2DX4: Microprocessor Systems

Final Project

Instructors: Drs. Haddara, Shirani, and Doyle

Lab TAs: Saif Aziz Absar (absars1@mcmaster.ca), Yiqiong Miao (miaoy16@mcmaster.ca)

Abhay Raina – L07 – rainaa4 – 400254138

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by **Abhay Raina, rainaa4, 400254138**.

Google Drive Links

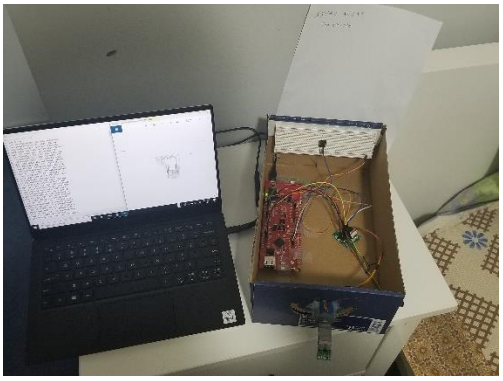
Project Demo:
https://drive.google.com/file/d/1nCJHGQqhe2yrO_l2BcQwVviDwF8us_oc/view?usp=sharing

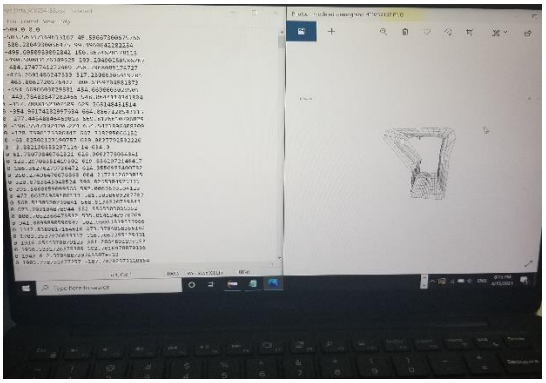
Question 1:
https://drive.google.com/file/d/1avmn_uviuPqZzhm9e9bcIHVeY4VVUCdg/view?usp=sharing

Question 2:
https://drive.google.com/file/d/1eKmWt__p90FUW8pzaiVF1Q_NISxYnG5e/view?usp=sharing

Question 3: https://drive.google.com/file/d/1wvWJe72VrEN4-MvzKTCOhelIaVpR0NsQ/view?usp=sharing

Device Overview





a) Features

This device is an integrated LIDAR sensor with single planar 360-degree distance measurement capability. It utilizes a TI MSP432E401Y microcontroller (MCU), in coherence with a Time-of-flight sensor (VL53L1X) and a stepper motor/controller. The relevant features for this project were:

- Adjustable bus speeds, set to 120 MHz by default and adjusted to 30 MHz for customization.
- High speed UARTs (8 in count), with 15 Mb/s baud rate in High-Speed mode and 7.5 Mb/s for regular mode.
- Has an operating voltage range between 2.5-5.5 V.
- 2 Ms/s, two SAR ADCs optimized for 12-bit data
- Memory capacity of 1024 KB for flash and 256 KB SRAM
- ToF has an effective distance range of 4m and 50 Hz ping frequency, under operating voltages of 2.6-3.5 V an the UNL2003 provides 512 steps during a 360-degree rotation, with LED indication.

