Generation of 3D Point Cloud using 2D LIDAR

Presented by: Group 1

Ajith Sylvester
Kamalnath Bathirappan
Ket Ashvinbhai Bhikadiya
Richik Sinha Choudhury
Athmavidya Venkataraman

Background and Problem Statement

Overview of the Problem:

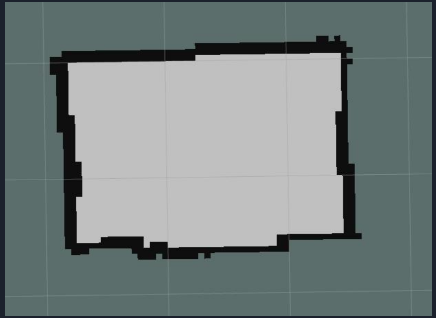
- Most mobile robots using ROS rely on LiDAR-based SLAM.
- **2D LiDAR Limitations:**
  - Detects only in a single horizontal plane.
  - Misses vertical obstacles such as chairs, hanging objects, or shelves.
  - Results in potential collisions even with strategic sensor placement.

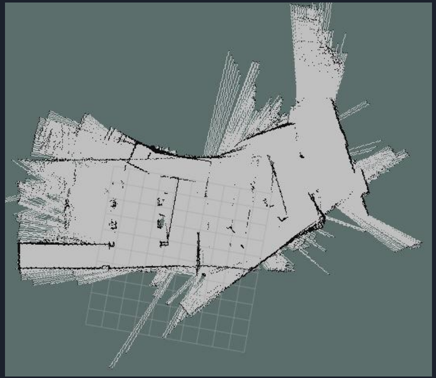
Current Solutions:

- 3D LiDAR provides a comprehensive 360° view.
- **Challenge:** High cost of 3D LiDAR limits its practicality.

Objective of the Project:

- Develop a cost-effective solution using 2D LiDAR to generate a pseudo-3D point cloud.


2d Map of elevator at EXP


2d Map near makerspace

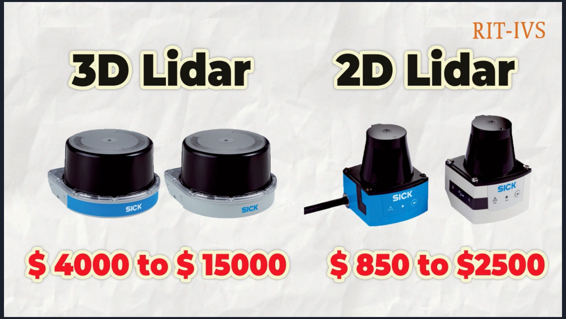
Why 2D Lidar????

Price Difference Between 2D and 3D LiDAR:

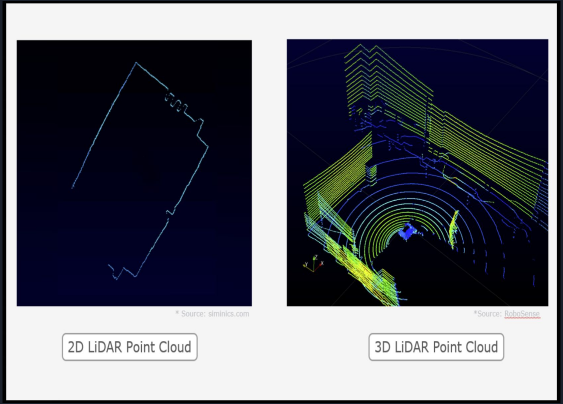
- **3D LiDAR Costs:**
  - High-end systems range from $10,000 to $50,000 per unit.
  - Mid-range systems commonly used in autonomous vehicles cost several thousand dollars.
- **2D LiDAR Costs:**
  - Affordable options range from $100 to $500.
  - Significant cost savings make 2D LiDAR a more accessible solution for various applications.

Advantages of Using 2D LiDAR to Generate a 3D Point Cloud:

- **Cost-Effective:**
  - Offers a practical alternative to expensive 3D LiDAR systems while achieving similar mapping capabilities.
- **Customizable Mapping Solutions:**
  - Enables dynamic mapping by integrating motion (e.g., rotation or vertical translation) to create pseudo-3D data.
- **Scalable Applications:**
  - Useful for indoor navigation, agricultural robotics, and logistics without increasing hardware costs.
- **Reduced Complexity:**
  - Simplifies hardware requirements by repurposing existing 2D LiDAR units with added motion mechanisms.

