Microprocessor System Data Acquisition
User Guide

1. Device Overview

1.1 Features

- Includes 2 components:
  - Texas Instrument MSP-EXP432E401Y microcontroller
  - VL53L1X long distance ranging Time-of-Flight (ToF) sensor
- Device operating voltage for all components: 3V3
- MSP-EXP432E401Y
  - C programming language with Keil IDE
  - Core processor is the Arm Cortex-M4F, a 32-bit architecture optimized for small-footprint embedded applications
  - 13 general purpose registers and 3 special registers each 32-bits
  - Bus clock frequency: 30 MHz
  - Programmable bus clock frequency allowing to change stepper motor angular speed
- 2 LEDs for tracking motor rotation (external LED) and data acquisition (integrated with microcontroller)
- 1 user push button for data acquisition initiation
- Serial communication configuration:
  - Python programming language
  - 115.2k baud rate
  - 8 data bits and 1 stop bit

1.2 General Description

The integration of the MSP432E401Y microcontroller, VL53L1X sensor, and serial communication to the PC provides a means to capture distance measurements in a single plane, send the acquired data through serial communication to a target device and recreate the data through 3D visualization.

The MSP432E401Y component is the center of the device which is responsible for important tasks such as initialization of components and configuration of ports and external devices such as the ToF sensor, stepper motor and establishing a connection between the controller and the PC for data transmission.

The VL53L1X sensor is connected to the MSP432E401Y via its I^2C bus. The sensor ranges data continuously with a clear interrupt required for new data to be processed. Otherwise, the device ranges continuously until the ranging sequence finishes. The sensor captures distance measurements for 360 degrees of rotation (y-z plane) with the aid of the stepper motor. Displacement measurements (x-axis) is done manually by repeating the distance measurement process until a defined fixed displacement is reached. All data is stored on the microcontroller.

Data is then transferred from the microcontroller to the PC through half-duplex communication with the aid of python programming language. The microcontrollers UART sends the data at a specific baud rate through a port on the PC while the python program configures that port for receiving data. The dataset is formatted for 3D visualization.

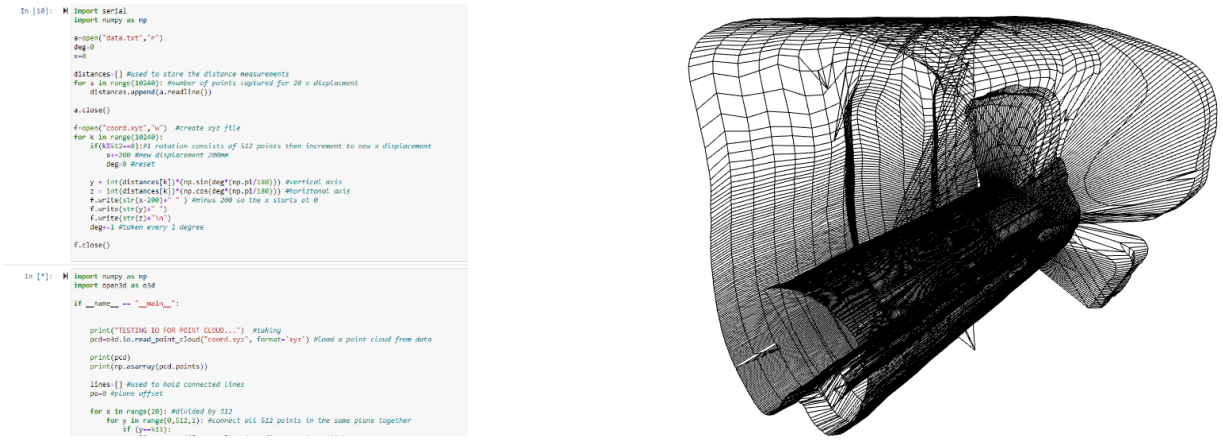
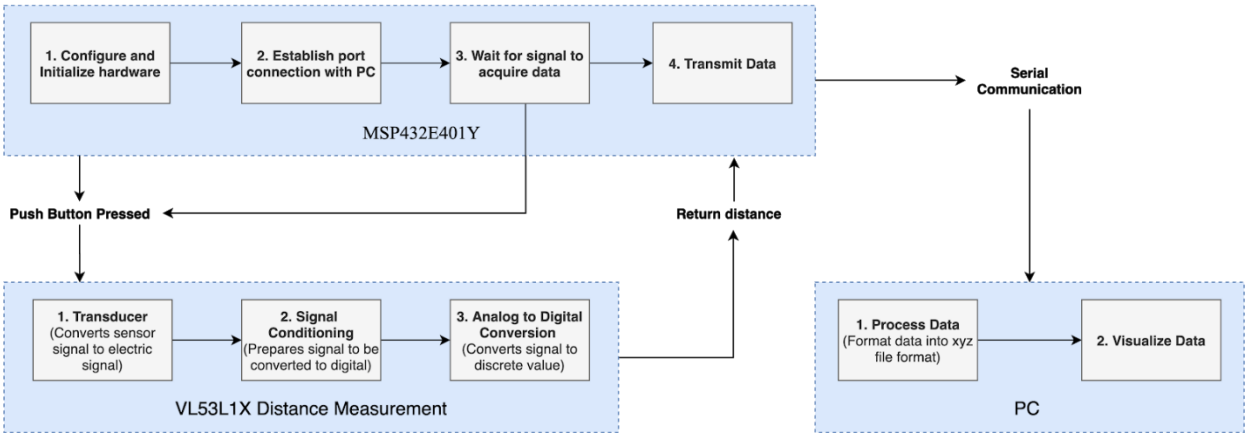


