HS-400516296

LiDAR Room Scanner

Features

- A push button for motor rotation/measurement and another for data acquisition
- Power supply range: 3.3-5V
- I2C distance transmissions (ToF Sensor to Microcontroller)
- UART data acquisition via button press (Microcontroller to PC)
- Status LEDs (Measurement success, UART transmission, returning home status)
- 14 MHz bus speed
- 3D automated plot generated via Open3D on all scans' completion

Description

The HS-400516296 is a LiDAR room scanner that consists of various components. The scanner uses a ToF sensor mounted on a stepper motor to provide full 360° measurements, 2 buttons for manual user control and an automated plotting system.

MSP-EXP432E401Y Microcontroller	
Parameter	Value
Measurement Status LED	PN0
UART Transmission LED	PN1
Return Home Status LED	PF4
Start/Stop Motor Rotation & Measurement Push Button	PJ1
Send Collected UART Data Push Button	PJ0
Bus Speed	14 MHz
Baud Rate	115200 bps
Serial Port	COM5

VL53L1X ToF Sensor	
Parameter	Value
Vin	3.3V
GND	GND
SCL	PB2
SDA	PB3

28BYJ-48 Stepper Motor	
Parameter	Value
IN1	PH0
IN2	PH1
IN3	PH2
IN4	PH3
V+	5V
V-	GND

Device Overview – Features

Texas Instruments MSP-EXP432E401Y features [1]:

- Used for transmitting sensor measurements, stepper control, and LED user interfacing
- 3 on-board LEDs used for statuses of measurement taking, UART transmission and status when returning home
- Interfaces with on-board push buttons
- ARM Cortex-M4F 32-bit CPU
- 14 MHz bus speed
- 1 MB flash, 256 KB SRAM

28BYJ-48 Stepper Motor features:

- Driven via ULN2003 driver board using GPIO signals
- 5V unipolar stepper motor
- 64 steps per revolution motor
- 512 steps for full rotation

VL53L1X ToF Sensor features [2]:

- Time-of-Flight distance sensor with up to 4 m range
- I2C interface with the microcontroller
- Can operate at voltages between 2.6-3.5V

2 Onboard Push Buttons feature:

- Integrated on MSP432E401Y LaunchPad
- Button 0: Transmit all collected data via UART
- Button 1: Start/Stop the stepper motor
- Debounced in software for reliable operation
- Enables real-time control of data acquisition process

Transmission features:

- UART communication at 115200 baud rate
- Transmits all stored measurements as a single line on button press
- Data formatted with spaces as delimiters for easy parsing
- Ensures compatibility with external plotting/visualization tools

Plotting program features:

- Implemented using Python and Open3D
- Parses a single line of incoming data into separate measurements
- Converts distance and angle data into 3D coordinates
- Enables visual inspection of scanned environment post-processing

