COMPENG 2DX3: Microprocessors System Project Final Report

Instructor: Dr. Yaser Haddara, Dr. Shahrukh Athar, Dr. Thomas Doyle
Alexander Diab-Liu – L03 – diabliua - 400370788

Device Overview

Features

- Fully integrated hallway mapping system with autonomous control and real-time 3D visualization
    - Performs 360° horizontal spatial scanning using a VL53L1X Time-of-Flight (ToF) distance sensor mounted on a stepper motor
    - Stepper motor controlled by a 4-phase sequence through Port H outputs, allowing precise incremental rotation
    - Entire frontal plane is mapped with adjustable timing and motor speed
- VL53L1X Time-of-Flight (ToF) distance sensor
    - Compact module: 4.9 × 2.5 × 1.56
    - 940 nm Class 1 invisible laser for safe ranging in public spaces
    - Ranging distance up to 4 meters, with typical accuracy of ±20 mm in dark indoor settings
    - Supports short, medium, and long-range modes
    - Operates via I²C at up to 400 kHz
- MSP432E401Y microcontroller
    - Clocked via Phase-Locked Loop (PLL) with final system clock derived from 120 MHz base
    - I²C bus configured to 26 MHz
    - Polling-based firmware for timing-critical control of motor and sensor
    - Onboard buttons (PJ0, PJ1) used to continue/stop scans
    - Multiple status LEDs (PF0, PF4, PN0, PN1) used to indicate sensor state and scanning status
- Stepper motor with ULN2003 driver (emulated logic in software)
    - 4-phase stepping sequence across PH0–PH3 controls rotation in 11.25° increments
    - Rewinds automatically after full scan to prepare for next cycle
    - Configurable step delay for balancing speed and precision
- UART data logging and PC integration
    - Transmits live [x, y, z] Cartesian coordinates from microcontroller to host computer
    - Data visualized using a Python script with Open3D to generate 3D point cloud maps
    - Z-axis (height) updated in discrete 30 cm steps after every 3 full scans
    - Python-based GUI or plot can be extended for user interaction or additional filtering
- Compact, low-power, and adaptable embedded system
    - Powered via USB
    - Designed for real-time hallway mapping, people detection, and environment visualization
    - Easily configurable and extensible for use in robotics, smart infrastructure, or other research

General Description

The spatial mapping system is a microcontroller-based embedded device designed to perform automated 3D scanning of indoor environments. The platform combines a VL53L1X Time-of-Flight (ToF) sensor with a stepper motor to capture horizontal spatial information in a full 360° rotation. Vertical displacement is introduced manually at fixed increments, allowing reconstruction of entire sections of hallways or rooms.

At the core of the system is a MSP432E401Y microcontroller, clocked with a bus speed of 26 MHz, configured via PLL from a 120 MHz system clock. The microcontroller manages all I²C communication with the VL53L1X sensor, handles real-time motor control through GPIO ports, and transmits scan data over UARTto a host PC.The VL53L1X sensor features a compact form factor and emits a safe 940 nm Class 1 infrared laser. It can perform distance measurements up to 4 meters with sampling frequencies as high as 50 Hz, and supports multiple distance modes (short, medium, and long). All ToF data are retrieved via I²C communication, with eventual visualization occurring on the connected PC.

Rotational control is implemented in software using a 4-step excitation pattern on PH0–PH3, advancing the stepper motor in 11.25° increments. A full sweep consists of 32 steps, covering 360°, and the angle resets once a complete scan is complete. Scan initiation and termination are controlled by momentary push buttons located on PJ0 and PJ1 respectively. Visual feedback is provided using onboard LEDs (PN0, PN1, PF0, PF4), indicating scan state, UART transmission, and other system-level diagnostics.

Scan data are streamed to a Python script running on a PC, which parses incoming UART packets, reconstructs each polar measurement into [x, y, z] Cartesian coordinates, and generates a real-time 3D point cloud using the Open3D visualization library. After every 3 horizontal scans, the vertical z-height is incremented by 30 mm, allowing stacked depth slices to be rendered as a full 3D representation of the environment.

