2DX3: Microprocessor Systems
Final Project

Instructors: Drs. Boursalie, Doyle, and Haddara

Jashanjyot Randhawa – randhj13 – 400337963 – 2DX3 – Monday afternoon – L01

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres tothe Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by **[Jashanjyot Randhawa, randhj13, 400337963]**

1. Device overview

a. Features

The system used to create this project was made by the use of the TI MSP432E401 microcontroller and the VL53L1X time-of-flight (ToF) sensor. The TI MSP432E401 has the following features:

- 120-MHz Arm Cortex-M4F Processor Core
- 120 MHz bus speed
- 8 UART's, each with baud rates up to 7.5 Mbps (regular speed) and 15 Mbps (high speed)
- 10 I2C's Modules with high-speed support
- 1024KB of Flash Memory
- 256 KB of SRAM
- 6KB EEPROM
- Two 12-bit SAR-based ADC modules, each supporting up to 2 Msps
- Temperature range of -40 deg to 105 deg (Celsius)
- Operating voltage range of 2.5 – 5.5V
- Uses the English language
- Is $73.40 Canadian

The VL53L1X has the following features:

- An emitter with a 940 nm invisible laser
- Up to 400cm distance measurements
- Up to 50 Hz ranging frequency
- Typical full field-of-view (Fov) is 27 degrees
- A single power supply (2v8
- I2C interface (up to 400kHz)
- Operating voltage of 2.6 – 3.5V
- Is $27.69 Candain

b. General description

The device used in this project is an embedded system which uses a microcontroller, stepper motor and time-of-flight sensor to map and graph an indoor environment, for example a hallway. The systems core components include:

- Digital I/O: On-board push button
- Digital I/O: LED (distance measurement status)
- Transducer: ToF sensor which measures the distance in the y-z plane
- Data processing: collection of coordinates, computation and then storage of distance and displacement
- Data processing: manual collection of new data, 360 degrees with 32 samples at each 11.25 degrees. This happens once the defined fixed displacement is reached
- Implementation: Interrupt method
- Control: Stepper motor. Rotation of the stepper motor is controlled to control data collection with the ToF sensor
- Communication: ToF sensor and microcontroller communicate data between each other
- Communication: Data collected communicated with the PC applications such as python
- Control: Data read from .xyz file and visualized using Open3D

There are many steps of the systems operation. The system operates in interrupt mode using a button press to signal the start of operations. The stepper motor rotates a total of 360 degrees while taking measurements at every 11.25 degrees. The ToF sensor acquires distance data in the y-z plane via transducer. The signal is then processed and transformed from an analog to a digital signal. The data is then communicated to the PC using serial communication. Still operating in interrupt mode, the sensor sends data via UART, one byte at a time. The PC then receives this data by the use of a python script which the values are then stored in a .xyz file. That same .xyz file is opened in a different python script where it is visualised with the use of Open3D.

c. Block diagram (data flow graph)

