Comp Eng 2DX3 Final Report
Time-of-Flight Spatial Mapping System

Dhruv Anand
400535222
anandd4@mcmaster.ca

Instructors: Sharuk Athar, Thomas Doyle, Yasser Haddara

April 9th, 2025
COMP ENG 2DX3

Table of Contents

List of Figures

List of Tables

Design Overview

Features

This project features a compact, portable, and cost-effective LIDAR system designed to create 3D maps of indoor environments. At its core is the Texas Instruments MSP-EXP432E401Y microcontroller, which operates alongside two on-board buttons, a ULN2003 driver board, a 28BYJ-48 stepper motor, on-board LEDs, and a VL53L1X time-of-flight (ToF) sensor.

Texas Instruments MSP-EXP432E401Y has features**:**

- Communication hub between sensor, motor, and computer for visualization
- 32-bit ARM Cortex-M4F CPU
- 120MHz max clock frequency
- 1MB Flash Memory
- 256kB SRAM
- GPIO pins with interrupt, PWM, and ADC support
- 2 onboard buttons
- 4 onboard user LEDs
- JTAG, Micro USB, Ethernet ports for data transfer
- UART communication at 115200BPS to PC
- I2C module for peripheral communication

VL53L1X Time of Flight (ToF) Sensor has features:

- 100 kbps communication via I2C protocol
- Distance modes:
  - 130cm
  - 300cm
  - 400cm (maximum)
- 50Hz frequency
- 940nm laser emitter
- 2.6V to 3.5V operating voltage

28BYJ-48 Stepper Motor has features:

- 512 steps for 360° rotation
- 4 phases per step
- 4 onboard LEDs to indicate phase
- 5V to 12V operating voltage

Serial Communication has features:

- I2C protocol for communicating between VL53L1X Time of Flight (ToF) Sensor and Texas Instruments MSP-EXP432E401Y
- UART protocol for communicating between Texas Instruments MSP-EXP432E401Y and PC
- 115200BPS baud rate

Data Visualization has features:

- MATLAB
    - Used for serial communication between Texas Instruments MSP-EXP432E401Y and PC
    - Used for interpreting and plotting data

General Description

This embedded system provides the user with a cheap and easy to operate indoor 3D visualization method.

Pressing onboard button PJ1 activates the ToF sensor. When successfully activated, onboard LED D3 (port PF4) will turn on to indicate status. Onboard button PJ0 will execute a full rotation of the stepper motor, alternating between clockwise and counterclockwise rotation.

The stepper motor rotates 11.25 degrees at a time, pausing momentarily to allow the ToF sensor to take a measurement. The ToF sensor captures distance data in the YZ plane every 11.25 degrees (in polar coordinates). Each time a distance measurement is taken, onboard LED D2 (port PN0) flashes to indicate the data capture.

After each measurement, distance data is sent to the microcontroller using I2C protocol and then sent to the PC through UART protocol. Each time data is to be sent to the PC, onboard LED D1 (port PN1) flashes to indicate transmission. This data is then captured and interpreted via MATLAB.

The system is operated through the MSP-EXP432E401Y microcontroller, which is running a 20MHz bus speed for this particular application.

Distance data is captured by the ToF by emitting a photon and measuring the time for the photon to reflect back to the ToF. The exact formula is

$$distance = \frac{time\ of\ travel}{2} \times (light\ speed)$$

Data is sent to MATLAB through UART protocol. Distance measurements are converted from polar to cartesian coordinates to obtain Y and Z components. X components are hardcoded by the user to emulate a specific depth. Data is plotted on the XYZ plane.

