Microprocessor System Project 2DX3
Project Report

Instructor: Dr. Haddara, Dr. Doyle, Dr. Athar

Jibin Mathew – L03 – mathej30- 400303976

Date of Submission: April 17th, 2023

Table of Contents

1. Device Overview

1.1. Features

The micro controller Texas Instrument MSP432E401Y and the time of flight (ToF) sensor were two of the essential hardware components required for the implementation of this project. The features of the devices are as follows.

Texas Instrument MSP432E401Y:

- Processor – Cortex-M4F Processor with Floating - Point Unit (FPR), Harvard Architecture
- Bus Speed – 120MHz
- Memory – 1024 KB flash memory
- Operating Voltage – 2.5V to 5.5 V
- SRAM – 256KB
- EEPROM – 6KB
- Analog to Digital – SAR based two 12-bit ADC modules.
- Programing Language – C or Assembly Language using Keil to interface.
- Communication: I2Cs and UART – Serial Communication
- Baud Rate – 115200bps (between microcontroller and laptop)
- Cost – CAD $73.40 (Mouser electronics)

Time of Flight Sensor (ToF) - VL53L1X

- Maximum Measurable distance – 360cm
- 3 distinct distance – short range, medium range, and long range
- Operating Voltage – 2.6V to 3.5 V
- Communication: I2C with a maximum frequency of 400KHz – Serial Communication
- Cost – CA $27.16 (Mouser electronics)

Other hardware and software components included:

Stepper Motor Controller

- Rotation – 512 Steps
- Operation – Clockwise and counter clockwise
- Operation Voltage – 5V

Visualization

- Python 3.10 – used to perform the calculation and process the data.
- Open 3D – used to built and plot the processed values to create the 3D model.

Miscellaneous Components

- Push Button – used to control the state of the motor.
- Resistor – 1KΩ (pull down resistor) used to implement the push button.
- Onboard LED – PF4 (measurement LED) all the onboard LED was used to signify completion of different tasks.

1.2. General Description

The purpose of this project is to map out a space (hallways, small rooms, or any space) and create a 3D model of that space. To achieve this, MSP432E401Y was used as the microcontroller and the VL53L1X time of flight sensor was used to acquire the measurement, this raw data is processed using python 3.10 to find the z and y coordinated of that space. This is then plotted using Open 3D to acquire a 3D model of the measured space.

VL53L1X ToF uses LIDAR to acquire the measurements. The onboard emitter sends a light beam which is reflected of an object and is read by the onboard receiver. The ToF sensor is mounted on the stepper motor using a customed 3D printed mount. The stepper motor stops at every 11.25 degrees allowing the ToF sensor to take a measurement. There are 32 measurements in 1 rotation. This analog measurement is converted in to digital measurement by the ToF sensor module is transmitted to the microcontroller using I2C communication protocol. This is then packed and serially communicated to a laptop's serial port (COM4) with the use of UART. The baud rate during the communication is 115200 bps. The z and y distance are calculated and stored on .xyz file using python 3.10. Since the movement along the corridor cannot be measured using the sensor it is manually incremented (20cm at a time) in the python code while calculating the Z and Y coordinate. The process of measure and sending the data to the laptop takes place after each 11.25 degrees. The data is however only written to the file after one whole rotation.

Open 3D uses the points that are saved on the xyz file to plot and built a 3D model. Each line on the xyz file has an X co-ordinate, Y co-ordinate and a Z co-ordinate. After each of the point is plot in the 3D space lines are used to connect each of the points together thus creating a 3D model that represents the measured space.