Device Overview – General Description

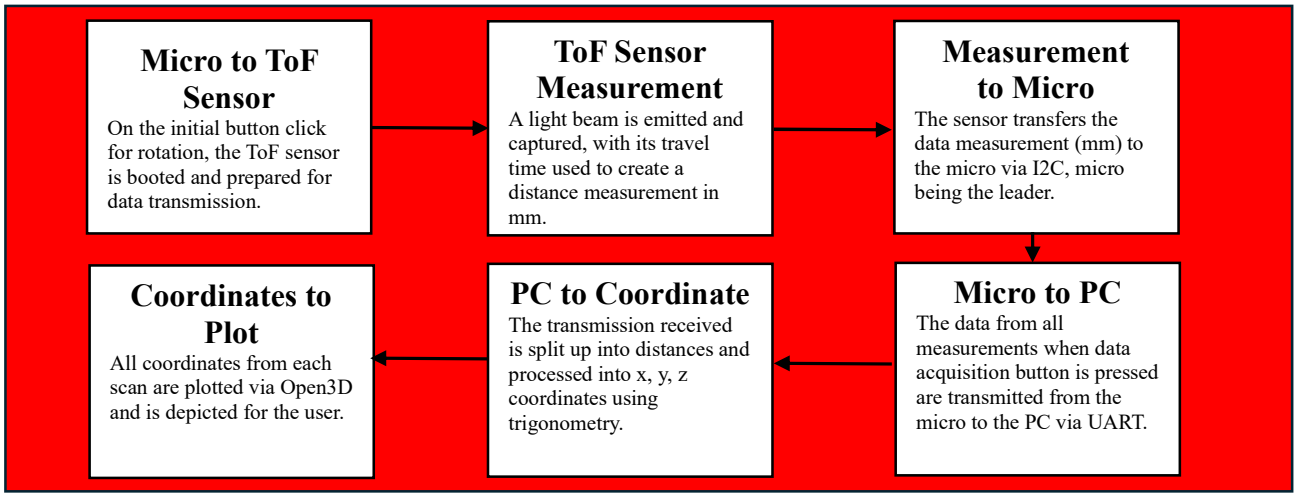
The room scanner is a low-cost, embedded spatial mapping system designed to capture 3D representations of indoor environments using a Time-of-Flight (ToF) sensor mounted on a stepper motor. The system integrates a VL53L1X ToF sensor, a 28BYJ-48 stepper motor, and a Texas Instruments MSP-EXP432E401Y microcontroller operating at a 14 MHz bus speed, customized per requirements. The scanner operates by rotating the ToF sensor 360° within the y-z plane, collecting radial distance measurements at regular angular intervals of 11.25° (32 measurements per scan). After a full rotation (one scan), the user must manually translate the scanner along the x-axis by a fixed displacement (e.g., 300 mm). This process is repeated to build a full 3D point cloud of the scanned environment, one vertical slice at a time.

Control is handled via two onboard push buttons: Button 0 initiates UART transmission of all data collected up to that point to a connected computer. Button 1 toggles the stepper motor and ToF scanning process, with stop functionality implementing return home.

LED indicators provide real-time feedback on system status: LED D1 (UART Transmission) briefly flashes at the start of a transmission sequence (when button 0 is pressed). LED D2 (Measurement Status) flashes with each distance measurement for visual confirmation. LED D3 (Return Home Status) toggles on while returning home.

Data is sent over UART at a 115200 baud rate, formatted as a single line containing all distance readings from a complete scan. On the host PC, a Python-based plotting program utilizing Open3D parses this data, reconstructs individual (x, y, z) coordinates, and renders the environment as a 3D point cloud. This visualization proceeds scan-by-scan, updating as new data is acquired and transmitted. The system offers a portable and scalable alternative to commercial LIDAR systems, enabling students to explore embedded data acquisition, real-time processing, and 3D visualization.

Device Overview – Block Diagram



Device Characteristics

MSP-EXP432E401Y Microcontroller		VL53L1X ToF Sensor		28BYJ-48 Stepper Motor	
Parameter	Value	Parameter	Value	Parameter	Value
Measurement Status LED	PN0	Vin	3.3V	IN1	PH0
UART Transmission LED	PN1	GND	GND	IN2	PH1
Return Home Status LED	PF4	SCL	PB2	IN3	PH2
Start/Stop Motor Rotation & Measurement Push Button	PJ1	SDA	PB3	IN4	PH3
Send Collected UART Data Push Button	PJ0			V+	5V
Bus Speed	14 MHz			V-	GND
Baud Rate	115200 bps				
Serial Port	COM5				

Detailed Description – Distance Measurement

The core of the room scanner's functionality lies in its ability to accurately acquire spatial distance data using the VL53L1X Time-of-Flight (ToF) sensor. This sensor operates by emitting infrared light pulses from its laser emitter, which reflect off a surrounding surface and return to the sensor's receiver. The key principle used in determining the distance is the time-of-flight measurement (where the name ToF comes from), where the time it takes for a photon to travel to the object and back is recorded. The distance is then calculated using the formula:

$$Distance = \frac{Photon\ Travel\ Time}{2} \times Speed\ of\ Light,$$

where dividing by two accounts for the round-trip nature of the signal. The VL53L1X communicates with the microcontroller via the I²C protocol. In the system, the ToF sensor is mounted on a 28BYJ-48 stepper motor, which performs a full 360° rotation to scan a complete vertical slice of the environment in the y-z plane. A full scan consists of 32 distance readings taken at 11.25° angular intervals, where the mounted ToF sensor is stopped to take each measurement.

