COMP ENG – 2DX3

Spatial Mapping Using Time-of-Flight

By: Erion Keka, 400435050

Professors: Sharukh Athar, Thomas Doyle, and Yaser Haddara

04/17/2024

Table of Contents

Table of Figures

Features

The Integrated Lidar System features a simple interface. The system features two on-board LEDs and an on-board button for the microcontroller. The LEDs are used to indicate a) the state of the system and b) the step of the stepper motor's rotation whereas the button is used to initiate the system.

- Texas Instruments (TI) MSP432E01Y Microcontroller
  - 32-Bit ARM Cortex-M4F Processor Core
  - 1024 kBytes of Flash Memory
  - 256 kBytes of SRAM
  - 80 MHz Bus Speed
  - GPIO Pins, On-Board Buttons and LEDs
- VL53L1X Time of Flight Sensor
  - Range: Up to 4 Meters
  - Ranging Error: ±20 mm
  - 940 nm Laser Emitter
  - Ranging Frequency: Up to 50 Hz
  - Operating Voltage: 2.6 – 3.5 V
- 28BYJ-48 Stepper Motor
  - 512 Steps per 360° Rotation
  - 4 Phases per Step
  - 4 On-Board LEDs
    - State of Current Phase
  - Operating Voltage: 5 – 12 V
- Serial Communication
  - I2C Protocol
    - Communication between Time of Flight Sensor and Microcontroller
  - UART Protocol
    - Communication between Microcontroller and PC
  - Baud Rate: 115200 BPS
    - Utilized in UART Communication
- Data Visualization
  - Open3D Python Library
    - Data processing and visualization

General Description

The Spatial Mapping using Time-of-Flight Project is an integrated 3D scanning system. The system is an economic, efficient, and simple system that can acquire distance every 11.25° of a full 360°rotation. The system consists of the TI MSP432E01Y Microcontroller, VL53L1X Time of Flight Sensor, and the 28BYJ-48 Stepper Motor. At the heart of the operation is the TI MSP432E01Y Microcontroller, it is responsible for ensuring that each operation proceeds and ensuring an accurate bus speed of 80MHz. The 28BYJ-48 Stepper Motor allows for a 360° range of motion to enable the time-of-flight to record distance measurements at each angle. The VL53L1X Time of Flight Sensor is responsible for recording the distance measurements at each 11.25° interval. The sensor calculates the distance by emitting and receiving an infrared beam of light and measuring the time it took the beam to reflect off a surface and return to the sensor. The sensor performs the following calculation: $Distance = \frac{Flight\ Time}{2}\ X\ Speed\ of\ Light$ to determine the distance. The sensor then transfers the data to the microcontroller through I2C communication which is then transferred to the PC via UART communication. The data transferred onto the PC is initially converted into Cartesian Coordinates through trigonometric formulas then is visualized using the Open3D library in Python. This data is plotted in a mesh-like point cloud and displayed to the user.

