2DX3 – Final Deliverable Report
JARV0203

By: Vraj Patel – patev61 - 400434795

I hereby declare that the work presented in this report is original, and any external sources or materials used have been duly cited in accordance with the accepted standards of academic integrity and ethical research practices.

DEVICE OVERVIEW – JARV0203

GENERAL DESCRIPTION

Introducing the JARV0203 LiDAR 3D Hallway Scanner, an innovative solution tailored for comprehensive spatial mapping of indoor environments, particularly corridors. This state-of-the-art device optimizes the scanning process with a simple keystroke, automating data capture and reducing manual intervention for heightened efficiency. Powered by the robust Texas Instruments MSP432E401Y microcontroller, acclaimed for its high-speed performance and seamless communication capabilities, the JARV0203 ensures reliable operation.

The JARV0203 scanner integrates intuitive keyboard controls, furnishing users with a user-friendly interface to initiate scanning sequences and effortlessly log data. By incorporating keyboard operations, the device eliminates the need for additional manual controls, enhancing both functionality and aesthetic appeal. Equipped with advanced components including the VL53LIX Time of Flight Sensor and the 28BYJ-48 Stepper Motor, the JARV0203 guarantees precise and dependable scanning results. Moreover, it supports serial communication using both I2C and UART protocols, enabling seamless data transfer and seamless integration with external systems. Additionally, leveraging Python for 3D visualization simulation, the device offers users a comprehensive and interactive mapping experience.

FEATURES

Automatic Scanning – From only a click on a button on your keyboard, the JARV0203 will automate the scanning process reducing any manual work needed.

Technical Parts – Utilizes strong, yet accessible, hardware components such as the MSP-EXP43E401Y (Microcontroller), the VL53L1X (ToF sensor), and 28BYJ-48 (Stepper Motor).

Amazing Visualization - By using Python packages like Open3d, visualizing data has become automated as well, showing precise plots from precision mapping utilizing the ToF sensor.