Block Diagram

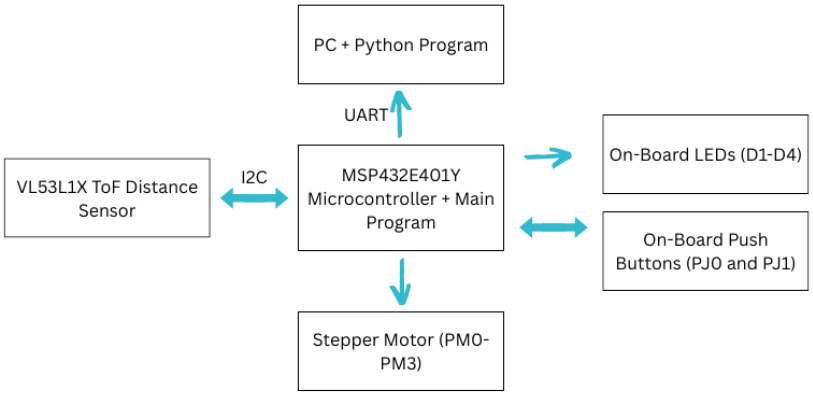

Image 2: Block diagram showing data flow for the designed 3D mapping circuit.

Device Characteristics Table

Component / Parameter	***Specification***
Microcontroller	MSP432E401Y
MCU Clock Source	PLL configured from 120 MHz system clock
Bus Speed (System)	26 MHz (based on student number LSD = 8)
Communication Protocols	I²C (sensor interface), UART (PC communication)
I²C Speed	Fast-mode up to 400 kHz
UART Configuration	8N1 protocol, 115200 baud
Communication Port	Manually entered by user in Python (e.g., COM6)
ToF Sensor	VL53L1X Time-of-Flight sensor
ToF Sensor Range (Dark)	Up to 360 cm
ToF Sensor Range (Ambient Light)	Up to 73 cm @ 200 kcps/SPAD
Laser Emitter	940 nm infrared, Class 1 eye-safe
Sampling Rate	5 samples per 11.25° rotation (approx. 160 points per full sweep)
Stepper Motor Step Resolution	11.25° per scan step (32 steps per full revolution)
Vertical Displacement per z-slice	30 cm manually after 3 scans
Scan Output Format	[x, y, z] Cartesian coordinates transmitted over UART
Visualization Software	Python with Open3D library for 3D point cloud reconstruction
Firmware Control Method	Polling-based main loop (non-interrupt driven)
Power Supply	USB powered (typically 5V input, regulated on board)
Baud Rate	115200
User Controls	PJ0 = Next scan, PJ1 = Stop scan
LED Indicators	PN0, PN1, PF0, PF4 (status, transmission, and diagnostics)

Pins Used	PB2-3 (I²C), PM0–PM3 (motor), PJ0-1 (buttons), PN0, PN1, PF0, PF4
Assumed Displacement per Slice	30 cm (z-axis increment)

Detailed Description

Distance Measurement

The core functionality of the spatial mapping system is driven by the VL53L1X Time-of-Flight (ToF) sensor, a compact and high-precision laser-based distance sensor manufactured by STMicroelectronics. The VL53L1X utilizes a 940 nm Class 1 eye-safe infrared vertical cavity surface-emitting laser (VCSEL) to emit light pulses, which are reflected off nearby surfaces and measured upon return. Unlike traditional IR sensors that rely on reflectivity or ambient light intensity, the VL53L1X measures the actual time of flight of the reflected signal to determine absolute distance, independent of the target's reflectance or color.

The sensor is mounted on a rotating platform controlled by a unipolar stepper motor. The stepper motor is controlled using a 4-phase excitation pattern driven via digital outputs on PH0–PH3, simulating the ULN2003 driver logic in software. Each scan step corresponds to an 11.25° rotation, meaning 32 distinct angle samples form a full 360° frontal scan of the surrounding environment. At each angle, the system performs 5 ToF samples, averaging them for greater reliability. The number of samples and the angular resolution can be adjusted in software to optimize for speed, accuracy, or environmental conditions.