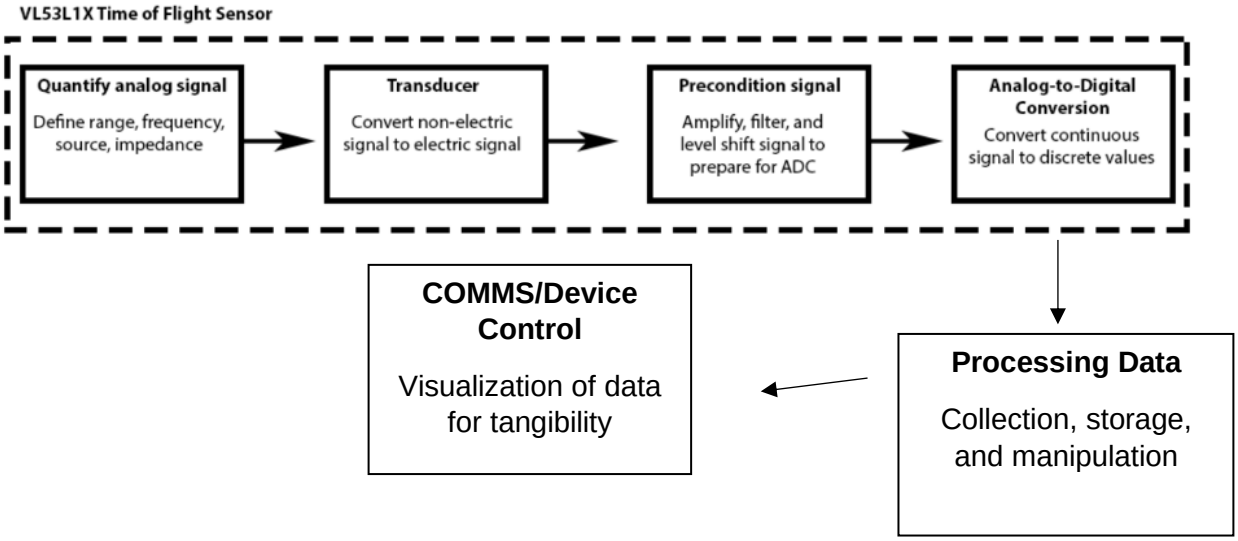
b) General Description

The assembled device is a space mapping embedded system, useful in indoor scenarios. The device can be mounted in various apparatus/chassis, such that it can form a part of a larger system, such as house layout mapping systems, navigation systems, etc.

The components and their functions in the system are as follows:

- IO: Push button and LED status for measurement and angle incrementation
- Transducer: Time of flight (ToF) sensor slicing in y-z axis plane
- Processing Data: Data collection and processing into distance, angle, coordinates, etc.
- Software design implementation: Implements an interrupts design philosophy
- Communication and device control: Communication between ToF and MCU, as well as communication between computer reader program and local storage. Then .xyz coordinate file is 3D visualized

The ToF handled a lot of discretization processes such as ADC conversion. The sensor captures analog data using a transducer, the uses the inbuilt ADC to convert to digital signal. The digital signal, once in a computer readable format, gets ported from MCU to a local computer communications port, through UART, at a set interrupt, byte by byte. This data is then received by a Python data receiving and conversion script, which accounts for delta-x, which was done by physically moving the circuit and spliced the distance data into yz coordinated with respect to the delta-x at that plane. The visualization Python script (uses open3D) then takes this data and uses libraries/packages to get a 3-D representation of captured data.



Device operability block dia.

Device Characteristics Table

UNL2003		VL53L1X		MSP432E401Y	
Pins	MCU pin	Pins	MCU pin	Spec	Value
VDD	5V	VIN	3.3V	Distance LED	PF4
GND	GND	GND	GND	Baud	115200
A	PM0	SDA	PB3	Comms port	COM6
B	PM1	SCL	PB2	Bus speed	30 MHz
C	PM2			Ext. Push button	PE0
D	PM3				

The above table was used for setting up the pins and features of device appropriately. The Comms port will be dependent on the user's computer and the Circuit diagram was used to assemble it. Standard operational steps such as flashing code to MCU and initializing serial ports for UART communication is to be followed, as usual.

Detailed Description

a) Distance Measurement

The operation device conducting raw distance measurements is the VL53L1X ToF sensor in millimeters (mm). This sensor combines a transducer, pre-conditioning circuit and an ADC. The transducer converts non-electrical signals into electric. The pre-conditioning circuit aids in digitization process and the ADC (Analog to Digital converter) translates the pre-conditioned signal into digital data. Due to the light sensitivity of the sensor and infrared laser implementation, a preferred scenario as chosen in the demo video, would be to use the device in a darker environment away from direct light sources. The ToF measures distance by calculating the time it takes the ping to go back and forth. In terms of communications, whenever the motor steps, the ToF measurement is sent to the computer through UART and the MCU receives the data from ToF using I2C. The communications port is set to 115.2 Kbps, 8N1 at COM6. The Python script "read.py", initializes the port setting and actively listens. It receives data in bytes in a specific delimited format at each rotation increment which it converts to integer value. These distance inputs getting received by computer from UART, alongside the expected motor step values, were used to calculate the relative angle, which was then split into y and z coordinates. For the axes, the x-axis was user defined, which manifested itself physically. The user would have to define the delta-x in the code and move the sensor by that much amount for each measurement. For the y-axis, since the sensor started facing the ground, it was set to $y = -d*\cos(\theta)$ and the z-axis was defined as $z = d*\sin(\theta)$. For calculating the angle (θ) for splitting each distance into its z and y components the formula $\text{angle}(\theta) = \text{steps}/64 * 2*\pi$, was used and the total steps in a full rotation in this case is 64 steps, since 64 data points were recorded per rotation in the demo. The measurements were kept in mm and the angles in radians, for further processing in visualization script. The logic flowchart (Fig 2) gives an appropriate representation as to how MCU serves its purpose, and the data conversion process in Fig 3, aids in visualizing this data processing step.