Block Diagram

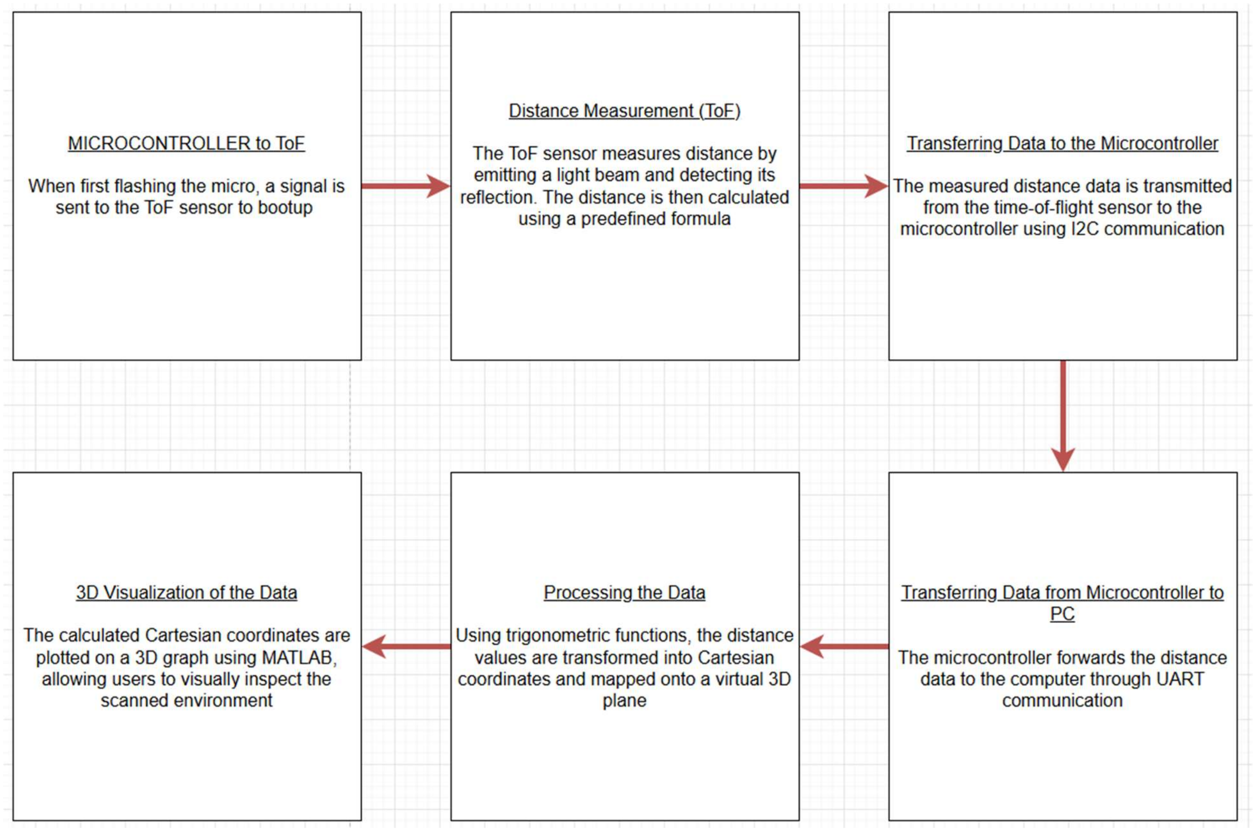

MICROCONTROLLER to ToF
When first flashing the micro, a signal is sent to the ToF sensor to bootup
Distance Measurement (ToF)
The ToF sensor measures distance by emitting a light beam and detecting its reflection. The distance is then calculated using a predefined formula
Transferring Data to the Microcontroller
The measured distance data is transmitted from the time-of-flight sensor to the microcontroller using I2C communication
Transferring Data from Microcontroller to PC
The microcontroller forwards the distance data to the computer through UART communication
Processing the Data
Using trigonometric functions, the distance values are transformed into Cartesian coordinates and mapped onto a virtual 3D plane
3D Visualization of the Data
The calculated Cartesian coordinates are plotted on a 3D graph using MATLAB, allowing users to visually inspect the scanned environment

