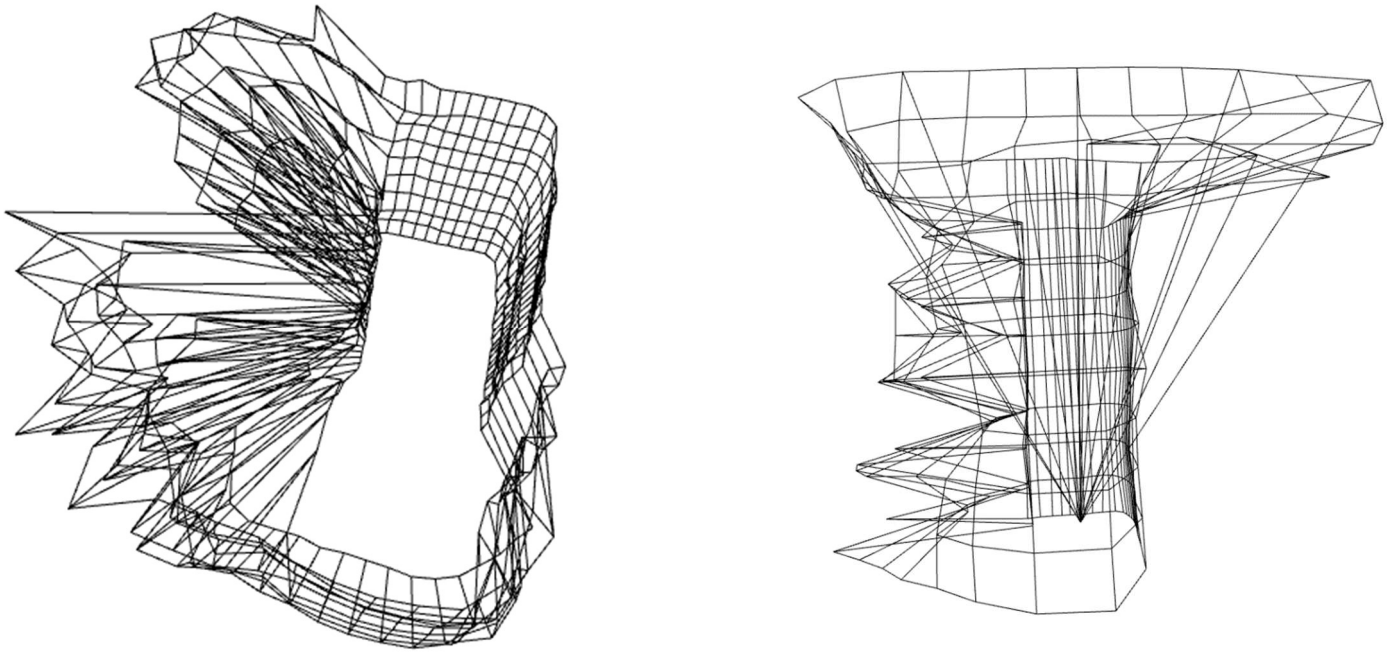

Fig 4: 3D scan of Location H

Limitations

1. The Floating-Point Arithmetic Unit in the microcontroller has restricted processing capabilities, which can lead to errors and slow performance when handling polar coordinates directly. To overcome this, the data is processed using Trigonometric functions in the Python script. Python offers more robust computational resources, enabling it to perform these calculations more quickly and accurately.
2. The Maximum Quantization Error for the TOF sensor is $0.9766\ mm$.

$$Maximum\ Quantization\ Error = \frac{Range}{2^{bits}}$$

$$Maximum\ Quantization\ Error = \frac{4000\ mm}{2^{12}}$$

$$Maximum\ Quantization\ Error = 0.9766\ mm$$

3. For this device, the UART is set at a baud rate of 115,200 bps because it is the fastest speed that is compatible with most devices. Although the PC is capable of operating at a higher baud rate of 128,000 bps, the maximum limit is set to 115,200 bps to ensure that both the PC and the sensor device are synchronized at the same baud rate. To verify the highest Baud rate a PC can achieve, look at port properties in Device Manager.
4. The microcontroller communicates with the Time-of-Flight sensor using the I2C protocol at a speed of 100kHz. However, the TOF sensor operates with a capture rate of 50Hz, reducing the overall speed of the device.
5. The major limitations on this system's speed are the time required for the stepper motor to complete all measurements and the initialization time for the TOF sensor to begin ranging. Although we cannot modify the sensor, we can adjust the motor's speed to achieve quicker outcomes. This issue is crucial because, despite the TOF module's capability to take rapid measurements, the motor must pause between rotation steps to guarantee that the sensor accurately receives and transmits measurement data. To optimize the motor's speed, the delay can be reduced to 2 milliseconds. Decreasing the delay beyond this threshold would compromise the motor's operational functionality. This was verified through a series of tests where the delay was incrementally shortened until the motor stopped rotating.
6. The Assigned Bus Speed is 16MHz. To obtain this speed, use the following formula to find PSYDIV.

