COMPENG 2DX3

Final Deliverable

Harikashan Thayeswaran - thayeswh - 400326364 - L07

Instructors: Dr. Yasser M. Haddara, Dr. Thomas Doyle, Dr. Shahrukh Athar

Date of Submission: April 17, 2023

Device Overview:

The device created is a commercial light detection and ranging (LIDAR) system. LIDAR devices are low-cost, bulky and effective systems that can scan the area around the device, obtain necessary data and display a 3D of the observed area. The system consists of numerous components that all play a specific role in capturing and displaying a 3D image. The LIDAR system created contains an MSP432E401Y microcontroller, a VL53L1X ToF sensor, a 28BYJ-48 stepper motor, a button, and a PC.

Features:

- MSP432E401Y microcontroller
    - Highest operating voltage is 5V
    - 12 MHz bus speed (according to student number)
    - 4 internal LEDs (state LEDs)
    - Cost: $53.53 [1]
    - Texas Instrument, 32-bit Arm Cortex-M4F
    - 1024 KB for the flash storage
    - 256 KB for static RAM storage
    - Uses internal ADC
- VL53L1X ToF sensor
    - I2C communication protocol with microcontroller
    - Ranges 50 Hz in frequency
    - 2.6V to 3.5V for operation
    - Cost: $14.95 [2]
    - Can collect measurements up to 4 meters
    - Shoots light pulse (photon), waits for it to travel back, and measures distance between sensor and object using the formula, distance = (photon travel time/2) * speed of light
    - Built-in ADC
- 28BYJ-48 Stepper Motor
    - Unipolar motor
    - 512 steps for one rotation
    - 5 wires (2 north, 2 south and 1 voltage supply)
    - Cost: $12.22 [3]
    - Operates at 5V
- Button
    - Requires power supply and input
    - Needs to be pressed for functions encoded to work
    - Cost: ($7.99/100) = $0.08 per button [4]
- PC:

- Uses python libraries
- Receives data from the microcontroller through UART communication protocol
- Cost Estimate: $1000-$2000
- Gathers data, places into one text file, and displays 3D image
- Operates at baud rate of 115200

General Description:

The objective of this device is to measure distances of the area around the device and create a 3D visual of the area through a graphical representation. The device is able to complete this objective through the use of a MSP432E401Y microcontroller, a VL53L1X ToF sensor, a 28BYJ-48 stepper motor, a button, and a PC. The microcontroller connects all the devices together through certain GPIO pins and USB ports for efficient communication. The microcontroller sends signals of data between all devices and effectively communicates with all of them. The I2C communication protocol is used for the data transmission between the microcontroller and ToF sensor, while the UART communication protocol is used between the microcontroller and PC. The button is placed on the external circuit board, which is connected to the microcontroller. The button is pressed to allow the stepper motor to rotate and the ToF sensor to collect data. The stepper motor is connected to the microcontroller and is used to effectively spin the ToF sensor while it collects data. The stepper motor makes a full rotation in 512 steps and stops every 16 steps to take a measurement, which results in 32 measurements for one full 360-degree rotation. The ToF sensor has an emitter and receiver. The emitter shoots a photon, which hits a surface and comes back to the sensor in the receiver. In order to get distance measurements, the ToF sensor collects the time it took for the photon to travel back to the emitter, divides it by 2 and multiplies it by the speed of light. The ToF sensor has a built-in ADC circuit which allows the measurements in non-electric form to be sent to the microcontroller in digital form. The steps for this would be sending the signal in the transducer, using signal conditioning, and completing the analog to digital process. This gives an accurate distance measurement, which is then transferred to the microcontroller using I2C protocol. The PC is used after the ToF sensor sends the measurement data to the microcontroller. The PC uses the UART communication protocol with the microcontroller through Python in order to receive the data. The PC, using Python, converts the measurements into x,y, and z measurements using mathematical operations such as cos and sin. The mathematical operations are carried out with the This process is used within the 2DX3 class and is used in this system as well. The data is then displayed in a 3D dimension using specific Python libraries.

Block Diagram:

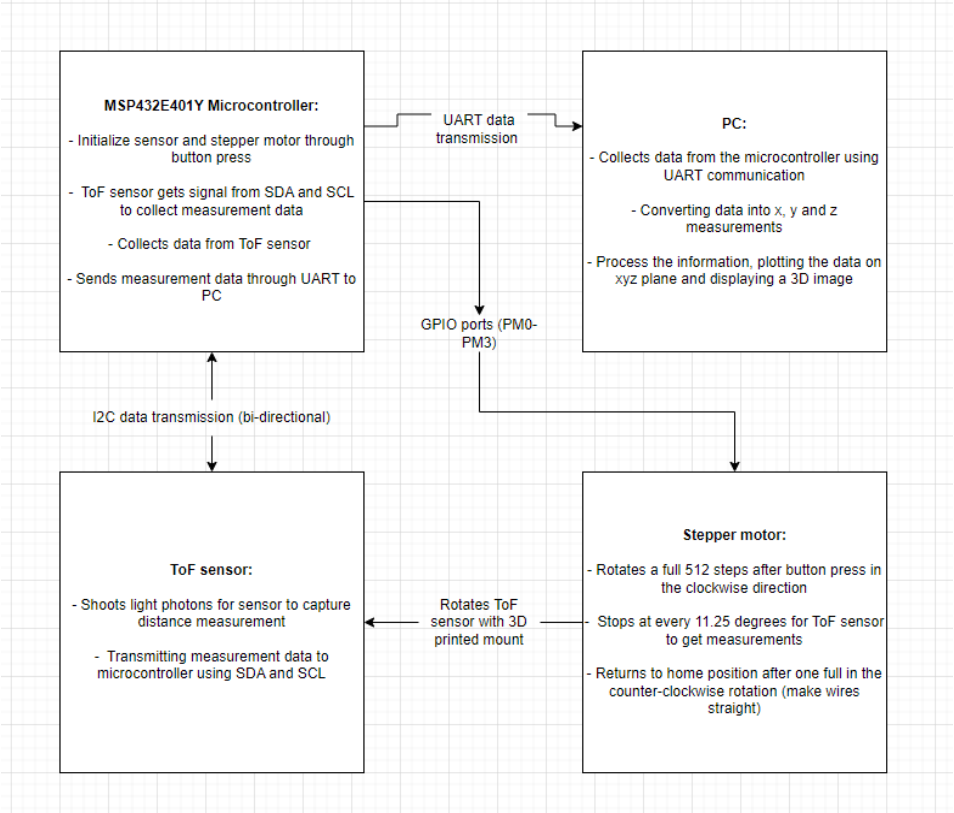

Figure 1: Block Diagram for System

Built Circuit:

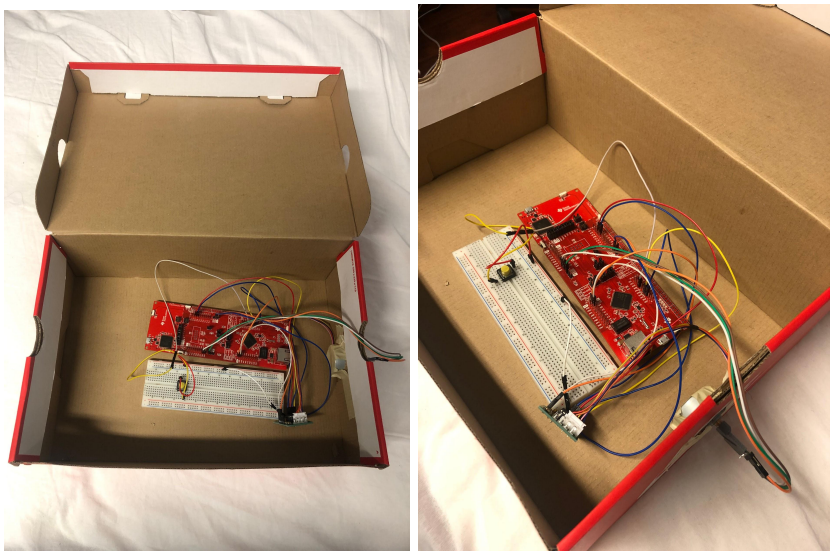

Figure 2,3: Physical Circuit for System

Device Characteristics:

Texas Instruments MSP432E401Y microcontroller	**Characteristic**	**Specification**
	Serial Port	COM4
	Bus Speed	12 MHz
	Measurement LED	PN1
Stepper Motor	**Pin**	**Microcontroller pin**
	IN1	PH0
	IN2	PH1
	IN3	PH2
	IN4	PH3
	- pin	GND
	+ pin	5V
ToF sensor	**Pin**	**Microcontroller Pin**
	SCL	PB2
	SDA	PB3
	Ground	GND
	VDD	3.3V
External Button	**Pin**	**Microcontroller Pin**
	Input	PM0
	Ground	GND
PC	**Characteristic**	**Specification**
	Baud Rate	115200
	Python Libraries	Pyserial, Open3D, Math, NumPy

Detailed Description:

Distance Measurement:

The distance measurements require the microcontroller, the 28BYJ-48 stepper motor, and the VL53L1X ToF sensor to function properly in order to be obtained. The PC then needs to be placed in the right COM for the measurements captured to be displayed in a 3D model. In order to complete this process, the system needs to be able to collect data in a non-electric form, covert the data and display the data on a digital system [5]. The first step in the process is using a transducer to be able to collect non-electric data and convert it to an electric signal. The electric signal is then prepared to be converted to digital form through signal conditioning. Finally, the electric signal is converted to digital form using ADC (analog to digital conversion) and displayed on a digital system. In order to start this process.