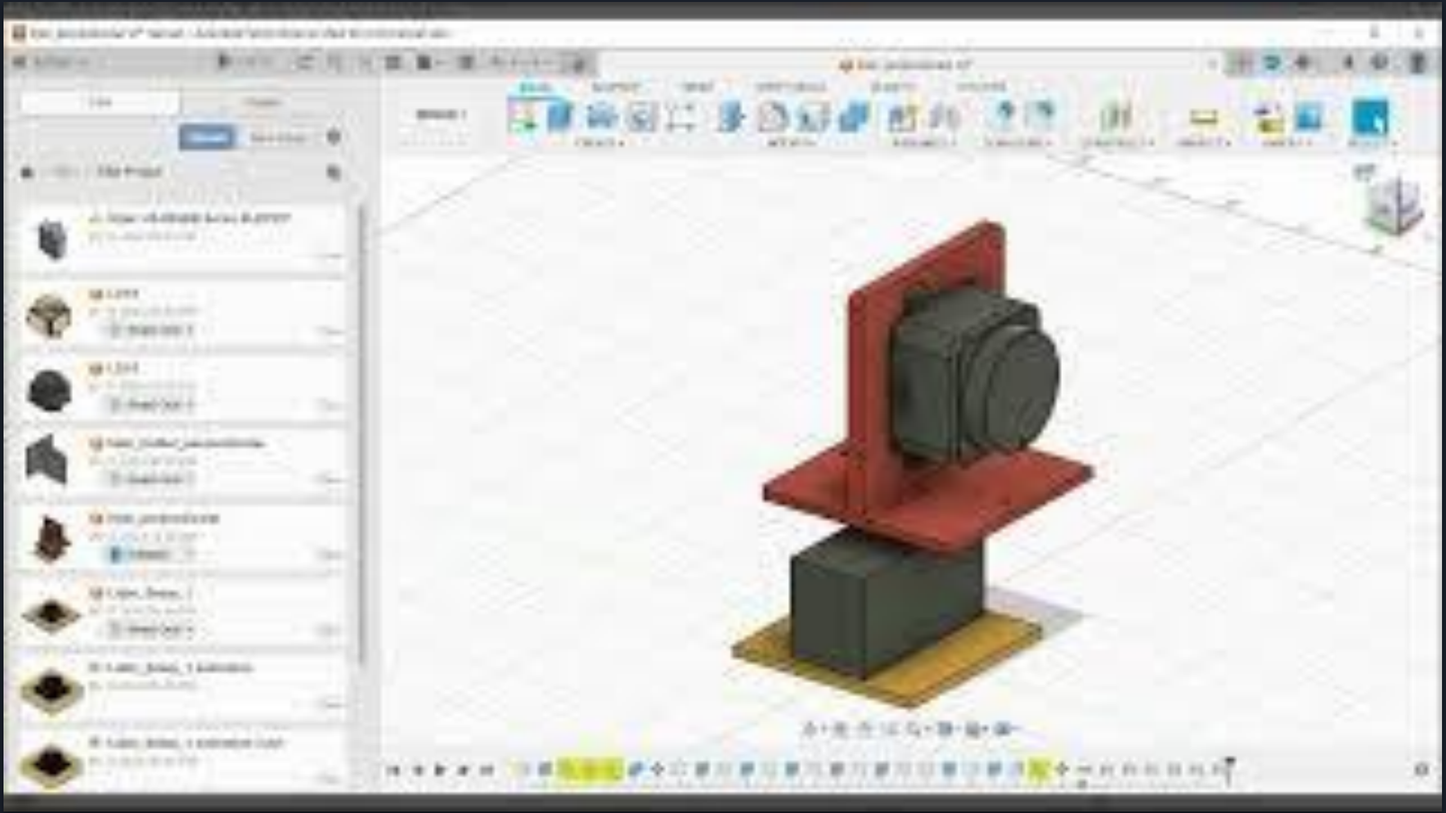


Cost of 2D Vs 3D Lidar



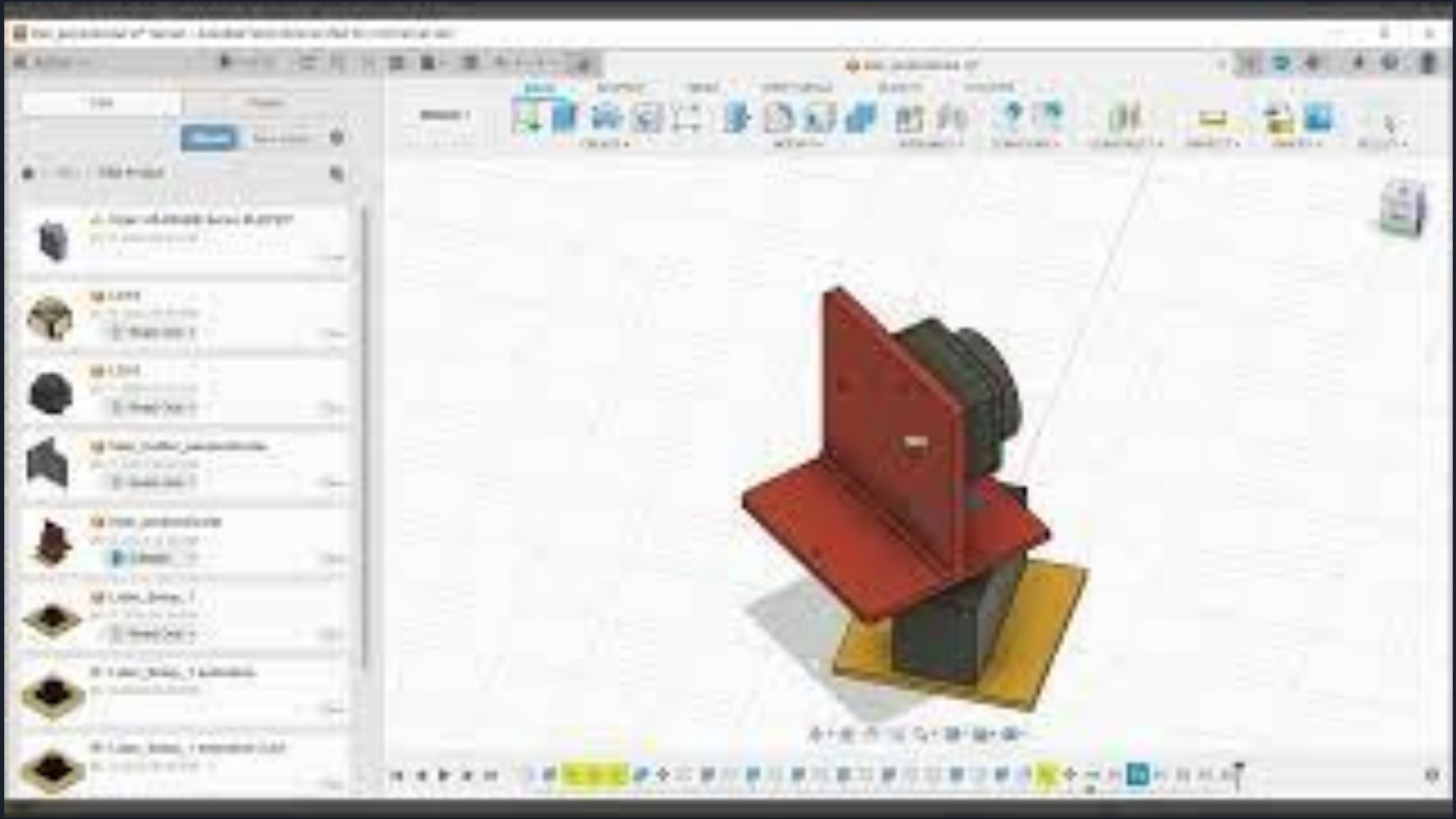
2D Lidar vs 3D Lidar

Implementation



Perpendicular setup with offset

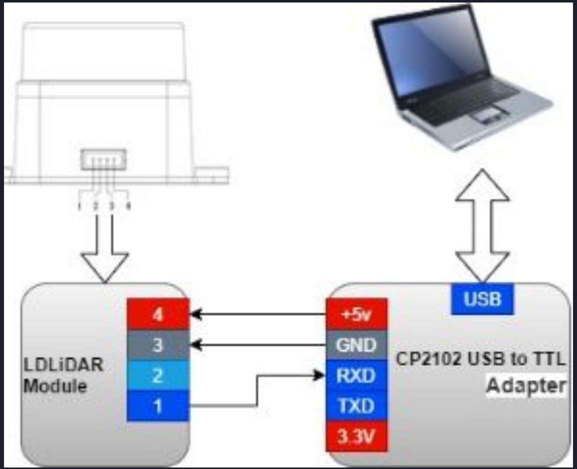
Implementation



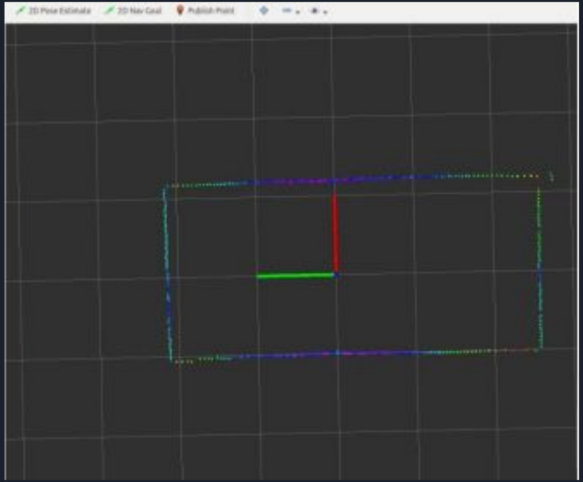
Perpendicular setup without offset, laser beams perfectly aligned with angle of rotation, same setup is rotated to record data in parallel

