2DX3 FINAL REPORT

GORDON TAM
TAMG3
400380714

2023-04-09

TABLE OF CONTENTS

DEVICE OVERVIEW

Features:

- Texas Instrument MSP432E401Y Microcontroller features:
    - Operating voltage range of 2.5-5.5 V
    - Bus/clock speed of 120 MHz
    - 1024 KB flash memory
    - 256 KB SRAM
    - 6 KB EEPROM
    - 2 12-bit SAR-based ADC modules
    - 16 digital comparators
    - C/C++ compiler
    - 8 UARTs with independent transmitter and receiver
    - 10 I2Cs with high-speed support
    - 115200 baud rate
    - Cost of approximately 45-70 CAD

- VL53L1X Time of Flight Sensor features:
    - The VL53L1X ToF Sensor operates within a voltage range of 2.6-3.5 V and uses a single power supply of 2v8.
    - It emits a 940 nm invisible laser and has a SPAD receiving array with an integrated lens.
    - The sensor can measure distances up to a maximum of 400 cm and has a ranging frequency of up to 50 Hz.
    - It uses an I2C interface that can handle up to 400 kHz.
    - The cost of the sensor ranges from 6-25 CAD.

Description:

The system utilizes the MSP432E401Y microcontroller along with the VL53L1X Time of Flight sensor and stepper motor to measure distance values along the y-z plane. The distance values then go through data manipulation to get coordinates which are plotted and visualized in 3d graphing software. The code on the microcontroller is written in C which controls the stepper motor and Time of Flight sensor via push buttons on a breadboard. Upon turning on the stepper motor and Time of Flight sensor, measurements will be taken every 11.25 degrees indefinitely until the ToF button is pressed again to stop measurements. Each measurement is transmitted to the PC through a serial UART port. A Python code will convert the data into y-z-x coordinates which are then saved and visualized using the open3D Python library.

Block diagram:

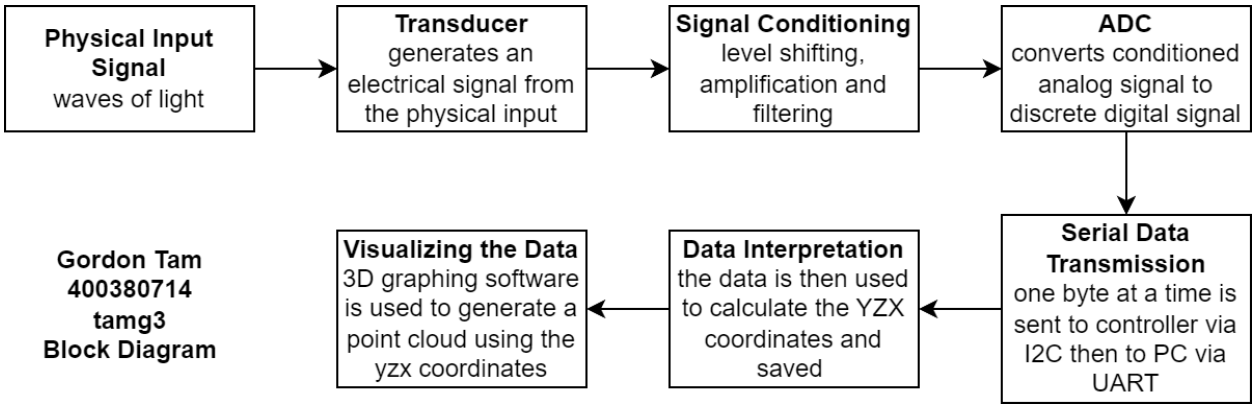

Figure 1 *block diagram*

DEVICE CHARACTERISTICS TABLE

Table 1 *characteristic table*

CHARACTERISTIC	VALUE/CORRESPONDENCE
Time of Flight Sensor (VL53L1X)	$V_{in}(3.3V)$, GND, SCL→PB2, SDA→PB3
Imported Python Libraries	Import serial, math, numpy as np, open3d as o3d
Serial Communication Speed	Baud Rate = 115200 bps
Stepper motor (2	PH0-PH3, GND, $V_{in}(5V)$
Bus Speed	24MHz
2x Active Low Push Buttons	Located on a breadboard, PM0, PM1
Required C Header Files in Keil	PLL, SysTick, UART, tm4c1294ncpdt, VL53L1X_api, stdint, onboardLEDs