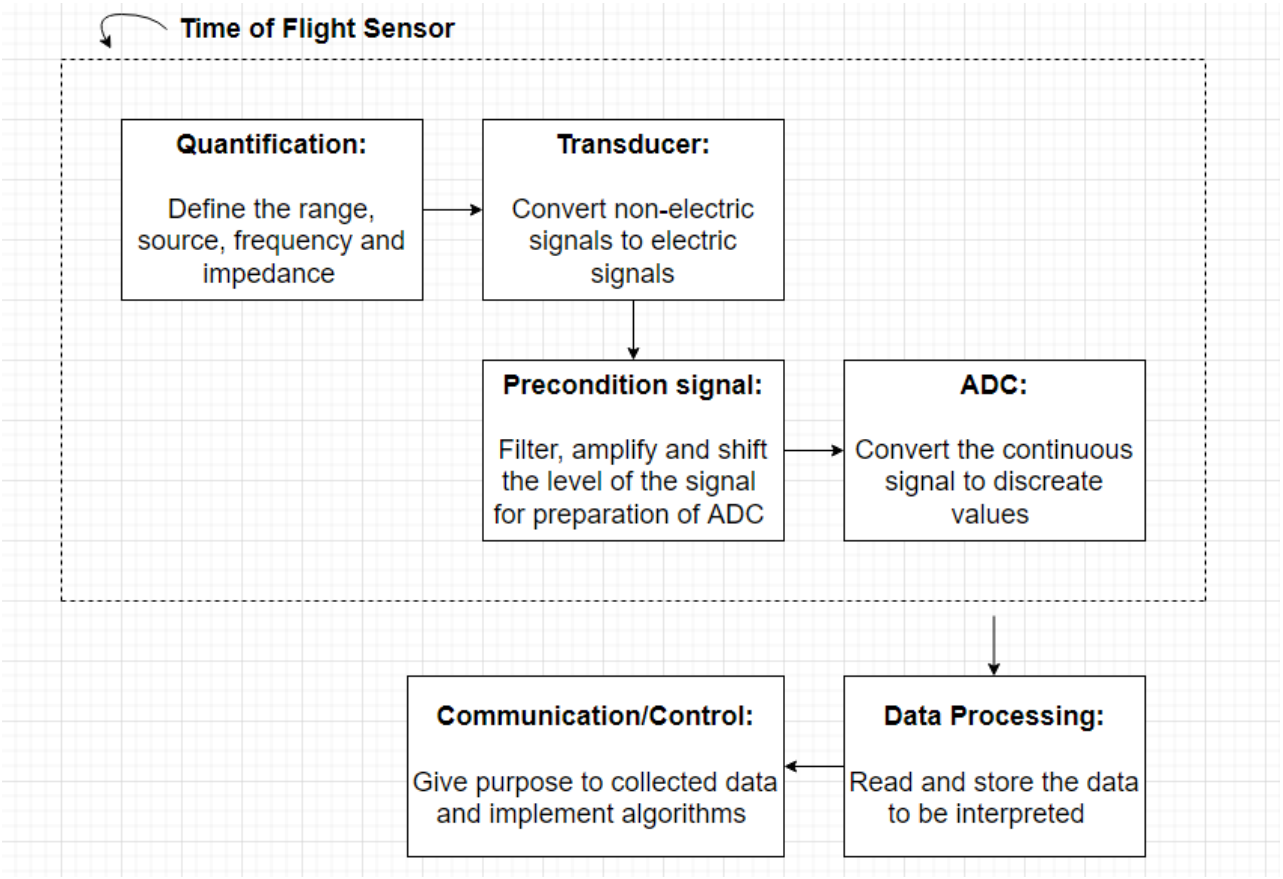


Figure 1: Block Diagram

2. Device Characteristics

Element	Location/ Value
On-board push button	PJ1
On-board LED	PF4
Stepper Motor Driving Pins	PH0 - PH3
ToF pins	PB2 (SCL), PB3 (SDA),
Bus Speed	12MHz
Communication speed	115200 bps
Communication port	COM4 (On PC) COM3 (On Laptop)
Python libraries	pySerial, Open3D, math, NumPy

3. Detailed Description

a. Distance measurements

When collecting distance measurements which would in turn become useful outputs from out system, there are 3 main components which we would have to consider: a transducer, a pre-conditioned circuit, and an ADC (analog-to-digital converter). The time-of-flight sensor has all three features integrated in its design which allows it to output distance values, in millimeters. It achieves this by outputting a light that will reflect off of a surface and bounce back to its receiver. It then calculates the distance by considering the time taken for the light to travel and return. This is done after the signal has been preprocessed and made into a discrete value via the ADC. The distance value obtained is outputted at each 11.25 degree of the motor and is sent to the PC with the use of UART. This accounts for a total of 32 values at 16 steps each. The PC then receives this outputted data though a python script which opens the serial communication port COM3/COM4 (depending on the device used) at a baud rate of 115200 bps. It then converts this distance measurement into y-z plane values. This is done in two steps. The first step being solving for the current angle of the motor:

$$angle = \frac{motor\ postion}{512} * 2\text{pi}$$

The motor position refers to the steps of the motor. We understand that in our case if we would want 32 values out of 512, we would need 16 steps as 32 * 16 = 512. The value of motor position is initially already 16 as we take the first distance measurement at that angle value. The use of 2pi allows us to get the angle value in radians as it will be used in a python script with python's math library.

The angle value is then used to solve for each y and z value:

$$y = distance * \cos(angle)$$
$$z = distance * \sin(angle)$$

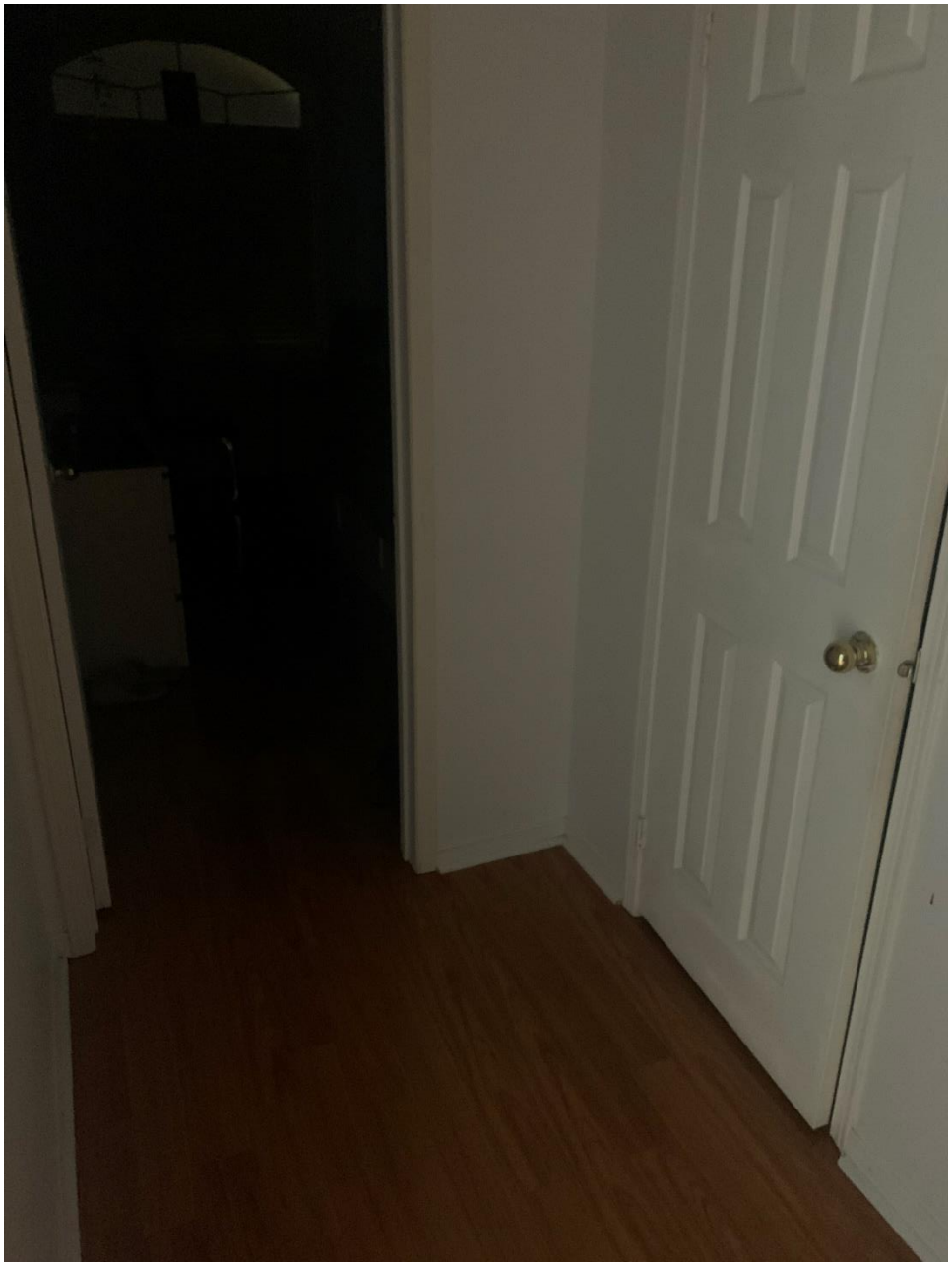
As the y axis will be pointing straight up and the z axis will be pointing to the right-hand side, their distance measurements are found by the use of sine and cosine. Each of these distance measurements were taken at a manual determined displacement, starting at x = 0 and incrementing by a value of 200mm each scan. With 10 scans there was a max x value of 1800mm. At each displacement a scan took place which considered 32 distance measurements. The data points create a vertical slice of the hallway in the y-z plane at each value. These planes are then visualized.