The sensor supports three distinct ranging modes: short, medium, and long, each balancing distance capability and ambient light resilience. In this implementation, long distance mode is used to maximize coverage, yielding up to 360 cm range in ideal (dark) indoor conditions. In ambient light conditions up to 200 kcps/SPAD, the maximum reliable range decreases to approximately 73 cm, a known and expected tradeoff due to signal attenuation from background infrared noise.

Communication between the sensor and the microcontroller occurs over the I^2C interface using PB2 and PB3. The I^2C bus is configured for Fast-mode (up to 400 kHz), and the overall I^2C module operates on a 26 MHz bus

Upon initiating a scan (via PJ0 button), the system performs a full 360° horizontal sweep. After each sweep, the user may manually adjust the sensor height (z-axis) by a fixed displacement — 30 cm per 3 scans — to construct a vertically stacked 3D model of the surrounding environment. At the end of each scan, the motor rewinds to its original position using a full reverse sequence, ensuring consistent alignment between layers. If the user presses PJ1, the system exits the scanning loop and completes its UART transmission cycle.

The system is designed around a polling-based firmware model: rather than relying on hardware interrupts, the firmware loops continuously, checking sensor state, distance availability, and user input in a non-blocking fashion. This choice simplifies timing management and debugging while maintaining responsiveness in a single-threaded embedded environment.

Real-time feedback during operation is provided using onboard status LEDs:

- PF4 (D3): flashes with each measurement sample
- PN0 (D2): toggles to indicate active ranging
- PN1 (D1): lights up to show scan completion
- PF0 (D4): indicates user intervention status (online requires either PJ0 or PJ1 click)

The combination of precise angular control, high-frequency distance sampling, and robust microcontroller-based data handling allows the system to perform low-cost spatial scanning comparable in principle to commercial LIDAR systems.

3D Visualization

Following data acquisition, distance measurements and angle data are converted from polar to Cartesian coordinates within the microcontroller. For each measurement, the angle (tracked in software) is converted to radians, and the following formulas are applied to calculate horizontal positions:

- $x = \text{distance} \times \cos(\theta)$
- $y = \text{distance} \times \sin(\theta)$
- z = vertical level, held constant during a scan, and incremented every 3 sweeps

These 3D points are immediately transmitted via UART over USB to a connected PC. The UART protocol is configured with 8 data bits, no parity, 1 stop bit (8N1) at a 115200 baud rate. Transmission occurs in human-readable ASCII format using standard serial output statements.

On the PC side, a Python script utilizing the pyserial library receives these serial data packets. The script parses the incoming [x, y, z] points and stores them in a dynamic point cloud structure. Once enough points have been collected to form a full scan or z-slice, the script employs Open3D, an open-source 3D visualization library, to render the environment as a 3D model in real time.

This visualization provides a high-level overview of the scanned space, including:

- Wall alignments
- Corners and protrusions
- Major obstructions (e.g., columns, doors)
- Transparent or reflective surfaces (inferred by missing or sparse points)

Open3D allows users to rotate, zoom, and inspect the model interactively. While the system does not provide an absolute scale reference or perform SLAM (simultaneous localization and mapping), it enables users to generate a discretized point cloud of hallway segments or small rooms, useful for navigation, object detection, or experimentation.

The scanned hallway and resulting point cloud are shown side-by-side in the report (see Image 1). Notably, while the right wall appears well-defined in the 3D output, transparent surfaces such as windows result in signal passthrough, a known limitation of ToF sensors that cannot detect returns from low-reflectance or glass targets.

This real-time streaming and reconstruction pipeline demonstrates how low-cost microcontroller platforms can perform meaningful spatial measurements and 3D modeling, providing a foundation for robotics, environmental sensing, or human-assistive technologies.

Application Note, Instructions, and Expected Output

Hardware Setup Instructions

1. Position the device on a level surface within the assigned hallway or indoor environment. Ideally with minimal windows. Ensure the ToF sensor has an unobstructed 360° field of view at its height level.
2. Connect the USB cable from the MSP432E401Y microcontroller to a laptop or desktop PC. This supplies power and establishes UART communication.
3. Launch the Python visualization script provided on the laptop. The script uses pyserial to connect to the microcontroller and Open3D to visualize the 3D point cloud in real-time.
    - Manually select or enter the correct communication port (e.g., COM6 or /dev/ttyUSB0).
    - Set the baud rate to 115200.
4. Confirm sensor readiness: When powered on, the onboard LEDs should briefly flash. This indicates the system is initialized and waiting for user input.