DETAILED DESCRIPTION

Distance Measurement:

Upon building the circuit as shown under the circuit schematic section, we can begin the process. Once the code is loaded onto the microcontroller and reset, the system will initialize Port H for the stepper motor, port M for the pushbuttons, port B for the Time of flight sensor, port N for the onboard LED, PLL, SysTick, I2C and UART. The program will stay idle until the user activates the ToF sensor or stepper motor via the active low bush button on the breadboard. This is due to the implementation of a polling system in the program which is constantly checking if either of the two buttons are pushed. The ToF sensor has a 940 nm laser that is transmitted as a wavelength of light from the emitter into the environment which will then reflect off the nearest surface and return to the receiver part of the sensor. The sensor can then easily compute the distance by multiplying the travel time by the speed of light and dividing the result by two (represented by the formula below).

$$Distance = \frac{speed\ of\ light \times \Delta T}{2}$$

After each measurement is done, an LED on the microcontroller is flashed to indicate that the ToF has taken a measurement. The distance measurement is saved to a variable of type float and then printed to UART via the agreed baud rate of 115200 bits per second for the Python program to read from. At the same time, the stepper motor rotates 11.5 degrees before repeating the process of taking measurements. The program tells the stepper motor to move by sending logic 1 to the correct port H pins. The stepper motor has a total of 512 individual steps and represents a full 360 degrees. Instead of taking a measurement of every single step which would take too long, 32 measurements were utilized which means each step size would be calculated by:

$$degrees = \frac{360}{32} = 11.25^{\circ},\ step\ size = \frac{512}{32} = 16\ steps$$

Once a full rotation or 360 degrees is reached, the stepper will swap directions from clockwise to counterclockwise or vice versa to avoid the wires becoming tangled. Unfortunately, this method only produces coordinates in the y-z plane, thus the user needs to physically move forward to get the coordinates of the next plane after each full rotation.

Visualization:

The entire visualization process is done using Python. To begin we import pyserial, math, numpy as np and open3d as o3d. Next, the program creates variables for serial port COM6, containing a baud rate of 115200 bps and a timeout of 5 seconds. Following this UART ports are reset to delete remaining data in the buffers. The first of 2 functions called 'read_sensor_data()' is responsible for reading the distance measurement from the ToF sensor. If ToF is turned off then the function returns None, otherwise we store the distance value as a float inside the 'distance' variable and return it. Moving on, the second function called 'convert_to_coordinates(distance)' checks if the input is 'None' which will then return 'None, None' (will be explained later). If a valid distance is entered the function will calculate the y and z coordinates using the following formula:

y = distance * math.sin(math.radians(angle))
z = distance * math.cos(math.radians(angle))

The calculations are also based on the angle which is initialized as 0 and will be updated later in the code. Moreover, the angle is turned into degrees because that is how the math library does trigonometric calculations. Next we create a new .xyz file to write the data points to and a list called 'planes' that contains sublists containing data points for each plane ultimately storing each plane in a sublist. Under an infinite while loop, 'convert_to_coordinates(read_sensor_data())' is called to get the 'y' and 'z' coordinates which are added to the .xyz file. The program then checks if the angle reaches 360 or 0 degrees and changes the direction to match the rotation of the stepper motor. Furthermore, every full rotation increments 'x' which represents moving forward. The main loop will end when 'y' and 'z' equal 'None'. After the while loop concludes the program beings to fabricate the linset for the point cloud. The first for loop connects the lines of each individual plane forming a closed plane while the second for loop connects each plane together. To connect the planes together, the program looks at a point on one plane and finds the closest point on the next plane. An algorithm is used to find the straight-line distance between these two points in a Euclidean space. In a three-dimensional space the Euclidean distance between two points $A(x_1, y_1, z_1)$ and $B(x_2, y_2, z_2)$ is calculated using the following formula:

$$distance(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Finally, open3d combines the data points from the .xyz file, lines that close the planes and lines that connect the planes together and visualizes it as the final result before the program ends the closes 'COM6'.