1.3. Block Diagram

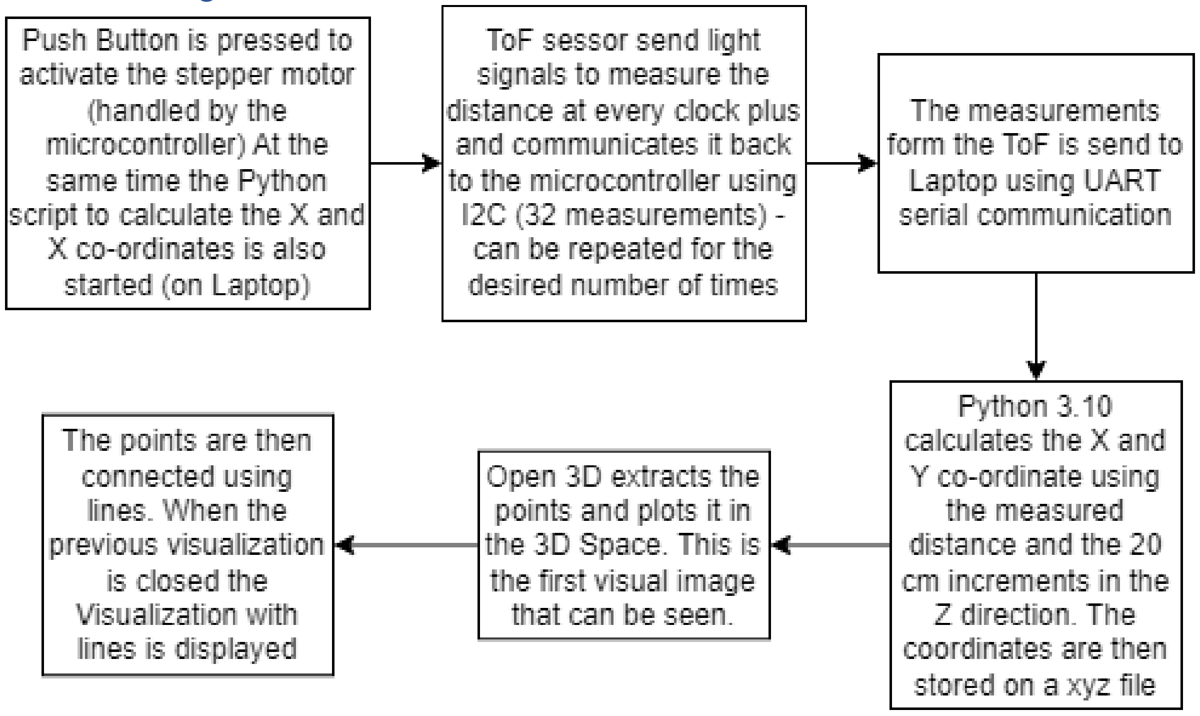

Push Button is pressed to activate the stepper motor (handled by the microcontroller) At the same time the Python script to calculate the X and X co-ordinates is also started (on Laptop)
ToF sessor send light signals to measure the distance at every clock plus and communicates it back to the microcontroller using I2C (32 measurements) - can be repeated for the desired number of times
The measurements form the ToF is send to Laptop using UART serial communication
Python 3.10 calculates the X and Y co-ordinate using the measured distance and the 20 cm increments in the Z direction. The coordinates are then stored on a xyz file
Open 3D extracts the points and plots it in the 3D Space. This is the first visual image that can be seen.
The points are then connected using lines. When the previous visualization is closed the Visualization with lines is displayed

Figure 1 Block Diagram

2. Device Characteristics Table

Component	Connection	Port	Pin
Stepped Motor	IN1	H	0
	IN2	H	1
	IN3	H	2
	IN4	H	3
Time of Flight Sensor	VIN	3.3V	-
	GND	GND	-
	SDA	B	3
	SCL	B	2
Push Button	Signal	M	0
LED	Onboard	F	4
External Applications			
Application	**Language**	**Purpose**	
Keil	C and Assembly	To program the microcontroller MSP432E401Y	
Python 3.10	Python	Serial Communication	
		Open 3D	

3. Detailed Description

3.1. Distance Measurement

The VL53L1X time of flight sessor uses a LIDAR to measure the distance between itself and an object. Lidar works on the principle of emitting a photon particle periodically and then measuring the time it takes for that photon to be sensed by a sensor. The distance is then calculated using the measured time and speed of light. The following formula is used to calculate the distance in millimetre.

$$Distance\ (mm) = \frac{1}{2}(measured\ time\ \times speed\ of\ light)$$

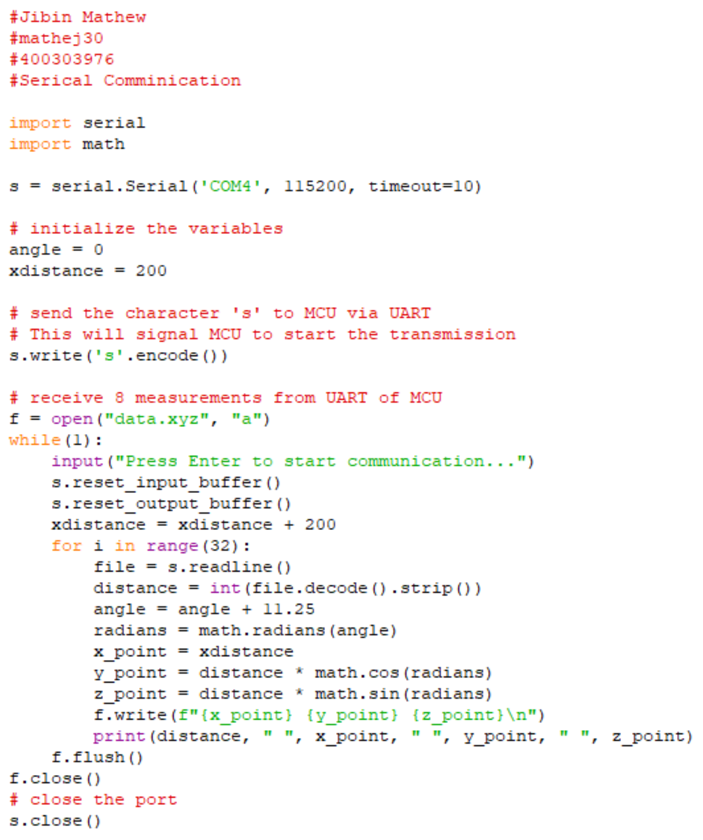
We must divide the product of speed of light and measured time by two because the measured time is for the light to go to the object and come back. If the division is not done, we get the double the distance between the sensor and the object. In the above equation speed of light is $3x10^8$ m/s. in our specific case the sensor can accurately measure approximately 360cm.

The VL53L1X ToF sensor user I2C communication to interact with the microcontroller, since I2C is a synchronous communication, the number of measurements that are taken can be controlled. In my specific case in one 360-degree rotation the sensor takes 32 measurements. Each measurement is 11.25-degrees apart. We obtained 32 measurements by performing 360/11.25. There is a 10ms delay between each 11.25 deg rotation during this delay the sensor collected the measurement for that angle. This data is then transmitted through I2C to the microcontroller. I2C protocol has bidirectional data transfer, therefore, two communication lines are required for I2C. The Serial Clock Line (SCL) keeps track of the clock and starts the measurement process, and the Serial Data Line (SDA) transmits the data to and from the ToF sensor. This data is then transmitted serially to the laptop’s serial port (COM4) using UART. This is an asynchronous data transfer method; in this project the data is transmitted after each measurement is take. Only the raw distance data is packed and transmitted to the laptop. The other available data packets are not measured or transmitted from the ToF sensor at all. This will help save a bit of time, this will be helpfully if more data needs to be collected. For this data to be transmitted over UART a baud rate of 115200 bits per second.