Detailed Description – Visualization

Once the collected data for the slice is transmitted by the user's button press via UART, the python code is able to receive the data via PySerial. The transmitted data consists of a single line of text containing all radial distance measurements from a full 360° scan, separated by spaces as delimiters. This line is parsed by the application, which converts each distance value into a coordinate in 3D space using trigonometric transformations:

$$y = Distance \times \sin(\theta)$$

$$z = Distance \times \cos(\theta)$$

It can be noted that θ comes from the stepper motor angle relative to home positioning, while x is constant for all coordinates in each slice as they're always considered to be vertical slices on the plane and in real life. This results in a full (x, y, z) point cloud that represents the spatial geometry of the scanned environment after all the vertical slices are taken.

Each complete scan represents a vertical slice of the space. As the user manually shifts the scanner along the x-axis between scans, the software updates the cumulative point cloud with each new batch of data. This scan-by-scan accumulation gradually builds a detailed 3D representation of the surrounding area. Open3D's rendering engine displays this data, allowing the user to rotate, zoom, and inspect the model after all scans are inputted. The visualization pipeline ensures that each new scan integrates seamlessly into the global point cloud, simulating a full 3D scan using relatively simple hardware.

This approach provides a cost-effective and flexible alternative to commercial LIDAR systems, particularly suited for mapping indoor environments such as hallways, rooms, or lab spaces. The modular design of the software also allows for further enhancements, such as exporting the generated point cloud for use in other applications like CAD software.

Application Note

This application note describes the design, operation, and practical implementation of a low-cost, embedded 3D spatial mapping system using a VL53L1X Time-of-Flight (ToF) sensor, a 28BYJ-48 stepper motor, and a Texas Instruments MSP-EXP432E401Y microcontroller. The system offers reliable, slice-based depth scanning that allows users to construct accurate three-dimensional models of indoor environments such as hallways, rooms, or lab spaces. By combining sensor readings taken during rotational sweeps and stacking them with incremental displacement, the device reconstructs a full 3D map using simple and readily available components.

The scanning system is ideal for indoor mapping and layout reconstruction, robot navigation mapping, environmental monitoring and spatial awareness in enclosed spaces and educational use in embedded systems and sensor fusion courses. Unlike commercial LIDAR systems, which can be costly and bulky, this system is compact, manually operated along one axis, and offers excellent educational value by providing full visibility into each subsystem with sensor interfacing, motor control, data transmission, and 3D rendering.

Instructions

As part of the following instructions, the assumption is that the necessary components and softwares are already installed. Also, the environments must be configured correctly. Note that python versions 3.6-3.9 are compatible with Open3D, and behaviour cannot be predicted for

other versions. Make sure the python environment used has installed numpy, serial and open3d (can be done with `pip install pyserial numpy open3d`).

1. Connect COM Port
    - Plug the MSP432E401Y microcontroller into your computer via USB.
2. Find the COM Port
    - Press the Windows key and type “Device Manager”, then press Enter.
    - Under the Ports (COM & LPT) section, locate “XDS110 Class Application/User UART (COM #)” and note the COM port number.
3. Configure Python Serial Port
    - Open the Python script (Project_Deliverable.py) provided for visualization with the correct python interpretation environment (right version, libraries, etc.)
    - In the script’s line 8, `with serial.Serial('COM5', 115200, timeout=2) as s:`, change `COM5` to match your COM port number from Step 1.
4. Assemble Circuit
    - Follow the pin mapping found in the Device Characteristics and Schematics sections of this manual to connect:
        - ToF sensor
        - Stepper motor driver
5. Load Code onto the Board
    - Open the Keil project.
    - Click Translate → Build → Load to compile and flash the program.
6. Press the reset button on the LaunchPad to begin execution.
7. Configure Distance Between Scans & Number of Scans in Python
    - In the Python script line 13, `x_increment = 500`, change `500` to reflect the amount of movement between scans you want to simulate (mm)
    - In the Python script line 17, `num_of_scans = 3`, change `3` to reflect the number of slices (scans) you plan to take along the x-axis.
8. Run Python Script
    - Start the Python script to begin listening for serial data and prepare for 3D plotting.
9. Scan a Slice
    - Press Button 1 to rotate the motor and take one full scan (360°).
    - The Measurement LED (D2) will flash to indicate the measurement being taken at each step.
    - The Return Home LED (D3) will turn on to indicate return home action being taken after the scan as to prevent the tangling of the ToF wires.
10. Send Data Transmission
    - Press Button 0 to send the measurement data from the scan via UART.
    - The status LED (D1) will flash to confirm UART transmission being sent.
    - If there was some issue with the scan, the python terminal will let you know. Otherwise, continue with the next step.
11. Displace and Complete Scans
    - Manually move the scanner along the x-axis by a fixed amount (e.g., 30 cm).
    - Repeat the process for each scan necessary.