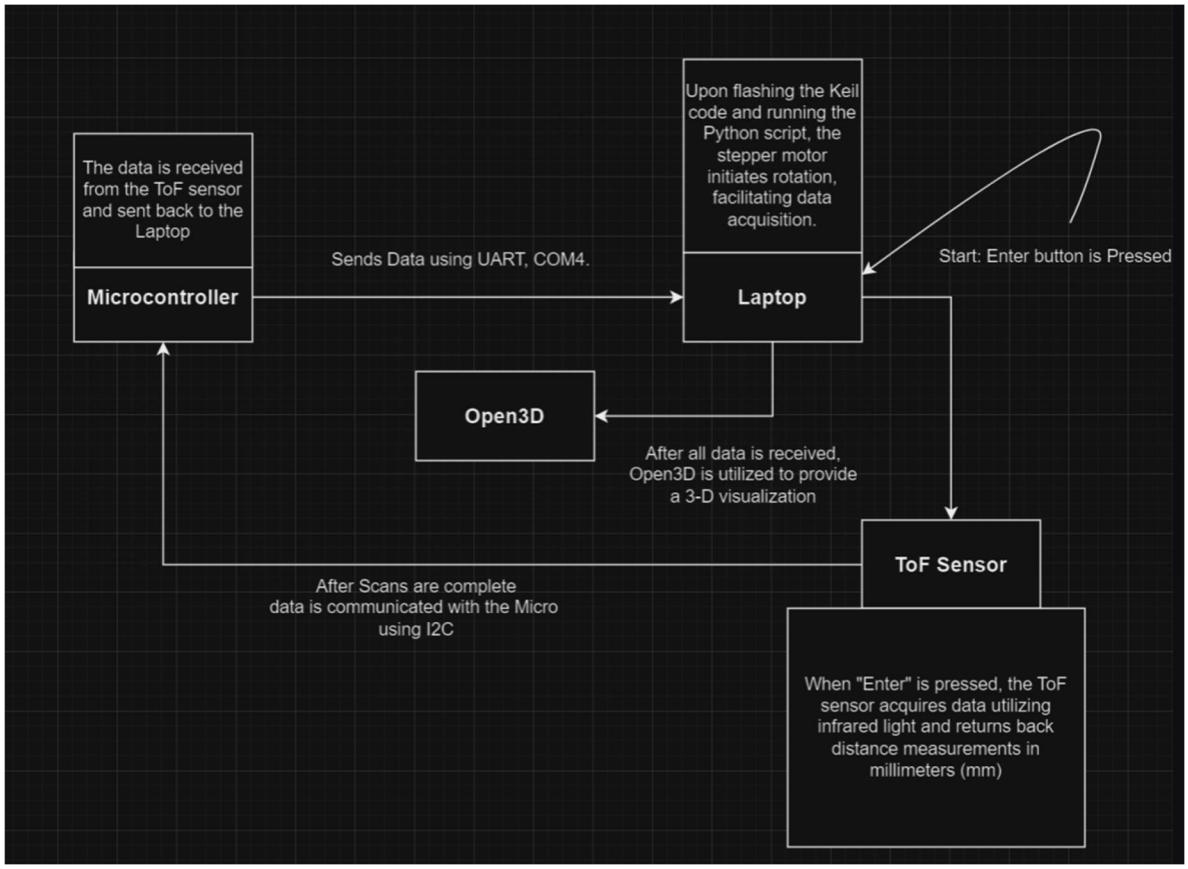
HARDWARE SPECIFICATIONS

MSP-EXP43E401Y (Microcontroller) – ARM Cortex M4F processor, 256KB SRAM, 6K EEPROM, accepts input from 3.3V-5V, 30MHz bus speed, Onboard LEDs, 115200 Baud Rate

VL53L1X (ToF sensor) – Measures up to 4000mm, Utilizes Infrared Light, Communicates with Micro using I2C Protocol

28BYJ-48 (Stepper Motor) – Uses electromagnets to move, 2048 total steps, 11.25 deg scan step

BLOCK DIAGRAM



Block Diagram for JARV0203

Device Characteristics

Microcontroller (MSP-EXP43E401Y)		Stepper Motor (28BYJ-48)		ToF (Time of Flight) Sensor (VL53L1X)	
Clock Speed	30 MHz	**In 1 – In 4**	PH0 - PH3	**V_{in}**	3.3V
Baud Rate	115200 BPs	**V_+**	3.3V	**GND**	GND
Serial Port	COM4	**V_-**	GND	**SDA**	PB3
Measurement & Status LED	PN1, PF4			**SCL**	PB2

Processor	Cortex M4F				

DETAILED DESCRIPTION – DISTANCE MEASUREMENT

The Time-of-Flight (ToF) sensor operates by utilizing an infrared laser light emitted from one of its components, often denoted as the "yellow block," while the other block receives the reflected light. This mechanism enables the sensor to calculate distances in millimeters. Distance calculations are based on the time it takes for the emitted light to reflect back to the sensor. Utilizing the speed of light, the sensor employs the formula $D = \frac{Reflection\ Time}{2} c$, where c represents the speed of light, to determine the distance from the object being measured.

The sensor is typically equipped with four separate connections: Vin (Voltage In), GND (Ground), SDA (Serial Data Line), and SCL (Serial Clock Line). These connections facilitate communication between the ToF sensor and the microcontroller, often adhering to the I2C protocol. Within this protocol, data exchange occurs via the SDA line, while synchronization of clocks is ensured through the SCL line. Following data acquisition, the sensor proceeds through transduction, conditioning, and Analog-to-Digital conversion processes to translate the collected data into a digital format suitable for effective utilization by the microcontroller. To configure the sensor, VL53L1X_SensorInit is called upon and to start data acquisition, the code enters a loop to collect distance measurements and waits for transmission initiation code “s” from the PC.

To facilitate the room scanning process, the ToF sensor is mounted onto a stepper motor using a custom-designed 3D-printed attachment. The stepper motor board, featuring a ULN2003 controller, is integrated with the microcontroller via specific pins outlined in the Device Characteristics table. The stepper motor employs a set of four electromagnets, housed within the motor assembly, to enable rotation. These electromagnets are strategically activated and deactivated sequentially to induce movement. By alternating the polarity of these magnets, the motor can rotate either clockwise or counterclockwise. A full-stepping technique is employed, ensuring that two adjacent magnets with alternating polarities are consistently energized. This method facilitates smooth and controlled movement of the motor, enabling precise positioning during the scanning process.