Figure 1: System block diagram

Device Characteristics

MSP-EXP432E401Y		**28BYJ-48 Stepper Motor (driver board)**		**VL53L1X Time of Flight**	
Clock Speed	20Mhz	**V+**	3.3V	**Vin**	3.3V
Measurement LED	D2 – PN0	**V-**	GND	**GND**	GND
UART LED	D1 – PN1	**IN1**	PH0	**SDA**	PB3
ToF Status LED	D3 – PF4	**IN2**	PH1	**SCL**	PB2
ToF activation button	Button 2 – PJ1	**IN3**	PH2		
Motor 360° rotation button	Button 1 – PJ0	**IN4**	PH3		
Baud Rate	115200 BPS				

Table 1: Device pin assignments

Detailed Description

Distance Measurement

The VL53L1X Time-of-Flight sensor works by emitting a photon from its "emitter" electrode. This photon reflects off the most nearby object nearby object (normal to the ToF) and is received back by the sensor's "receiver" electrode. By tracking the time it takes for the photon to travel to the object and return, the sensor determines the distance using the formula: $distance = \frac{time\ of\ travel}{2} \times (light\ speed)$.

Once the measurement is taken, the sensor processes the signal through several stages, including transduction, signal conditioning, and analog-to-digital conversion. This process results in a digital distance value, which is sent to the microcontroller through I2C.

To initialize the ToF sensor, the ToF boot function is called within the system's C program. The system then waits for a trigger through polling methods to begin taking measurements. Pressing onboard button 2 (PJ1) readies the ToF to take measurements by toggling a variable "scanReady". This subsequently toggles the ToF status LED D3 (PF4).

Pressing onboard button 1 (PJ0) begins a 360° rotation cycle of the stepper motor, moving 11.25° at a time. The program checks if the motor has moved by this angle using modulus 16. Since we know 512 steps is 360deg, we can determine 11.25deg is 16 steps. When the motor completes 11.25deg of rotation, the program flashes led D2 (PN0) to indicate a measurement will be taken. A measurement will then be taken, and using the ToF's API functions, distance data is retrieved.

At the same time, depth and steps (of the motor) travelled are being tracked within the program. Depth increments each time the motor completes a 360deg rotation to simulate movement in the X-axis, and steps increments each time the stepper motor goes through a single step sequence. The steps data is used to check for 11.25deg of movement, but also to calculate polar displacement of the motor using a positive reference, which will later be converted from steps to degrees. If moving in the clockwise direction, displacement is directly equal to the step count, whereas in the counterclockwise direction, displacement is the total steps for 360 ° rotation (512), minus the step count.

Finally, each time a distance value is retrieved through I2C, LED D1 (PN1) will flash to indicate UART transmission will take place. The microcontroller then transmits distance data, the current displacement, and the current depth to the PC via UART protocol. This cycle repeats 32 times, for one complete 360° rotation.

Visualization

Once distance, displacement, and depth are transmitted, MATLAB is used to store and process the data. MATLAB connects to the PC UART port and opens for serial communication at 115200BPS baud rate with a 100-second timeout. Within the program, the user sets "maxDepth", which is the number of 360deg stepper motor rotations intended to be measured. The program stores all distance, displacement, and depth into a matrix using a while loop that runs maxDepth*32 times (total number of measurements). The displacement values are converted to angles using the knowledge that 16 steps is 11.25 degrees, and then appended into the matrix. The program also interprets each increment of depth as 20cm in the X-axis. Using trigonometry, polar distance is converted to cartesian coordinates in the YZ plane. The formulas used are:

$$y = distance \times \cos(angle)$$

$$z = distance \times \sin(angle)$$

Each point is then plotted in a 3-dimensional XYZ scatterplot. To better represent the area being scanned, points within each depth are connected using black lines. Points are also sorted by their angle to connect each point to its corresponding point with the same angular component between adjacent layers of depth using blue lines. The final output is a helical structure representing the scanned area.

Application Note

Lidar scanning devices use light emission to create 3D maps of environments, making it a powerful tool for sensing and mapping with a broad range of possible real world applications.