- Once all scans are complete, the Python script will automatically generate a full 3D model from the received data.

12. Review Output
    - The resulting point cloud will appear in Python as a 3D plot.
    - You can rotate, zoom, and explore the scan within Open3D's viewer.

Expected Output

The expected output can be thought of by looking at the overall shape of the hallway, as well as any obstacles or contours seen. My assigned scan location was G, which was 2nd floor ETB. The below figures include the actual hall and the simulation plot.

Figure 1: Actual Hallway (Start & Full Hall)

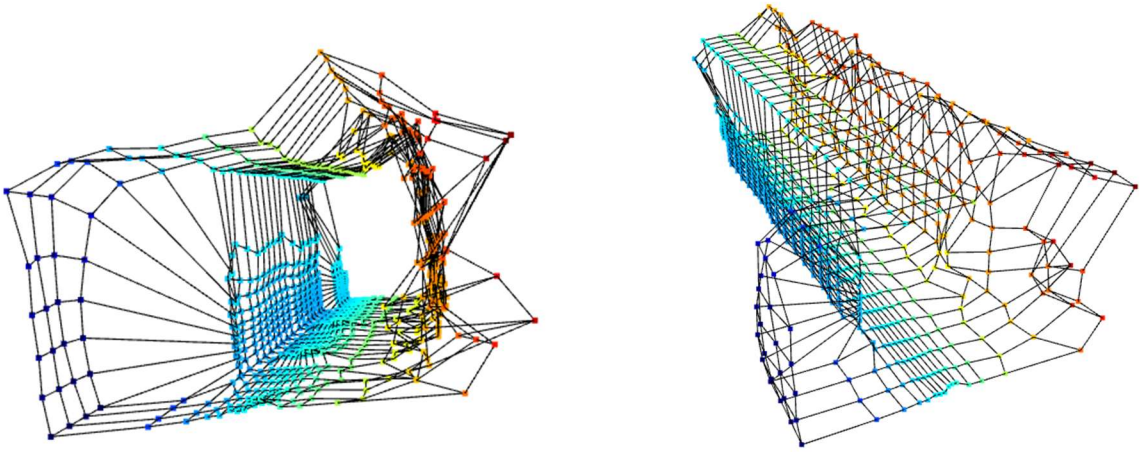


Figure 2: 3D Visual Representation (Front & Isometric)

The first thing to note is the coordinate representation is inverted, as the door from the start seen in figure 1 actually appears on the left in the representation. As can be seen in the 2nd figure, the general hallway shape was mapped out, with the exception of a few outliers. The start of the hall introduced a rectangle-like shape with the door, and the side from which the picture in figure 1 was taken from was completely open. This made the sensor capture some random looking values, while also allowing it to capture individuals walking by at the start. Preceding into the hall, the pillar was skipped due to the increment, but the rest of the walls and roof can be seen to be mapped accurately, with certain things like the roof height change being mapped as a triangle in the figure due to the angle at which the sensor was located. It can be noted that the side of the hall with tables looks very irregular due to the presence of students sitting there at the time of the scan, as well as the many gaps in the tables themselves.