Figure 1. PC display of captured data using python (left) and the visualized data (right).

1.3 Block Diagram



**Hardware issues were encountered with the sensor mounting on the motor due to the stiff wires. The use of the slip ring would solve this issue as the wires would connect to the ring instead of the breadboard directly allowing for sensor to rotate properly with stepper motor*

2. Device Characteristics

- Bus Speed: 30 MHz
- Serial Port: COM3
- Communication Speed: 115.2 kbps with 8N1 configuration

Device	Port	Pins	Application
MSP432E401Y	P	0:3, 5	Stepper Motor, LED
	B	2:3	I2C with VL531X
	M	0	Push Button
	N	0	LED

3. Device Description

3.1 Distance Measurement

The distance measurement module captures 512 points in the y-z plane on a 360-degree scan. The MSP432E401Y contains the main program that governs the behaviour of the device and the flowchart of the program can be found in section 7 – Programming Logic Flowcharts figure 3. The microcontroller is continuously waiting for user input and is where all data is processed and transmitted. To initiate distance measuring, the push button is pressed. The displacement LED will turn off while the distance LED will turn on and flash every time a distance measurement is made.

The first step in distance measurement is the acquisition of the distance from the sensor. The VL53L1X sensor communicates with the MSP432E401Y via I2C connection which can be seen in section 6 – Circuit Schematic figure 2. The I2C interface uses 2 signals, the serial data line (SDA) and the serial clock line (SCL). Each device connected to the bus uses a unique address and a simple master/slave relationship. Clock signal (SCL) generation is performed by the master device (MSP432E401Y) and it initiates data transfer. The sensor uses a device address of 0x52.

The MSP432E401Y will initiate the sensor to measure distance. In the sensor, the laser is beamed out and is received back at the lens. ADC is performed and the information is packed into bytes followed by acknowledge bit, Ac for VL53L1X acknowledge and Am for master acknowledge. A message contains a series of bytes preceded by a start condition and followed by either a stop or repeated start. The first byte of contains the device address (0x52) and specifies the data direction. If the LSB is low (0x52) the message is a master-write-to-the-slave. If it is high (0x53) the message is master-read-from-the-slave. As data are received by the slave, they are written bit-by-bit to a serial/parallel register. After each data byte has been received by the slave an acknowledge is generated and the data are then stored in the internal register. When the data on distance is desired, the read message (0x53) is employed and the contents of the register holding the data are parallel loaded into the serial/parallel register and clocked out of the sensor by the falling edge of SCL returning the distance in millimeters.

This stage corresponds to the third last box in the flowchart seen in figure 3 between the diamonds. The distance measurement is stored on the microcontroller and is now ready to be outputted. Transmission of data to the PC from the MSP432E401Y is done serially through a port called COM3 on the PC. *Ports will vary with different computers*. Establishing a connection with the port and synchronizing the baud rate of 115.2k is necessary before transmission. The UART of the MSP432E401Y will send the distance measurements to the PC after each measurement capture. The measurements are stored in an array upon arrival to the PC.

After data transfer to the PC occurs, coordinates need to be obtained. The following formulas are trigonometric equations used to convert the distance into y-z coordinates.

$$\boldsymbol{y = distance * \sin(degree)}$$

$$\boldsymbol{z = distance * \cos(degree)}$$

Example. 1

The distance measured is 1.6m at 45 degrees from the initial starting position and no displacement has occurred in the x-axis. Then y = 1.6*sin (45) and z = 1.6m*cos (45). The corresponding xyz coordinates are approximately (0,1.13,1.13) m.