To begin, the microcontroller needs to sense when the external button is pressed in order to allow the stepper motor and ToF sensor to function accordingly. The button is coded to have parameters that allow the stepper motor to rotate a full rotation, and the ToF sensor collects 32 measurements of distance for the area around the system. The process of checking if the button is pressed was implemented using a polling method rather than interrupts. The polling method was used because, in this case, the event of a button being pressed occurs often, so a while loop would check the condition. Polling also uses less amount of code and wires to implement compared to interrupts. This polling method then allows the stepper motor to rotate and the ToF sensor to collect data.

The stepper motor is connected to the pins PH0-PH3 while being connected to a 5V power source and ground to function. When the stepper motor starts, the motor rotates a full 512 steps in the clockwise direction. In order to collect accurate measurements, the ToF sensor is attached to the stepper motor to allow the sensor to rotate as well as to observe the world around it. To allow the ToF sensor to collect accurate measurements, the stepper motor needs to stop for a certain amount of time to allow the sensor to retrieve data. The stepper motor, in this case, had a delay of 10ms between every stop, and the stepper motor stopped 32 times for 32 measurements to get an accurate representation of the 3D display. To calculate the degrees that the stepper motor will be stopping at in order to get the 32 measurements, the following formula was used,

$$Degree\ increment\ =\ 360\ degrees\ /\ 32\ measurements\ =\ 11.25\ degrees$$

$$Step\ increment\ =\ 512\ steps\ /\ 32\ measurements\ =\ 16\ steps$$

This degree and step increment means the stepper motor will stop at every 11.25 degrees (16 steps) to allow the ToF sensor to calculate the distance of the surface in front of it. After the stepper motor makes a full rotation clockwise, the motor rotates back to its home position in the

counter-clockwise direction, mainly so the wires connecting the ToF sensor can untangle and the measurement process can restart with minimal discrepancies.

In addition, the ToF sensor plays a major role in obtaining the distance measurements for the system. The ToF sensor has probes that help it function, one is the emitter, and the other is the receiver. The emitter shoots a light pulse, or specifically a photon, which travels from the sensor to the surface and it reflects back into the receiver. The distance is then measured with the following formula,

$$Distance = \frac{Travel\ time\ of\ photon}{2} * Speed\ of\ Light$$

To send the data to the microcontroller, the ToF sensor and the microcontroller use an I2C communication protocol. This protocol is effective within single-circuited boards with multiple devices, so it is efficient with the ToF sensor and microcontroller. The I2C communication protocol has an SDA (Serial Data Line), which is used for data transmission, and an SCL (Serial Clock Line), which synchronizes the communication between the devices. After the ToF sensor uses the I2C communication protocol, the data measurements are then stored in the microcontroller and can be sent out to the PC using another communication protocol.

Finally, the data is sent to the PC through UART communication. The PC, in this case, runs at a baud rate of 115200 bps. In order to connect the microcontroller and PC, the USB port needs to be matched with the correct COM in the serial communication code (in this case, COM4). In order to get a 3D display of the area, the distance measurements need to be converted into x, y, and z components, which was done in the serial communication code. To convert the data into xyz measurements, the angles from the microcontroller data were used. The angles needed to be converted into radians as the Python libraries operate within radians rather than degrees, so the conversion of angles to radians was done by the formula,

$$Radians = Angle * (\frac{\pi}{180})$$

One example calculation of this would be with 11.25 degrees, which comes out to be 0.1963 in radians. The x-distance measurements were calculated manually within the code and incremented. The y and z distance measurements were calculated using trigonometry, and the formulas,

$$y = ToF\ sensor\ distance * sin(radian)$$

$$z = ToF\ sensor\ distance * cos(radian)$$

An example calculation would be converting a measurement of 200 in 11.25 degrees to a y and z measurement. This would create a y measurement of 39 and a z measurement of 196.16. When running the serial communication code and enter is clicked before the button press on the circuit,

the data is sent to another file called “measurements_ToF.xyz” file, which contains all measurements in the xyz plane. After the data is collected, the 3D diagram code is then used to display the distance measurements on a 3D plane.

In order to get accurate distance measurements with the device, the device should be placed on the ground with a clear path forward. The displacement of the device is crucial in order to create an accurate representation of the area. The displacement would need to be forward in the x direction as the user is moving the device in the x-axis. To obtain the distance measurements, the device should be moved forward an appropriate amount (around 300mm). This was implemented within the code by creating a counter for x measurements and incrementing that value by 300mm after every measurement. This would allow accurate measurements to be captured by the ToF sensor.

Visualization:

In terms of visualization, the main code that allows the PC to display a 3D image in the xyz is the serial communication and 3Ddiagram python code. To get the distance measurements through UART, convert the distances to an xyz plane, and plot them on 3D visual, the Python 3.10.9 version was installed onto the PC. This would allow the use of libraries such as serial, math, numpy and open3d. After Python was installed, the PC and the microcontroller were connected using the correct serial port (COM number).