As part of the measuring process, the chair seen near the bins in figure 1 had the system in figure 3 mounted on top for consistent successive scanning.

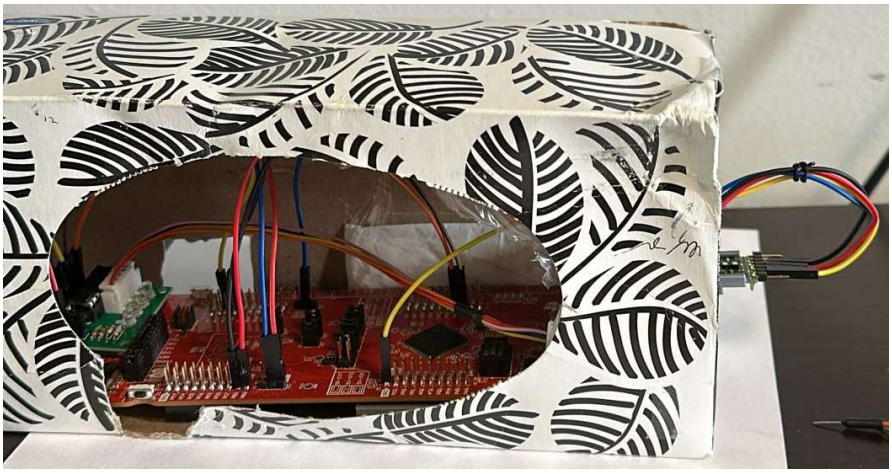


Figure 3: Physical Room Scanner Device

Limitations

The MSP-EXP432E401Y microcontroller uses an ARM Cortex-M4F core, which includes a single-precision Floating Point Unit (FPU). While this allows for efficient 32-bit floating point calculations, it is still not as fast or precise as double-precision floating point. Functions like sin and cos can introduce minor rounding errors and latency in execution, especially during real-time computations of Cartesian coordinates from polar data. Additionally, using trigonometric functions within tight timing loops (e.g., real-time motor stepping) can lead to noticeable slowdowns.

The VL53L1X Time-of-Flight sensor provides digital distance data to the microcontroller and doesn't use a traditional ADC. Due to this, the maximum quantization error is 1mm as this is the same as the resolution.

The maximum serial baud rate supported by the PC was determined by checking the device manager properties of the COM port. The XDS110 Debug UART (used by the MSP432) supports up to 128000 bps. However, for better reliability and buffer handling across platforms,

the project uses 115200 baud. This was verified by Setting the baud rate in the terminal, observing correct and complete data transmission with no dropped packets or garbage characters and ensuring the UART LED indicator flashes only once per transmission.

The communication method used between the microcontroller and the VL53L1X ToF sensor is I²C (Inter-Integrated Circuit). I²C allows for the use of the same data and clock lines with multiple devices, employing a leader-follower relationship to establish transmissions. The system used has an operating data transfer rate of 100kbps. The I²C speed was sufficient for single sensor reads at intervals between motor steps.

The primary bottleneck in the system was the stepper motor speed. Each full scan involved 512 steps per rotation. A delay of ~2 ms was required between steps to ensure proper rotation, and attempting lower delays led to missed steps or erratic motion. This was tested by gradually reducing delay intervals between steps in the motor control loop until instability was qualitatively observed. Even though ToF initialization and data reading add overhead, the motor's mechanical response time dominated total scan duration. As part of the bottlenecks, another primary limitation came from the timing budget of the ToF sensor which was required to be 100 ms [2].