Hardware Setup

youyeetoo fhl-ld19 lidar

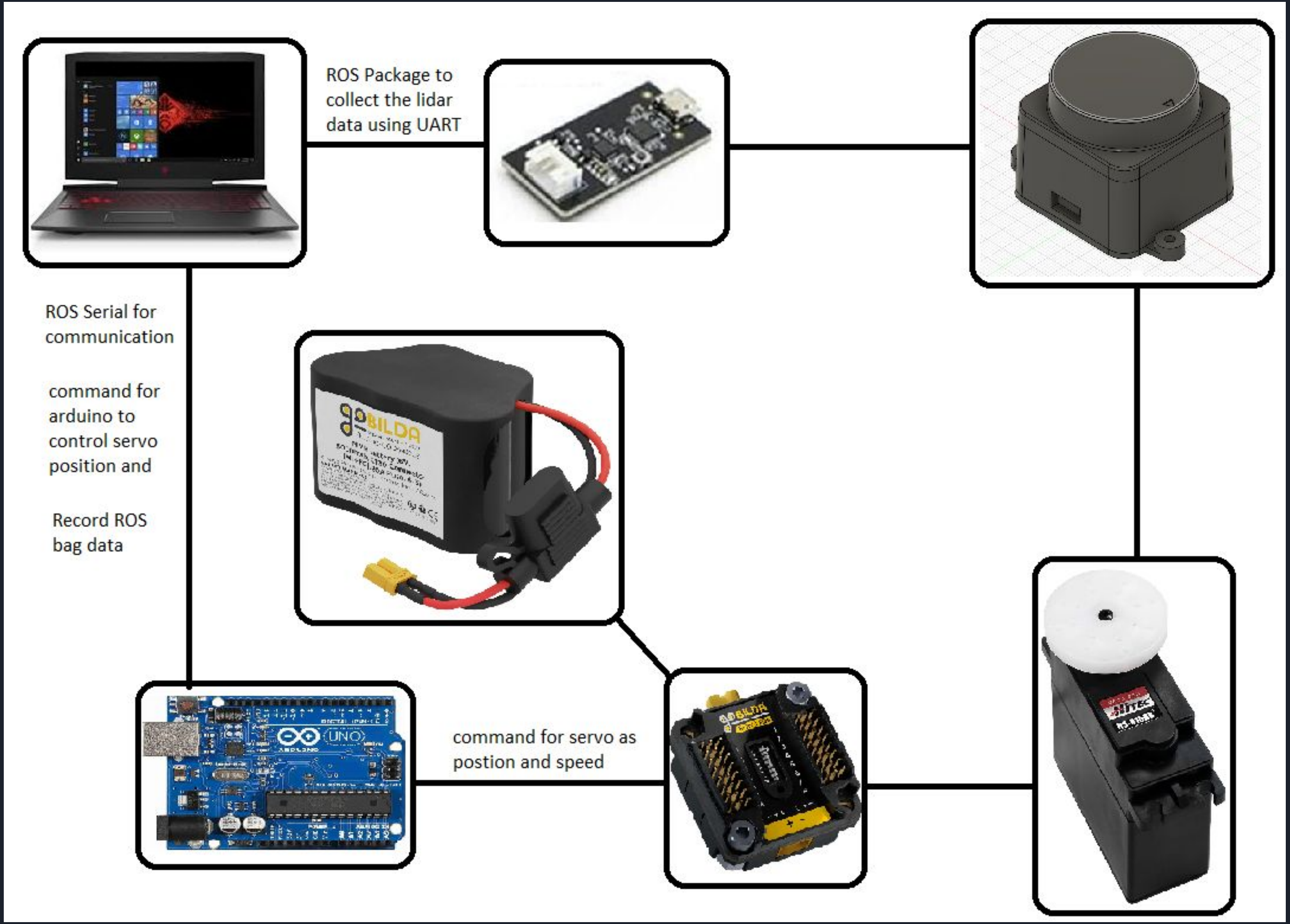


Lidar UART module

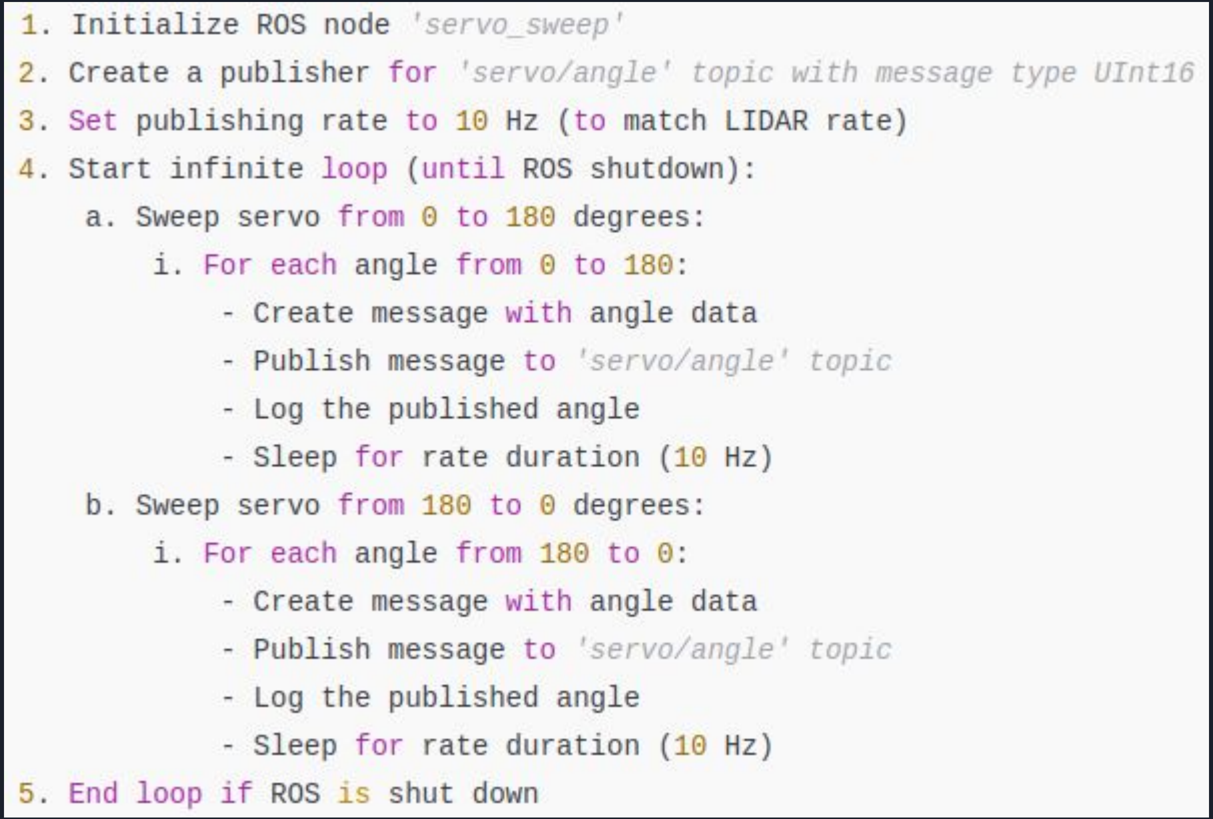


2D Laser points

Hardware Setup

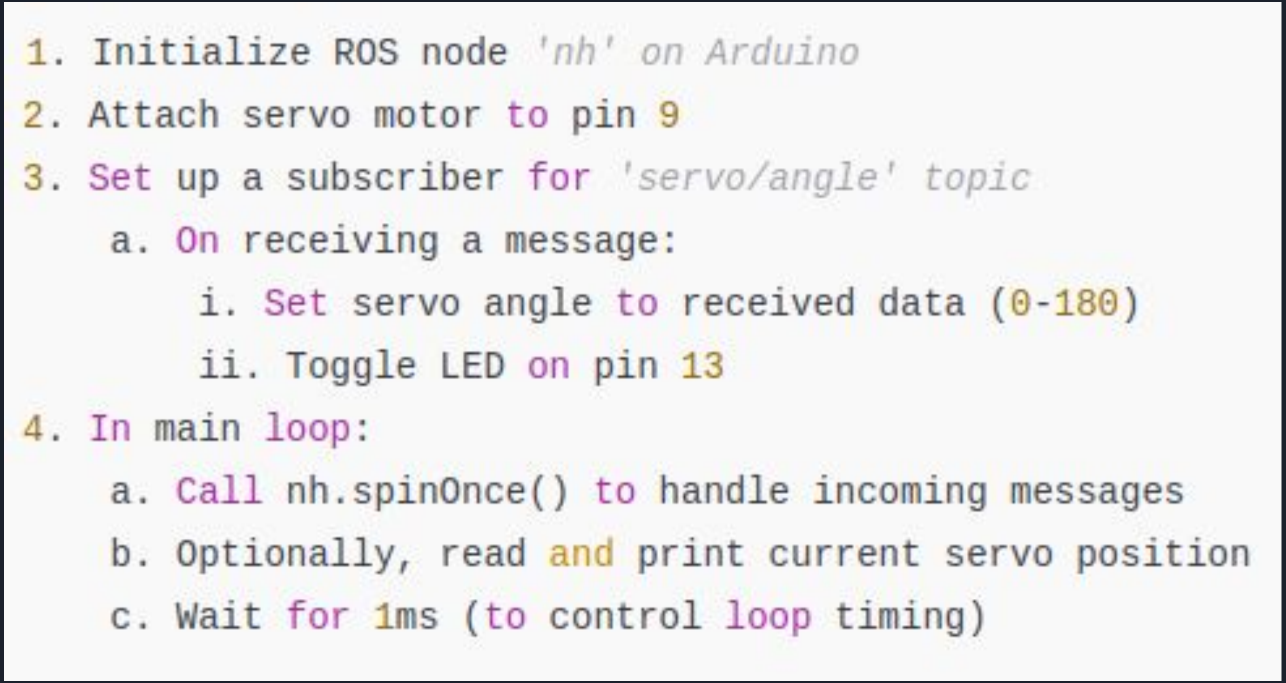

ROS Package to collect the lidar data using UART
ROS Serial for communication
command for arduino to control servo position and
Record ROS bag data
command for servo as postion and speed

Data Collection



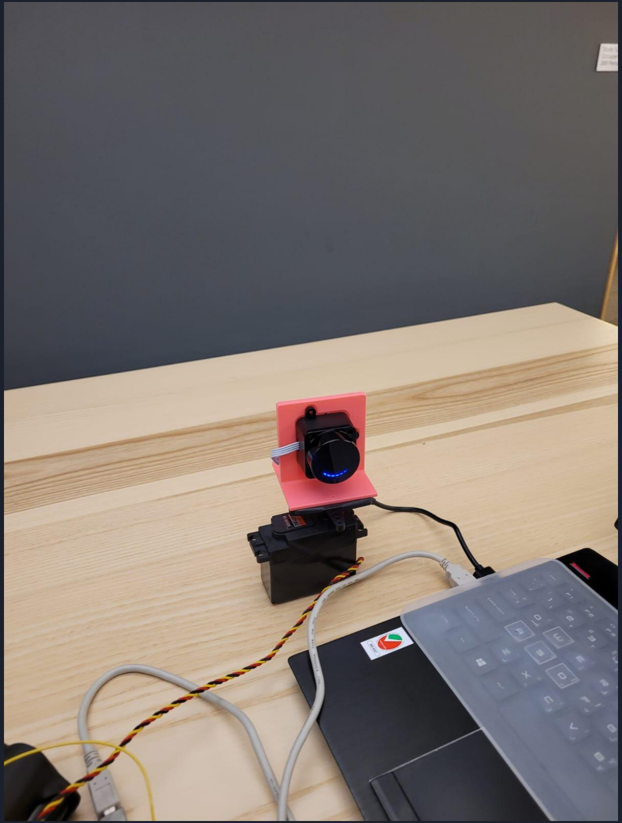
```
1. Initialize ROS node 'servo_sweep'
2. Create a publisher for 'servo/angle' topic with message type UInt16
3. Set publishing rate to 10 Hz (to match LIDAR rate)
4. Start infinite loop (until ROS shutdown):
    a. Sweep servo from 0 to 180 degrees:
        i. For each angle from 0 to 180:
            - Create message with angle data
            - Publish message to 'servo/angle' topic
            - Log the published angle
            - Sleep for rate duration (10 Hz)
    b. Sweep servo from 180 to 0 degrees:
        i. For each angle from 180 to 0:
            - Create message with angle data
            - Publish message to 'servo/angle' topic
            - Log the published angle
            - Sleep for rate duration (10 Hz)
5. End loop if ROS is shut down
```