To begin, in the serial communication code, the math and serial library were used. The serial library allowed the data transmission to happen through UART and travel through the correct port at a baud rate of 115200. The file “measurements_ToF.xyz” was then created to store all the data measurements taken from the microcontroller. The data is then transferred, and the variables “x_measurement”, “y_measurement”, and “z_measurement” are created to plot the points on a 3D plane. The math library allows the functions such as math.radians, math.cos, and math.sin, which are to be used to convert the angles and measurements accordingly, as mentioned in the distance measurement section. All of the calculations and conversions are done within a for loop that contains a counter parameter, which is in the range of 32 as this system will collect 32 measurements. The x, y, and z values are generated for each measurement within this counter, which creates an accurate xyz plot of the ToF sensor measurements. When running the serial communication code, there will be a prompt that requires the user to press the key “Enter” to start the communication process. The key is pressed before the external button that allows the stepper motor and ToF to function is pressed in order to get the measurements transferred to the PC correctly.

After the data is collected in the “measurements_ToF” file, the 3Ddiagram code then runs to display the data in an xyz plane. The numpy library is used to allow the use of arrays and mathematical operations on arrays. The open3d library allows the PC to read the measurements in point clouds and place the point clouds onto a 3D display. When running the 3Ddiagram code,

the point cloud data from the measurements_ToF file is first manipulated using arrays and functions. The data is then plotted onto a 3D display in points on the xyz plane. The code displays the 3D image and allows the user to rotate the graph to get a full perspective of the measured area.

Application Instructions and Example:

Instructions:

In order to have the code run on the computer, the downloads that the PC should have to run the device are Keil, Python 3.10.9 and Open3D. The first step in the setup is finding the correct port that will connect the PC to the microcontroller for the serial communication Python file to run. The correct port that the microcontroller will be plugged into is found in the device manager, ports, and then the port number is located at the user. This port needs to be used when connecting the microcontroller and the PC together.

Furthermore, the Keil program needs to be downloaded onto the corresponding PC. In order to download the Keil program, the website is found online through the link provided in Studio 0 [6]. The arm Keil website then leads the user through the correct steps to create a license from McMaster and have the program on the PC. The necessary library needs to be downloaded for the program, which is the "MSP432E401Y" pack. After completing these steps, the Keil code can be used for the device. In addition, Python needs to be installed on the PC for the serial communication and 3D diagram code to function. The Python 3.10.9 version is installed through the website found online. After Python is installed, the command prompt needs to be opened, and the command "pip3 install pyserial" needs to be typed to install the necessary libraries [7]. The open3D library then needs to be installed onto the PC through the command prompt as well. The command for this installation is "pip install open3d", and this allows the PC to use open3d libraries for the 3D display with point cloud data. Another library that needs to be installed is numpy. This is done by typing in the command "pip install numpy", which allows mathematical operations to be used on arrays.

Application Process:

To begin, plug the PC into the microcontroller with a USB port. After connecting the device to the PC through the microcontroller, the port on the serial communication code should be changed to the correct port. The corresponding port number (COM), according to the computer, should be replaced in the code line, "s = serial.Serial('COM4', 115200, timeout = 10)". After connecting the microcontroller to the PC, the circuit should be built following the circuit schematic below. The circuit schematic shows how to connect the components necessary for the device to function. The device should then be placed in a box with the stepper motor on the side of the box pointing outwards. The stepper motor should be able to rotate outside the box with the ToF sensor mounted onto the motor with a 3D-printed component.

After connecting the computer to the microcontroller and building the physical circuit correctly, the Keil code is then flashed to the microcontroller. To flash the Keil code to the board, the process is to translate, build and then download the code onto the controller. In this process, the 4 internal LEDS on the microcontroller should turn on and off, indicating that the code was flashed onto the microcontroller successfully. After flashing the code onto the board, the Python serial communication and 3Ddiagram code should be opened in any IDLE that supports Python. The user then needs to run the serial communication code first in order to get the PC and microcontroller to communicate effectively in UART. The code will then display a prompt that will ask the user to press “Enter” to start the communication process. To get the data measurements transferred to the new “measurements_ToF” file, the key “Enter” needs to be pressed first, and then the external button on the circuit is pressed right after. It is crucial to click these buttons in order because if the stepper motor were to rotate before the UART communication is enabled, the data would not be able to transfer correctly to the PC.