b) Visualization

The computer was a Dell XPS 13, running Windows 10 Pro 64-bit. Python version running was 3.7.7 (chosen because of compatibility with packages) and the libraries used are listed as follows:

- Open3d: for 3d representation of data and wireframing
- Numpy: for 3d matrix conversion
- Math: for math function such as pi and trig. Functions
- PySerial: for serialization and UART communication

After the reading process is complete and the data has been converted into xyz format in a .xyz file. The visualization script (visuals.py) is described in steps as follows:

- Create an array of lines/wireframes
- Setup a perfect 3D rectangular wireframe
    - Create the line segments for joining points in the single plane
    - Create line segments for joining the adjacent point in interplanar fashion

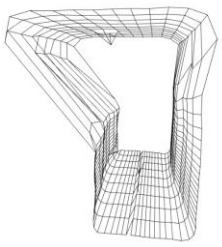
- Open the .xyz file and convert it into a point cloud using open3d
- Bind those xyz coordinates to the wireframe and its respective default point in each plane using manipulations of numpy
- Utilize the draw function to output the updated and molded wireframe defined by the point cloud

Measurement area sketch:

Application Example

Steps are listed as follows:

1. Setup the device on a movable platform such that delta-x can be incremented
2. Flash the code on to the MCU using keil and appropriate debugger settings
3. Run the read.py script and press the reset button on MCU
4. Wait for 10s for initialization
5. Then press the external push button to start measuring
6. After first measurement is complete the motor will reset its position to start position and increment delta-x
7. Repeat steps 5-6 until 10 measurements are done
8. A dataset has been written in an .xyz format, which can be parsed by visuals.py
9. open visuals.py and run it, the expected 3D view of the .xyz file will appear.



Example 3D graph from Demo

The xyz planes are defined in the Distance measurement section. Refer to that.

For setup:

- Follow Fig 1 to setup the connections
- Attach ToF sensor to the motor shaft
- Flash project onto MCU
- Run the read.py for measurements and xyz file generation and employ the visual.py script to 3D model as mentioned previously

Limitations

1) The FPU in MCU can support 32-bit add, sub., multi, div., multiply and accumulate, sq.root functions. The FPU can also perform conversions on floating point data formats. In relation to this program and device, the math.h library can perform trig. Functions using 32-bit (single precision floating point numbers) with no issues. However, python for math functions was preferred in this use case due to added support and processing abilities.

2) Max quantization error is Vfs/2^m equal to resolution

ToF resolution = 5/2^8 = 19.53 mV; 8bit ADC

MCU resolution = 5/2^12 = 1.22 mV; 12bit ADC

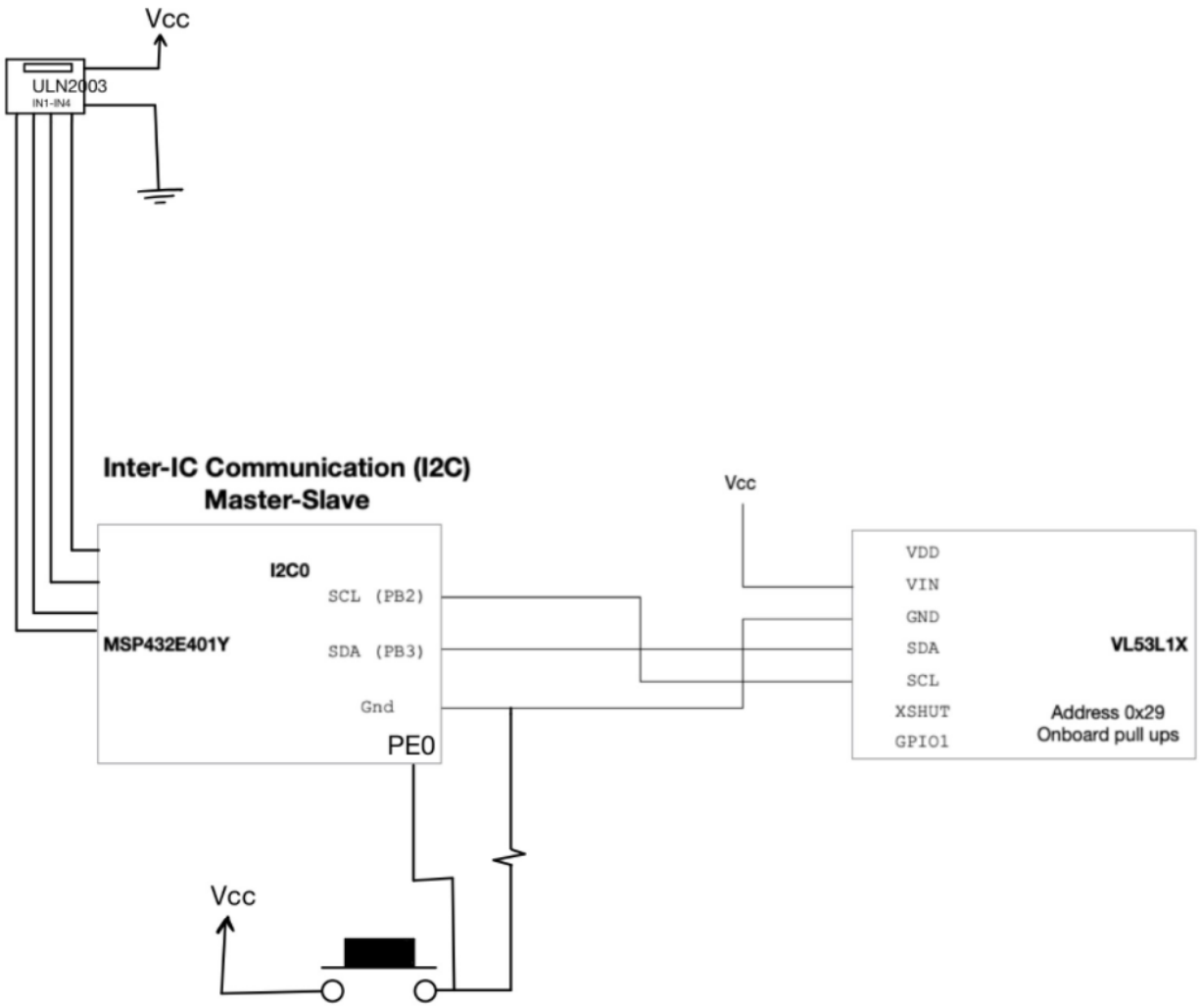
3) Unable to complete due to time restriction