Block Diagram

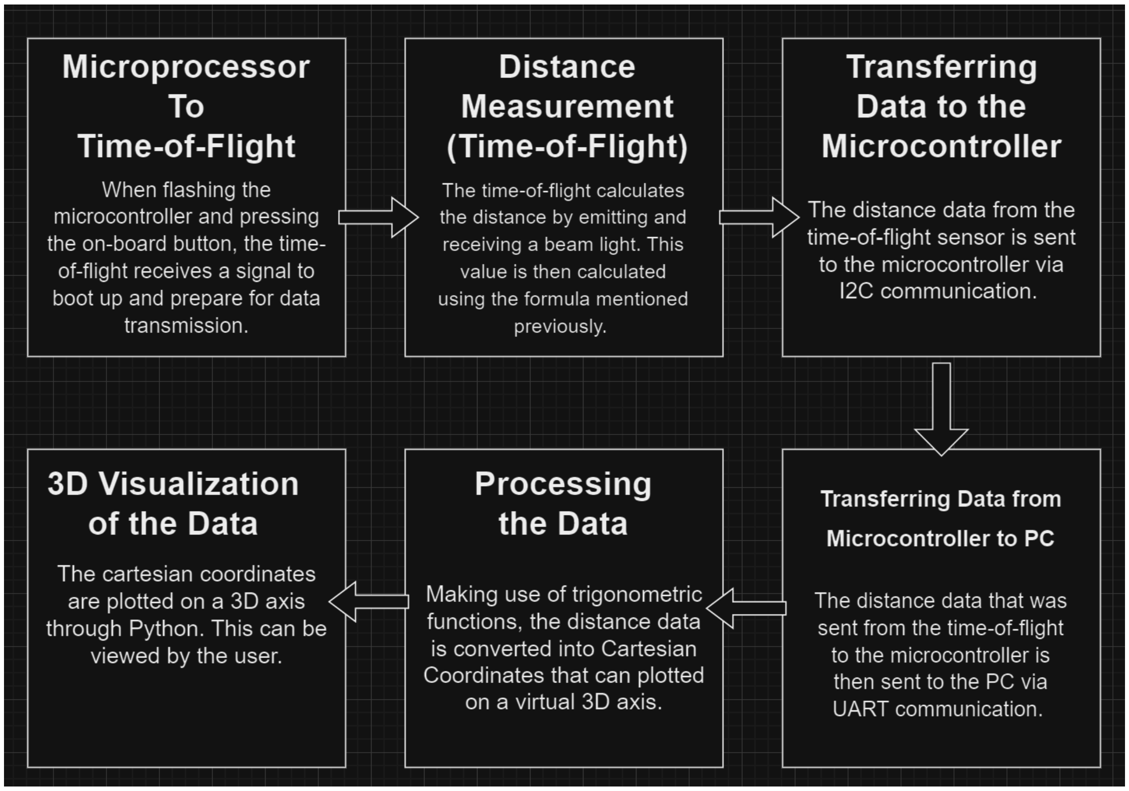
Microprocessor To Time-of-Flight
When flashing the microcontroller and pressing the on-board button, the time-of-flight receives a signal to boot up and prepare for data transmission.
Distance Measurement (Time-of-Flight)
The time-of-flight calculates the distance by emitting and receiving a beam light. This value is then calculated using the formula mentioned previously.
Transferring Data to the Microcontroller
The distance data from the time-of-flight sensor is sent to the microcontroller via I2C communication.
Transferring Data from Microcontroller to PC
The distance data that was sent from the time-of-flight to the microcontroller is then sent to the PC via UART communication.
Processing the Data
Making use of trigonometric functions, the distance data is converted into Cartesian Coordinates that can plotted on a virtual 3D axis.
3D Visualization of the Data
The cartesian coordinates are plotted on a 3D axis through Python. This can be viewed by the user.

Figure 1: Block Diagram

Device Characteristics

TI MSP432E01Y Microcontroller		**28BYJ-48 Stepper Motor**		**VL53L1X Time of Flight Sensor**	
Clock Speed	80 MHz	**V+**	3.3 V	**VDD**	N/A
Measurement LED	PN1	**V-**	GND	**VIN**	3.3 V
Status LED	PF4	**IN1**	PH0	**GND**	GND
On-Board Button	PJ1	**IN2**	PH1	**SDA**	PB3
Serial Port	COM4	**IN3**	PH2	**SCL**	PB2
Baud Rate	115200 BPS	**IN4**	PH3	**XSHUT**	N/A
				GPIO1	N/A

Figure 2: Device Characteristics

Distance Measurement

The distance measurements are gathered through the VL53L1X Time-of-Flight Sensor. The sensor takes a measurement at each step of the stepper motor, this is done as the stepper motor performs a full rotation. The VL53L1X Time-of-Flight Sensor emits an infrared beam of light through its “Emitter”, this beam reflects off an object placed within close proximity to the sensor. Upon the reflection of the beam of light, the “Receiver” of the VL53L1X Time-of-Flight Sensor receives the beam. The VL53L1X Time-of-Flight Sensor measures the delay between the emission and the receiving of the infrared beam. Using this measurement, the distance between the sensor and the object in-front is determined through the following formula: $Distance = \frac{Flight\ Time}{2}\ X\ Speed\ of\ Light$. Following the completion of this step, the VL53L1X Time-of-Flight Sensor performs the necessary operations to allow for the data to be transmitted to the microcontroller via the I2C Communication Protocol. These operations include the transduction stage, conditioning stage, and the ADC (Analog-to-Digital Conversion) stage. These various stages allow for a digital representation of analog distance measurement. To utilize the VL53L1X Time-of-Flight Sensor, the activation of the sensor and the initiation of the ranging process must be performed within the Keil C program. Initially, the sensor’s boot function is called within the main program of the C program. This allows for the sensor to become active and ready to calculate distance measurements; however, the C program has been programmed to wait for an interruption by the on-board button, PJ1. This has been programmed through a combination of polling and interrupt-based programming. The on-board push button is utilized to initiate the system and begin scanning. Upon the press of the push button, the data transmission begins in which we utilize the API functions from the time-of-flight sensor to retrieve data from the sensor; however, both the depth and step are determined through the program in which the step is incremented along the step. This allows a calculation of the angle of the motor. The depth is determined through an incrementation of each full rotation of the motor. This emulates the scanning depth. Upon the completion of this process, the data is transmitted to the microcontroller through I2C communication then, from the microcontroller to the PC through UART communication.