Operational Instructions

1. Start a Scan:
    - Click the Reset Button on the side of the Microcontroller
2. Allow the system to rotate:
    - The stepper motor will rotate in 11.25° steps, collecting 5 samples per step, and averaging the readings.
    - This results in 32 angles × 5 samples = 160 points per sweep.
    - After a full 360 degree scan, the system will rotate counterclockwise, resetting its position
3. Start Next Scan:
    - Press PJ0 (Button 1) to initiate the next 360° scan at the current height (z-level = 0 cm initially).
    - The onboard LEDs will indicate activity:

        - PF0 (D4): toggles during system operation
        - PF4 (D3): flashes with every ToF distance sample
        - PN0 (D2): lights during UART transmission
        - PN1 (D1): lights once a full scan is complete, waiting for user interaction
4. Vertical Scanning (Z-stack):
    - After 3 scans, manually raise the sensor by 30 cm.
    - Press PJ0 again to initiate the next height scan.
    - This process can be repeated to build a full 3D profile of the space.
5. Stop the System:
    - Press PJ1 (Button 2) at any time to gracefully end scanning.
    - The system will rewind the stepper to the starting position, send any final data, and transmit a termination signal ("@") over UART.

Expected UART Output

The system transmits [x, y, z] formatted distance points over UART during operation. An example output stream in the tof_radar.xyz file:

[102.3, -51.4, 0.0]
[97.6, -64.2, 0.0]
[88.1, -80.0, 0.0]
...
[30.4, 95.2, 30.0]
[14.6, 99.7, 30.0]
...
@

- Each coordinate set corresponds to a polar-to-Cartesian transformed point.
- The z value reflects the vertical layer and increments by 30 cm after every 3 scans.
- The final @ symbol indicates the system has finished scanning and all data has been sent.

Expected Visualization Output

The Python script reconstructs the 3D point cloud using Open3D, producing a live, rotatable visualization of the scanned space. This visualization should reveal:

- Wall geometry: straight edges and corners of the hallway
- Protrusions: such as columns or lockers, visible as clusters of dense points
- Openings: like doorways or arches, shown as voids in the point cloud
- Reflective/transparent surfaces: like glass doors, where points may be sparse or missing

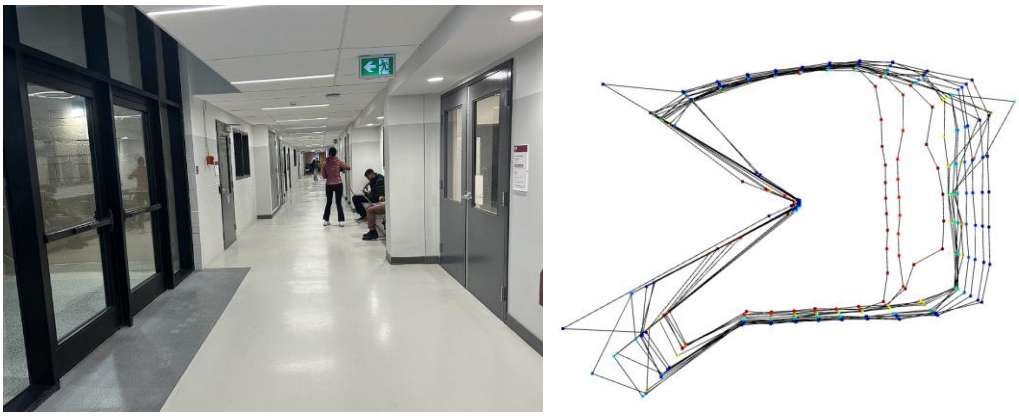

Image 1: Mapped hallway side by side with 3D scanning visualization. The right side of the graph depict the solid wall, with a column poking out at the back. The left side of the visualization falls victim to a key limitation of the device, that being glass windows, as the light passes through the doors with the exception of the handle.