4) Unable to complete due to time restriction

5) Unable to complete due to time restriction

Circuit Schematic

Fig 1: Circuit setup



Logic Flowcharts

Fig 2: MCU flowchart:

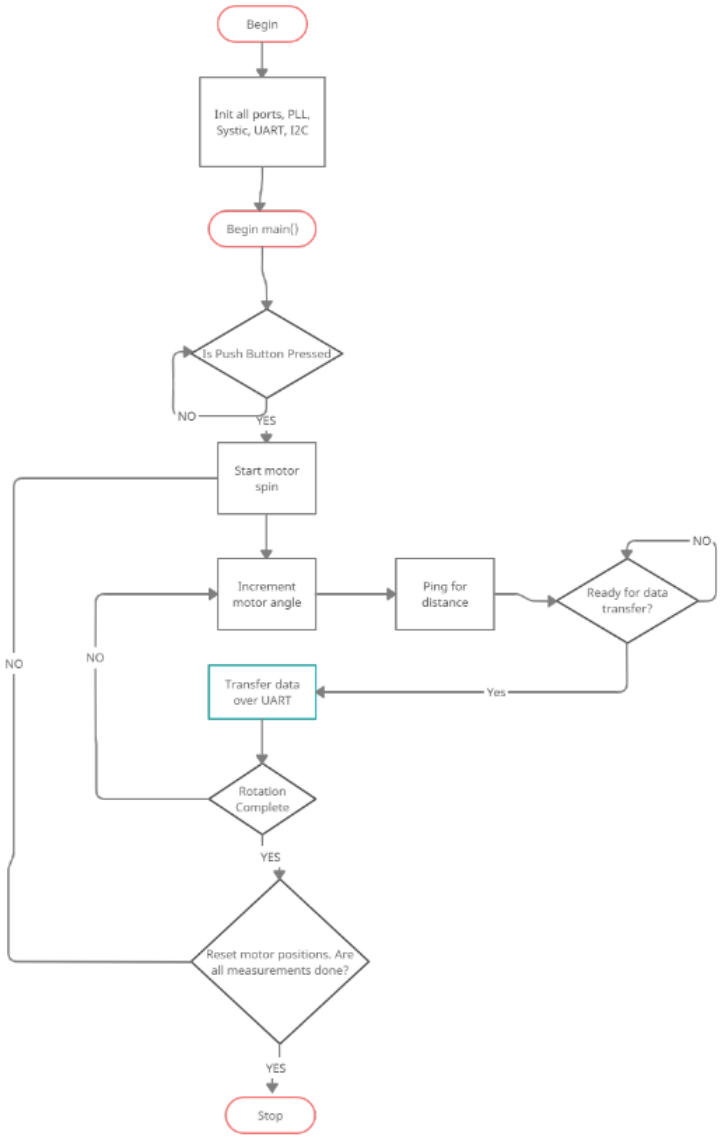


Fig 3: Python (read.py) flowchart: