2DX4: Microprocessor System Project

Final Project

Instructors: Dr. Bruce, Dr. Haddara, Dr. Hranilovic, Dr. Shirani

Yuying Lai – L02 – lay24 – 400268588

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 to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by [Yuying Lai, laiy24, 400268588]

Folder Link for all Videos:

https://drive.google.com/drive/folders/1Obn6psX0NrS9YNtKdTUlqDWPN8ayW2mN?usp=sharing

Demo Video Link:

https://drive.google.com/file/d/1pX9EV01eQ89-2GAe-2M_AQZ0HRGwgWZK/view?usp=sharing

Question 1 Answer Video Link:

https://drive.google.com/file/d/1aygwhOH1fFjJCJJmcZDtGOAw_WoPufTN/view?usp=sharing

Question 2 Answer Video Link:

https://drive.google.com/file/d/1II-CFBMZJe_lQAYLezthSRNQRcgzxNvX/view?usp=sharing

Question 3 Answer Video Link:

https://drive.google.com/file/d/1e6l9pM0Yq1yK100yoYWxGuYnLBmZnA_F/view?usp=sharing

Table of Contents

1. Device Overview

a) Feature

- Texas Instrument MSP432E410Y
  - Processing embedded system and transmitting data between sensor and PC
  - 30MHz clock speed (Max 120MHz)
  - ARM Cortex-M4 Core
  - 1024KB of Flash memory, 256KB of SRAM, and 6KB EEPROM
  - ADC speed up to 2Msps
  - Program with C programming langrage
- 28BYJ-48 Stepper Motor & ULN2003 Stepper Motor Driver
  - 4 pins for data input, 1 for voltage supply, and 1 for GND
  - Gear ratio 64:1
  - 512 full steps per rotation
  - Operating voltage 5V to 12V
- VL53L1X Time-of-Flight Sensor
  - Sending and sensing laser
  - Measure the distance using (photon travelling time * speed of light /2)
  - Max ranging frequency of 50Hz
  - Max measured range of 4m
  - Operating voltage 2.6V to 3.5V
  - I2C interface up to 400kHz
- 3D Plot Visualization
  - Program with Python 3
  - Acquire data using pyserial package
  - Arrange data using numpy package
  - Generate 3D plot using open3d package
- I2C
  - Communicate between VL53L1X sensor and MSP432E410Y
- UART
  - Communicate between MSP432E410Y and PC
  - Physical connected via USB-to-microUSB line
  - Baud rate of 115200 bps

b) General Description

This measurement system is used to acquire the information of the layout of the area around this system. The distance data is acquired with VL53L1X laser sensor. The sensor is attaching to 28BYJ-48 motor to give a 360-degree of measurement angle, and it measured in geometric z-y plan. Multiple rotations of measurement are allowed with manually move the device 15cm after each rotation in x-axis. The information will stored in MCU memory and send to PC, so PC can map each data of point and plot the 3D graph of layout.

To use this system, push reset to start the initializing process. After initialization, run Python file and push PJ1 onboard button to start measuring. After each rotation, manually move the device and push PJ1 again for measurement of another rotation. If the complete measurement is done, push PJ0 onboard button to stop. The 3D graph will automatically appear in the screen.

Before the measurement start, time is needed for initialize VL53L1X sensor. In terms of the process of VL53L1X ToF sensor, the distance is calculated based on the time difference between emitting and sensing the laser. The distance is calculated by half the time times the speed of light, and this data will be decode from analog to digital output and sent to MSP432E401Y via I2C. For more information of the decoding process or the basic working function of sensor, please refer to the datasheet or Detailed Description. The I2C clock speed used is 25MHz. For the calculation, please refer to Limitation parts.

The stepper motor 28BYJ-48 is controlled by MSP432E410Y via GPIO port H0-H3. Before each measurement is taking, the motor will spin 2 full steps, so there will be 64 measurement per rotation. After each rotation, the motor will spin in the opposite direction (counter clockwise) for 1 revolution to put wire in original position.

Python program run at PC is used to acquiring measurement data and plot 3D graph. Before the python is used, please install pyserial, numpy, and open3d package. Data is read from serial input send from MSP432E410Y via UART with baud speed 115200 bps, and they will be written into file "tof_radar.xyz" in xzy format. Using open3D library to create point cloud of data stored in the file, and use numPy library to create vector line between each point. All points and lines will be plot at screen after the program completed. For more information of data acquiring and visualization, please refer to the Detailed Description part.

c) Block Diagram

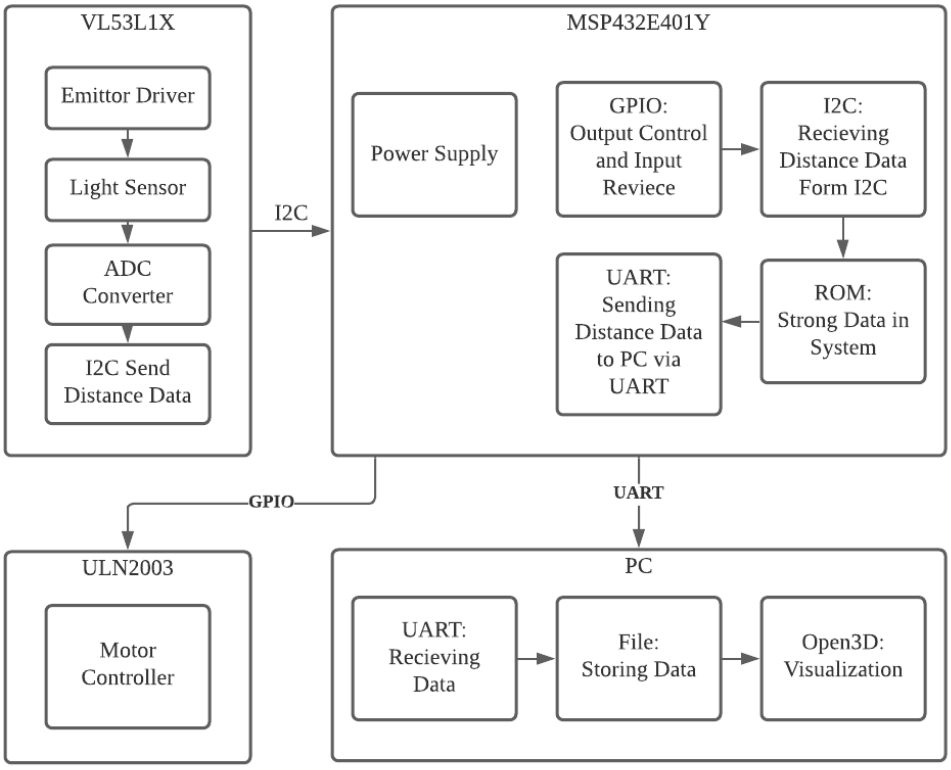

Figure 1: Block Diagram

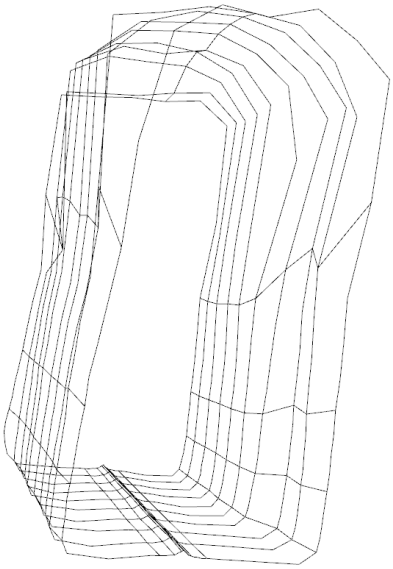

Figure 2: Output 3D Visualization

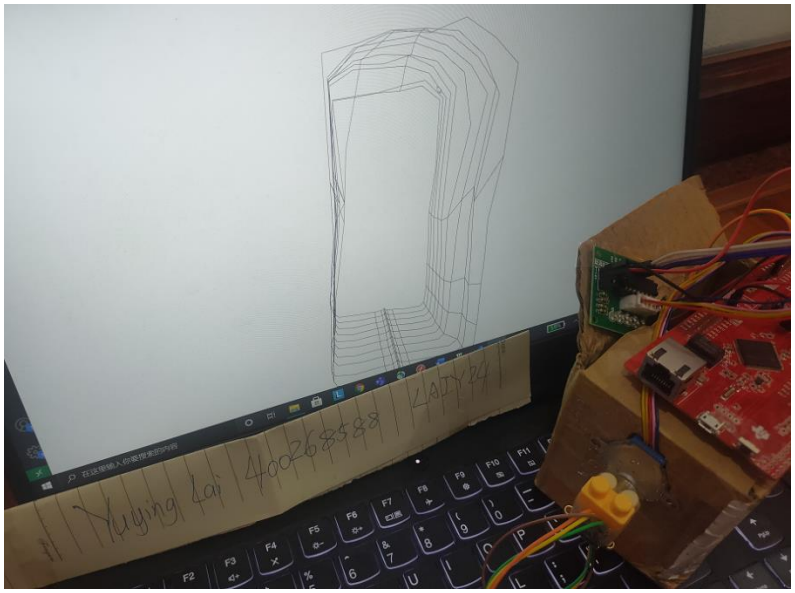

Figure 3: System Device and Output

2. Device Characteristics

Table 1: MSP432E410Y System Setting

Parameter	**Setting**
Bus Speed	30MHz
Baud Rate	115200 bps
PJ1	Active-Low Input Button (Start)
PJ0	Active-Low Input Button (End)
PN1	Onboard LED for Distance Status

Table 2: Pins Connection

Device	**Pin at Device**	**Pin at MSP432E410Y**	**Functionality**

ULN2003 Stepper Motor Driver	IN4	PH0	Motor Control
	IN3	PH1	
	IN2	PH2	
	IN1	PH3	
	+	5V	Voltage Supply
	-	GND	
VL53L1X	SDA	PB3	I2C Data
	SCL	PB2	I2C Clock
	VIN	3.3V	Voltage Supply
	GND	GND	
PC	USB (Serial: COM7)	SimpleLink microUSB Port	UART

3. Detailed Description

d) Distance Measurement

VL53L1X time-of-flight sensor is used for distance measurement in this system. With 360 degrees rotation angle provided by the movement of motor, the sensor can acquire information about the area around it.

In terms of the working process of sensor, the functioning part of sensor can be divided into two part, the emitter, and the detector. The emitter send light to the detecting object. When the light is reflected, the detector will sense it. The time difference between sending and sensing the light is the time of flight. Theoretically, the light travel twice the distance with the detected time of flight. Therefore, the distance is half the time times the speed of light with formular $D = c * \frac{t}{2}$, where D is distance, t is the time, and c is the speed of light. In terms of the digital process of each measurement, some steps are required. After the initialization of sensor, interrupt is enabled at the sensor. For each measurement, the sensor will begin at Ranging. When sensor sensed light, interrupt will raise, and data will be acquired. After that, interrupt will be clear to be ready for another

ranging. ADC component inside the sensor module will convert the analog to digital data. Then, the 16 bits data will send to microcontroller via I2C.

Before the measurement is taking, some initialization mush be complete. System, GPIO, I2C, and UART should be initialized. LED flashing will indicate the process of initialization. After that, python code should be run to successfully set communication between PC and board. 4 onboard LED will flash when python code is successfully run.

The data acquired at the microcontroller board is cylindrical coordinates. When the program starts, pushing PJ1 will allow the motor to rotate, and sensor will start to take measurement. In this system, 64 data will be collected for one rotation, so the angle between each measurement is 5.625 degrees. If more precise angle is needed, please refer to the Application Example to see the instruction of modifying number of tests per revolution. Every time MSP432E410Y receive distance data from sensor via I2C transmission, the data will be sent to PC via UART. Then, the motor will rotate clockwise for 8 full steps to prepare for another measurement. After one revolution, a complete data at y-z plane with fixed x-coordinate will be collected, and motor will rotate 360 degrees counter clockwise to rearrange the wire.

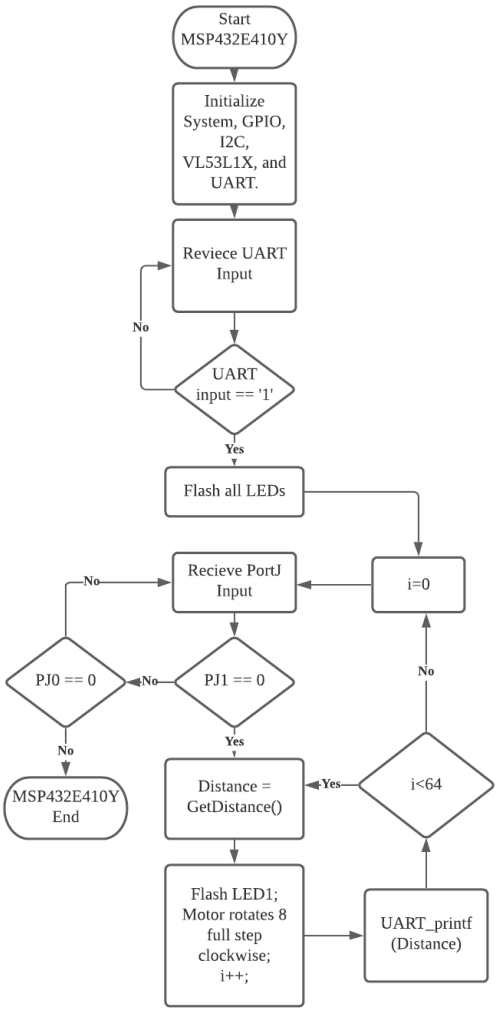

Figure 4: Program Processing Flowchart

There are four state in this machine. State 0 is waiting for python, which will loop itself if communication between python and board is not set. If python file ran, it goes to state 1 to receive input. State 1 will get input from onboard button. If no button is pressed, it will loop itself. If PJ1 pressed, it goes to state 2 to rotate motor. If PJ0 pressed, the measuring program end, and state 3 will loop empty progra. State 2 is where the program is taking one rotational of measurement. When state 2 is completed, it goes back to state 1 automatically to wait for the other input. Therefore, to start another rotation, press PJ1 if the last rotation has completed.

Rotational at y-z plane provide the information of y and z-coordinate. Manually move the device provide the movement of x-coordinate. However, this onboard system only sends distance data via UART in measurement order. C program will not process coordinate data, so it can save time and space for execution. Please refer to the next

section, Visualization, for how python code will process these data. For a brief explanation, d*cos(angle) will give y-coordinate where d is the measured distance. The z-coordinate is d*sin(angle).

e) Visualization

PC with Python can be used for this system. Windows 10 with Python 3.8 is recommended. The only limit is that the PC and Python must be 64 bits in order to use Open3D package. Python program and embedded program work synchronize. Please refer to Programming Logic Flowchart for the information.

Python with package PySerial, NumPy, and Open3D will be used for this visualization. The python code contains two modules. The first module will be acquiring data from serial via UART and write point data into file in xzy. The second module will read data from file contains point xyz data and plot the 3D graph using Open3D. These two modules can also represent two state.

At the first module, PySerial package will be used to send and receive data. It begins with enable serial at correct XDS110 COM port and open the xyz data file. Please refer to the Application Example for the step to set appropriate port and file path. It will first send char of 1 via UART to enable the program at MSP432E410Y. After that will be the acquiring process. It will then read one line from serial input via UART using **serial.readline()**. After removing '\n' char, if the remaining data is digit, it means this data is distance data. According to the number of tests that have been receive, the program will calculate the angle information of this distance data. Using angle and trigonometric function, it can calculate the y and z coordinate. The y-coordinate is equal to the distance times cosine of the angle, **y = Distance*cos(angle).** The z-coordinate is equal to the distance times sine of the angle, **z = Distance*sin(angle).** Then, data of coordinate of x, y, and z will be written into 'tof_radar.xyz" file using **file.write()**. The total number of points will be counted during this process. Each times the program complete receiving data for one rotation of measurement, x-coordinate will increment for a set value (15cm specifically in the default setting). This can be achieved by formula numTest % numTestPerRotation, where no remainder means one rotation is completed. If there are already data in file and the next receiving data is not digit, it means the 'end' massage is send by MSP432E410Y. The serial and file can closed and acquiring data stage is end.

The second state will be plotting 3D graph. After importing required package, it read data from 'tof_radar.xyz' file in xyz format and create point cloud using **open3d.io.read_point_cloud()**. Then, NumPy is used to create vector line between adjacent point using **array.append()**. For points in each plane, the connected graph should be a closed graph. Each point will connect to two adjacent points that is close to it. After closing every plane, connections between each plane are needed. At default setting, points are connected every 22.5 degrees. For example, the file contains two planes of data. Program will connect the points at 22.5 degrees from the first plane to second plane. It will also connect points at 45, 67.5, 90 degrees, and so on. To modify the default

setting of angle here, please refer to the steps in Application Example. Then, vector will be converted to line set using **open3D.geometry.LineSet()** and **open3d.utility.Vector3dVector().** After points and lines are all set, the program will plot 3D graph of these points and vectors using build-in function **open3d.visualization.draw_geometries([line_set])** from Open3D library.

All distance units are in millimetre. The default view angle is at x-y plane. Angle of view can be adjusted using mouse.

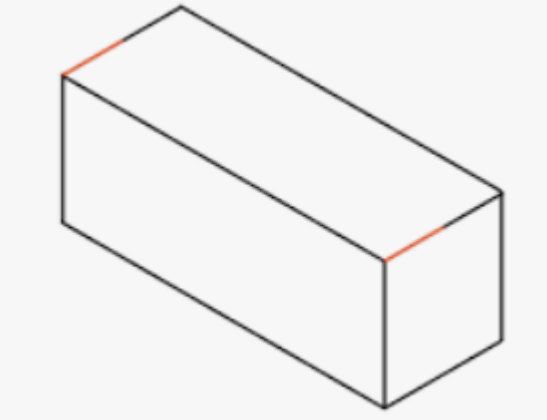

Figure 5: Rough space detected from figure 2.

4. Application Example

The manually movement of device is in positive x-axis direction. Plane of y-z is the rotation plane where the measurement is taking. Generally, horizontal line is y-axis, and vertical line is z-axis. x-axis is perpendicular to y-z plane, with positive direction pointing by the motor shaft.

In order to use the distance measurement system, some setup steps are required.

1. Download Python if you are not already having that in your PC. Python 3.8 is recommended. Make sure you are using 64 bits version. Link for downloading is at https://www.python.org/downloads/ .
2. Connect the circuit according to the Circuit Schematic. Load C program into MSP432E410Y board. Using Keil is recommended. If the program is loaded, you can push reset button to start the program for initialization.
3. Make sure you have installed PySerial, NumPy, and Open3D package for Python. The following instruction can run on Windows terminal for installation.
    a. pip install pyserial
    b. pip install numpy
    c. pip install open3d
4. Modify python file “dataVisualize.py” at line 13. Change “COM7” with your USB COM port number used for connecting MSP432E410Y. Modify line 19 and 63 to change the path to where you want to store file “tof_radar.xyz”. If you want the file to be at the same directory as the python file, path information is not necessary. The following instruction can help you with choosing the write COM.
    a. Connect the board to PC

   b. In terminal run “python -m serial.tools.list_ports -v”. You can try which one is the right port to use.
   c. If success, you will see output of distance data on screen in the following steps.
5. If step 3 is completed, you can run python file.
   a. Make sure you run the file when initialization on MSP432E410Y is complete. The initialization is completed when you will not see any onboard LED is on.
   b. You will see all LEDs flash when you successfully run Python file. It indicates that the connected between PC and board is success. If not, please check the connection and run python file again.
6. Push onboard button PJ1 to start measurement. The measurement will start immediately, and you will see onboard LED1 is flashing when the measurement is taking.
7. After a rotation completed and the motor is back to its original place, manually move device 15cm at positive x-axis direction(orthogonal to the rotation plane). Push PJ1 again for another rotation of measurement.
8. Push onboard button PJ0 to stop the program. You will see the output 3D plot on screen. Make sure you push when the motor has rotated back. When the program stop, LEDs on motor drive will turn off.
9. If you are experiencing any problem during the test, or you need to start a new measurement, please push reset button anytime.

Here is some instruction if you need to change the precision of the result plot.

1. To change the number of tests per rotation, open “main.c” file to change the value at line 143, and open “dataVisualize.py” file to change the value at line 26. Please make sure that these two values are the same. The maximum value if 512.
2. To change the x-axis movement value, open “dataVisualize.py” file and modify at line 49. The unit is millimetre.
3. To change the connecting point between each plane at output graph, open “dataVisualize.py” file and modify **vectorAngle** value at line 83. This means the connecting point will be per that amount of angle. The unit is degree.

5. Limitations

1) Floating point unit used in MSP432E410Y is 32-bit instructions for single-precision (C float) data-processing operation. It takes less time to execute a float than double, but float have less precision than double. The Coretex-M4F floating point instruction does not support all operations in IEEE 754-2008 standard, like calculate Remainder, round float to integer-valued float, and Bin-to-dec conversion. Compiler building head file <math.h> using and return double type data for trigonometric functions. It required convention between float and double. Including “arm_math.h” file allows user to use cos and sin function for float32_t type parameter and returned value. The specific function is “arm_cos_f32(float32_t x)” and “arm_sin_f32(float32_t x)”.
2) The maximum range is 4m. The return data is 16 bits.

$$Max(quantization\ error) = \frac{4000mm}{2^{16}} = 0.061mm$$

3) The maximum serial communication rate for PC is 128000 bps. To check this property, go to setting **Device Manager > Port (COM & LPT).** Right click the correct **XDS110** used for UART and select **Properties.** At **Port Setting** tab, all available baud rate is listed. The maximum available rate is 128000 bps. The maximum baud rate for regular speed at MSP432E410Y is 7.5Mbps, which is listed at page 1621 at "MSP432E410Y SimpleLink Microcontrollers – Technical Reference Manual".
4) The communication method used between MSP432E410Y and ToF sensor is I2C. The timer period written into I2C_MTPR_TPR is 0x3B(59). The system clock speed is 30MHz. SCL_HP is fixed at 4, and SCL_LP is fixed at 6. According to this formula, the SCL line period is 400000ns, and the frequency is 25kbps. SCL_PERIOD = 2*(1+TRP)*(SCL_LP+SCL_HP)*CLK_PRD = 2*60*10*(1/30M)
5) The primary limitation of speed is the ranging speed of the sensor. To take one measurement of distance, the program needs to check if data is ready before the measurement is taken. This step generally takes times since the max frequency of ranging is 50Hz. The minimum timing budget used in short distance mode is 20ms. Generally, 140ms is the timing budget to detect 4 m. The rotation of motor may also spend times, but the delay time is adjustable.

6. Circuit Schematic

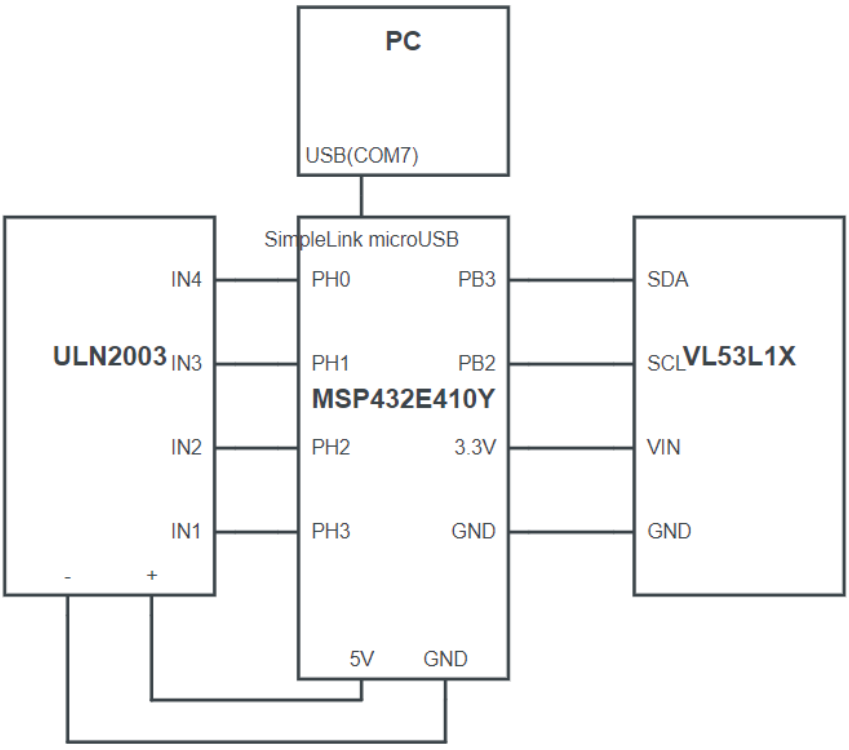

Figure 6: Circuit Schematic

7. Programming Logic Flowchart

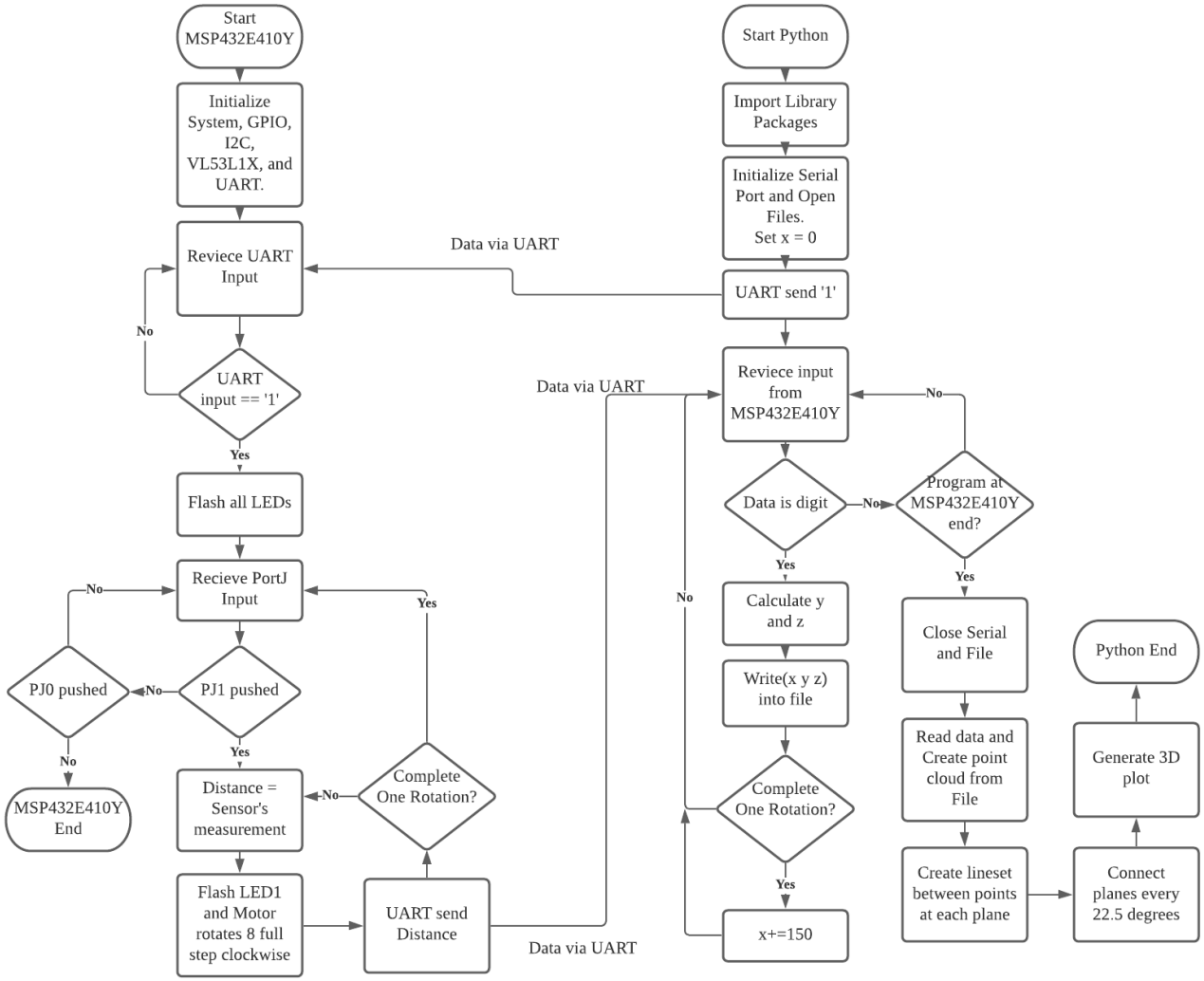

Figure 7: Synchronize working flowchart of Embedded and Python program with default setting.

*Data via UART only represent the data transmission direction. It does not represent working process.