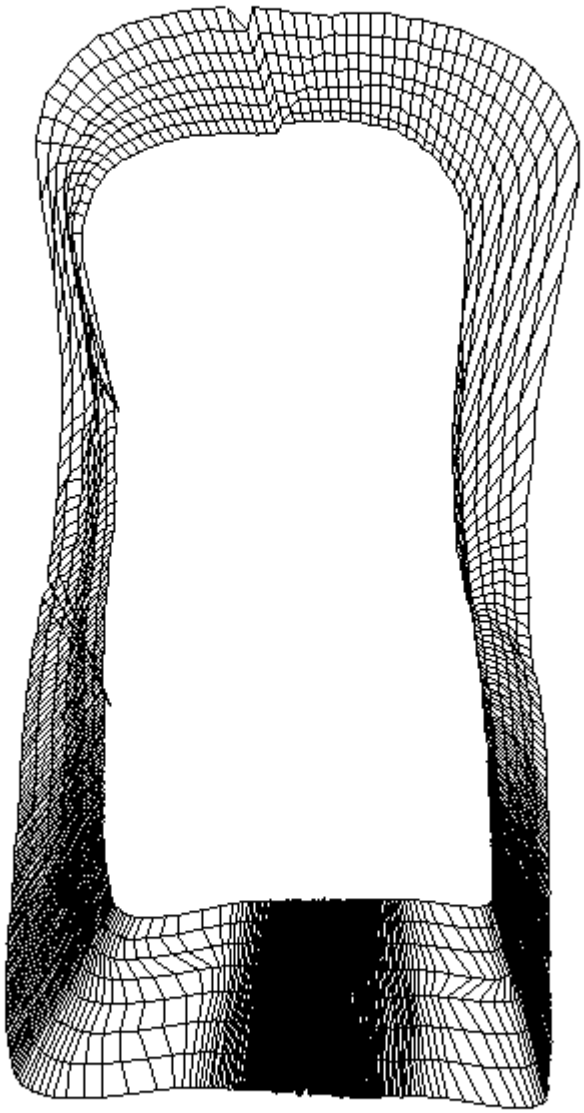
b. Visualization

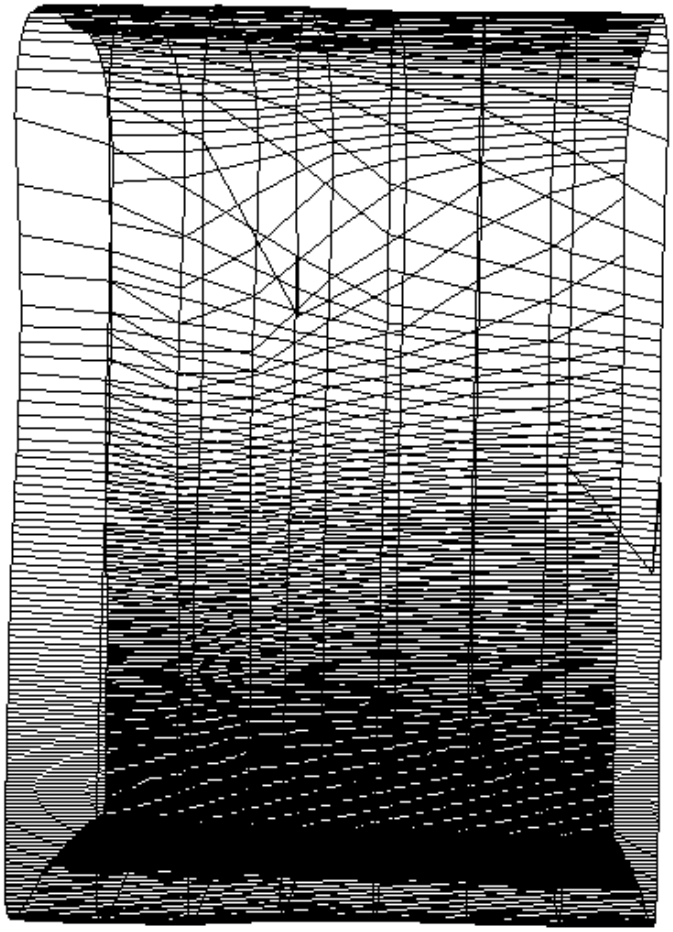
The visualization of the data points was achieved on a MacBook Pro (Intel Core i5, 8 GB of memory and Intel Iris Plus Graphics 635) running windows 10 and python 3.8.6. The python libraries used include:

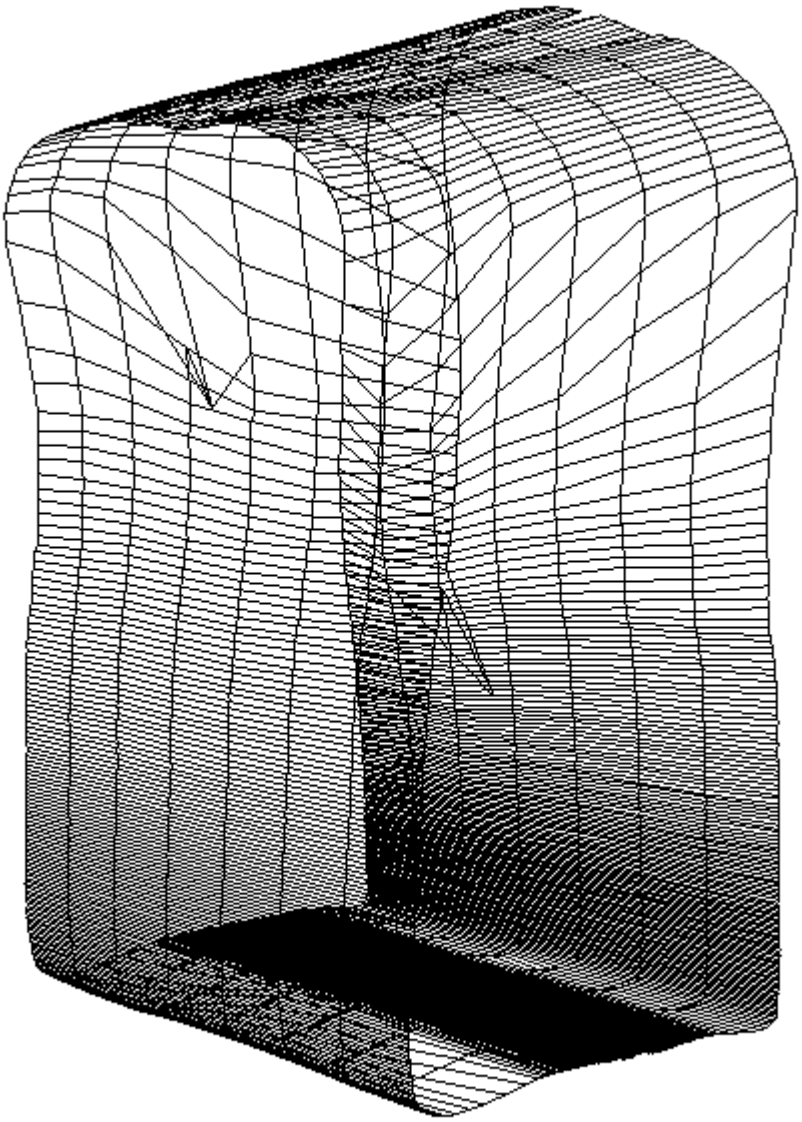
- PySerial: used for serial communication
- Pythons' math library: used for math functions such as trigonometric functions
- Open3D: used to represent the data as point cloud
- NumPy: used to represent the point clouds as arrays

The distance measurements are converted into x, y and z coordinates and are then stored in a .xyz file. The x value is consistence, depending on the manually determined displacement value from before. That same x value is consistent for each y-z plane for that displacement; the y and z coordinates are determined with the use of trigonometric functions by the use of the number of steps and the distance measured. This same process is repeated for each plane, 10 planes in total. A point cloud is then read form the new .xyz file with the use of Open3D. A line set is then created for the visualization. For every point in a plane, the next point in the plane is linked to the previous point, making the plane a continuous surface. Also, a point in a plane is linked to that same point in the next plane thus in turn linking the planes together. Finally, the lineset and point cloud are used to create a 3D image of the hallway.









4. Application example

a. Example
   1. Place the device in a hallway. Make sure there is no direct light sources in the way of the sensor
   2. Run the I2C python script for receiving data on PC
   3. Press the on-board push button which will begin the collection of data for the initial plane.
   4. Wait for device to complete one full rotation
   5. Move the device forward in the positive x direction by a value of 200mm
   6. Repeat the steps 2-5 for 10 sets of measurements while making sure to move the device 200mm between each set
   7. A full set of 10 different [x y z] values are now stored on the PC. Run open3dproject.py to display the creation of the 3D representation of the hallway.

5. User Guide

a. Setup
   1. Assembly of the circuit and device are shown in figure 1. The pin connection are described in Section 2
   2. Connect the ToF by the use of a plastic wall plug that hugs the tip of the motor. Then connect specific wires.
   3. Build and load the project code onto the TI MSP432E401Y. Flow chart shown later
   4. Run the python script data_collect.py. the script will open at default settings of a value of COM3 and a baud rate of 115200 bps and now wait to receive the data. Flow chart later
   5. The device is now ready to use. See above to run device

b. Defined axes

The coordinates used are the x, y and z coordinates. The positive x direction is pointing away from the user, the positive y direction is facing upwards, and the positive z direction is facing to the right. For every set of xyz values, the displacement is with respect to the positive x direction.

