2DX3: Microprocessor Systems
Final Project

Instructor: Drs. Boursalie, Doyle, Haddara
Lab TAs: Han Zhou (zhouh115), Junran Chen (chenj425),
Danial Sadeghpour (sadeghpd)

Marcus Cohoon – cohoom1- 400297985 - 2DX3 - Friday Afternoon - L05

Table of Contents

Device Overview

Features

Texas Instrument MSP432E401Y Microcontroller:

- Bus/Clock Speed is 120 MHz
- 2.5-5.5 V Operating Voltage Range
- 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 each with independent transmitter and receiver
- 10 I2Cs with high-speed support
- 115200 Baud rate
- Cost of approximately 45-70 CAD

VL53L1X ToF Sensor:

- 2.6-3.5 V Operating Voltage Range
- Single Power Supply (2v8)
- Emitter: 940 nm invisible laser
- SPAD receiving array with integrated lens
- Maximum of 400 cm distance measurement
- Maximum of 50 Hz ranging frequency
- I2C Interface for up to 400 kHz
- Cost of 6-25 CAD

General Description

The intelligent system utilizes the MSP432E401Y microcontroller and the VL53L1X Time-of-Flight (ToF) sensor with the help of a stepper motor to take distance measurements of its immediate surroundings, to be latter interpreted into coordinates and mapped utilizing 3D graphing software. The microcontroller was coded in C to rotate the sensor which is attached to the unipolar stepper motor after the push of an on-board button. The micro accepts distance measurements from the ToF via I2C every 22.5 degrees and stops after a full rotation is completed. This data is then sent to the PC via the serial UART port. The PC receives and manipulates this distance data with a python program that turns the data into xyz coordinates. This data is then saved to a '.xyz' file to later be displayed graphically. The user can then move the pre-decided distance in x-direction and press the onboard button once again to repeat this process until the desired number of planes has been reached. After the desired number of data points has been reached the program will utilize Open3D a python add-on to create the point cloud and then also connect lines between the points for a more easily interpreted 3D visual of the scanned environment.

Block Diagram

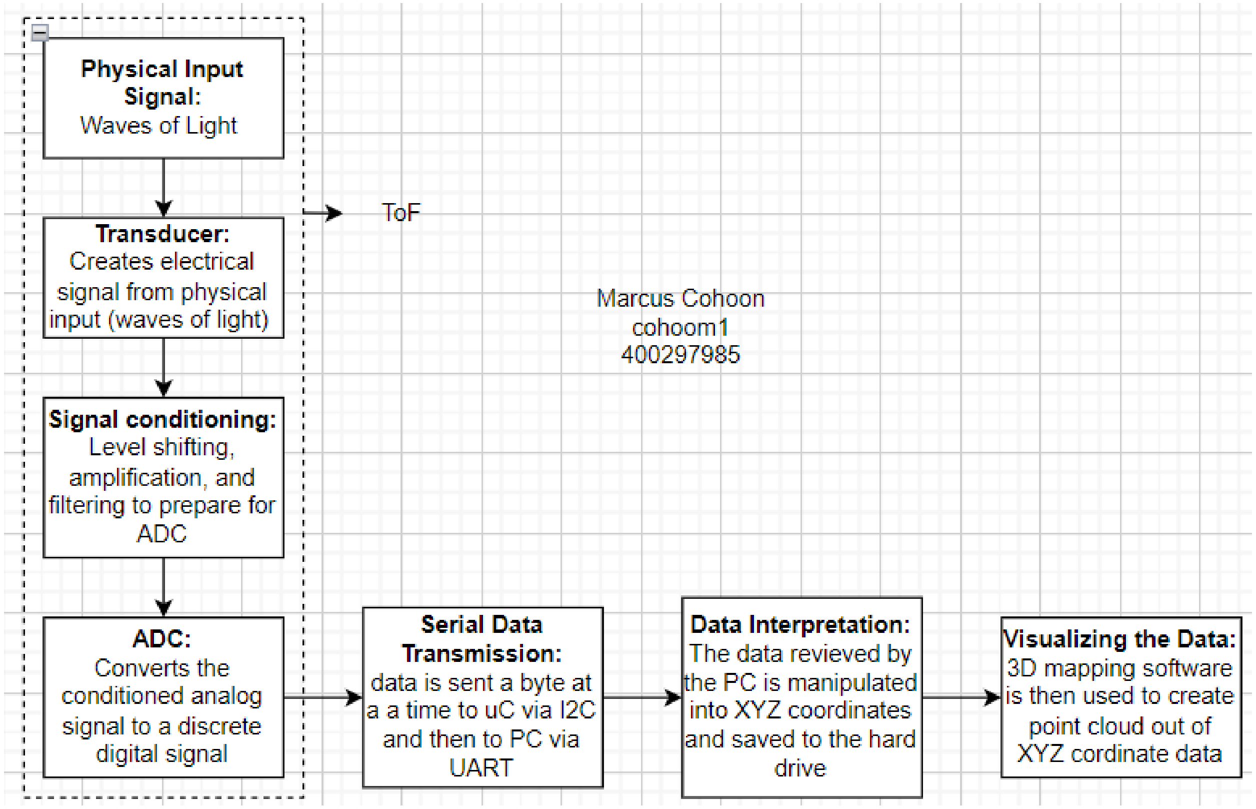
Physical Input Signal: Waves of Light
Transducer: Creates electrical signal from physical input (waves of light)
Signal conditioning: Level shifting, amplification, and filtering to prepare for ADC
ADC: Converts the conditioned analog signal to a discrete digital signal
ToF
Marcus Cohoon
cohoom1
400297985
Serial Data Transmission: data is sent a byte at a a time to uC via I2C and then to PC via UART
Data Interpretation: The data reviewed by the PC is manipulated into XYZ coordinates and saved to the hard drive
Visualizing the Data: 3D mapping software is then used to create point cloud out of XYZ cordinate data

Fig. 1. Block Diagram of Main System Processes

Device Characteristics Table

Table 1

System Characteristics

Characteristic	Value/Correspondence
Time of Flight Sensor	Vin (3.3 V), GND, SCL (Port B2), SDA (Port B3)
On-Board Push Button	Port J1 (Configured with Interupts)
Serial Communication Speed	115200 Baud rate (bps)
Bus Speed	48 MHz
Stepper Motor	Port H0:H3, Vin (5 V), GND
Special Python Libraries	Import math, serial, numpy as np, open3d as o3d
Required uC header files	PLL, SysTick, tm4c1294ncpdt, vl53l1x_api, uart, stdint

Detailed Description

Distance Measurement

After the circuit is correctly set-up as seen in the circuit schematic shown in Fig. 2 in the Circuit Schematic section, the system is ready to begin its work flow. The system begins the distance measurement process by first initializing and configuring all required ports including Port H for outputting signals to the stepper motor, Port N for LED indicator light, Port B for I2C, and Port J for on-board push button. After configuring the required values for each individual port interrupts are enabled as the system utilizes interrupts for Port J to sense the button press. All required steps for configuring interrupts are taken including as previously said arming the source of interrupt, enabling the interrupt in the NVIC, enabling global interrupts, setting the priority level, writing an appropriate interrupt service routine (ISR). Now the system is ready to begin the main body of the distance measurement process. This Portion of the system includes the first three sections of Fig. 6. Flowchart in the "Programming Logic Flowchart" section.

After this initialization is complete the system waits for the interrupt created from the push of the PJ1 on board button. When a button press occurs the system boots the VL53L1X ToF sensor and configures it with the default settings. The ToF is now ready to obtain measurements. To do this it utilize it's 940 nm invisible laser transmitter to send a wavelength of light into it's environment which will then reflect off the closest object and eventually return to the ToF being observed by the receiver. The chip on the ToF sensor's board then can determine the distance between itself and this object by doing some simple physics calculations with the time it just measured for this signal to return and the known value for the speed of light. These equations are as follows:

Speed of Light (in a vacuum) = 299792458 m/s

Measured Distance = (Photon Travel Time/2) x Speed of Light x 10^{-3} mm

The time it takes the photons to travel to the nearest object and return to the receiver as we only want the displacement between the two objects not the total distance traveled by the photon on its journey. This value was then multiplied by 10 to the power of -3 because the sensor outputs the distance measurement to the microcontroller via I2C in milli-meters as previously stated.

Returning to programming flow, once the ToF sensor data is ready, the distance measure in milli-meters is obtained from the sensor and saved to a long integer variable to latter be sent to the PC via UART. The system then flashes the on-board LED PN1 to signal a distance measurement has been observed. The system then checks that the PC is ready to accept the data via the UART port and then sends the data over at the specified baud rate. On the PC, a python program is running which receives and stores the data on the hard drive for latter use. This portion of the process will be further explained in the visualization sub-section below. The micro then calls a function to rotate the uni-polar stepper 32 steps or 22.5 degrees. It does this by utilizing the full step method and activating the proper coils in the stepper motor by outputting HI the correct pairings of Port H pins in order. After the rotation is complete the micro signals to the sensor and obtains a new distance measurement utilizing the same process outlined above. Once a full rotation is complete and 16 distance measurements have been taken and transmitted to the PC, the motor does a full rotation in the opposite direction to avoid wire entanglement. Now the system waits for the user to move forward the specified increment. The system was designed so that this forward movement is in the positive x direction. The user can now press the push button again and the entire above process will be repeated. The user can continue doing this until the intended number of cross sections have been obtained.

Visualization

The data visualization portion of the process is carried out by the PC using a python program. This process begins by importing the required additional libraries including serial, math, open3d as o3d, and finally numpy as np. The program then creates the variable s representing the serial port ‘COM6’ configured with a baud rate of 115200 bps, equal to that of the micro’s. This port is then opened and cleared of all potential data remaining. A ‘.xyz’ file is then opened and configured for writing, if the file exists it will be overwritten and if it does not exist it will be created. Variables for the number of steps the motor has taken and the initial displacement in the X (forward) direction are initialized. The program then waits for the user to input the number of full scans they intend to take. After an integer value has been obtained from the user the system is ready to begin receiving data from the micro and it signals this by sending the letter s to it via the open serial port. The program then reads the data coming from the micro, decodes it and drops the return and newline characters that were also sent. To ensure the data being written to the coordinate file is indeed numerical, a simple check is made utilizing the predefined ‘.isdigit()’ method. The angle of rotation the motor is at is then calculated by dividing the total number of steps taken by the stepper motor by the number of steps required to make a full 360 degree rotation and multiplying this value by 2π. The vertical y coordinate and horizontal z coordinate can then be calculated by multiplying the transmitted distance measurement by the sin and cos of the angle respectively. These formulas can be seen below:

Angle = (Number of Steps/Steps in 360 deg Rotation) x 2π

Y = Distance x sin(angle)

Z = Distance x cos(angle)

All three axis values have now been obtained and the program is ready to write them to the open ‘.xyz’ file. After writing the program checks to see if the motor has completed a full rotation and if it has it resets the step count to zero, increments the X value by the pre-set amount, and increments the number of scans taken by 1. This process continues until the number of scans the user said they would take is reached at which point the file closed saving the data.

From here the system is ready to create the visual representation of the scanned surroundings with the help of the open3d add on. The data points contained in the ‘.xyz’ file are then turned into a point cloud with ease after reading the data from the file. This point cloud is then displaced on the PC monitor and can be manipulated and investigated further by dragging the cursor around the window. Lines are then added to connect the vertices and various cross-sections for a better visualization of the scanned environment. This is done by first numbering each of the collected data points. After numbering the data points, the vertices of each slice are connected to one another, to create a clearer 360-degree cross-section. Finally, each cross-section connected by connecting the vertices at the same angle value together. The final product is then displayed to the user once they close the point cloud window. The final results of the test hallway scan can be seen bellow in figures 2 and 3.

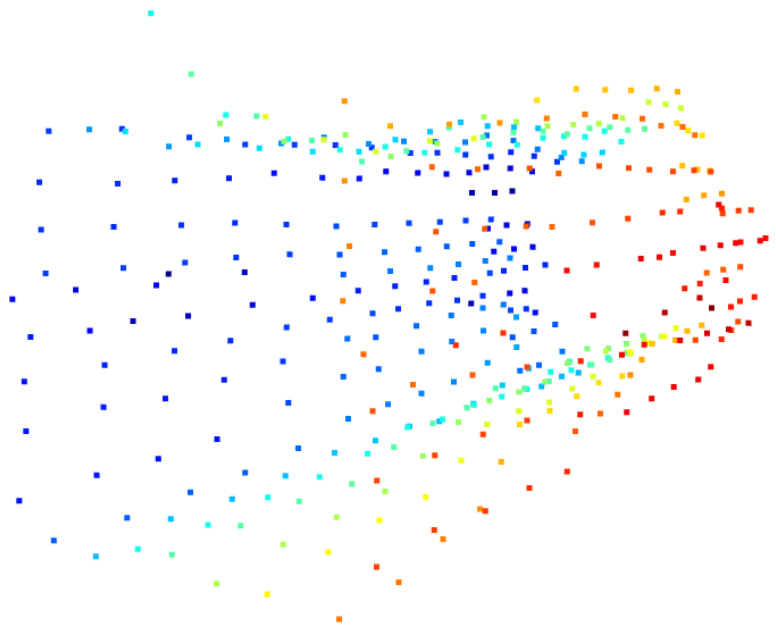

Fig. 2. Point Cloud of Scanned Empty Hallway

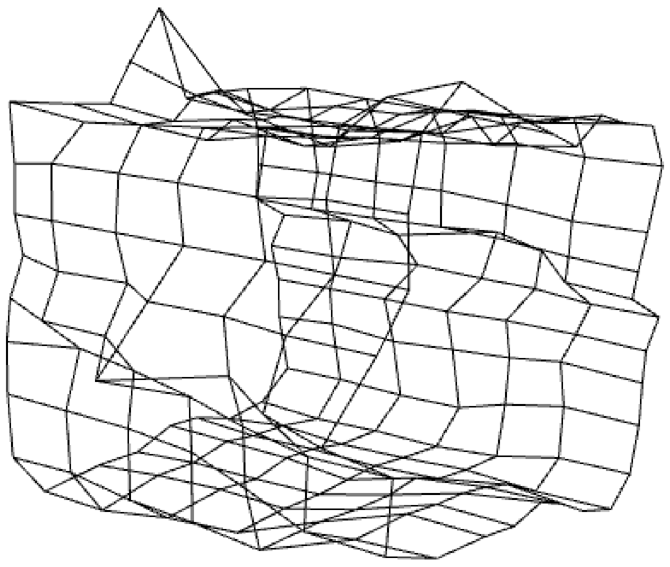

Fig. 3. Final 3D Graphic with Points from Cloud Connected

Users Guide

1. Connect the system hardware including the MSP432E401Y, VL531X, and stepper motor as outlined in the circuit schematic shown in Fig. 2. The user will also need to secure the motor and the sensor to the motor, see prototype picture in Fig. 5 for inspiration. Please note that the ToF sensor is oriented facing downwards parallel to the ground.
2. The C code is then flashed to the microcontroller utilizing an IDE such as Keil, which was the IDE used during the system design. Keil translates, builds, and then flashs the C code to the Microcontroller.
3. The user then loads the code onto the Texas Instrument MSP432E401Y by pressing the on-board reset button.
4. To set up the UART correctly the user must then check their device manger and search for the COM port using UART. This can be done by going to device manager > Ports (COM & LPT)> XDS110 Class Application/User UART (COM#). The number value in the # position is what is required. Line 8 of the python code should then be modified to have this number instead of the current value 6.
5. Ensure the baud rate is also set to the designed 115200 bps. This can be confirmed in line 8 of the python code.
6. The user should now run the python program utilizing IDLE or their IDE of choice and answer the question in the console “How many full scans are you going to take?” with an integer value.
7. The system is now ready for scanning and the user may go to the beginning of the area they wish to scan and press the on-board PJ1 push button to begin scanning.
8. After the full rotation is completed and the motor has completely stopped moving and is in it’s initial rest position the user may move forward the pre-set amount in the forward direction (positive X direction).
9. The User may now repeat steps 6 and 7 until the desired number of scans has been reached. At which point the program will open the point cloud. The view point of the point cloud can then be manipulated by clicking and dragging the cursor in the window.
10. The user may then close this window which will cause the final 3D interpretation to appear with the connecting lines. The view point of this image may also be manipulated in the same way.

Limitations

1. The MSP432E401Y microcontroller has a 32-bit Arm Cortex-M4F processor core which includes a floating-point unit or FPU. Therefore, floating point calculations will be limited to the available 32- bits.
2. Maximum Quantization Error for the System:

   Resolution = $V_{FS}/2^m$
   V_{FS} = Full Scale Voltage
   M = number of bits

   Texas Instruments MSP432E401Y microcontroller:

V_{FS} = 5 V
m = 12 bits
Resolution = 5/2^12 = 1.22 mV

VL53L1X Time-of-Flight:

V_{FS} = 3.3 V
M = 8 bits
Resolution = 3.3/2^8 = 12.89 mV

3. The maximum standard serial communication that can be implemented with the PC is 115200 bps. This was confirmed by attempting to increase the baud rate to 230400 bps and an error occurred upon attempting to open the UART port. This occurred because the micro is only able to handle 115200 bps.
4. The communication method utilized between the micro and ToF sensor was I2C meaning there was both a data line and clock signal line. There is the serial data line (SDA) for the transmission of data and the serial clock line (SCL) to synchronize the data transfer. The speed between these two devices is equal to that of the ToFs maximum transmission speed of 400 kbps.
5. The primary limitation on the speed of the system is the baud rate of the UART tansmissions between the micro and the PC. This caps the system to 115200 bps which is significantly less then the 400 kbps communication speed between the ToF and micro.

Circuit Schematic

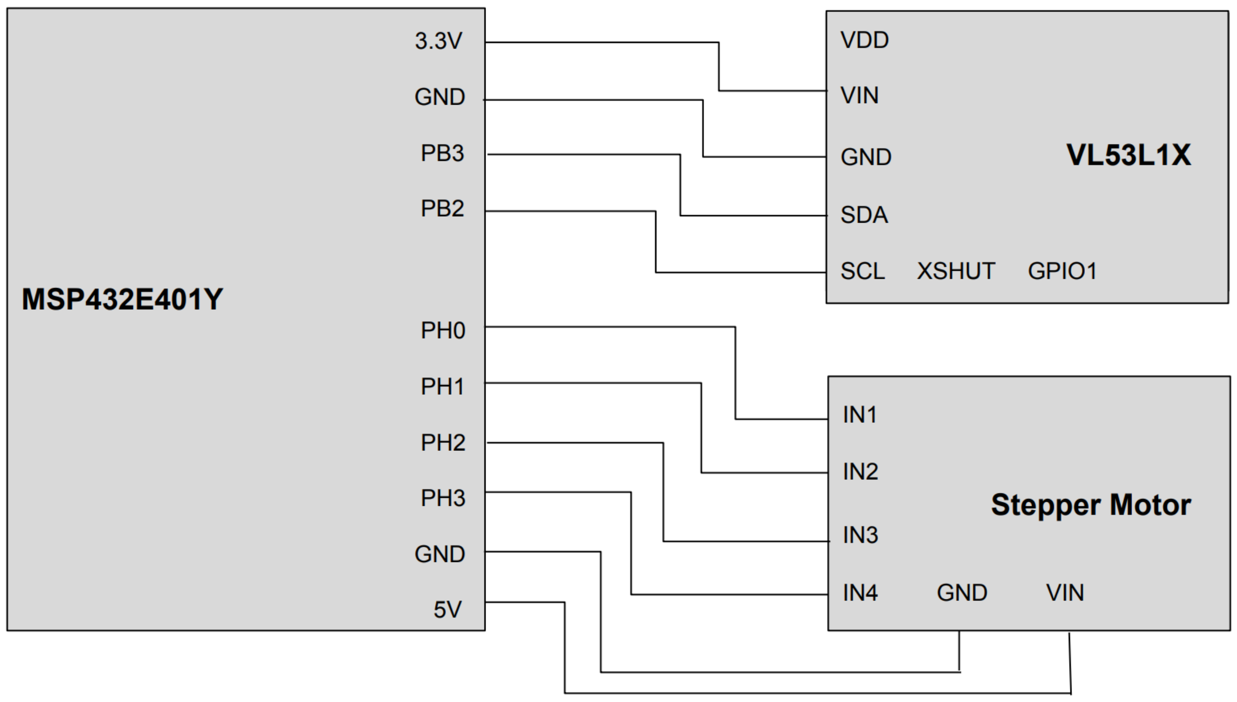

Fig. 4. Full Circuit Schematic

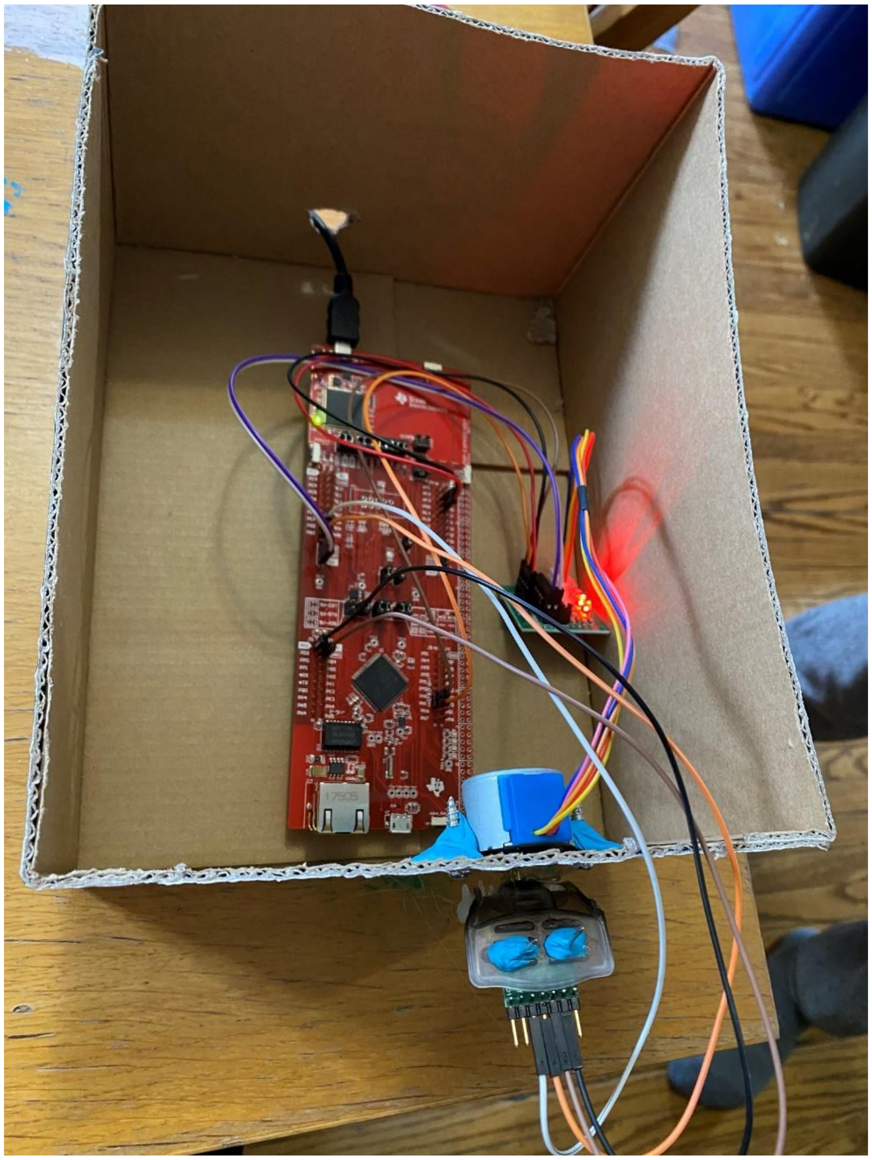

Fig. 5. Prototype System Setup

Programming Logic Flowchart

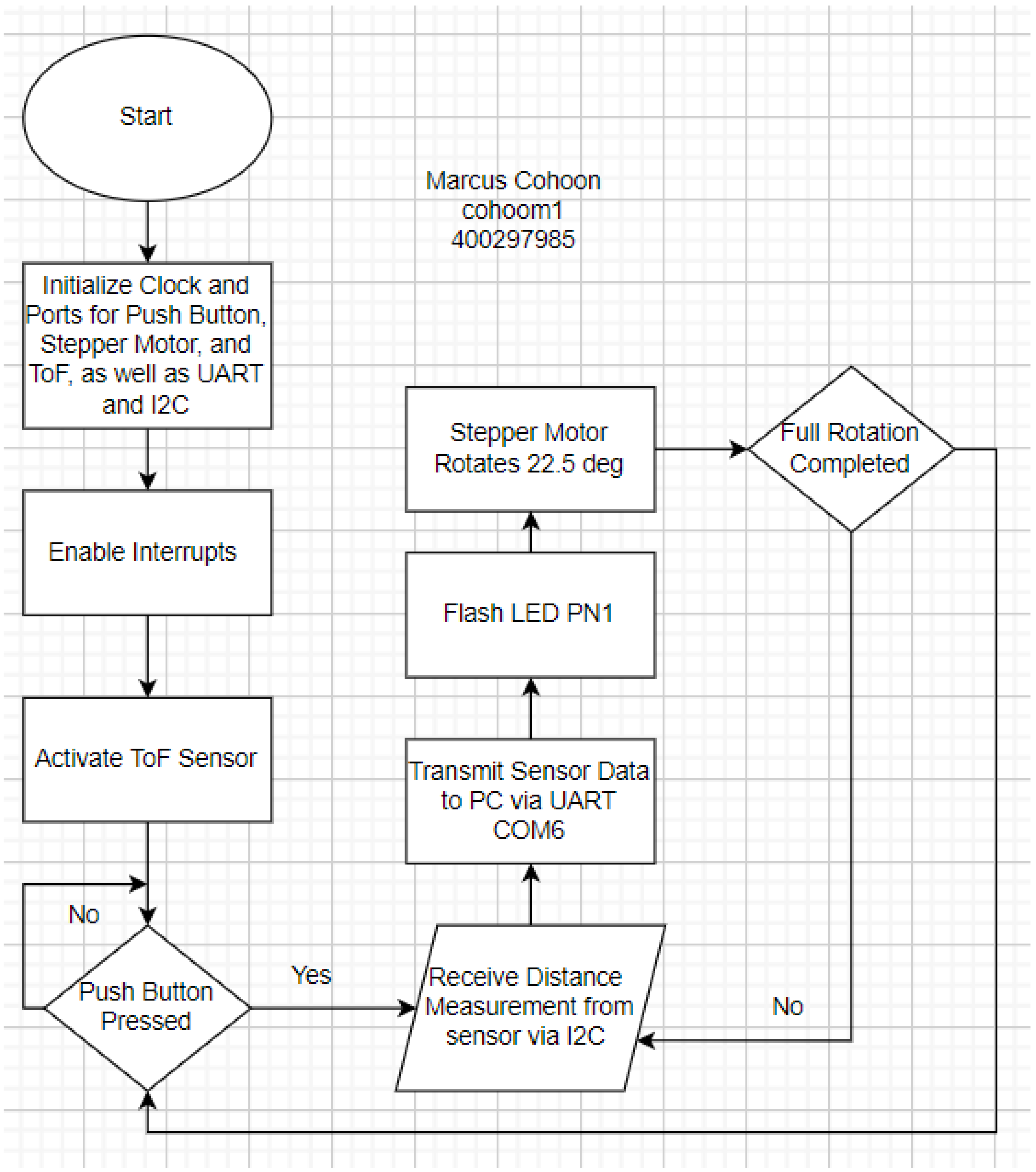
Start
Marcus Cohoon
cohoom1
400297985
Initialize Clock and Ports for Push Button, Stepper Motor, and ToF, as well as UART and I2C
Enable Interrupts
Activate ToF Sensor
No
Push Button Pressed
Yes
Receive Distance Measurement from sensor via I2C
Transmit Sensor Data to PC via UART COM6
Flash LED PN1
Stepper Motor Rotates 22.5 deg
Full Rotation Completed
No

Fig. 6. Microcontroller C Code Flowchart

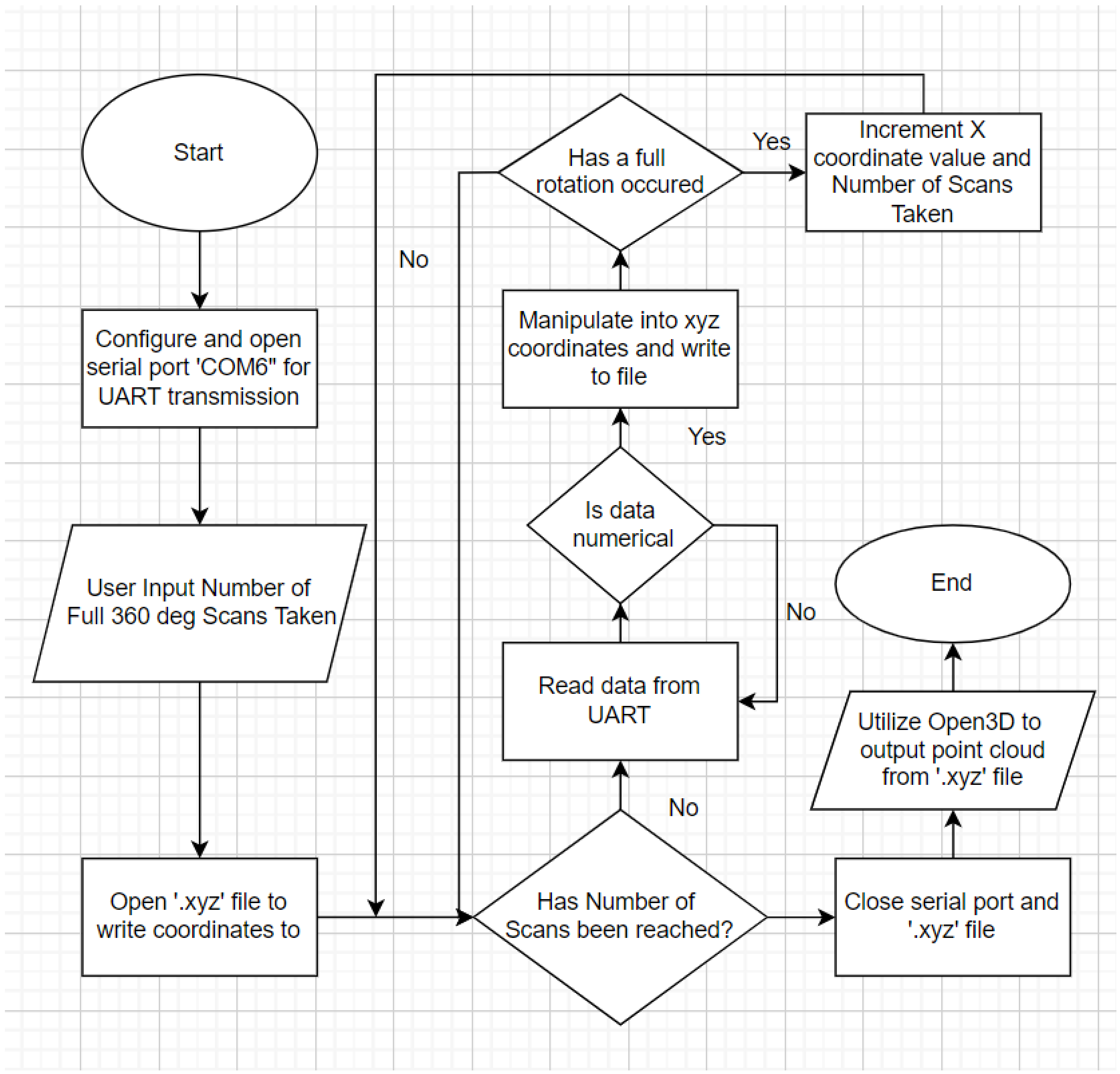
Start
Configure and open serial port 'COM6" for UART transmission
User Input Number of Full 360 deg Scans Taken
Open '.xyz' file to write coordinates to
Has Number of Scans been reached?
No
Read data from UART
Is data numerical
No
Yes
Manipulate into xyz coordinates and write to file
Has a full rotation occured
Yes
Increment X coordinate value and Number of Scans Taken
No
Close serial port and '.xyz' file
Utilize Open3D to output point cloud from '.xyz' file
End

Fig. 7. Python Code Flowchart