Data Collection

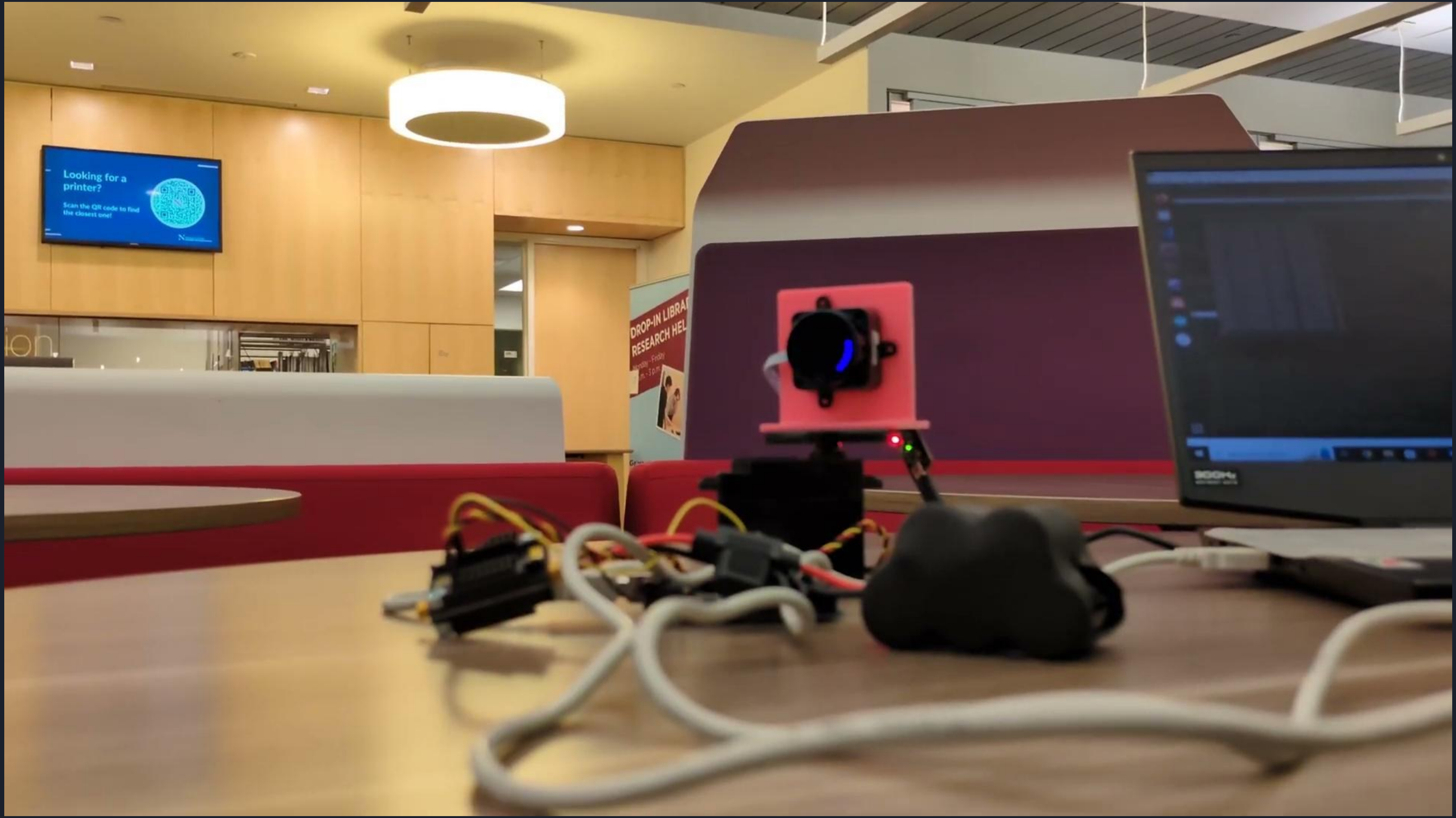


```
1. Initialize ROS node 'nh' on Arduino
2. Attach servo motor to pin 9
3. Set up a subscriber for 'servo/angle' topic
    a. On receiving a message:
        i. Set servo angle to received data (0-180)
        ii. Toggle LED on pin 13
4. In main loop:
    a. Call nh.spinOnce() to handle incoming messages
    b. Optionally, read and print current servo position
    c. Wait for 1ms (to control loop timing)
```