Once y-z coordinates are obtained the corresponding x coordinate will be generated according to the user's displacement distance. These coordinates are then written to a file in xyz format for 3D visualization.

3.2 Displacement Measurement

The IMU will communicate with the MSP432E401Y via I^2C serial interface like the ToF sensor. To measure the x-axis, the IMU's gyroscope feature will be used to obtain the xyz coordinates. However, the only coordinate of interest is the x coordinate. When the motor rotates the gyroscope will begin measuring the 3-axis around it and will employ UART functions to transmit the data from the microcontroller to the PC.

Displacement was implemented through manual movement of the device, including the PC to a fixed distance. After a distance measurement in the y-z plane was obtained, the device was moved to the next location and the distance process was repeated until enough samples were captured.

3.3 Visualization

The following computer was used to visualize data:

OS Name: Microsoft Windows 10 Home
System Model: HP Pavilion x360 Convertible 14
Processor: Intel® Core™ i5-8250U CPU
Python Version: 3.6.8
Serial Port: COM3
***Anaconda Jupyter** was used to develop Python code.

To visualize the xyz coordinates, the **Open3D** library was imported as it provides tools that deal with 3D data. **NumPy** was also imported as it provides trigonometric identities that will be used for distance conversion into coordinates. Configuring the serial port COM3 is required as it is where the MSP432E401Y sends the distance measurements.

Recall from section 3.1 Distance Measurement, the distance is sent to the PC right after it is obtained. Referring to figure 4 in *Programming Logic Flowcharts* section, the distances are stored in an array. This process is repeated 512 times for distance measurement and is repeated until the desired displacement distance has been reached.

Example. 2

The desired displacement distance is 8cm (x-direction). The desired x displacement is 1cm. For every displacement the device is moved 1cm until 8cm has been reached. Since the displacement

was done 8 times for every meter and each distance measurement is equal to 8 points, the array will contain 64 distance measurements.

Next, conversion from distance to coordinates is required before writing to an xyz file. The following explanation will refer to Example 2 as reference. Conversion can be done using the equations found in 3.1 employing NumPy math functions to achieve successful conversion. Since displacement is done manually, for every 8 elements in the array, the x coordinate is incremented by 1 to indicate different y-z plane slice. After a single xyz coordinate has been generated, it is then written to a file in xyz format. This process is repeated for every distance measurement in the array.

This stage in the visualization process corresponds to the last 3 boxes in the flowchart in figure 4. Using the xyz file, a point cloud is created using the Open3D library. These points are now in an empty space at the coordinates specified in the file. The next step is to connect the points in a single plane together using the *lines.append* command. After, each point is then connected to the next plane creating a 3D connection. All these lines are stores in the same array. Then, the command Open3D *LineSet* is used to define a set of line in 3D with the coordinate points generated in the point cloud. Lastly, this line set is visualized using the *visualization.draw_geometries command.*

4. Application Example

This section will walk the user through an example of acquiring a signal and visualizing the data through python.

Setup and Hardware:

- Python 3.6.8 or a version that is supported by Open3D
- Keil MDK
- MSP432E401Y
- VL53L1X
- 1 Push button
- 1 10kΩ resistor
- 1 1kΩ resistor
- 1 LED

Ensuring all the above items are achieved, refer to figure 2 for connection of the components for this device.

Testing and Establishing Connections:

1. Open the code provided with this device in Keil then translate, build, and load the code to the microcontroller. Ensure the target debug settings are correctly configured.

2. Running the code on the microcontroller results in the external LED to turn on. Confirm the LED turns on. If not, check wiring again.

3. Test the performance of the distance and displacement LEDs and motor by pressing the push button. Several actions will be seen:
   i. The motor should be seen rotating in the clockwise
   ii. The displacement LED (external LED) is turned off
   iii. The distance LED on the microcontroller or D1 is turned on

   Make sure the sensor lens is facing the floor as it will capture the y-z plane a vertical slice in space while the x displacement is captured through manual movement of device through the indoor space. In other words, this device will scan the walls, the floor and the ceiling in the room.