$$Clock\ Speed = \frac{480MHz}{PSYDIV + 1}$$

$$PSYDIV = \frac{480MHz}{16MHz} - 1$$

$$PSYDIV = 30 - 1$$
$$PSYDIV = 29$$

7. The Maximum range of the TOF sensor is 4 metres.
8. TOF can only communicate via I2C protocol.
9. The Open3D library that was used to generate the 3D scan is not compatible with Python 3.10 and newer versions.

Circuit Schematic

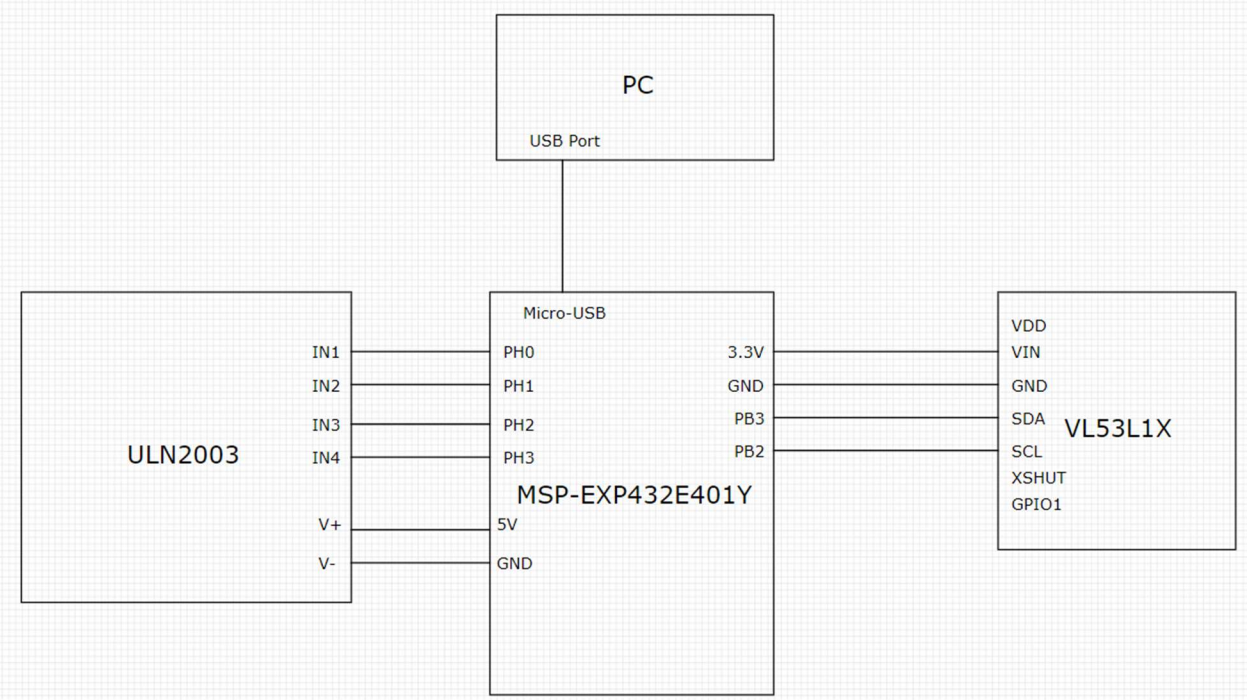

Fig 5: Circuit Schematic

Programming Logic Flowchart

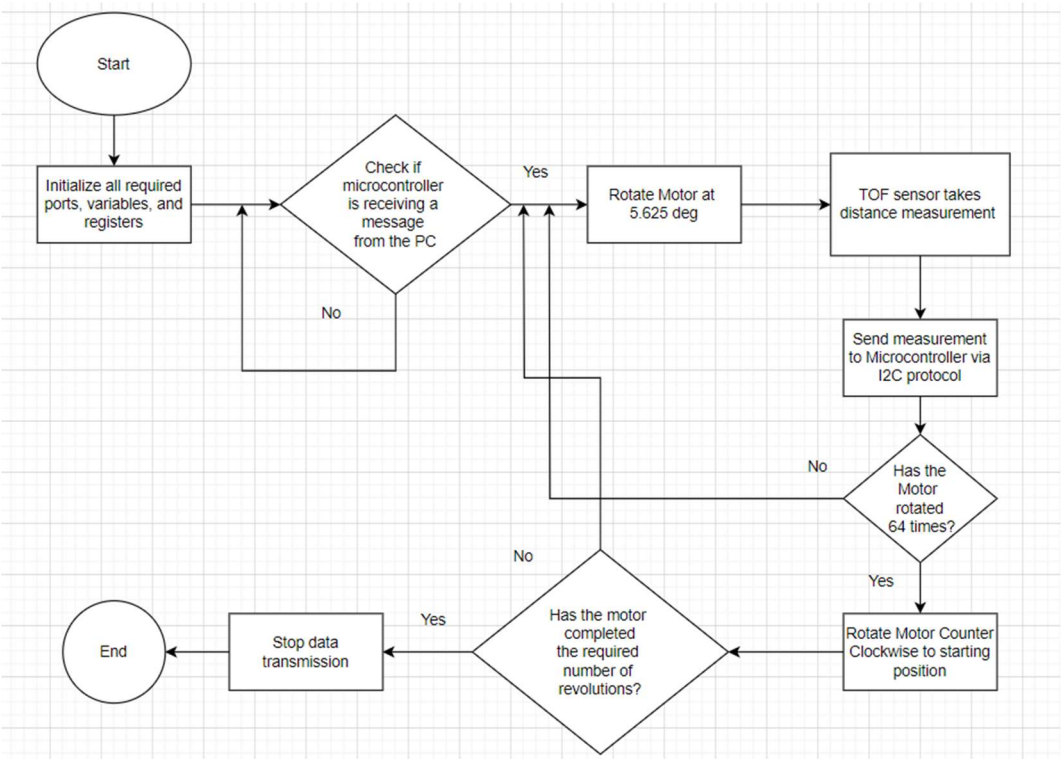

Fig 6: Keil Code Flowchart

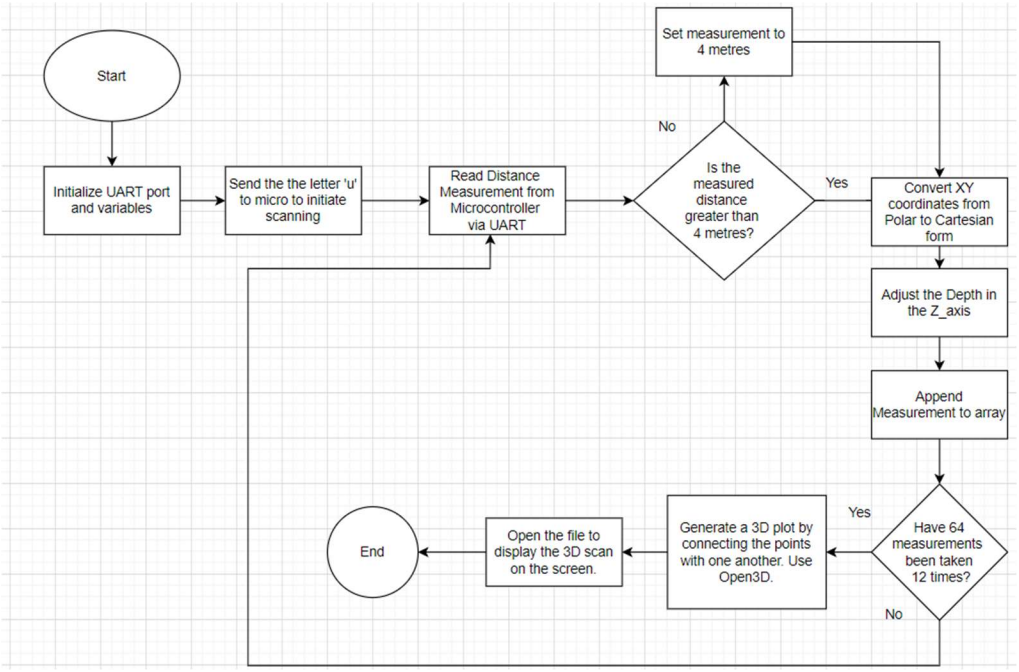

Fig 7: Python Code Flowchart