The assigned system clock speed is 14 MHz, based on the least significant digit of the student number and the project spec table. To configure this, the PSYSDIV value was changed from 3 to 33, which was found using the following formula:

$$By\ Default: SYSCLK = \frac{480MHz}{PSYSDIV + 1} = \frac{480MHz}{3 + 1} = 120\ MHz$$

$$For\ SYSCLK = 14\ MHz: PSYSDIV = \frac{480MHz}{14MHz} - 1 \sim \mathbf{33}$$

Circuit Schematic

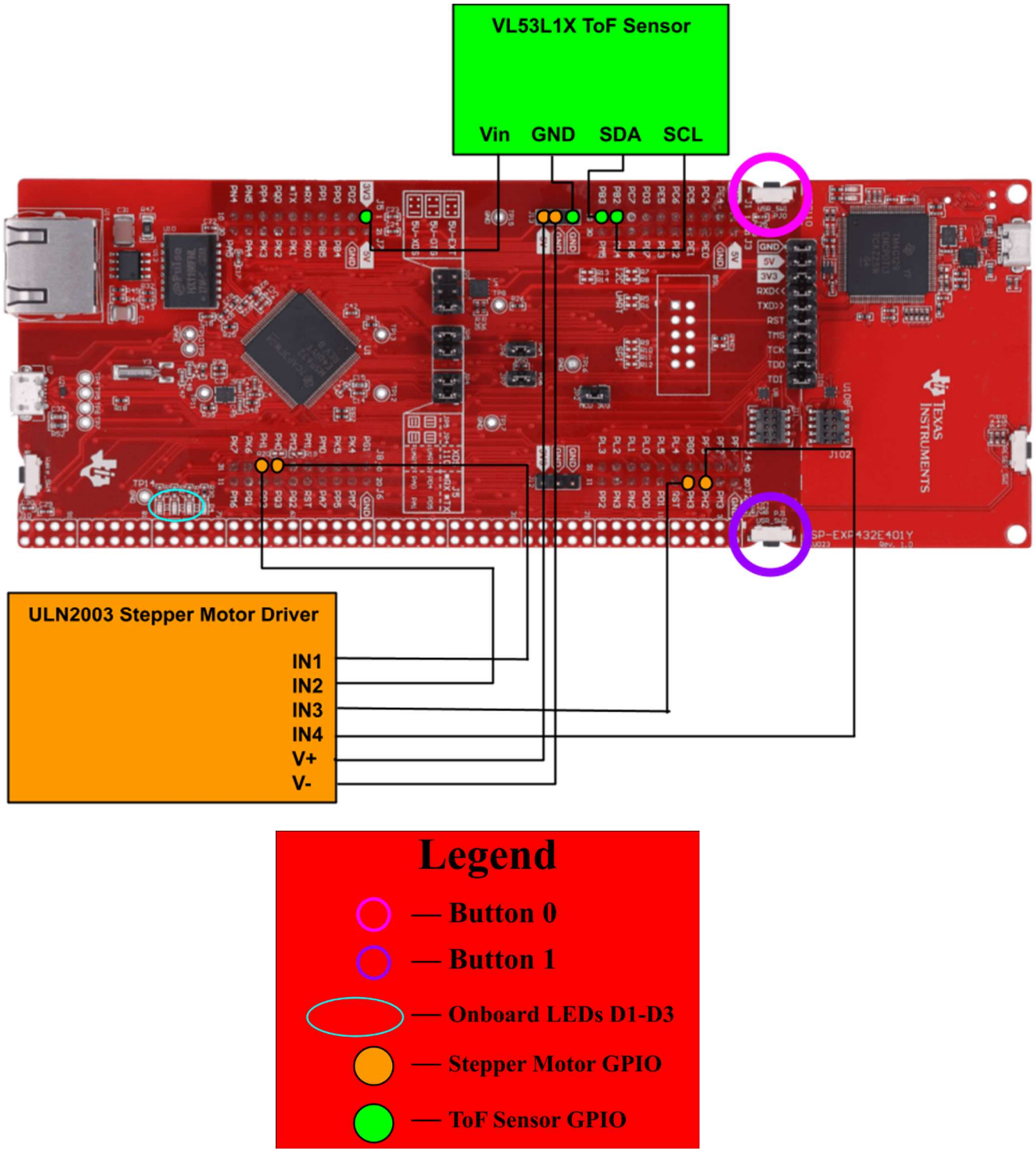


Figure 4: Circuit Schematic & Legend

Programming Logic Flow

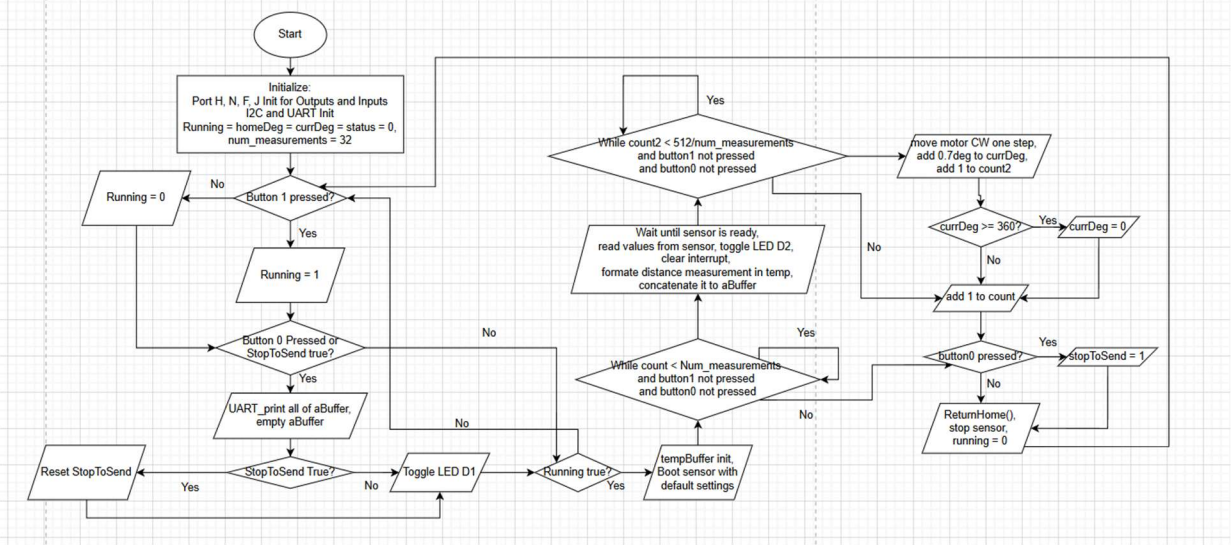