4. Now all the hardware has been tested except for the sensor and data transmission to the PC. To establish data transmission, the Python package PySerial must be installed. This will be used to list all the available serial port(s). By using the command in the CMD

   python -m serial.tools.list_ports -v

   a list of available ports can be seen. These are virtual com ports established over USB after the microcontroller is plugged and enabled. The port of interest is COM3.

5. Next, restart Python, import PySerial library, and open COM3 port referring to the following commands:

   import serial
   s = serial.Serial('COM3', 115200)
   print(s.name)
   Output - COM3

   The 115200 number refers to the communication speed or baud rate of the microcontroller. The default settings of 8 data bits and 1 stop bit will not be changed or 8N1. Confirming the output is the name of port, data transmission has now been established between the microcontroller and the PC.

6. Lastly, the sensor will be tested if it is capturing measurements. Open the python program called **Capture** provided with this document and run it. In the beginning the serial library is imported and COM3 was set as the serial port at 115.2kbs 8N1. It is important to configure the port in the program to ensure data is received. Press the push button and wait for the motor to rotate 360 degrees. Once finished, a file formatted as an xyz file should appear and contain the xyz coordinates. Confirm the file contains coordinates. This process should be repeated several times to capture a decent amount of distances throughout the room. Make sure to record the x displacement as the code may need to be changed depending on the displacement you choose. Run the python program **Visualize** and you should see a 3D model of the room captured by the device.

After all steps have been followed, the device is ready to use. To recap, run the python program **Capture** and press the push button to initiate data acquisition, making sure the sensor is facing the floor. This will capture the vertical slice data, the y-z plane. Manually, move the device a fixed amount in the room and repeat the process taking note the fixed amount moved. Once satisfied with the number of samples taken, run the python program **Visualize** to see the generated 3D model.

*IMPORTANT

Depending on how many displacement measures taken in the x-direction, numbers in the code may need to be changed to visualize data. For example, if 4 displacement measurements are taken, then 2048 points are taken, and the code should be modified accordingly.

5. Limitations

1) Limitation with the microcontroller floating point capability is it can only provide 2^{32} addresses for a system. The microcontroller can provide accurate ADC results with a quantization error of 0.0008057. The core also contains a IEEE 754-compliant single precision floating point unit.

2) **ToF maximum quantization error:**

V_{MAX} = 3.3V, V_{MIN} = 0V

Resolution = VFS / 2^m, where m is the number of bits used in ADC

VFS = Vmax – Vmin = 3.3 – 0 = 3.3V

Quantization error = $3.3/2^{16}$ = 0.000050354

Using the same equations from above the **IMU quantization error = 0.000054932**

3) The maximum standard serial communication that can be implemented with the PC is 122.5 kbps. This was verified by changing the baud rate at the serial port COM3. This was the max value in which data was transmitted as numbers and not random symbols.

4) The communication method between the microcontroller and the ToF sensor modules is through the I^2C interface and UART. It can either be simplex communication or half-duplex communication. The speed used between the microcontroller I^2C interface and the ToF modules is 400kHz.

5) The primary limitation of speed with the system is the MSP432E401Y bus speed. The bus speed is currently set to 30MHz. Changing this to a higher frequency will increase the speed of all devices and operations while lowering this frequency will slow down the devices. This was tested by changing the bus speed while keeping the SysTick wait timer the same. Changing the bus speed to a lower frequency would increase the time of NOP function in the microcontroller causing all functions using the SysTick command to increase the wait time and decrease speed of the system and vice versa.

6) For the IMU, the fast mode communication with the I^2C is 400kHz and the IMU sampling rate is 8kHz. The MSP432E401Y bus speed is set to 30MHz well above the speed of the IMU allowing for accurate sampling. According to the Nyquist Sampling Rate, the frequency of the sampler must be at least twice of the input signal. In this case, the necessary **sampling rate required is 16kHz**. If the input signal exceeds this frequency, then aliasing will occur, and the sampled data will be inaccurate.