The transmitted data needs to be processed to calculate the Z and Y coordinates, it must be noted that the Z is the horizontal component, Y is the vertical component and X is the depth component in the XYZ plane. These calculations are done with the help of trigonometric principles and were performed through a Python script which uses Python 3.10 using the math library. The equations are as follows.

- The Y coordinate is calculated using the formula:
  - **y_point = distance * math.cos(radians)**
    - y_point - calculated coordinate
    - distance – the measured distance from the sensor
    - radians – the radiant value of angle which is initially 0 but increments by 11.25 deg during each iteration.
- The Z coordinate is calculate using the formula:
  - **z_point = distance * math.sin(radians)**
    - z_point - calculated coordinate
    - distance – the measured distance from the sensor
    - radians – the radiant value of angle which is initially 0 but increments by 11.25 deg during each iteration.
- The X coordinate is incremented by 200 millimetres after each 360 rotation (32 measurements) through the formula:
  - **xdistance = xdistance + 200**

All the calculated data is written to a XYZ file called ‘data.xyz’ and stored in the file directory.

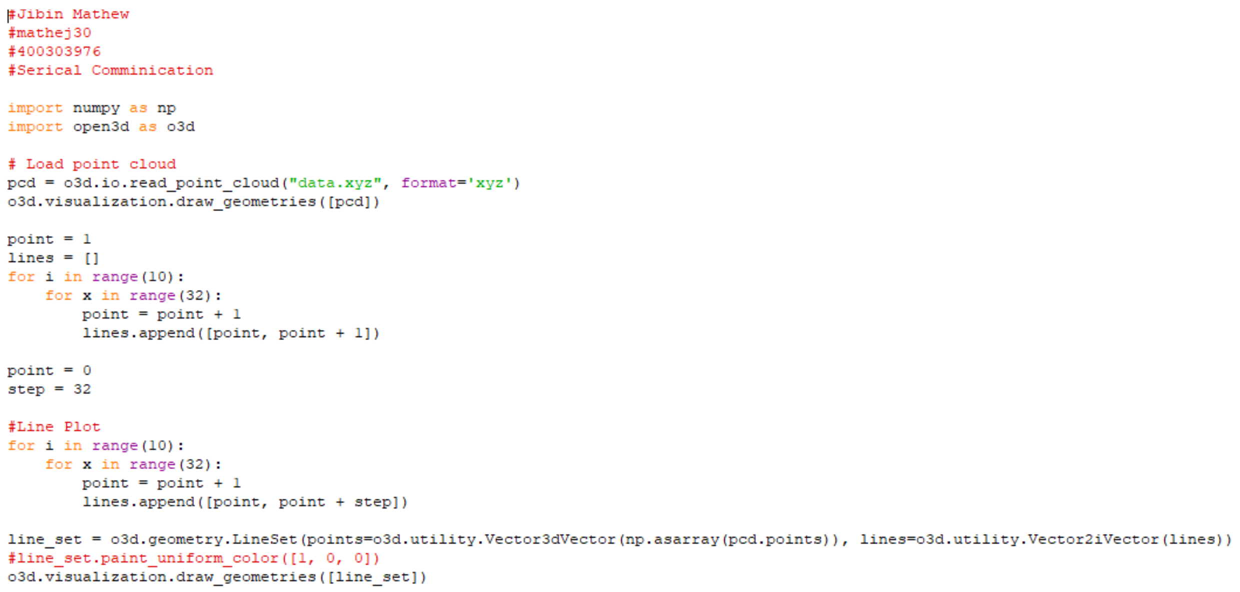


```python
#Jibin Mathew
#mathej30
#400303976
#Serical Comminication

import serial
import math

s = serial.Serial('COM4', 115200, timeout=10)

# initialize the variables
angle = 0
xdistance = 200

# send the character 's' to MCU via UART
# This will signal MCU to start the transmission
s.write('s'.encode())

# receive 8 measurements from UART of MCU
f = open("data.xyz", "a")
while(1):
    input("Press Enter to start communication...")
    s.reset_input_buffer()
    s.reset_output_buffer()
    xdistance = xdistance + 200
    for i in range(32):
        file = s.readline()
        distance = int(file.decode().strip())
        angle = angle + 11.25
        radians = math.radians(angle)
        x_point = xdistance
        y_point = distance * math.cos(radians)
        z_point = distance * math.sin(radians)
        f.write(f"{x_point} {y_point} {z_point}\n")
        print(distance, " ", x_point, " ", y_point, " ", z_point)
    f.flush()
f.close()
# close the port
s.close()
```

Figure 2 Python Script used to receive data from the microcontroller and process the data to obtain XYZ coordinates.

3.2. Visualization

The visualization is done through the Open 3D software with the help of the libraries numpy and open3d. From the previous section a file named data.xyz is stored in the file directory, each line of this file contains the X, Y and Z coordinate of a point. This extracted line by line and plotted in the point cloud. Since there are 32 measurements for each rotation the 32 lines are extracted per rotation. This is repeated for the number of rotations that was measured, in this specific project that was 10. This is why there is a for loop with 10 iterations which contains a for loop with 32 iterations. This creates a 3D space with just the points. The next block of code creates a 3D space in which all the points are connected with lines this helps with seeing the hallway more clearly. The o3d.io.read_point_cloud Function constructs point cloud from the data present in the data.xyz file.



```python
#Jibin Mathew
#mathej30
#400303976
#Serical Comminication

import numpy as np
import open3d as o3d

# Load point cloud
pcd = o3d.io.read_point_cloud("data.xyz", format='xyz')
o3d.visualization.draw_geometries([pcd])

point = 1
lines = []
for i in range(10):
    for x in range(32):
        point = point + 1
        lines.append([point, point + 1])

point = 0
step = 32

#Line Plot
for i in range(10):
    for x in range(32):
        point = point + 1
        lines.append([point, point + step])

line_set = o3d.geometry.LineSet(points=o3d.utility.Vector3dVector(np.asarray(pcd.points)), lines=o3d.utility.Vector2iVector(lines))
#line_set.paint_uniform_color([1, 0, 0])
o3d.visualization.draw_geometries([line_set])
```

Figure 3 Python script used to create the visualization.