Visualization

After transmitting the data from the microcontroller to the PC via UART Communication, Python completes the process. Within the Python program, various libraries are utilized to manipulate the data and transform it into a 3D visualization. Through the Serial Library in Python, the program begins by connecting to the port, COM4 on the PC. This port becomes open to Serial Communication. The data being transmitted must be manipulated prior to being plotted. Utilizing the distance measurement, we determine the angle by determining which increment of the scanning process we are in. The Z coordinate is determined by the increasing step value whereas to determine both the X and Y values, the distance and angle must be converted into Cartesian Coordinates. To convert the distance and the angle, the following formulas are used: $x = distance * \cos(angle)\ and\ y = distance * \sin(angle).$ Upon the completion of the scans, the serial connection is closed and the collected XYZ coordinates are converted into a point cloud through the Open3D library. A mesh-like structure is created by connecting each of the points within each of the scans and between the consecutive scans.

Application Note

To ensure successful utilization of the integrated lidar system, the user must be certain that the necessary software has been successfully installed on their PC. Without the installation of the necessary software, the integrated lidar system may not be capable of functioning correctly. Follow the steps below to ensure the user's PC is configured correctly. Please Note: The following procedure is performed on Windows 11 PCs.

1. Download and install the latest version of Keil. This is utilized as an environment for our C program. It also allows for the translation, building, and loading of the C program onto the TI MSP432E01Y Microcontroller. Keil can be downloaded from the following link: https://www.keil.com/download/.
2. Download and install version 3.8 of Python. The release can be found here: https://www.python.org/downloads/release/python-380/. Follow the setup process and ensure that Python 3.
3. 8 has been added to the Path. The Path can be accessed by searching for "Environment Variables" in the Windows' Search Bar then entering "Environment Variables…" then double-clicking "Path" under the "System Variables" tab.
4. Upon installing Python 3.8, we must also install Pip which is used to install the necessary Python Libraries. To install Pip, open Command Prompt and run it as an administrator. When prompted to enter a command, type the following "curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py" and hit enter. Upon the completion of this command, type the following command "python get-pip.py", this will install the Pip. Again, it must be ensured that Pip has been added to the Path.

5. Upon completion of the Pip installation, open Command Prompt again and run it as administrator. To install the required Python Libraries; pySerial, Math, NumPy, Open3D, and Time. These libraries can all be installed through Pip installation. To install pySerial, type the following command “pip install pyserial”. To install Math, type the following command “pip install python-math”. To install NumPy, type the following command “pip install numpy”. To install Open3D, type the following command “pip install open3d”. To install Time, type the following command “pip install Library-Time”. These installations cannot be completed simultaneously and must be done individually.

Instructions

Upon completion of the installation setup, the integrated lidar system is ready to be utilized alongside our PC. To utilize the integrated lidar system, the steps are as follows:

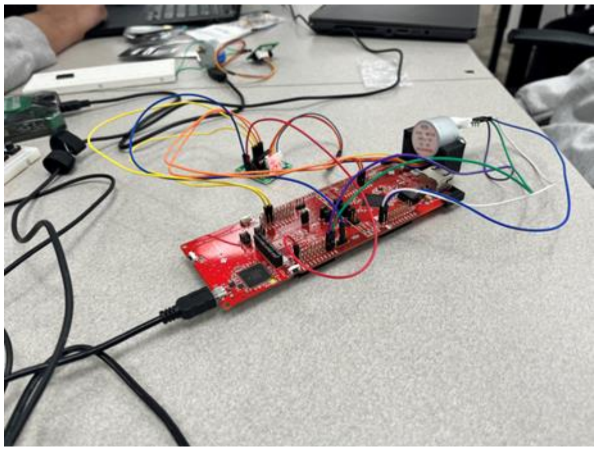
Figure 3: Physically Constructed Circuit