6. Circuit Schematic

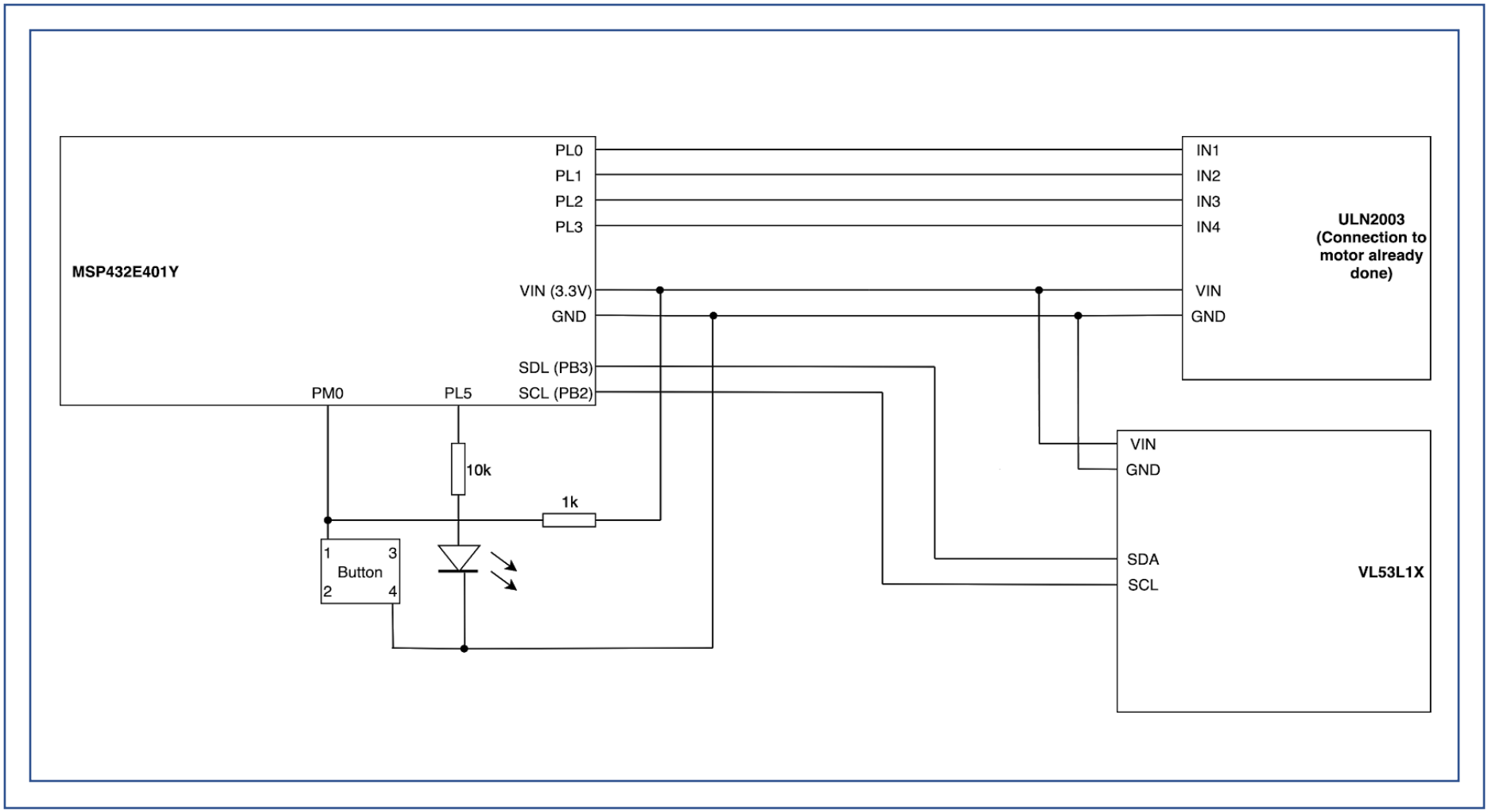


Figure 2 - Circuit schematic and wiring of device

7. Programming Logic Flowcharts

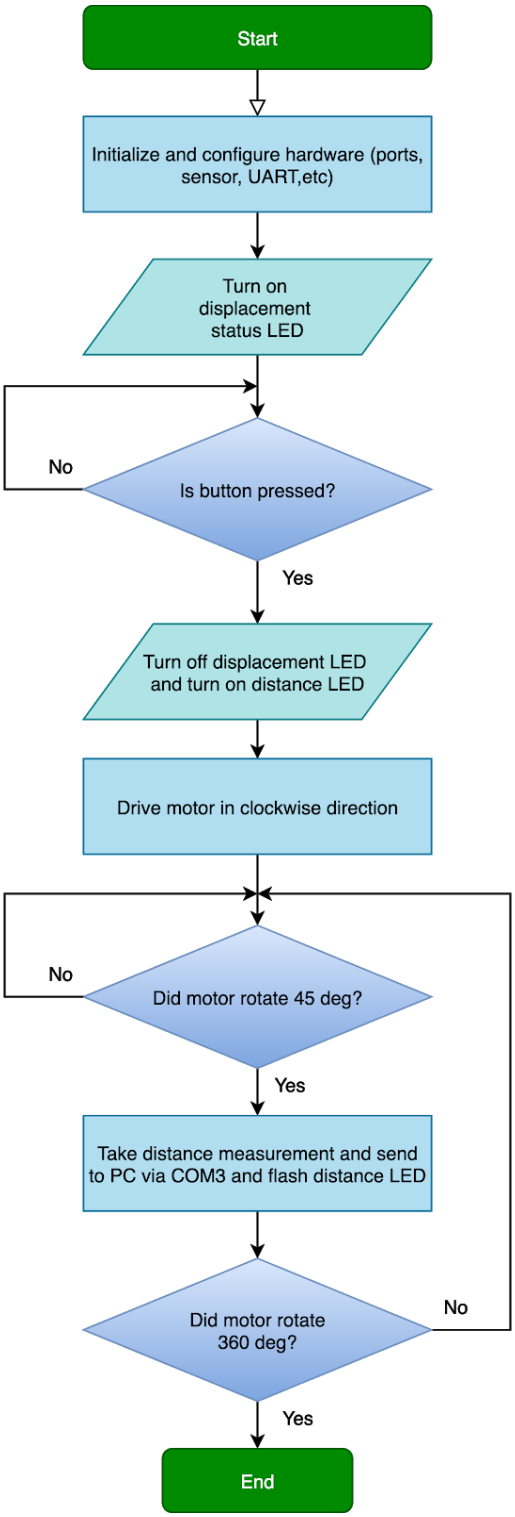