Mapping and Surveying

- Topographic mapping: LIDAR scanning can create high quality 3D maps of terrain, even through and around obstacles such as vegetation

Autonomous Vehicles

- Navigation: LIDAR scanning helps vehicle detect obstacles
- Environment detection: LIDAR scan to build 3D map of live surroundings for path planning

Forestry and Agriculture

- Canopy analysis: LIDAR scan can be used to measure tree heights and forest structure
- Precise agriculture: LIDAR can be used analyze crop density, and finding optimized terrain for cultivation

Archeology

- Site discovery: LIDAR can reveal hidden structures
- Safe documentation: LIDAR can be used to document at-risk locations without disturbance

The list of real-world applications for which LIDAR can be used continues to grow as the technology becomes more accessible and precise.

Instructions

The following instructions assume that the user has successfully set up all necessary software for their specific device (PC) and knows the corresponding UART port.

1. Connect the microcontroller into computer
2. Open the MATLAB script and uncomment line 5, "display(portList)", and run the script to determine which index of the array containing COM ports refers to the PCs designated UART port. Comment back this line, and edit the index of "portList" on line 6 to the index referencing the designated COM port for UART that is to be opened for communication.
3. On line 10, change "maxDepth" to the number of 360deg rotations/depth increments you wish to measure
4. Configure the circuit by following the pin assignments in Table 1 or the circuit schematic in Figure 4.
5. Open the Keil project. Open "options for target". Under C/C++, ensure optimization is set to "-O0".
6. Translate, build, and load the C program into the microcontroller, then hit the onboard reset button.
7. Wait for all 4 onboard LEDs to finish flashing twice. The system is now ready for operation.
8. Activate the ToF sensor by pressing onboard button 2 (PJ1). LED D3 (PF4) will light up to indicate the ToF is ready.
9. Run the MATLAB script. Note the script will timeout if no serial data is read for 100 seconds.
10. Use button 1 (PJ0) to begin rotating the stepper motor 360deg, taking measurements and sending data every 11.25deg
11. After every 360deg rotation, when ready to take another measurement (note you only have 100 seconds to initiate another cycle) click button 1 (PJ0) to initiate another 360deg cycle
12. Once the "maxDepth" number of rotations have been completed, press button 1 one last time. The MATLAB script will now end and output a 3D mapping of the scanned area

Expected Output

The expected output of the system can be visualized using the images below of a hallway and its corresponding scan.