The device needs to be kept in a stable position with a clear pathway forward. To have accurate measurements, make sure there are no moving objects that can possibly interfere with the 3D scan. Once the 2 buttons are pressed in the correct order, the stepper motor will start to rotate while the ToF sensor collects data every 11.25 degrees. The data is sent to the microcontroller from the ToF sensor with I2C, and the data is then sent to the PC using UART. The data will be written into the “measurements_ToF” file in order of x_meausurements, y_measurements and z_measurements. If there is an error while the code is running, in order to restart the process, the data in the “measurements_ToF” file needs to be erased and saved manually. This would allow the microcontroller to send a new set of data after the device runs again. If there is no error and the measurements were collected correctly, the user should keep the data in the original xyz file. After the first set of measurements is taken, the user then needs to move the device forward in order to get another set of distance measurements. The “Enter” key and the external button should be pressed again in the correct order for the device to collect the new measurements. This process needs to be repeated a number of times for an accurate diagram to be outputted. After the data is collected, the measurements should be saved automatically onto the xyz file, which will be used for the 3D display.

When the measurements are collected, the 3Ddiagram code then needs to be opened on the same IDLE. When running this code, the measurements should be displayed onto an xyz plane in point cloud data. The diagram will appear as dots on the 3D diagram, and the 3D image can be visualized while rotating the diagram. If the diagram does not come out as expected, this means the sensor may have collected the wrong measurements due to moving objects in its view. In order to fix the output, the data from the “measurements_ToF” sensor file should be erased and saved manually. The full process of collecting multiple measurements should then be repeated with minimal possible interference in the device's pathway. This would allow the 3D display to be approximate to the area around the device.

Application Example/Expected Output (hallway for least significant student number digit 4):

One application example of the device being used was within the ETB building in the furthest hallway to the right from the entrance. This hallway was scanned due to the corresponding student number ending with the number 4 (400326364). The device was set up on the edge of a table, and 4 sets of 32 measurements were taken to scan and display the hallway in a 3D plane. In this case, the x measurement is the displacement done by the code and user, while the y and z measurements are used for the vertical distances. This method is used and practised within the 2DX3 class.

Figure 4: Image of ETB hallway

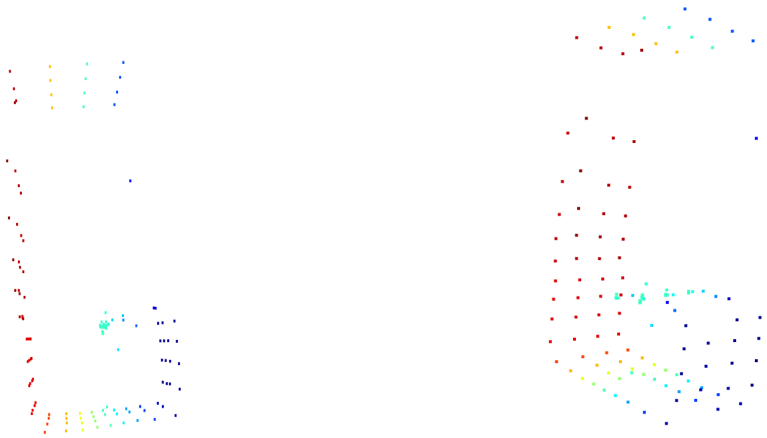

Figure 5,6: Image of 3D display of area

In the figures above, the hallway is displayed in a 3D model in points. The measurements for the right side of the hallway come out more curved than the other side, as the hallway has a curve on the right side of the wall. The measurements did get interfered with by the wires of the circuit because the ToF sensor wire connections often end up in front of the sensor while it is rotating.

Limitations:

The first limitation the device has is the serial communication rate. The communication rate controls how fast the data can be transferred to the PC from the microcontroller. The communication rate is limited by the laptop being used, in the application example case, the serial communication rate on the MSI PC used was 128000 bits per second. This was checked through the device manager and shown in the specific port properties. There are numerous options for the port speed; however, the maximum serial communication rate is 128 kilobits per second.

The second limitation would be the microcontroller floating point unit rate (FPU). The FPU in the microcontroller is set to a maximum of 32 bits in the specific Texas Instruments microcontroller in this system. The floating bits in the microcontroller limit the ability of the microcontroller to transmit accurate data to the PC as there is a fewer amount of digits used. This would affect the accuracy of the trigonometric data calculated in the Python code in xyz plane, as the values will not be exact. The more optimal floating point unit value that can be used is the 64 floating bits, which is in the double precision format.

The third limitation of the implemented system is the Maximum Quantization Error. The equation to find the quantization error is,

$$Max\ Quant.\ Error\ =\ \frac{Distance}{2^n}$$

When implementing the ToF sensor in the device, the maximum distance it can measure is 4m. This parameter would correspond to distance in the formula. The n parameter, which corresponds to the number of bits, is set to 16 [8]. This results in the maximum quantization error of

$$Max\ Quant.\ Error\ =\ \frac{4m}{2^{16}}$$

$$Max\ Quant.\ Error\ =\ 0.000061035$$

Therefore, the maximum quantization error of the ToF sensor is 0.000061035.