6. Limitations

a. FPU

For most microcontrollers, floating point values are supported. If they do have floating point values, they will have a floating-point unit (FPU) that is assigned to each task. For our microcontroller the hardware supports for an IEEE 754 FPU with 32 bits for single precision. This FPU is integrated into the processor, the 120 MHz Cortex-M4F core. This in turn means it can support floating points within 32 bits of precision. In our case the use of trigonometric functions were not used by the microcontroller as they were used in the python script. This was done because it provided a simpler and clearer understanding of how the program functioned as it allowed the microcontroller to focus on data acquisition and the python script to focus on data manipulation. The microcontroller is capable of performing for example trigonometric functions.

b. Quantization error

The maximum quantization error in this case is equivalent to the resolution. Resolution can be solved using:

$$\frac{V_{FS}}{2^m}$$

This is where m is equal to the number of bits and VFS is the full-scale voltage. In our project this would be 5 volts. Now regarding the modules of this project, we had a ToF sensor and the TI MSP432E401. The TI MSP432E401 has a 12-bit ADC and the ToF sensor has a 8 bit ADC. This gives us:

$$MCU\ resolution = \frac{5}{2^{12}} = 1.22\ mV$$

$$ToF\ resolution = \frac{5}{2^{8}} = 19.5\ mV$$

c. Serial communication rate

While regarding the serial communication rate used in this system, it was understood that there would have to be a maximum standard serial communication rate that would be implemented with the PC. For this project a maximum standard serial communication rate of 115200 bps was used. The reason this speed was used and was verified as due to the fact that when higher speeds were attempted to be used errors would be outputted during the transmission. This data was outputted on Realterm.