Figure 3 - Microcontroller distance measurement program flowchart

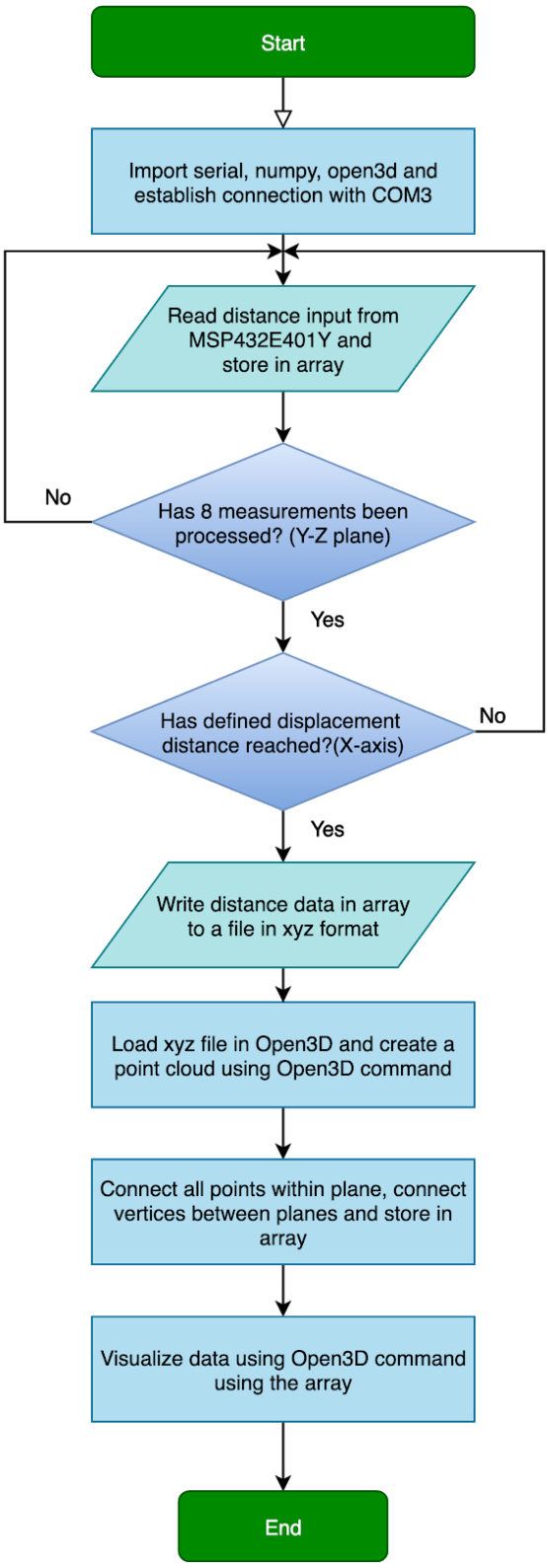


Figure 4 - Python visualization from transmitted data from microcontroller