1. Connect the microcontroller to the PC. Upon connecting the microcontroller, the COM port must be determined. To determine the COM port, enter Device Manager and locate “Ports (COM & LPT)”, expand the drop-down menu. Take note of the value in the # position in the following item: XDS110 Class Application/User UART (COM#). This is the COM Port that will be used.
2. Open the Python program, set the correct COM Port in the following line of code: “s = serial.Serial('COM#', 115200, timeout=30)”. Again, your value should be placed in the # position in the line of code. This completes the serial communication setup. Both the total amount of scans performed and the number of steps per rotation can be modified directly within the Python code by following the comments within.
3. Setup the physical circuit by following the pinout table under “Device Characteristics”.
4. Open the Keil Project. In the Keil Project, translate, build, and load the code onto the microcontroller. Then, flash the Keil Project onto the microcontroller by clicking the on-board reset button.
5. Upon running the Python file, a prompt will appear that will require a push of the “Enter” key on the keyboard. This initiates the Python program, the on-board push button of the microcontroller should also be pushed to initiate the rotation of the motor.
6. Throughout the process, the motor will spin incrementally to gather a measurement every 11.25°. Upon completion of a 360° rotation, the motor will spin rotate in the anti-clockwise direction to unwind the winded wires. At this time, the user should take a step forwards to determine to ensure they are not scanning the same region twice.

7. Upon completion of the scanning process, the data will automatically be transformed into a 3D visualization through Open3D. Within this visualization, users can interact with it.

Expected Output

The expected output of integrated lidar system can be visualized below. The assigned location can be compared to the 3D visualization generated. Location A was the assigned room to be scanned. Below images of the 3D visualization and the hallway in real life can be seen. These images can be compared side-by-side. From the 3D visualization, the lidar system is capable of scanning each of the details of the location.

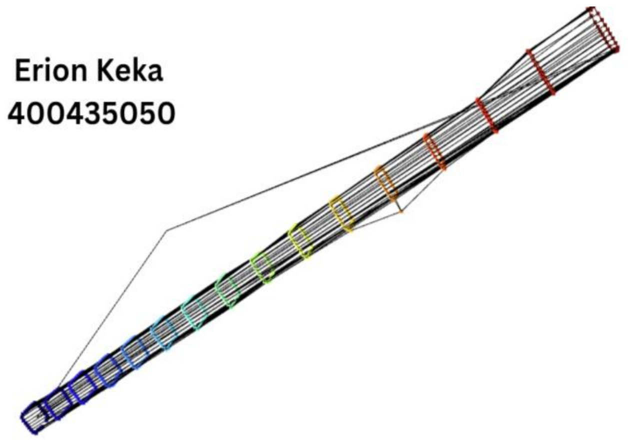

Figure 5: 3D Visualization (View 1)

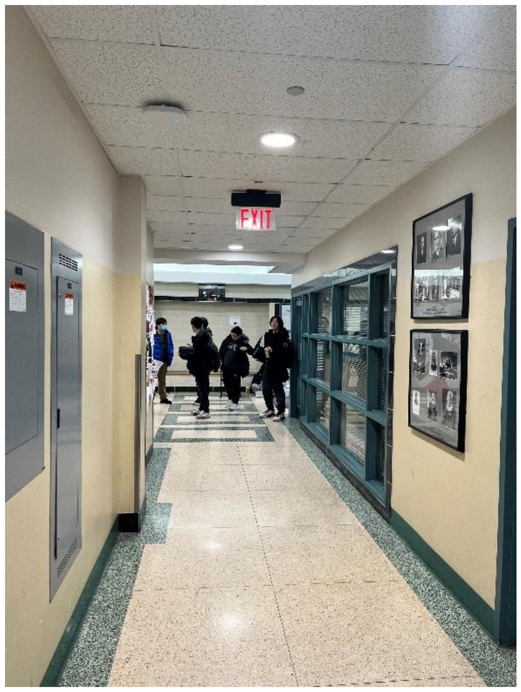

Figure 4: Photo of Location A

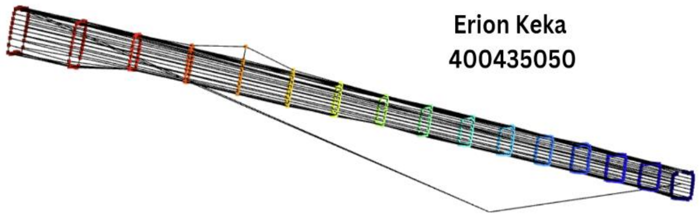

Figure 6: 3D Visualization (View 2)

Limitations

1. Since the data plotting requires the use of trigonometric functions, the outputs of the utilization of these functions tend to result in long and repeating decimal values. To conquer this challenge, the TI MSP432E01Y Microcontroller incorporates 32x32-bit single precision registers which can be found within the Arithmetic Logic Unit (ALU). They also make up the Floating-Point Unit (FPU) which enables the system to support single-precision floating point operations.
2. To calculate the maximum quantization error, we utilize the following formula: $Maximum\ Quantization\ Error = \frac{Maximum\ Reading}{2^n}$. In this formula, "Maximum

Reading" represents the maximum reading possible from the time-of-flight sensor which is 4 meters and since our microcontroller reads values in a 16-bit format, "n" is equivalent to 16. From these values, the maximum quantization error is calculated to be $\frac{4000\,mm}{2^{16}}$ = 0.06104 mm.