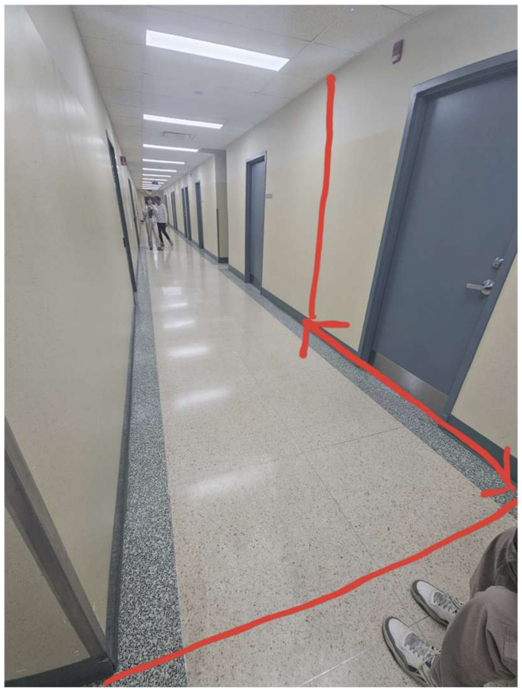


Figure 2: Example scanned location

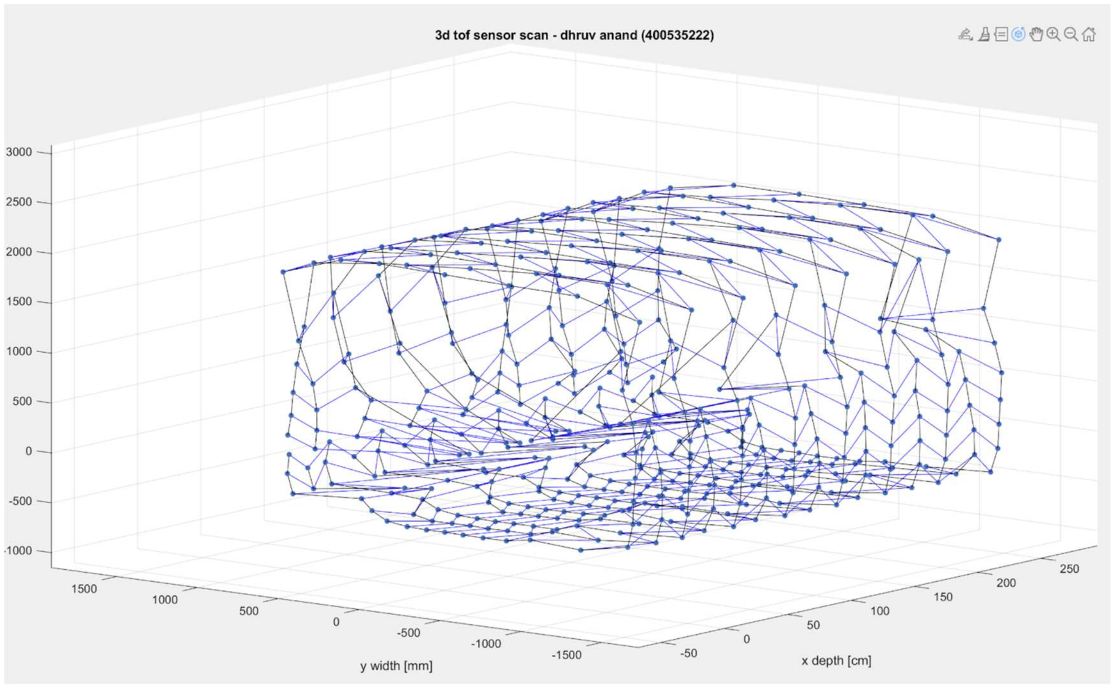


Figure 3: Scan output of example location

As can be seen from the final scan, the system is able to capture the scanned location. The scan depicts a uniform rectangular prism, with slight variations in width representing the depth of doorways in the hall.

Limitations

As this system is designed to be a cheaper alternative to industry LIDAR scanners, there are some inherent limitations to be considered.

While trigonometric functions are handled on the PC during data visualization, if these calculations were to be performed on the MSP-EXP432E401Y microcontroller, they would be limited by its single-precision floating-point capabilities. The microcontroller includes 32x32 bit "single precision registers" as part of its Floating-Point Unit (FPU), allowing it to perform basic floating-point operations.

These operations can introduce rounding errors due to limited precision, especially when dealing with the long decimal values that arise from trigonometric computations. Additionally, complex operations are computationally slower compared to integer math, potentially impacting real-time performance.

Quantization error is defined as the difference between the analog data value and the nearest digital value. The VL53L1X ToF provides digital distance readings with a 1mm resolution. Therefore, the maximum quantization error is equal to the resolution of the sensor, which is 1mm.

This means any measurements taken may be off by ±1mm. There is no ADC involved for this sensor, so no bit-based quantization needs to be considered.

The max standard serial communication rate that can be implemented by my PC is 128000BPS. This was verified by opening the properties of the "XDS110 Class Application/User UART" port within the device manager. A speed of 115200BPS was implemented for this LIDAR system.

The ToF communicates with the microcontroller using I2C protocol at a max transmission speed of 50Hz.

The primary limitation on system speed is the speed at which the stepper motor can rotate. Between each of the 4 phases required to rotate the motor, there is a delay present that inherently slows down the system. If the delay is too short, the motor will not rotate. To test the minimum delay permissible between phases, the delay was gradually decreased until the motor was no longer able to spin. The minimum delay permissible between phases was found to be about 2ms.

As per table 1 in the project specification, I had to set the bus frequency to 20Mhz.

The following formula was used to isolate for the PSYSDIV value required to achieve a 20MHz bus:

$$Bus\ Frequency = \frac{480MHz}{PSYSDIV + 1}$$

$$20MHz = \frac{480MHz}{PSYSDIV + 1}$$

$$PSYSDIV = 23$$

To reflect this change in the program, the value of PSYSDIV was changed to 23 inside the PLL.h file. Note that changing the bus speed also meant SysTick functions needed to be updated.

Circuit Schematic

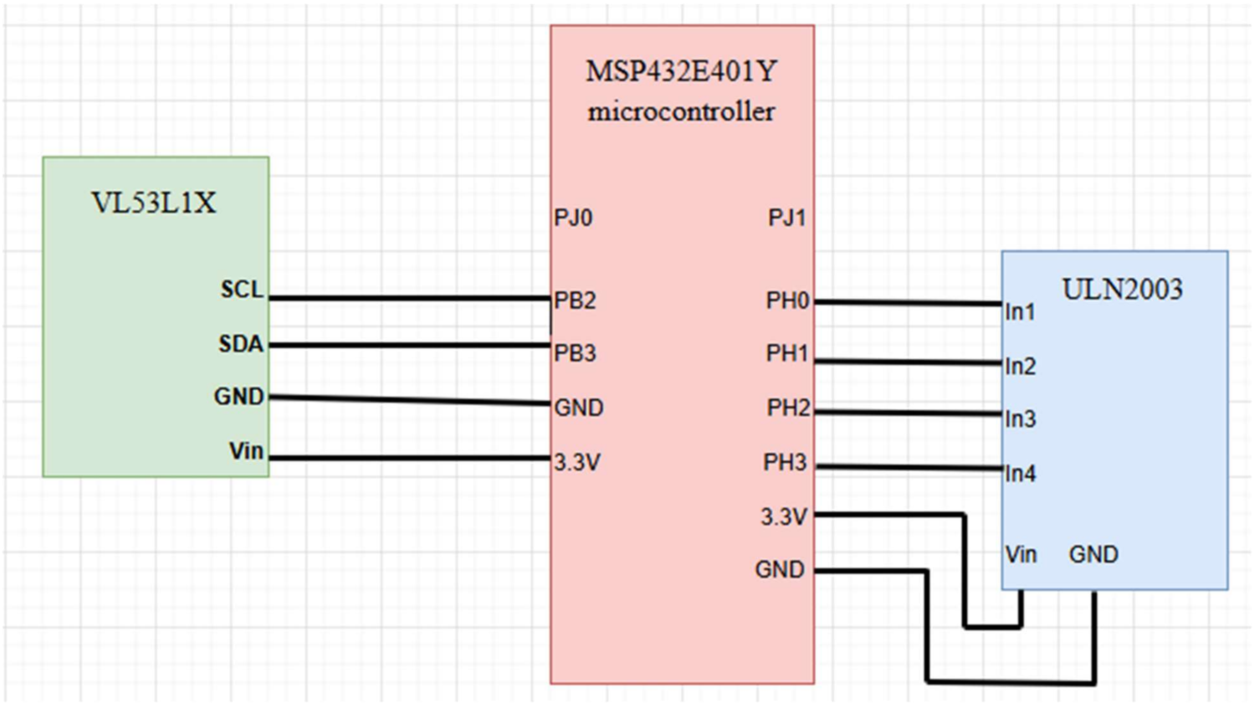
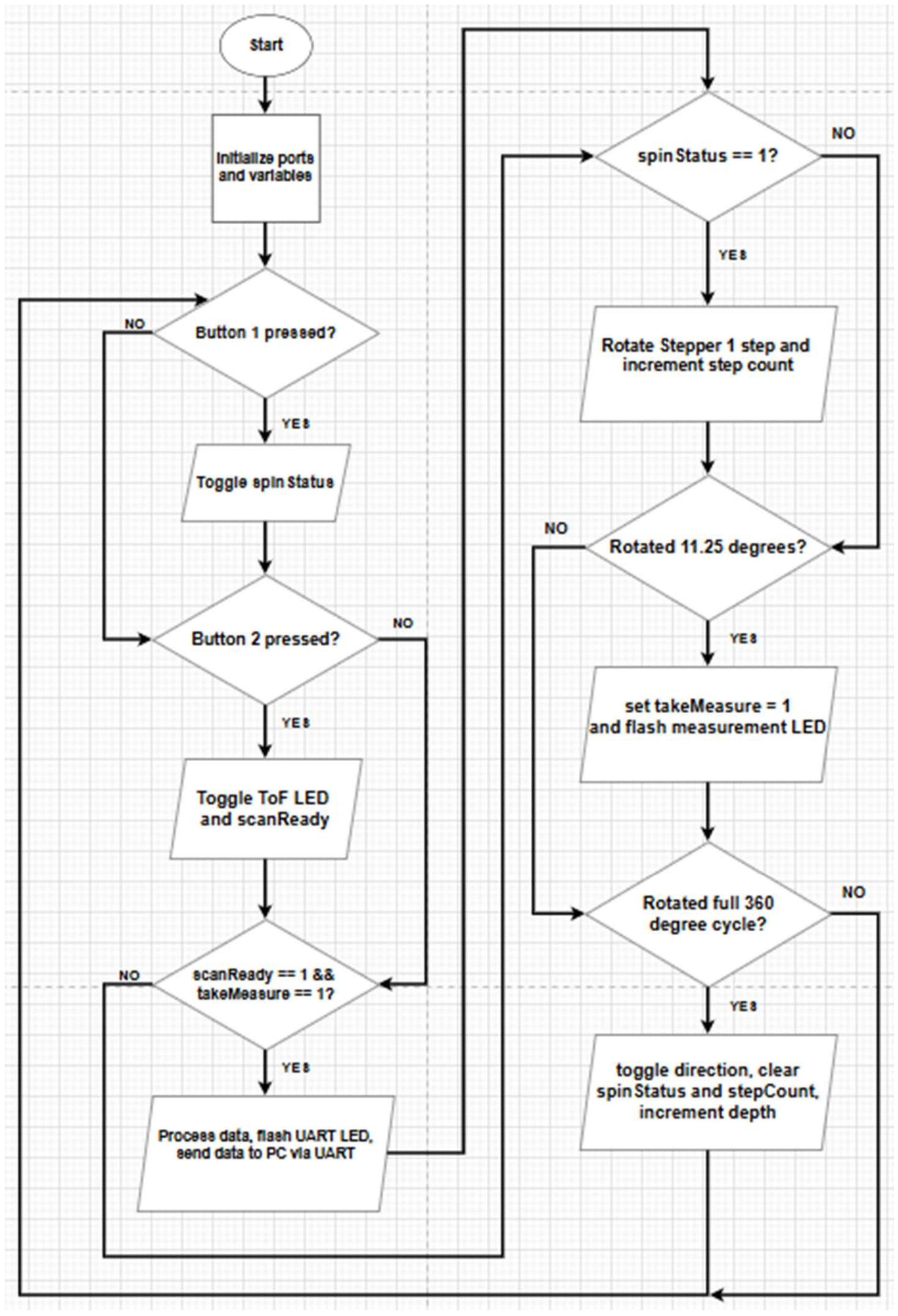


Figure 4: System circuit schematic

Logic Flowchart



Start
Initialize ports and variables
Button 1 pressed?
NO
YES
Toggle spin Status
Button 2 pressed?
NO
YES
Toggle ToF LED and scanReady
scanReady == 1 && takeMeasure == 1?
NO
YES
Process data, flash UART LED, send data to PC via UART
spin Status == 1?
NO
YES
Rotate Stepper 1 step and increment step count
Rotated 11.25 degrees?
NO
YES
set takeMeasure = 1 and flash measurement LED
Rotated full 360 degree cycle?
NO
YES
toggle direction, clear spin Status and stepCount, increment depth

Figure 5: Program flowchart