Once the microcontroller receives the transmission from the PC, which is done by clicking enter after the python script is ran, signified by the reception of the 's' character, the motor connected to the ToF sensor begins its rotation. The motor is programmed to rotate in steps of 11.25 degrees, pausing for 5 milliseconds at each step to ensure the ToF sensor has sufficient time to perform a scan of the surroundings. During each pause, the onboard LED, specifically designated

as PN1, flashes to indicate to the user that the ToF sensor is actively scanning the environment. This scanning process continues until the motor completes a full 360-degree rotation, allowing for a total of 32 measurements to be taken from different perspectives. Once the full rotation is achieved, the motor reverses its direction briefly to untangle any wires, ensuring smooth operation and longevity of the device for future use.

DETAILED DESCRIPTION – VISUALIZATION

Once the scans are completed and data is acquired from the Time-of-Flight (ToF) sensor, the microcontroller utilizes UART to establish communication with the laptop at COM4, initializing the port for data exchange. This data, consisting of distance measurements obtained from the ToF sensor during the scanning process, serves as the foundation for creating a comprehensive spatial map of the environment. By transmitting this raw data to the laptop, the microcontroller facilitates further processing and visualization, enabling a deeper understanding of the scanned area's layout and structure.

Upon receiving the data, the laptop initiates the processing phase using Open3D, a powerful Python package tailored for three-dimensional visualization tasks. Open3D offers a rich set of functionalities for processing and manipulating point cloud data, making it an ideal tool for interpreting the distance measurements obtained from the ToF sensor. Leveraging its capabilities, the laptop transforms the raw distance data into a structured point cloud representation, where each point corresponds to a specific location within the scanned environment. This structured point cloud serves as the foundation for generating a detailed spatial map, laying the groundwork for accurate visualization and analysis.

Within the Open3D framework, the processed point cloud data is utilized to create a visually compelling representation of the scanned environment. By connecting the points with lines or surfaces, Open3D constructs a three-dimensional model that accurately reflects the structure of the scanned area. This model provides valuable insights into the spatial layout, dimensions, and features of the environment, empowering users to make informed decisions and assessments based on the visualized data. Additionally, Open3D offers a range of customization options and visualization techniques, allowing users to tailor the output to their specific needs and preferences, thereby enhancing the overall interpretability and usability of the generated spatial map.

APPLICATION NOTE

This product leverages LiDAR technology, a method for determining distances by emitting laser beams toward objects or surfaces and measuring the time taken for the reflected light to return to the receiver. Implemented from scratch, this LiDAR application utilizes three key hardware components: the microcontroller, stepper motor, and Time-of-Flight (ToF) sensor.

LiDAR finds diverse applications across various fields, particularly in modern technology. In archaeology, for instance, LiDAR replaces traditional excavation methods by employing infrared light to scan and map cities accurately, revealing structures otherwise concealed. Similarly, in contemporary technology and Artificial Intelligence, LiDAR plays a pivotal role, notably in autonomous vehicles. Here, LiDAR systems are integral for accurately measuring the vehicle's surroundings, a critical aspect for autonomous operations. The precision demanded in such applications necessitates advanced LiDAR technology, ensuring precise distance measurements between the vehicle and surrounding objects.

INSTRUCTIONS

These instructions presuppose that the required software is installed on your device and that you possess the necessary equipment. Additionally, it is presumed that the circuit has been constructed according to the schematic diagram.

1. Begin by connecting the MSP-EXP43E401Y microcontroller to your personal computer via the USB-C port. Access the Device Manager to identify the COM port associated with both the microcontroller and the computer.

2. Launch your preferred integrated development environment (IDE) and open the necessary files, including the Python script and the Keil folder.

3. Adapt the Python script to utilize the appropriate COM port of your personal computer, ensuring compatibility with your system configuration. Make sure it is UART COM

4. Within Keil, proceed to translate the code, build it, and then flash it onto the connected microcontroller. Utilize the buttons located at the top-left corner of the screen to perform these actions. Once flashing is successful, press the reset button on the microcontroller.

5. After the onboard LEDs flash upon pressing the reset button, execute the Python script. Verify that no errors occur during execution.