The fourth limitation of this system is the limitations on the speed by the main elements of the system. After reviewing the entire system, one main element of the system that limits the speed of the I2C data transfer is within the software of the I2C protocol. The I2C protocol makes a maximum of 100000 bits per second for the data transfer due to device functionality with different components being connected together. The bus speed of the communication protocol was also set to 12 MHz according to student number, which also decreased the data transfer speed.

Finally, another limitation of the system is the parameters used for the communication methods in this system. The ToF sensor uses the communication protocols of the I2C with the microcontroller. The I2C communication is done using bi-directional signal lines, which are serial clock line and serial data line. The I2C communication protocol requires a larger amount of wires to be used and is more complex than UART; however, it is effective for a single-circuit system with multiple devices. The communication protocol from the microcontroller to the PC uses the UART communication protocol. The parameter for the UART communication protocol was set to 115200 bits per second in baud rate, which affects the performance of the system.

Circuit Schematic:

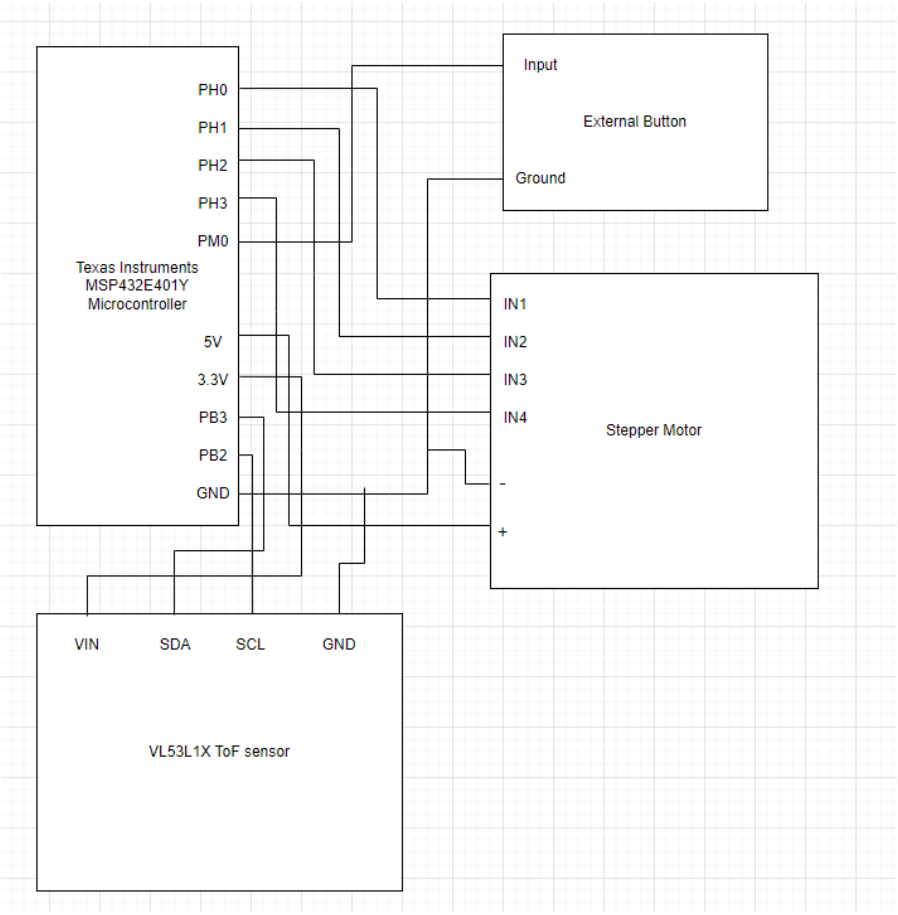

Figure 7: Circuit Schematic for System

Flowchart:

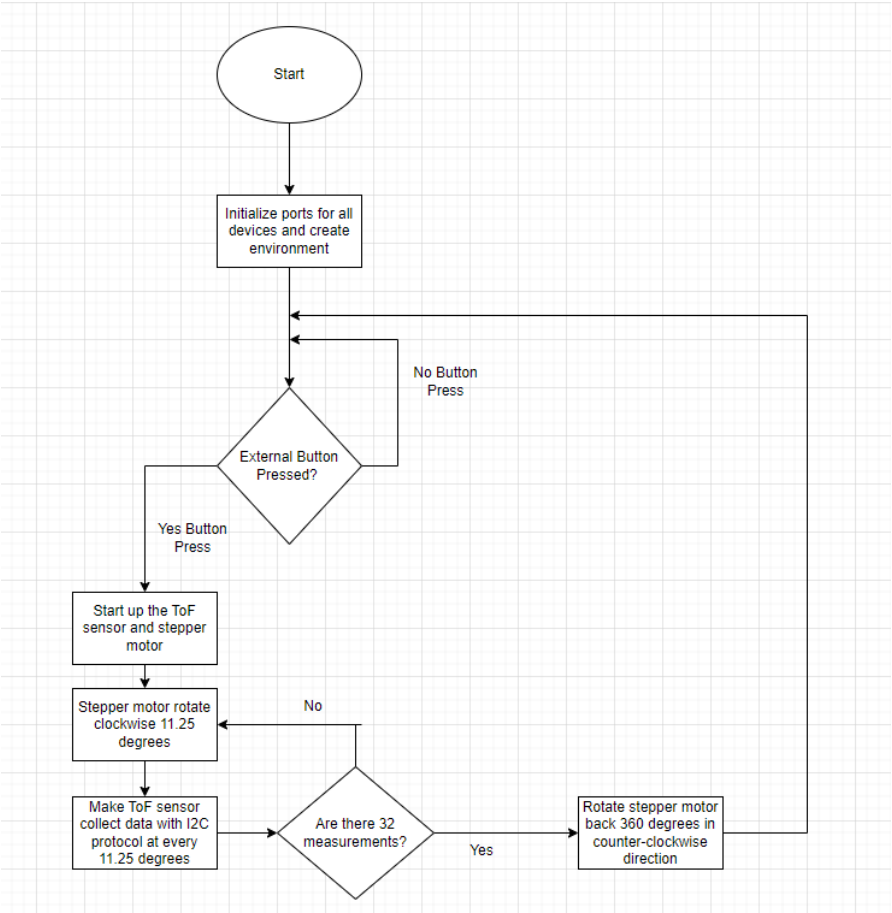

Figure 8: Flowchart for Keil Code

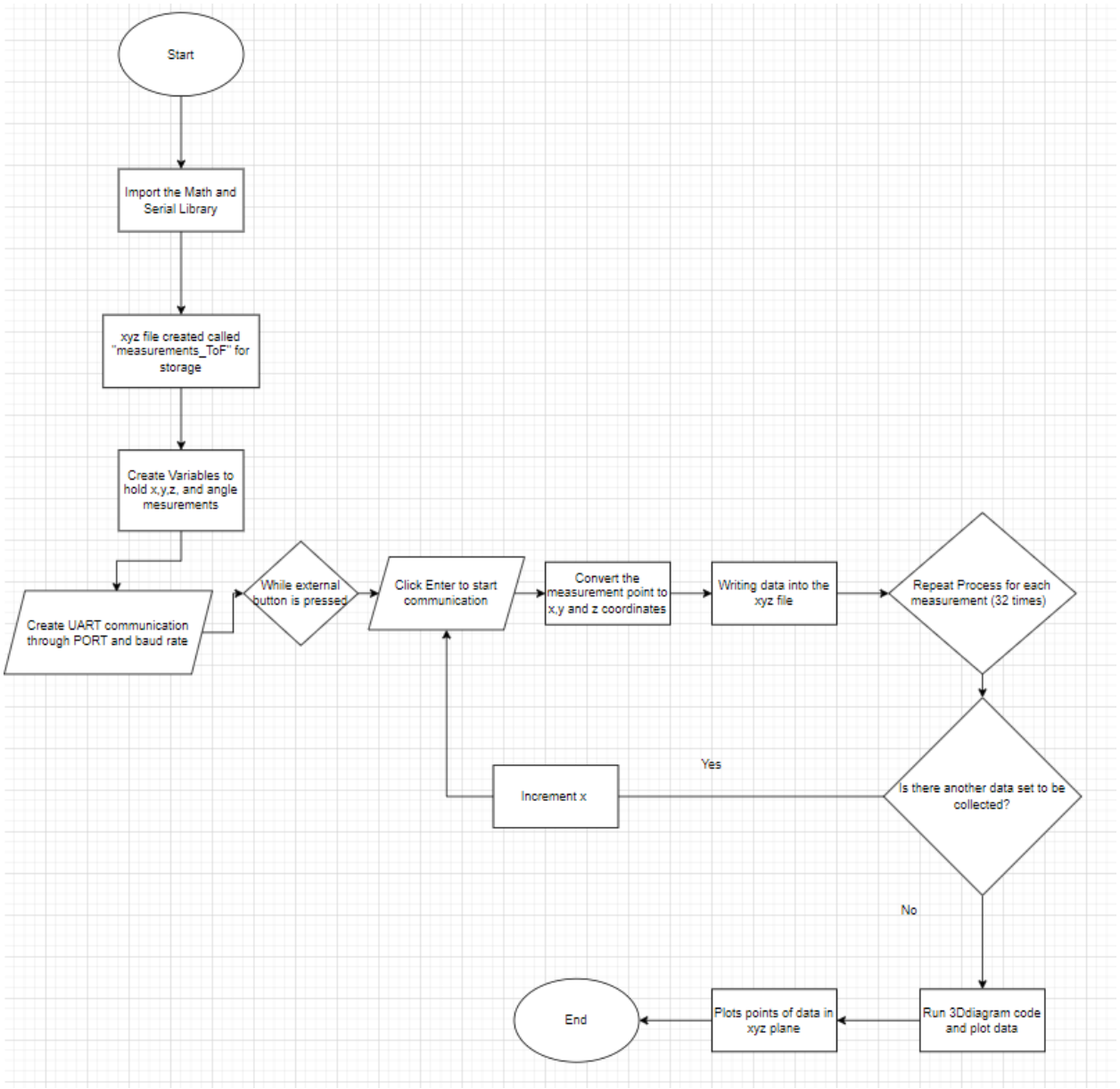

Figure 9: Flowchart for Python Code

References

[1] T. Instruments, “MSP-EXP432E401Y,” *MSP-EXP432E401Y | Buy TI Parts | TI.com*. [Online]. Available: https://www.ti.com/product/MSP-EXP432E401Y/part-details/MSP-EXP432E401Y?utm_source=google&utm_medium=cpc&utm_campaign=ocb-tistore-promo-epd_opn_en-cpc-storeic-google-wwe&utm_content=Device&ds_k=MSP-EXP432E401Y&DCM=yes&gclid=CjwKCAjw3POhBhBQEiwAqTCuBk7sHMXtVXQHjOnr2_QX1gh5kqnQt790WaFhqQNjruvLX7ZfTW7sNhoCPGwQAvD_BwE&gclsrc=aw.ds. [Accessed: 16-Apr-2023].

[2] A. Industries, “Adafruit VL53L1X time of flight distance sensor - ~30 to 4000mm,” *adafruit industries blog RSS*. [Online]. Available: https://www.adafruit.com/product/3967. [Accessed: 16-Apr-2023].

[3] Amazon, “Seiwei Stepper Motor, 28BYJ-48 Uln2003 DC 5V Stepper Motor compatible with Arduino,” *Amazon.ca: Electronics*. [Online]. Available: https://www.amazon.ca/SEIWEI-Stepper-28BYJ-48-Uln2003-Compatible/dp/B0962SPZHF/ref=sr_1_3_sspa?hvadid=232475450535&hvdev=c&hvlocphy=9000871&hvnetw=g&hvqmt=e&hvrand=7089621625355046785&hvtargid=kwd-297662253553&hydadcr=1530_9454494&keywords=28byj-48%2Bstepper%2Bmotor&qid=1681772204&sr=8-3-spons&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUExMzhMNFZPUkdURkYmZW5jcnlwdGVkSWQ9QTA4MjA0NjgyR1QyRTZVODZGWE1CJmVuY3J5cHRlZEFkSWQ9QTA3NTU2MjYyTTNWU0Y5NVExUU9FJndpZGdldE5hbWU9c3BfYXRmJmFjdGlvbj1jbGlja1JlZGlyZWN0JmRvTm90TG9nQ2xpY2s9dHJ1ZQ&th=1. [Accessed: 16-Apr-2023].