3. The maximum standard serial communication rate that can be implemented on my PC is 128000 BPS. To verify this value, we can determine the maximum standard serial communication rate by entering "Device Manager" then, entering the properties of "XDS110 Class Application/User UART" and viewing the "Port Settings".
4. The communication method utilized between the microcontroller and the time-of-flight module is I2C communication. The speed utilized was 50 Hz which is the maximum transmission speed of the sensor.
5. The primary factors that affected the speed of the system was the speed of the motor's rotation. Its speed was determined through the time intervals between each of the motor's four phases that are necessary for each rotation. With each rotation of the motor, there would always be an added delay that will negatively impact the speed of the system. To test this factor, I determined the minimum delay required between the phases of the motor which was approximately 2 ms.
6. To configure the Bus Speed that was assigned in Table 1 of the Project Specifications, the following formula had to be implemented: $Bus\ Frequency = \frac{480\ MHz}{(PSYSDIV+1)}$. If we utilize a value of PSYSDIV of 5, we are able to match the bus speed of 80 MHz. Thus, we are required to change PSYSDIV to 5 in the PLL.h file. Furthermore, in the Systick.C file, we must change the SysTick_Wait(1200000) line under the void SysTick_Wait10ms(uint32_t delay) function to match our bus speed. To do this, we change the 1200000 to 800000 which matches our bus speed.

Circuit Schematic

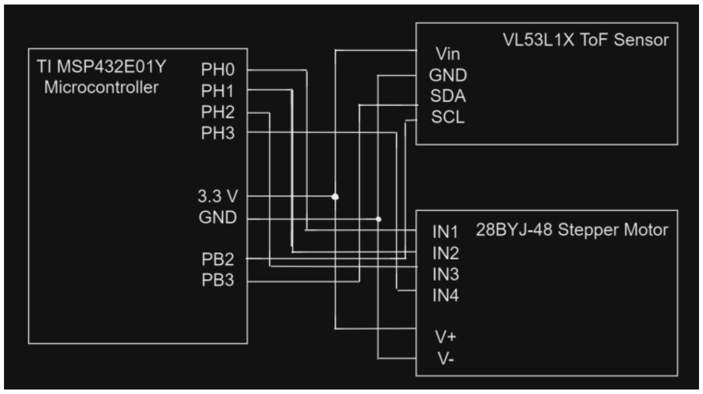

Figure 7: Circuit Schematic

Programming Logic Flowchart

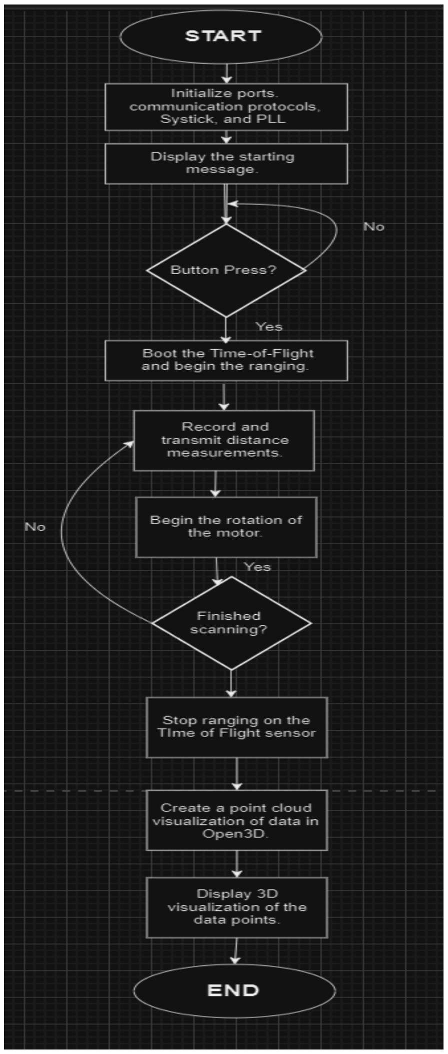
START
Initialize ports, communication protocols, Systick, and PLL
Display the starting message.
Button Press?
No
Yes
Boot the Time-of-Flight and begin the ranging.
Record and transmit distance measurements.
Begin the rotation of the motor.
Finished scanning?
No
Yes
Stop ranging on the TIme of Flight sensor
Create a point cloud visualization of data in Open3D.
Display 3D visualization of the data points.
END

Figure 8: Programming Logic Flowchart