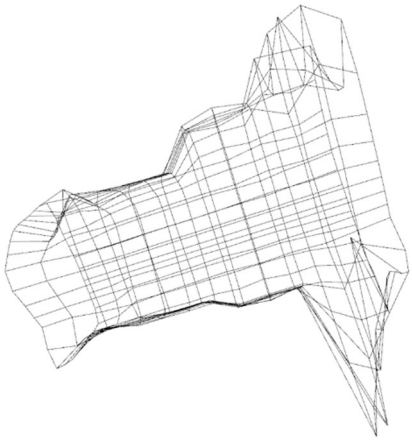
6. Follow the instructions provided by the Python script and press the "Enter" key as prompted.

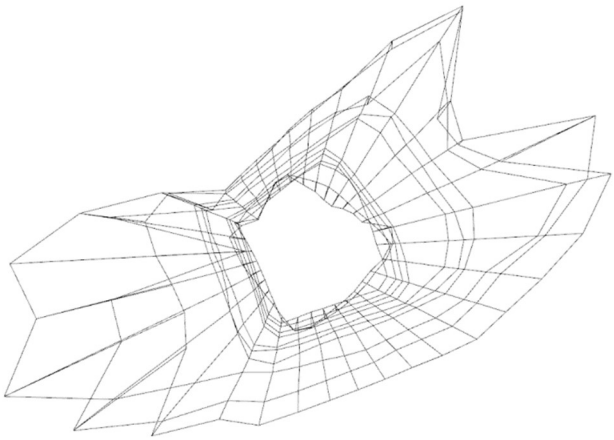
7. Position the microcontroller in such a way that the wires do not obstruct the scanning process. Allow the motor to complete its 360-degree motion and return to its initial position.

8. Repeat steps 6 and 7 as necessary, depending on the desired number of scans.

9. Upon completion of the scans, Open3D should automatically launch with the visualization of coordinates. If a visualization with lines is preferred, close the initial Open3D pop-up, and another window with the desired visualization should open automatically.

EXPECTED OUTPUT





Completed Scan vs Reference Picture

LIMITATIONS

While the MSP-EXP43E401Y microcontroller integrated LIDAR project offers advanced capabilities for precise environmental sensing and mapping, it is essential to understand the inherent limitations of the system. These limitations encompass various aspects, including computational constraints, memory limitations, mechanical inaccuracies, sensor noise, and environmental factors. Addressing these limitations is crucial for optimizing the performance and reliability of the LIDAR system. In this section, we explore the potential constraints and challenges that users may encounter when utilizing this integrated solution, along with recommendations for mitigating these limitations to achieve optimal results.

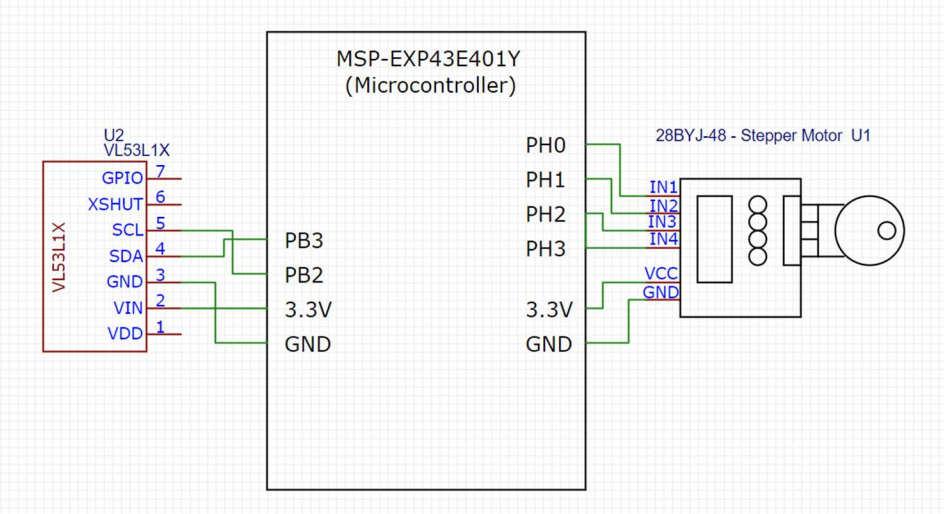
When the ToF sensor sends data to the microcontroller, it is in digital values such that the microcontroller can utilize the data properly. In order to covert digital values into analog, an Analog-to-Digital Converter (ADC) is used, which comes built into the microcontroller. When converting these values, there is a small discrepancy between actual continuous signal and its digital representation due to the finite precision of the ADC, this small discrepancy is called Quantization Error This error arises because the sensor's analog signals are converted into discrete digital values, leading to potential deviations between the measured distances and their true values. It's crucial to consider and minimize quantization error to ensure accurate spatial mapping and visualization of the environment during data acquisition and processing. In order to calculate the maximum deviation, the following calculations are done.