4. Application Example

4.1. Instruction

1) Download and install the latest version of Python, this project example is doe using Python 3.10.
2) Once Python has been installed procced to open command Prompt to install pySerial and open 3D.
3) Install pySerial and open 3D by entering pip install pySerial and pip install open3d. If these libraries are already installed a message will be displayed. If this is the first time the file will automatically be installed, and a prompt message will be shown when it has been successfully installed.
4) To run the project the microcontroller needs to be connected to the laptop and the ToF sensor need to be mounted to the stepper motor and placed on the floor. Figure 10 and 11 show a possible method.
5) Open Keil and load the project that contains required code for the project.

6) Ensure that all the hardware components are connected to the appropriate ports, and they match the one declared in the code.
7) Open the default Python IDE named IDLE and load the Python scripts called Serial Communication.py and Opean3d.py.
8) In the Serial Communication.py file change the COM port to the COM port to which the microcontroller is connected to (in my project its COM4).
9) When ever everything is setup as mentioned above build and load the code the is on Keil to the microcontroller.
10) Reset the microcontroller by clicking the reset button on the launch pad.
11) Run in the Serial Communication.py file by clicking on the run button present on the tool bar of the respective IDEL IDE.
12) A new window will open and a prompt to click on the enter button on the keyboard will be shown.
13) Hit the enter button on your keyboard and then press the physical button on the breadboard to start the stepper motor. It is important that both the button be pressed with in a maximum time of 10 seconds.
14) After one complete rotation, the python script will prompt you to press the enter button to measure another rotation. Before pressing enter the setup needs to be mover 200mm forwards.
15) After the displacement along the X axis is achieved (moving the set-up forwards) step 13 can be repeated.
16) This is done for a maximum of 10 rotations to generate the output shown in the next section.
17) After 10 rotation the python script call opean3d.py can should be open and run.
18) A new window will pop up displaying all the points plotted in 3D space. (Figure 5)
19) If popped up window is closed another window will pop up, this will display all the points connected with lines. (Figure 6)

4.2. Expected Output

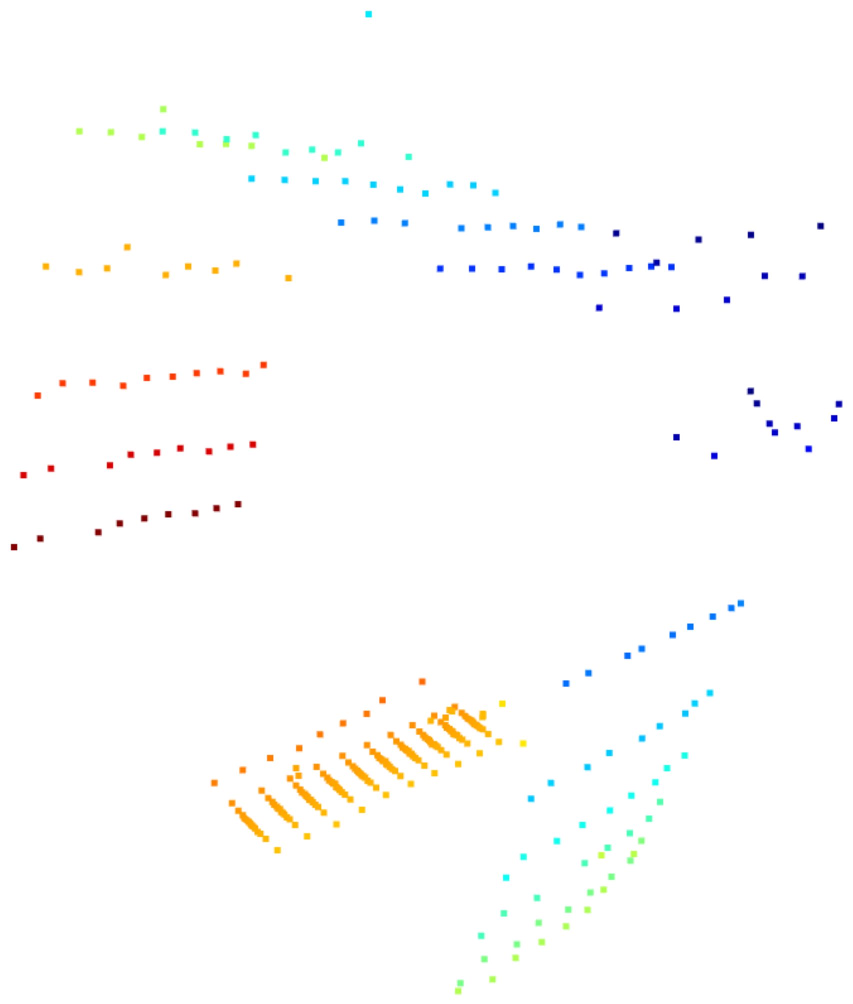


Figure 4 Points plotted in the 3D space.

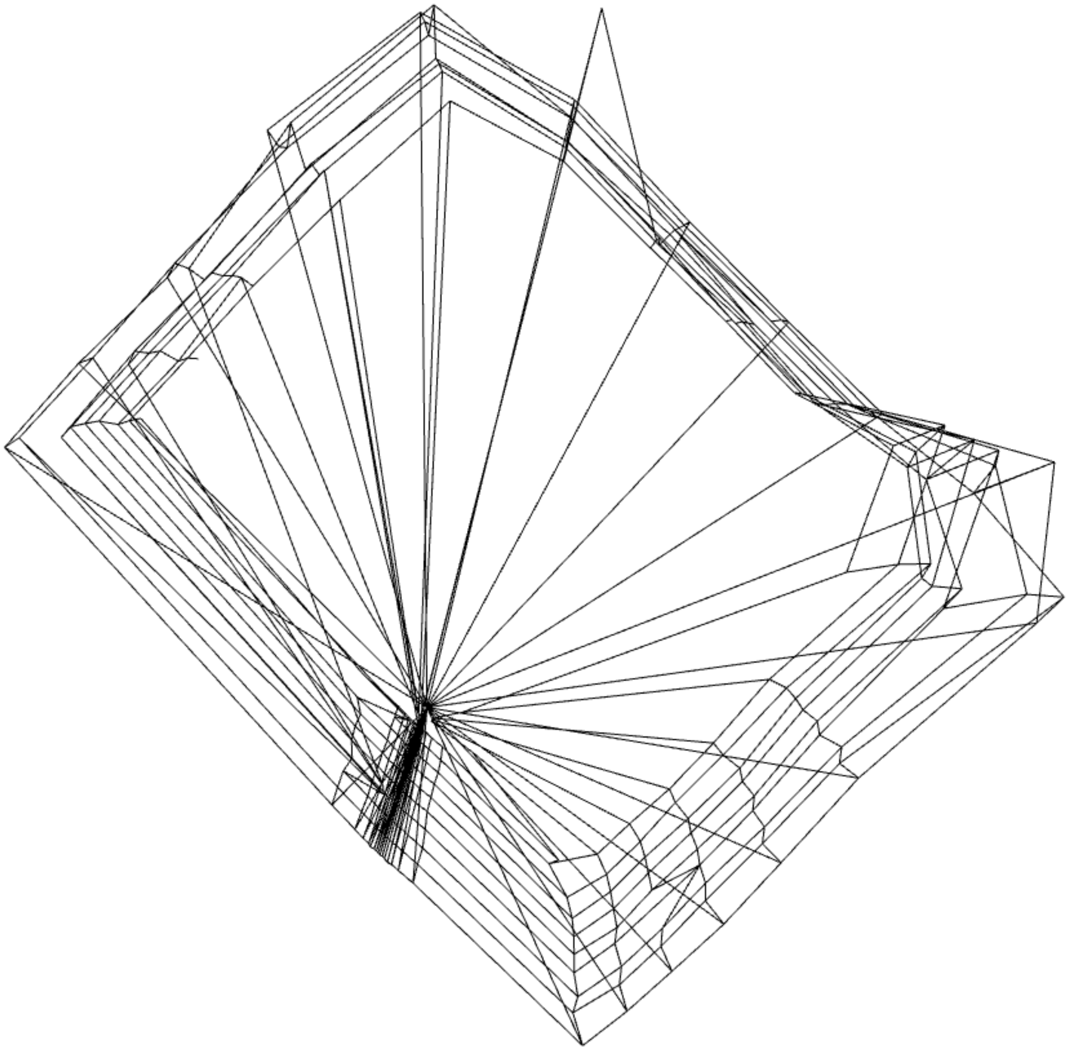


Figure 5 Plotted points connected with lines for better visualization.

5. Limitation

The Floating-point capability of the microcontroller and the use of trigonometric function is a limitation that is present within this design. The measured data that is return from the time-of-flight sensor is a 16-bit integer. The trigonometric functions return a floating value. If we were to do the calculation on the microcontroller, we will have to type cast the calculated values to integer, this will cause the data points to lose accuracy and converting to integer will cause rounding. To overcome this issue, the data is then communicated through UART to the laptop. Through Python and the trigonometric functions, we can calculate the Z (horizontal) coordinate and Y (Vertical) coordinate. The resulting calculation will not have to be type casted thus keeping the accuracy.

The quantization error is 4000/(2^16) = 0.061 mm per bit. This is based on the long range mode having a total distance of 4000mm.

The communication used between the PC and microcontroller is UART. 128000 bits per second (bps) is the maximum serial communication rate that is available. This was verified using the time-of-flight VL53L1X sensor and the capabilities of UART.

The communication used between the VL53L1X time-of-flight sensor, and the microcontroller is I2C protocol. And it is configured for 100 kilo bits per second (kbps) clock.

The stepper motor and the VL53L1X time-of-flight sensor are the hardware's that was the primary limitation on the speed of the system. The ToF sensor was limited to the speed of measurements when in long range mode. The maximum sampling rate in this mode was 30Hz. Even if a faster sampling sensor is available the stepper motor would limit the speed of measurements. The stepper motor cannot reach any speed that is close the rest of the system. This limitation can be test by setting all other hardware components to a fixed speed and then varying the data collection speed form the ToF sensor. By observing the collected data conclusion can be draw about the limitation of the ToF sensor.

6. Circuit Schematic

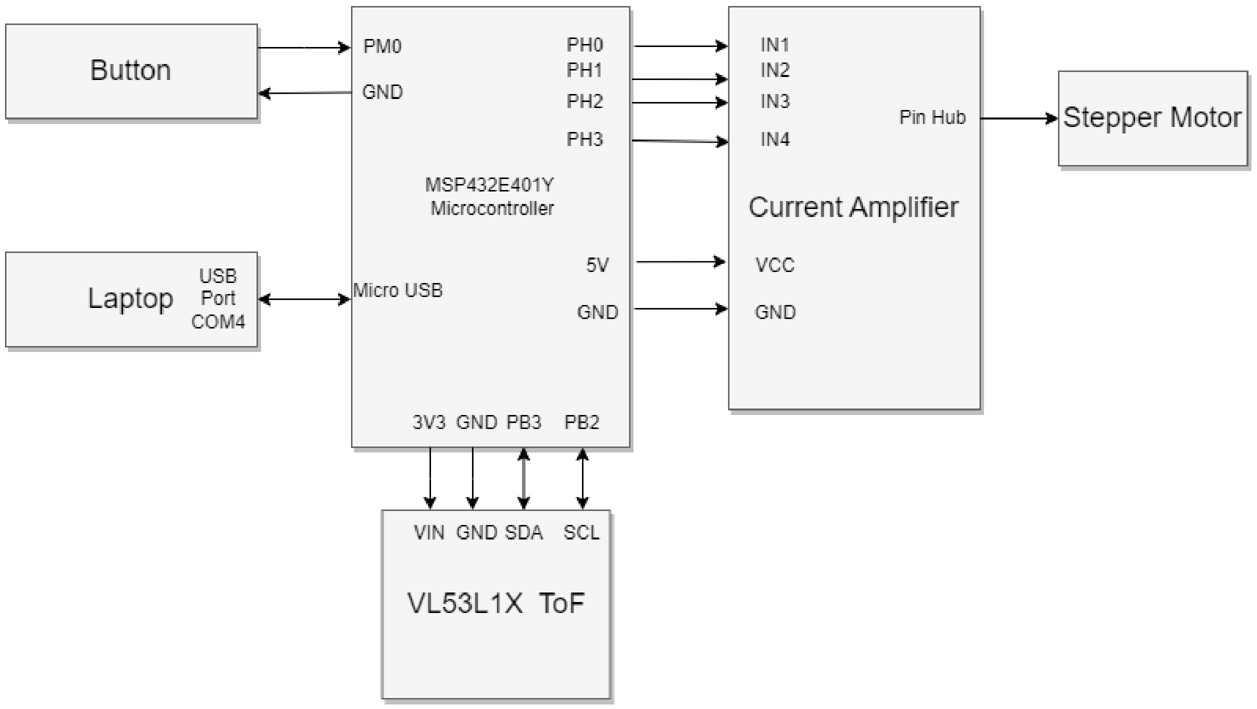


Figure 6 Schematic of the Circuit

7. Programming Logic Flowchart

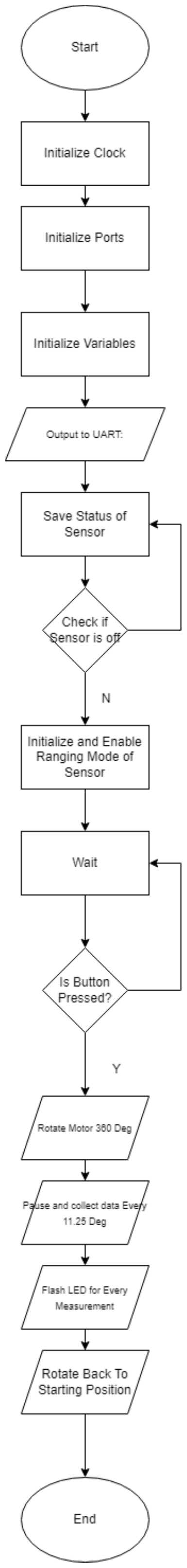


Figure 7 Keil flow Chart

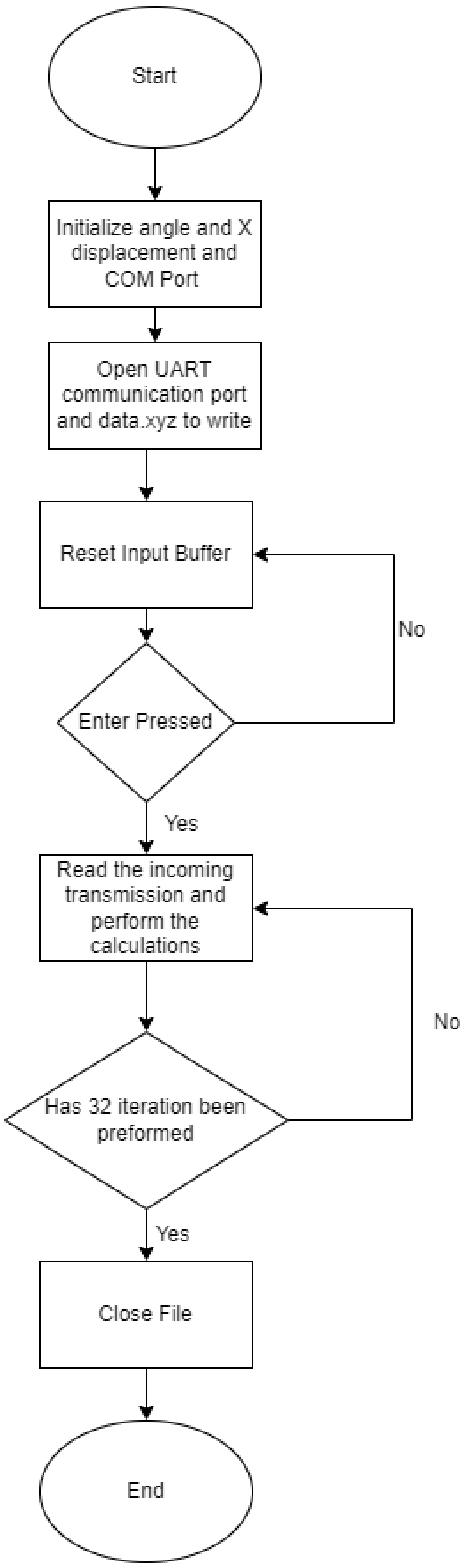

Start
Initialize angle and X displacement and COM Port
Open UART communication port and data.xyz to write
Reset Input Buffer
Enter Pressed
No
Yes
Read the incoming transmission and perform the calculations
Has 32 iteration been preformed
No
Yes
Close File
End

Figure 8 Serial Communication Flow Chart

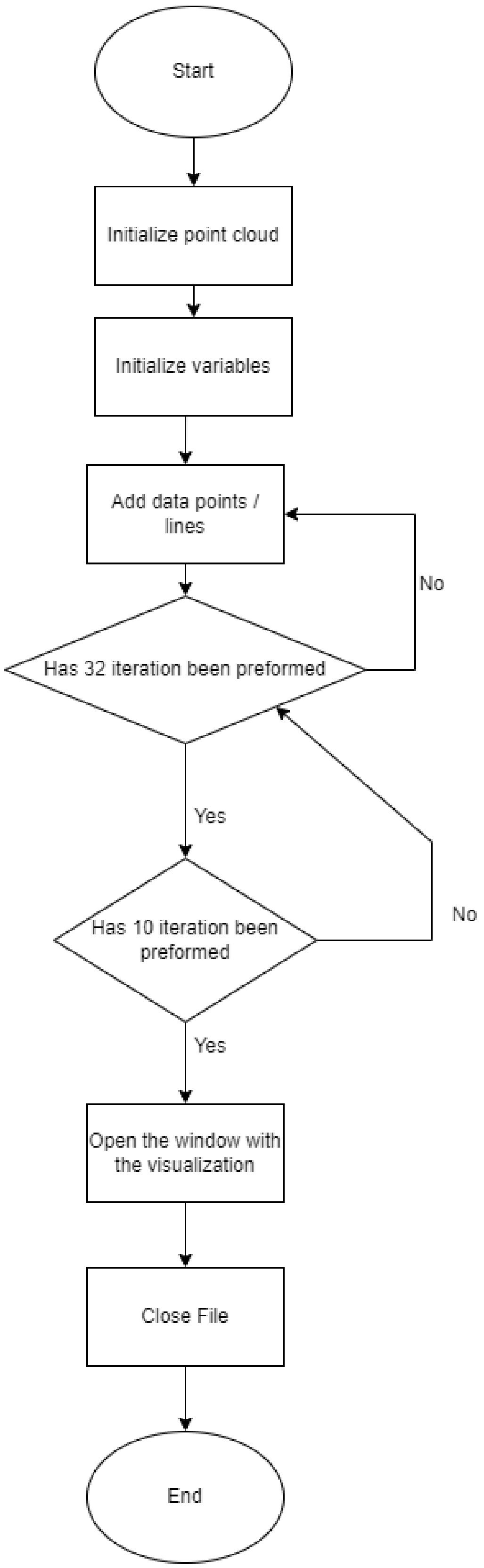

Start
Initialize point cloud
Initialize variables
Add data points / lines
Has 32 iteration been preformed
No
Yes
Has 10 iteration been preformed
No
Yes
Open the window with the visualization
Close File
End

Figure 9 Open 3D Flow

Appendix

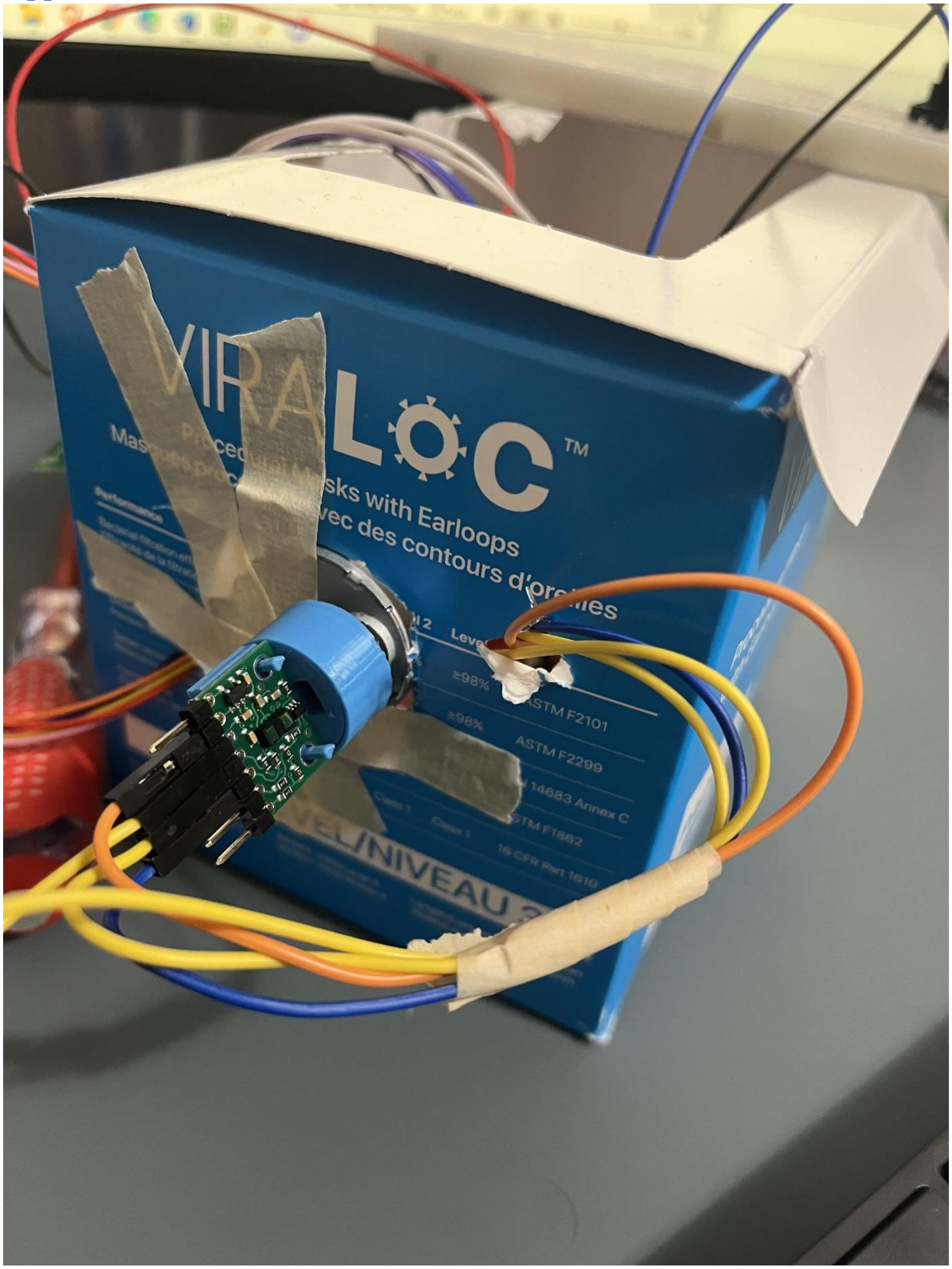


Figure 10 3D printed mount

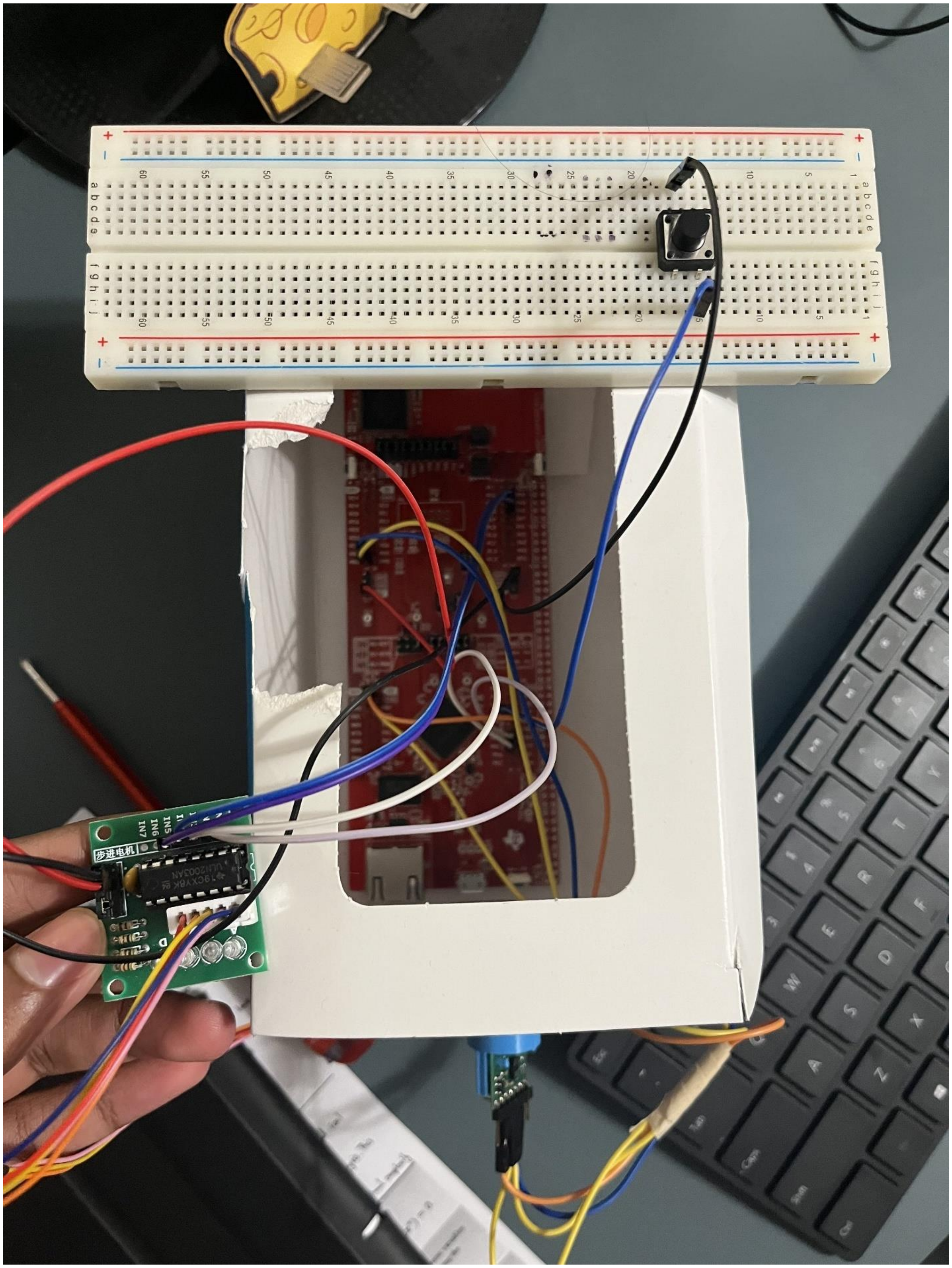


Figure 11 Combined Setup