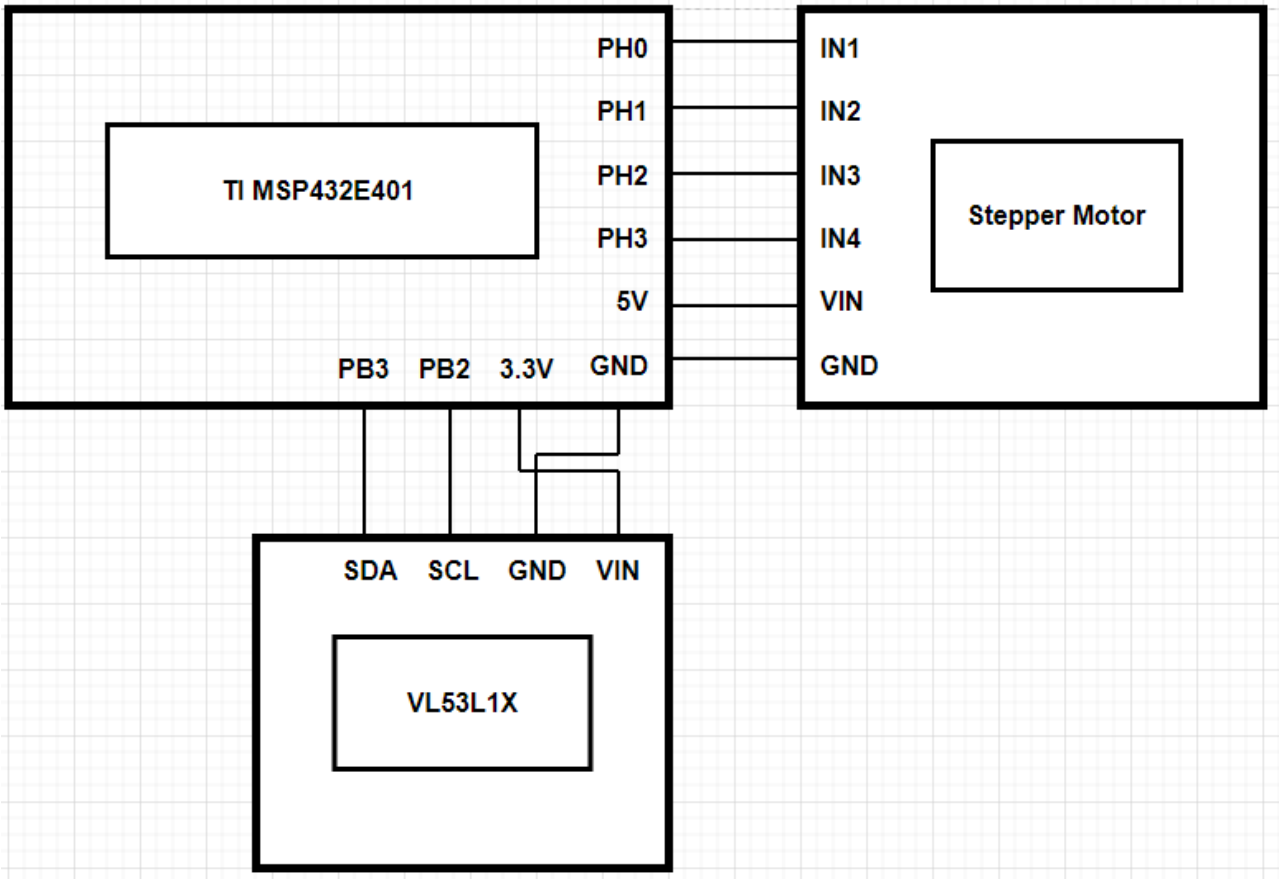
d. Communication methods

The communication method used were an I2C communication between the ToF sensor and the PC itself. One the TI MSP432E401 there are 8 UART's and 1 one of them were used. This implemented asynchronous serial transmission. The port that was used for communication was either COM4/COM3 depending on what device was running the program (PC or laptop). The baud rate that was set was 115200 bps. the TI MSP432E401 would send one byte of data at one time across the COM4/COM3 port which would then be inputted to a waiting python script. The python script in turn would receive and process that data.