$$Max\ Quantization = \frac{Max.Reading}{2^{ADC\ bits}} = \frac{4000mm}{2^{16}} = 0.061mm$$

According to the calculation above, it is now known that the quantization error is 0.061mm per bit. This is based on the fact that the ToF sensor has a maximum distance range of 4m which corresponds to 4000mm.

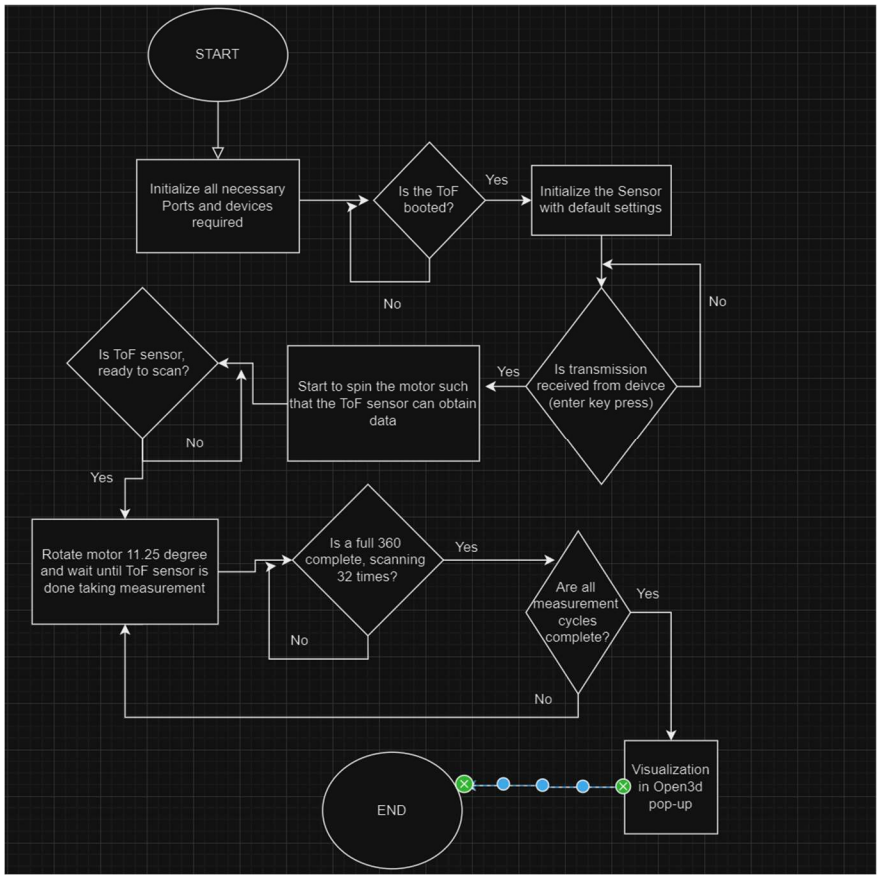
The efficiency of data transmission among devices is notably influenced by the maximum communication speeds supported by their respective protocols. For instance, the communication between the Time-of-Flight (ToF) sensor and the microcontroller is limited to 100 kbps, which significantly impacts the rate of data transfer between these components. Similarly, constraints are encountered in the baud rate utilized for communication between the two devices, with the microcontroller imposing a maximum baud rate of 115,200 bps due to its inherent capabilities. Additionally, timing constraints present significant challenges, as digital systems can swiftly handle data transmission, whereas physical components require more time for execution. This discrepancy becomes evident when attempting to increase the motor speed beyond its operational capacity, resulting in failure to spin.

CIRCUIT SCHEMATIC



Circuit Schematic

PROGRAMMING FLOWCHART



Programming Flowchart