Error Indicators and Troubleshooting

Issue	Likely Cause	Indicator
No data showing on Open3D	Wrong COM port or baud mismatch	No UART LED activity (PN0 off)
Scan hangs at one angle	Sensor I^2C stall or insufficient lighting	D3 LED remains off, no UART output
No power / lights	Faulty USB or board not powered	All LEDs off

Limitations

Trigonometric Functions

The MSP432E401Y microcontroller does not include a hardware floating-point unit (FPU) for trigonometric functions, meaning all such operations are executed in software using the standard C math.h library. As a result:

- Trigonometric functions like sin() and cos() are relatively slow, especially inside real-time polling loops.
- To minimize computation overhead, angles are calculated and converted to radians only once per scan step.
- Precision is acceptable for point cloud reconstruction, but not suitable for applications requiring sub-centimeter angular resolution or extremely rapid sampling rates.

- Integer-based lookup tables could be used to optimize performance in future versions.

Quantization Error of ToF Sensor

The VL53L1X outputs 16-bit distance values in millimeters. This means the sensor cannot distinguish between two objects that differ in distance by less than 1 mm. However:

- Real-world accuracy varies based on distance mode and environmental conditions.
- In **long distance mode**, the datasheet lists typical errors of **±20 mm** under dark ambient conditions and up to **±25 mm** in high ambient light (200 kcps/SPAD).
- Error is exacerbated by surfaces with low reflectivity (e.g., glass, black cloth) or extreme angles of incidence.

Measurement Interference

A significant environmental limitation of the system is its inability to accurately measure distances to transparent or highly reflective surfaces, such as:

- Glass windows or doors
- Glossy whiteboards
- Chrome or polished metal objects

This issue stems from the underlying Time-of-Flight (ToF) principle. The VL53L1X relies on detecting photons reflected back from a surface. Transparent materials (like glass) allow most of the laser light to pass through, while reflective materials can cause scattering or misalignment, resulting in:

- No return signal (measurement dropout)
- Incorrect distance values (often maximum range)
- Spatial gaps in the 3D point cloud visualization

Observed Behavior:

- In hallway tests, glass window panels behind interior doors did not register measurable distances.
- Door handles and opaque surfaces (e.g., drywall) were accurately captured.
- In the 3D output, affected areas appear as “blank spots” or sparse regions in the point cloud.

Workaround:

- This behavior is expected and documented in the VL53L1X datasheet.

- Users should interpret such gaps as either transparent objects or regions beyond measurable range.
- Future iterations may use multi-sensor fusion (e.g., ultrasonic, camera, or dual ToF) to resolve this issue.

Circuit Schematic

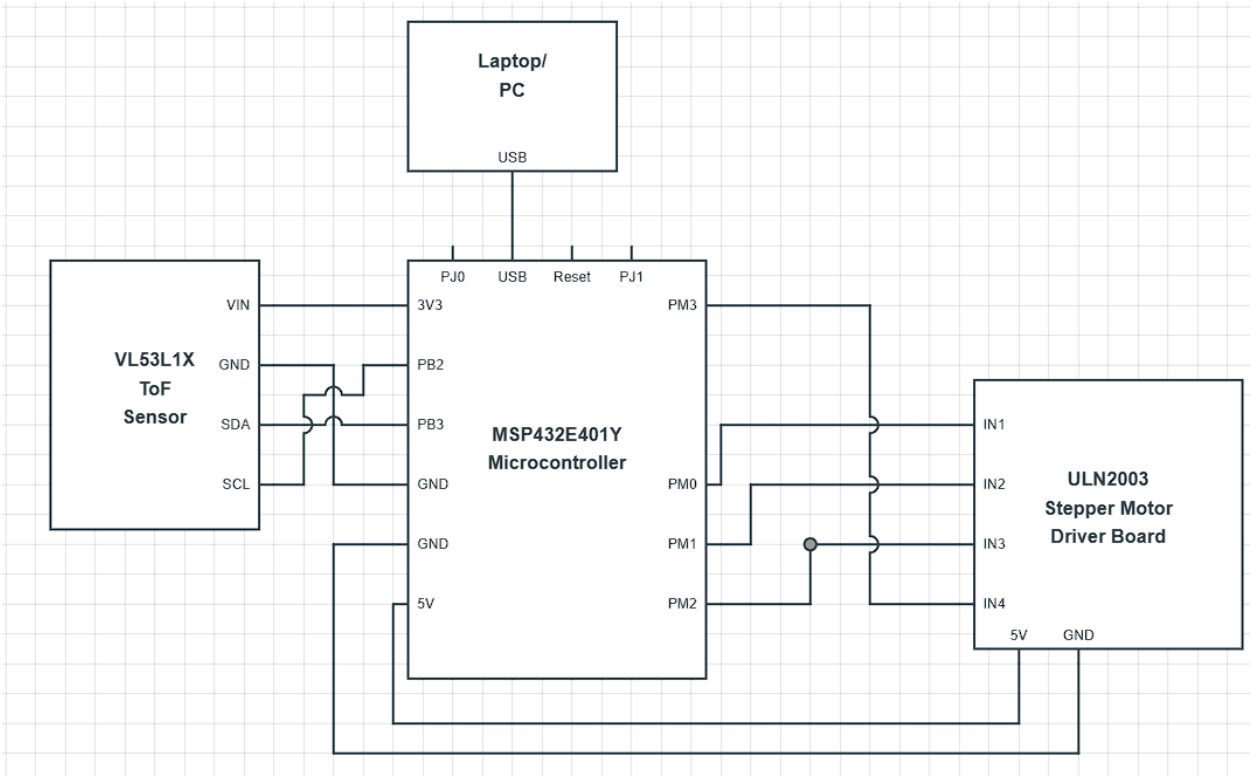

Image 2: Circuit Schematic showcasing the port connections on the device

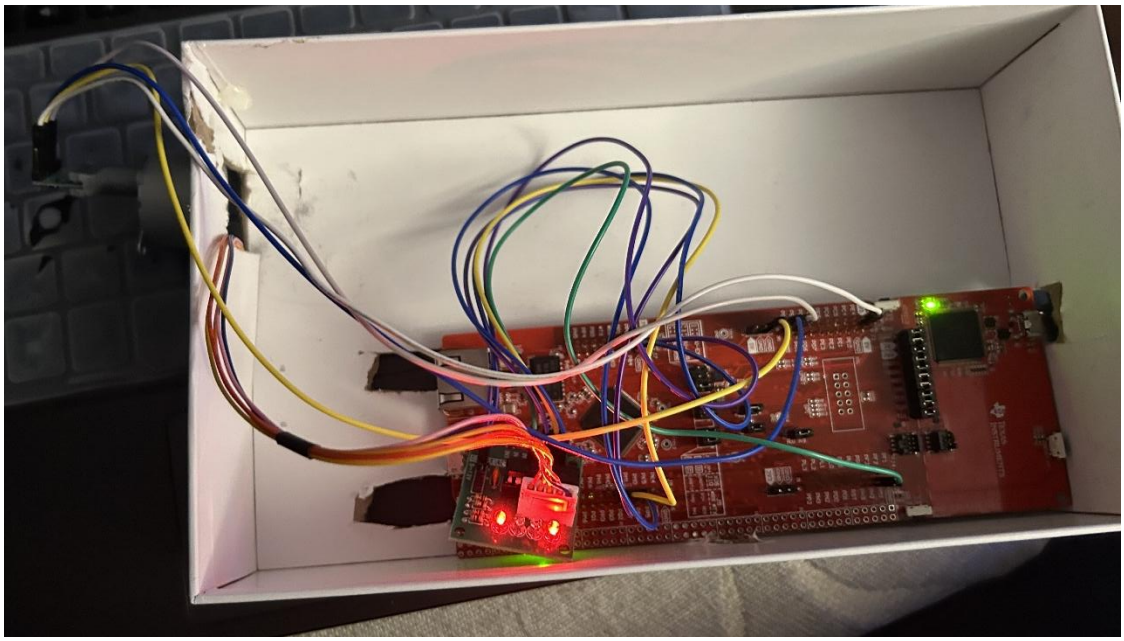

Image 3: Physical circuit build of above circuit schematic