e. Primary limitation on speed

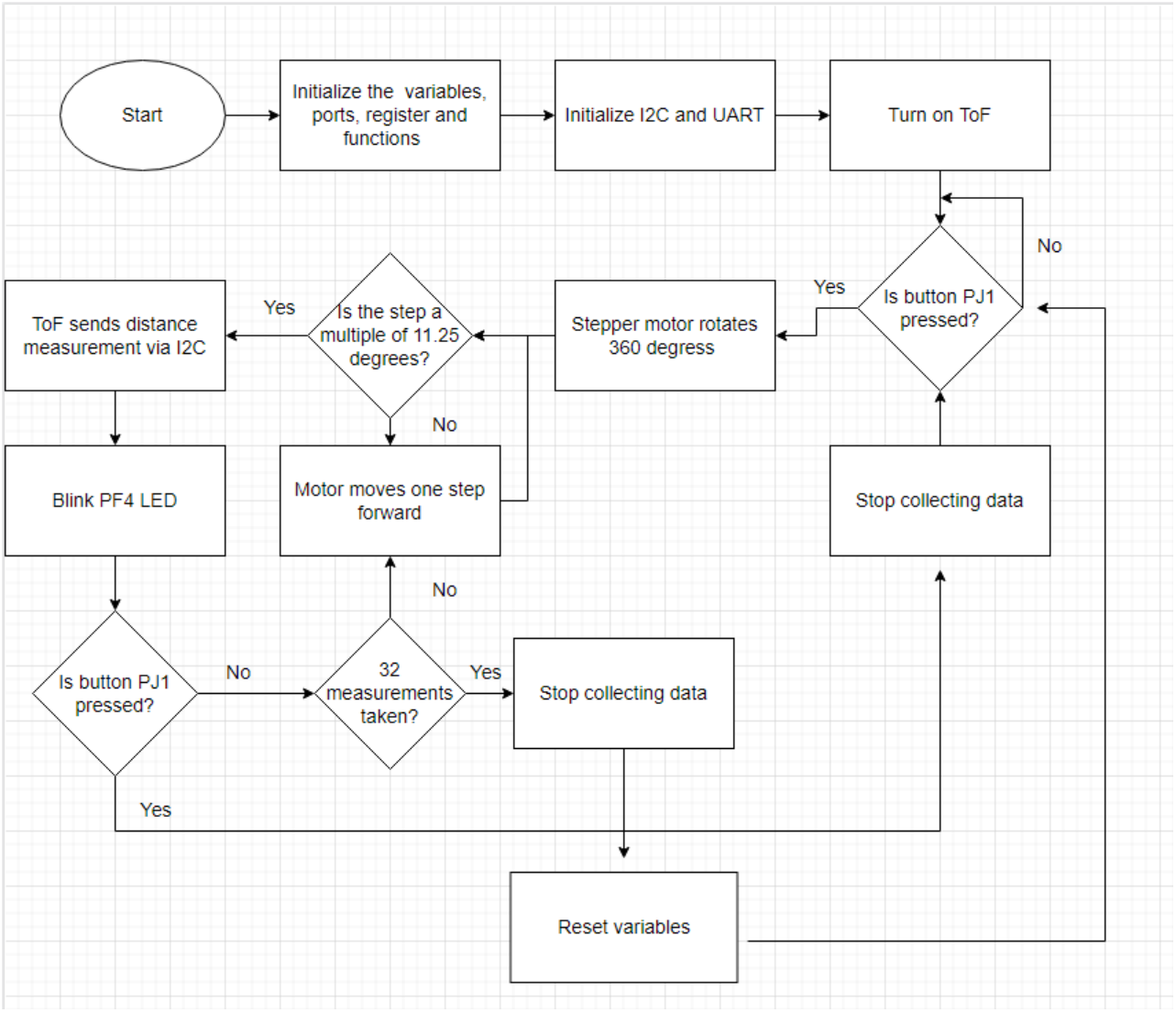
7. Circuit Schematic

a. Figure 1: Circuit Schematic

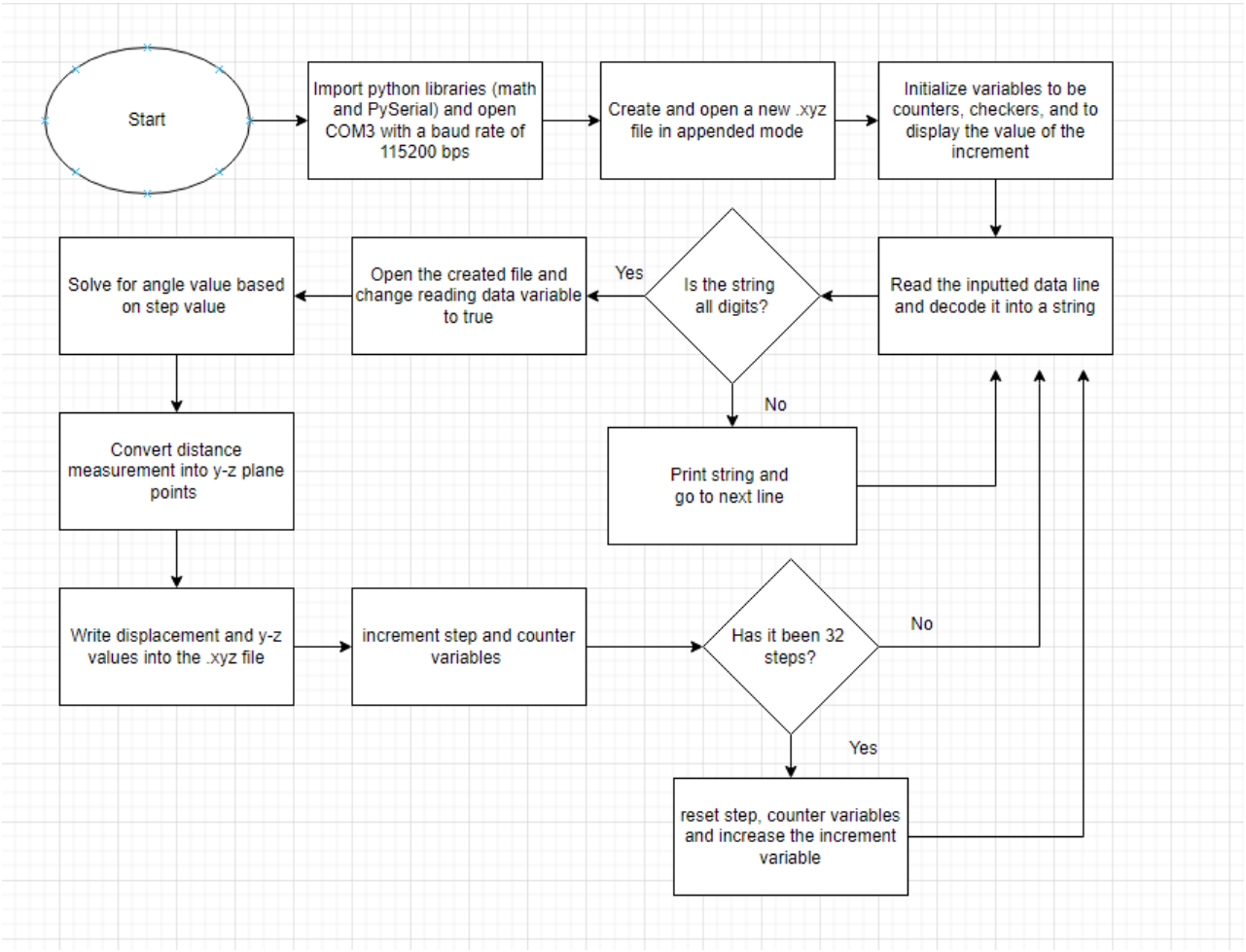


8. Flowcharts

a. Figure 2: Microcontroller flow chart



b. Figure 3: Receiving data python flow chart



c. Figure 4: Visualizing data python flow chart