Figure 5: C Programming Logic Chart

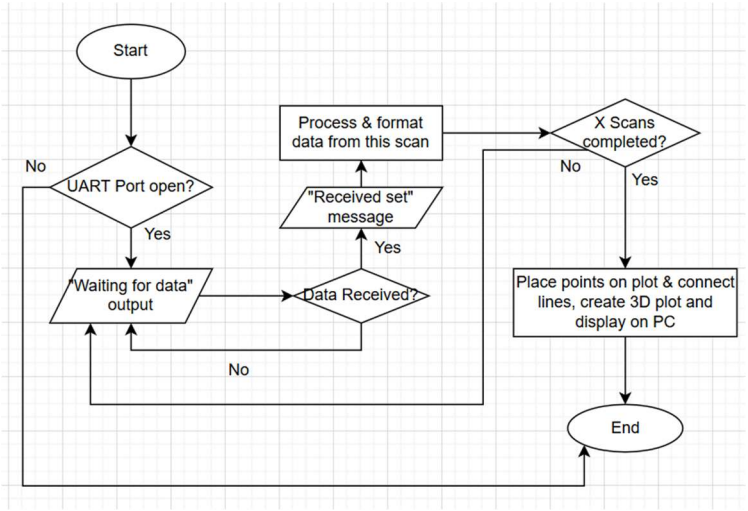


Figure 6: Python Programming Logic Chart

Reference List

[1] Texas Instruments, *MSP432E4 SimpleLink™ Microcontrollers Technical Reference Manual*, Rev. A, Literature Number SLAU723A, Oct. 2017, Revised Oct. 2018. [Online]. Available: https://www.ti.com/lit/pdf/slau723a

[2] STMicroelectronics, *VL53L1X: Time-of-Flight Ranging Sensor Based on ST's FlightSense™ Technology*, DocID031281 Rev. 3, Nov. 2018. [Online]. Available: https://www.st.com/resource/en/datasheet/vl53l1x.pdf