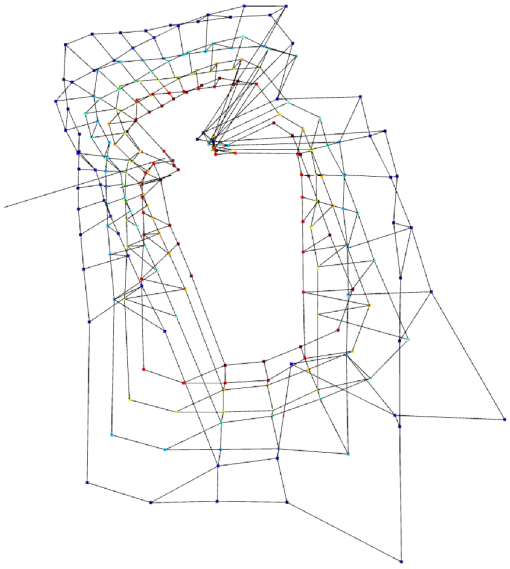

Figure 2 *3D point cloud of hallway*

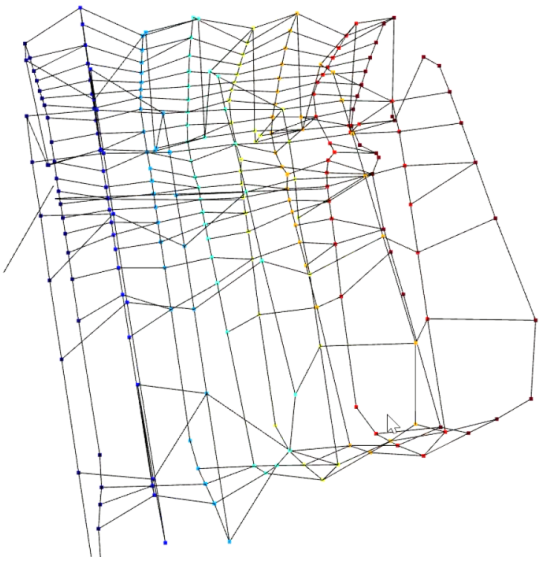

Figure 3 *Side view of 3D point cloud*

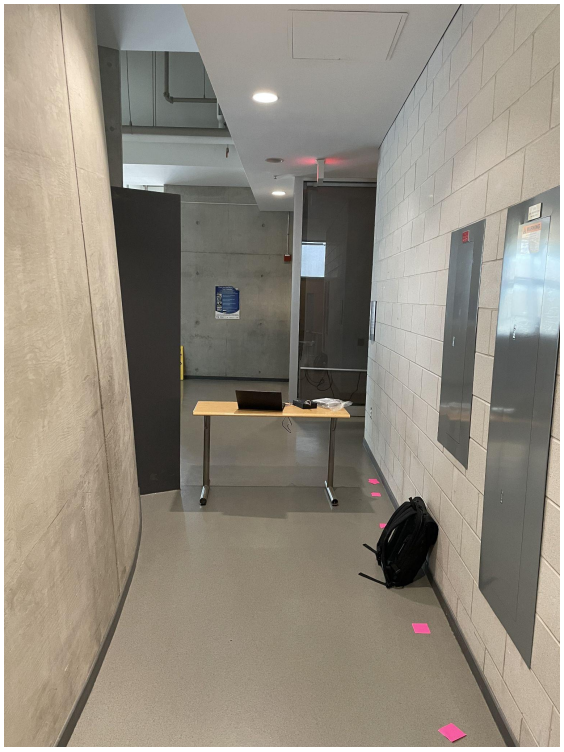

Figure 4 *Picture of hallway*

APPLICATION EXAMPLE, INSTRUCTIONS, AND EXPECTED OUTPUT

Assumptions: for the example and instructions, it is assumed that Python version 3.10.10 is installed along with pyserial and open3d.

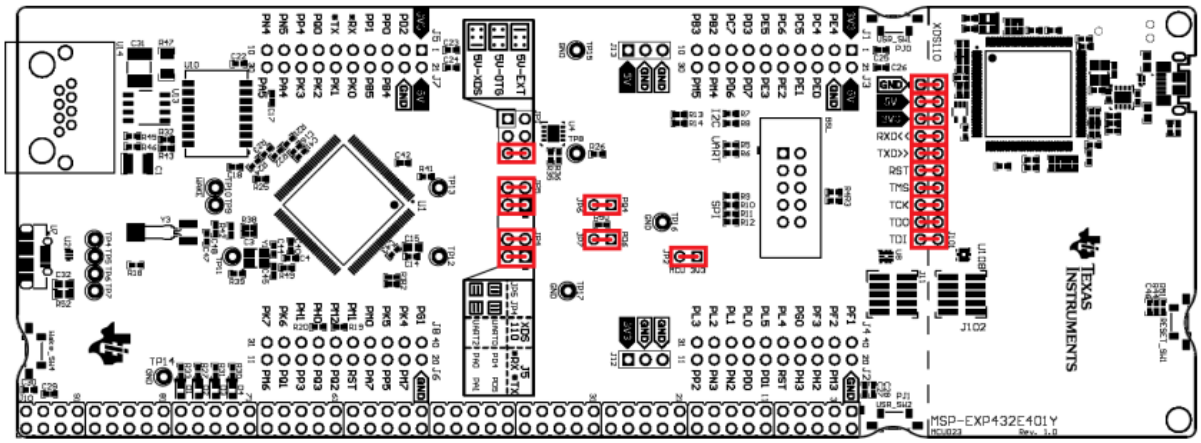

Figure 5 *MSP432E401Y Microcontroller*

1. Build the circuit schematic from ***Figure 6*** (connect MSP432E401Y, VL531X, and stepper motor) the user will also need to find a way to secure the time of flight sensor to the stepper motor.
2. Open the C code file in Keil, click translate, build and load onto the microcontroller (make sure to press the reset button after).
3. The user must make sure UART is set up correctly before continuing. To do this, open the device manager, under the ports tab should be 'XDS110 Class Application/User UART(COM#)' (# is different for every device). Now on line 6 in the Python program change 'COM6' to 'COM#'. Also, make sure that line 6 uses a baud rate of 115200 bps and a timeout of 5.
4. The user may now run the Python program with their IDE of choice. The console will output 'Press Enter to start communication…' which will start the program and save measurements after the 'Enter' key is pressed. Before pressing the key it is better to turn on the stepper motor first
5. Once the 'Enter key is pressed and the stepper motor is running, the user will have a 5-second window to start receiving measurements otherwise, the program will end so try to turn on the ToF sensor as soon as possible after starting the program.
6. When the ToF is turned on, no matter the current position, the stepper will move to 0 degrees before starting the first measurement. The user can tell that the ToF sensor has started taking measurements by observing the stepper moving smoothly to pulsing every measurement.
7. The user is required to move forward one human step every time the stepper motor completes a full rotation of 360 degrees and continues so until the user is satisfied with the number of data points taken.
8. When the user is finished taking measurements, press and hold the button for the ToF sensor for approximately one second before observing the stepper start to move smoothly without stopping. After 5 seconds an open3D visualization window will open containing the point cloud of the scan

LIMITATIONS

Quantization Error and baud rate

The MSP432E401Y microcontroller is equipped with a 32-bit Arm Cortex-M4F processor core that has a floating-point unit (FPU), allowing it to perform floating-point calculations within the 32-bit limit. The maximum quantization error for the system can be calculated using the formula:

$$Max\ Quantization\ Error\ = \frac{Max\ Reading}{2^{\#\ of\ ADC\ bits}}$$

Since the maximum working distance of the ToF is approximately 4 meters, the maximum quantization error can be calculated as:

$$Max\ Quantization\ Error\ = \frac{4000mm}{2^{16}} = 0.061035$$

The maximum standard serial communication speed that can be used with the PC is 115200 bps. Any attempt to increase the baud rate beyond this value will result in an error because the microcontroller can only handle 115200 bps. Moreover, the baud rate is how the devices communicate with each other, so if it were to change for one of the devices the system will no longer work. The communication method used between the microcontroller and ToF sensor is I2C, which utilizes a data line and a clock signal line. The serial data line (SDA) is used to transmit data, while the serial clock line (SCL) synchronizes the data transfer. The speed of communication between these two devices is equal to the ToF's maximum transmission speed of 400 kbps. The primary limitation of the system's speed is the baud rate of the UART transmissions between the microcontroller and the PC. The system is capped at 115200 bps, which is significantly lower than the communication speed between the ToF and microcontroller.

Stepper Motor and ToF sensor

After prolonged use the stepper motor suffers from overheating issues, thus the user may need to consider a cool down in between each complete run. There is also a 5% error in the accuracy of the steps which may affect the accuracy of the angle accuracy.