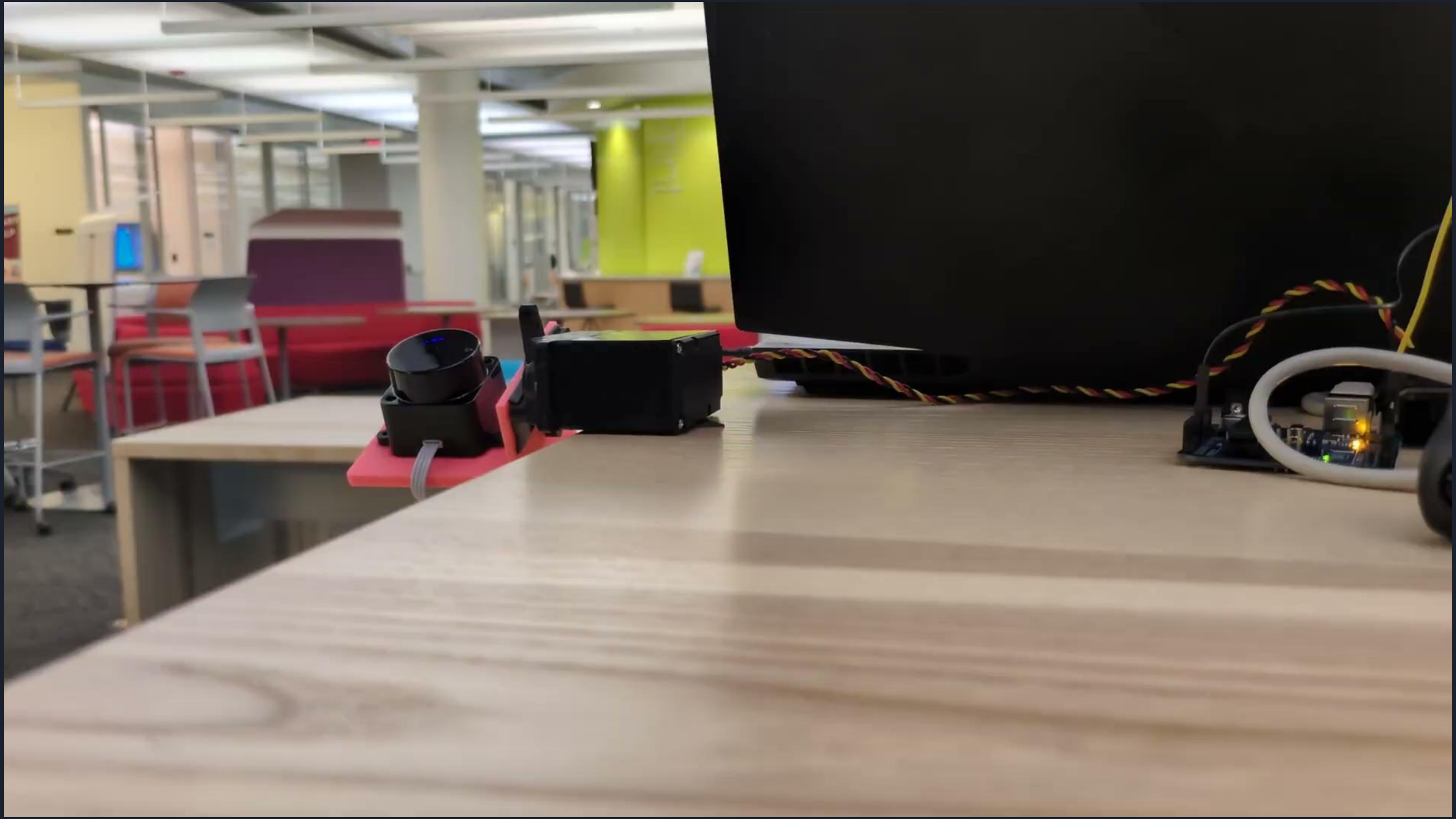
Data Collection



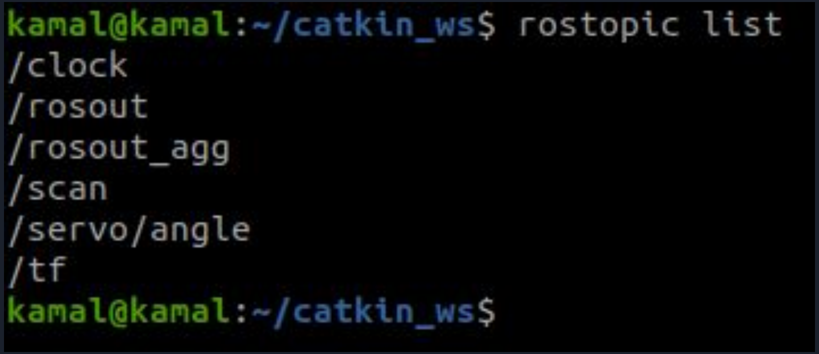
Data Collection



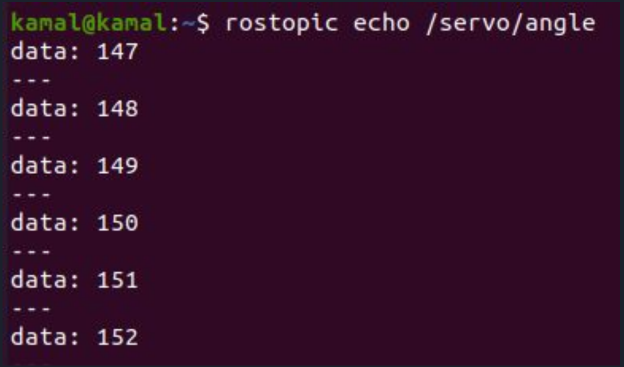
Data Collection



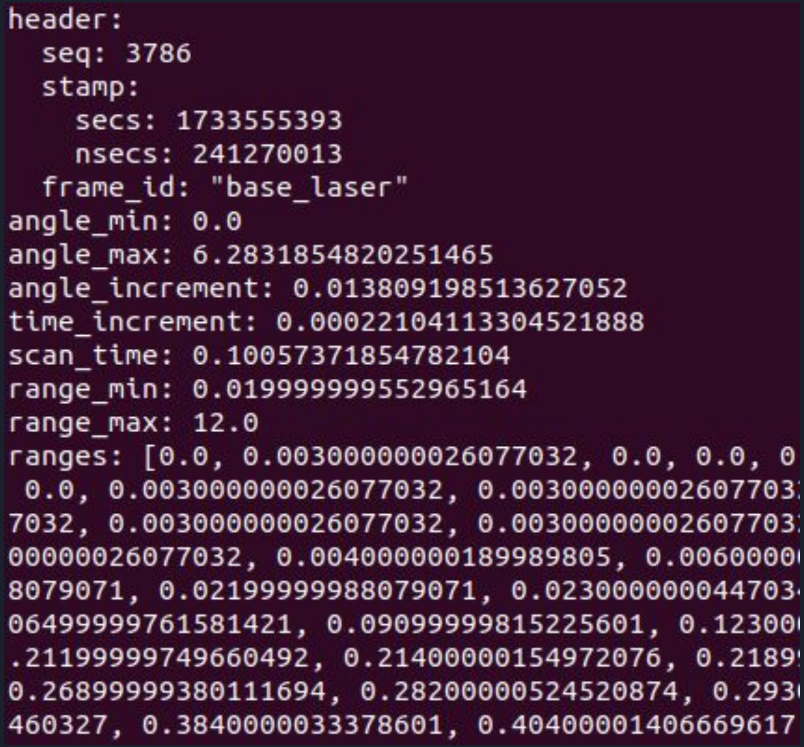
Data Collection



```
kamal@kamal:~/catkin_ws$ rostopic list
/clock
/rosout
/rosout_agg
/scan
/servo/angle
/tf
kamal@kamal:~/catkin_ws$
```