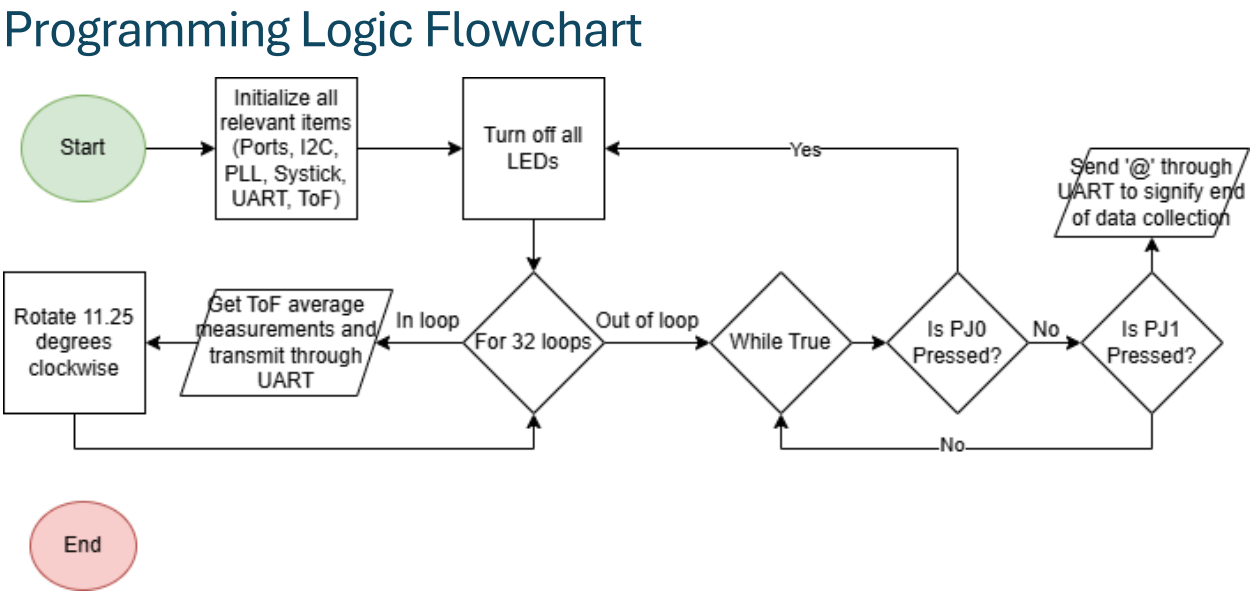

Image 4: Flowchart for microcontroller program

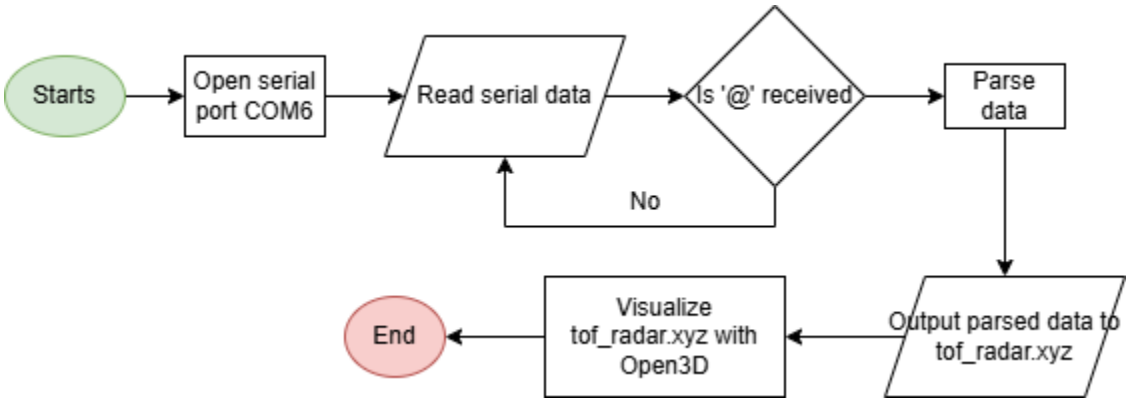

Image 5: Flowchart for Python program on Laptop/PC