The ToF sensor is unable to get proper readings from reflective surfaces or glass since the reflections are not received by the sensor resulting in outlier data points. Other sources of light may cause unpredictable measurements due to the noise from the environment entering the receiver.

CIRCUIT SCHEMATIC

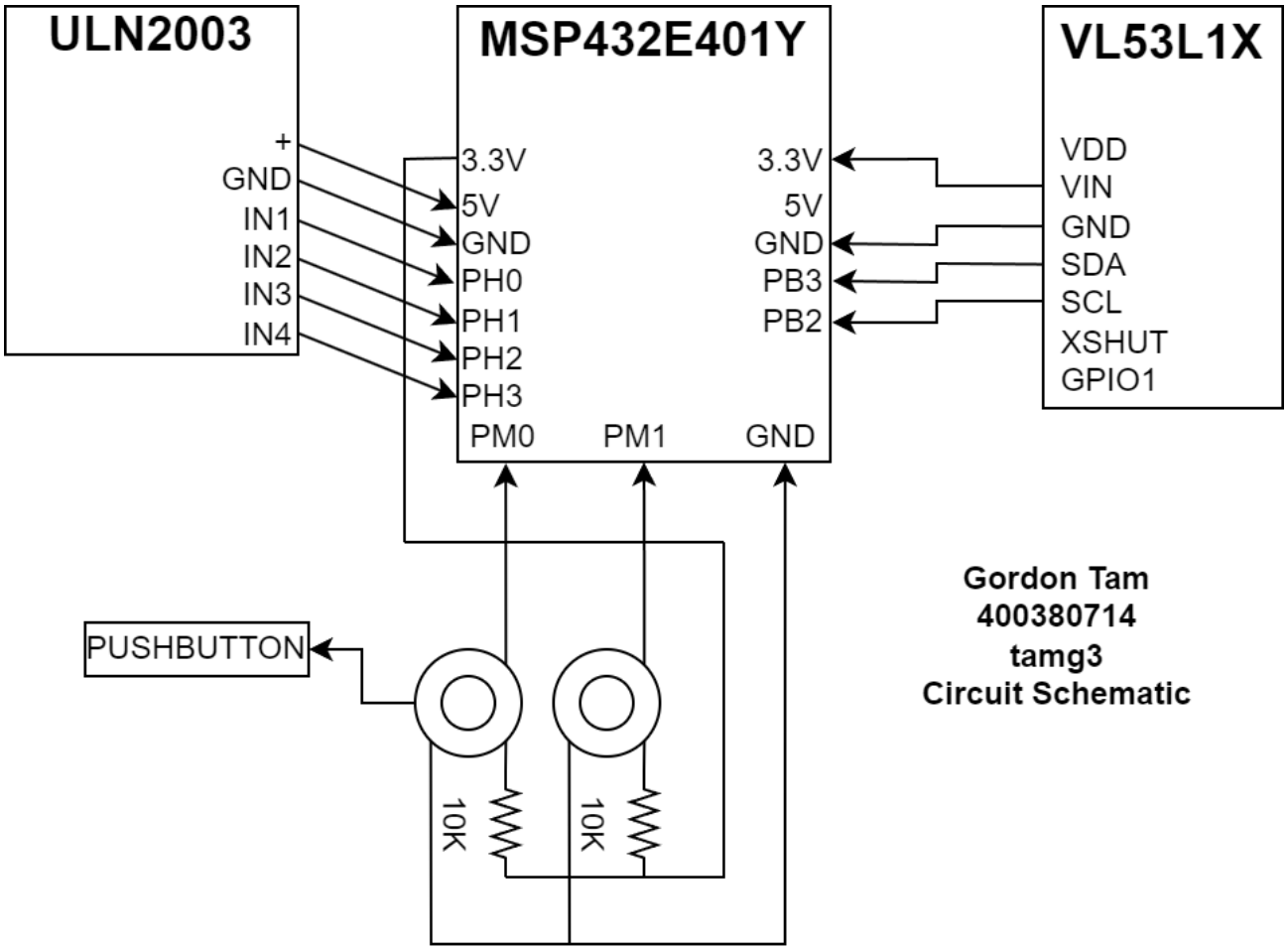

Figure 6 *Circuit schematic*

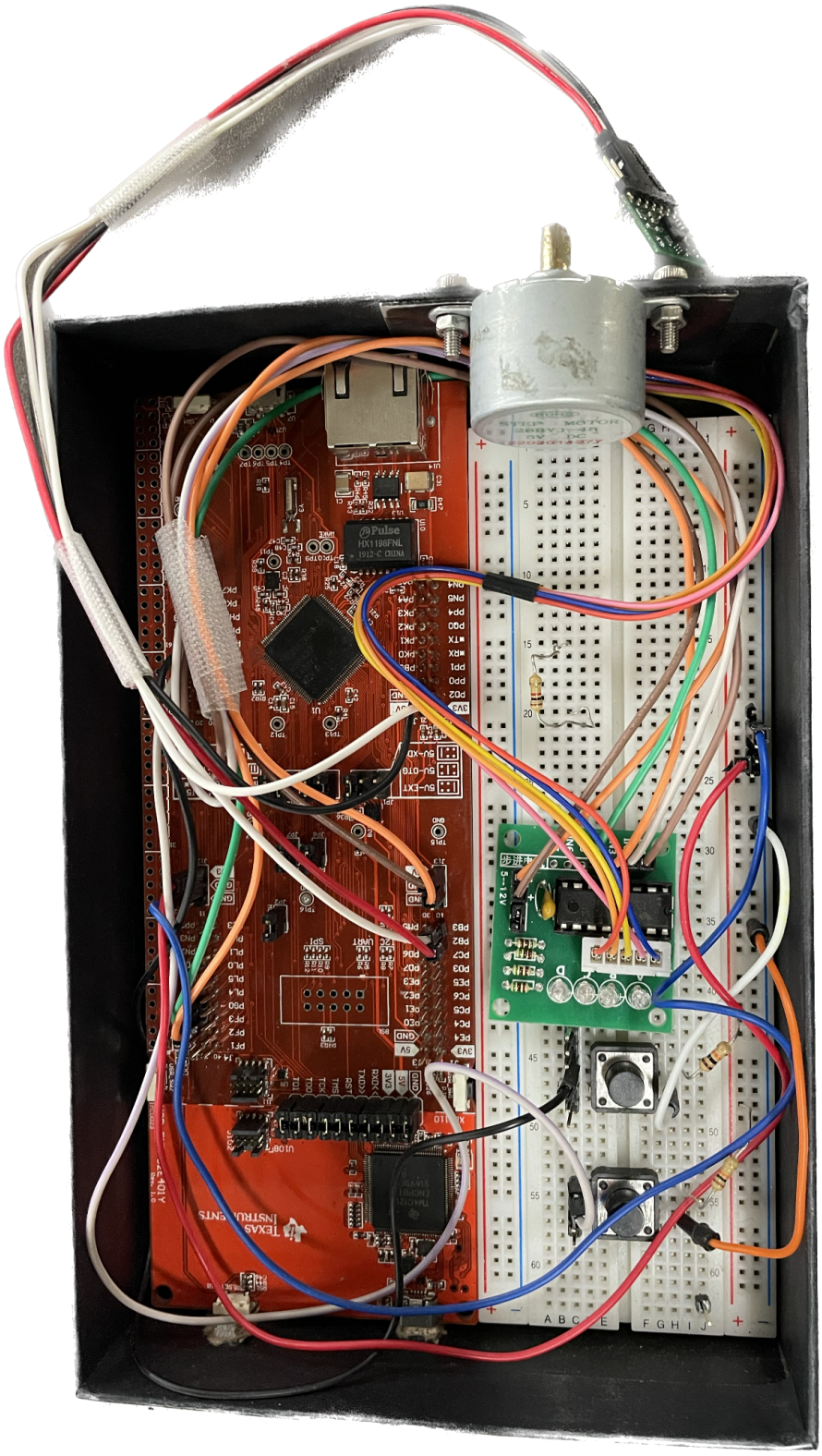

Figure 7 *Physical Circuit*

PROGRAMMING LOGIC FLOWCHART

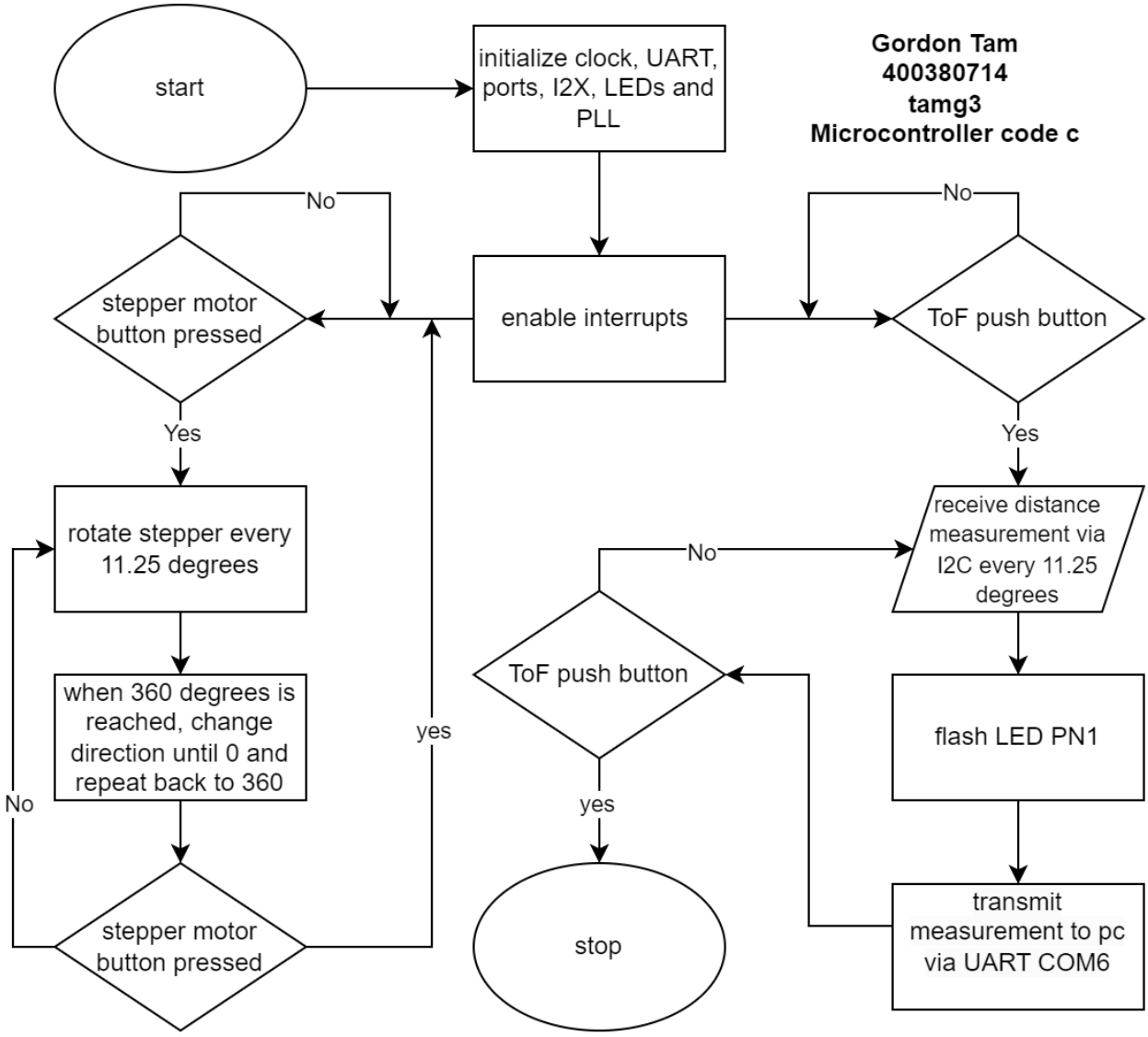

Figure 8 *Microcontroller code flowchart*

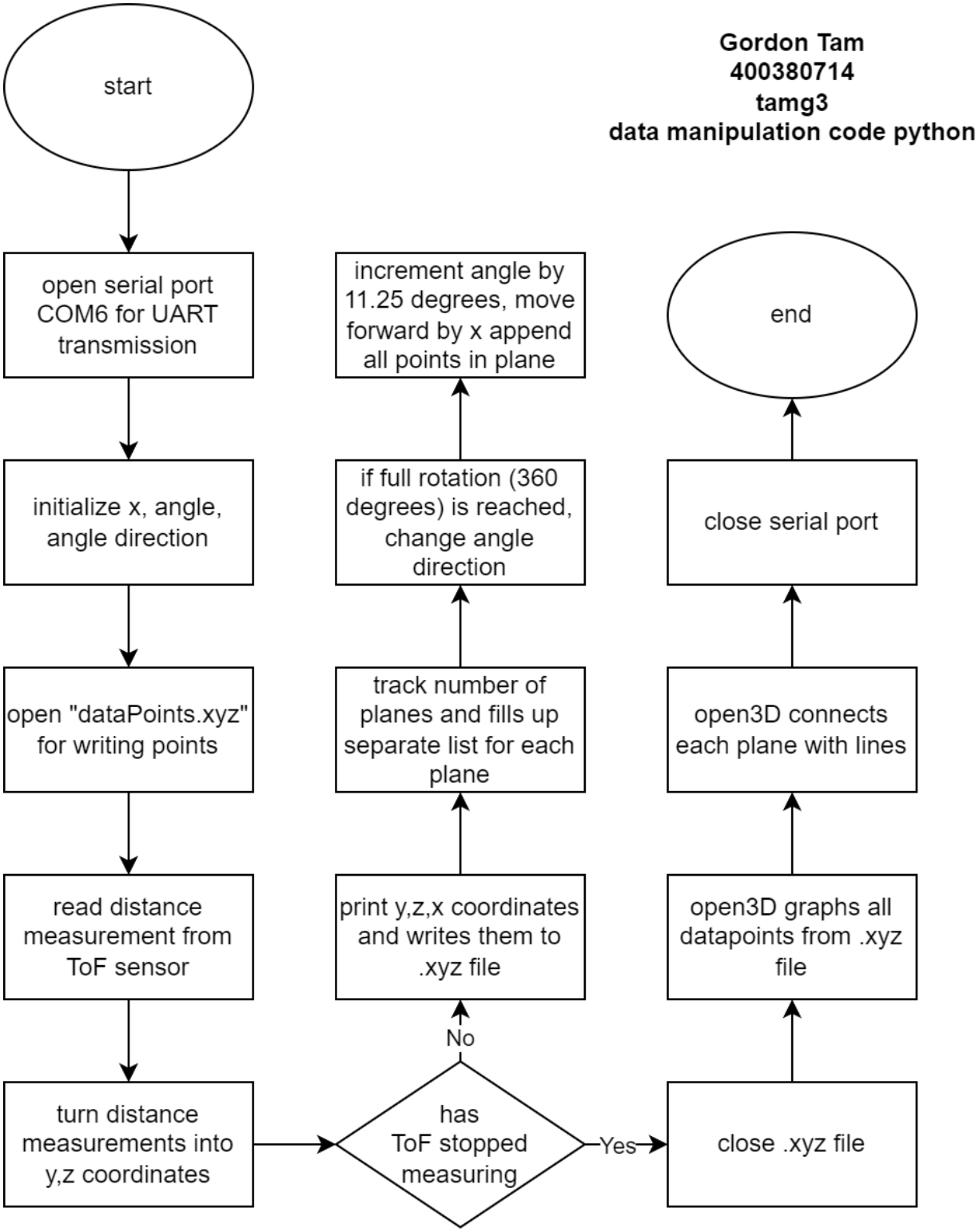

Figure 9 *Data manipulation flowchart in python*

CITATIONS

[1] “MSP432E401Y,” *MSP432E401Y data sheet, product information and support | TI.com*. [Online]. Available: https://www.ti.com/product/MSP432E401Y?utm_source=google&utm_medium=cpc&utm_campaign=epd-msp-null-prodfolderdynamic-cpc-pf-google-wwe_int&utm_content=prodfolddynamic&ds_k=DYNAMIC%2BSEARCH%2BADS&DCM=yes&gclid=Cj0KCQjwlumhBhClARIsABO6p-yfQcsZGTWHygTQoe1nGcOj56kL0uumN01qgiq44u9by5T2f5Kfan4aAiD_EALw_wcB&gclsrc=aw.ds. [Accessed: 15-Apr-2023].

[2] “VL53L1X datasheet - home - stmicroelectronics.” [Online]. Available: https://www.st.com/resource/en/datasheet/vl53l1x.pdf. [Accessed: 15-Apr-2023].

[3] “Studio 7C - Week 7 - UART and I2C.” [Online]. Available: https://avenue.cllmcmaster.ca/d2l/le/content/512368/viewContent/4091720/View. [Accessed: 15-Apr-2023].