[4] Amazon, “Daoki 100-pack 6x6x5 mm miniature micro momentary tactile tact touch push button switch quality switch SPST miniature/mini/micro/small PCB,” *Amazon.ca: Tools & Home Improvement*. [Online]. Available: https://www.amazon.ca/C-J-SHOP-Miniature-Momentary-Tactile/dp/B01CGMP9GY/ref=asc_df_B01CGMP9GY/?tag=googleshopc0c-20&linkCode=df0&hvadid=335390221329&hvpos=&hvnetw=g&hvrand=8864225760548024102&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9000871&hvtargid=pla-640514760452&th=1. [Accessed: 16-Apr-2023].

[5] “Week 1 - Signals around us - COMPENG 2DX3:Microprocessor Systems Project.” https://avenue.cllmcmaster.ca/d2l/le/content/512368/viewContent/3979012/View, class notes for 2DX3, Department of Electrical and Computer Engineering, McMaster University, Winter 2023 (accessed Apr 16 2023).

[6] “Studio 0B - Week 1 - Tools 1” https://avenue.cllmcmaster.ca/d2l/le/content/512368/viewContent/4044094/View, class notes for 2DX3, Department of Electrical and Computer Engineering, McMaster University, Winter 2023 (accessed Apr 16 2023).

[7] “2DX_2023_Studio 9” https://avenue.cllmcmaster.ca/d2l/le/content/512368/viewContent/4113074/View, class notes for 2DX3, Department of Electrical and Computer Engineering, McMaster University, Winter 2023 (accessed Apr 16 2023).

[8] “Week 2 - Transduction and analog signal acquisition” https://avenue.cllmcmaster.ca/d2l/le/content/512368/viewContent/3979013/View, class notes for 2DX3, Department of Electrical and Computer Engineering, McMaster University, Winter 2023 (accessed Apr 16 2023).