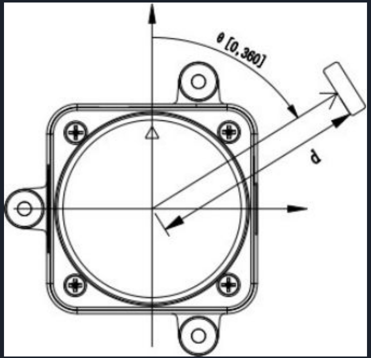


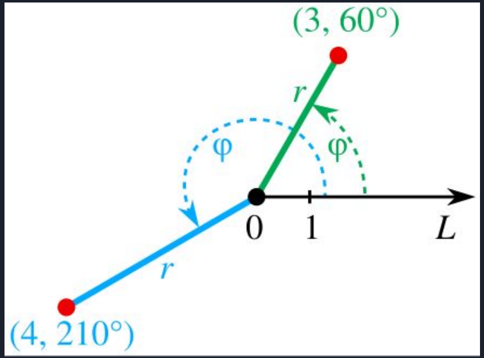
```
kamal@kamal:~$ rostopic echo /servo/angle
data: 147
---
data: 148
---
data: 149
---
data: 150
---
data: 151
---
data: 152
```

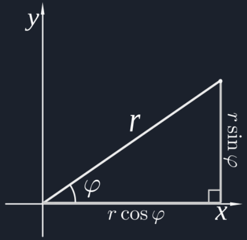


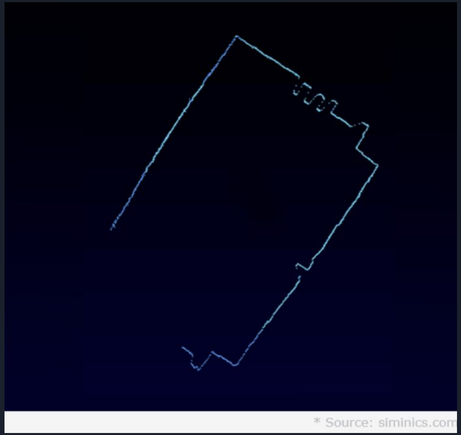
```
header:
  seq: 3786
  stamp:
    secs: 1733555393
    nsecs: 241270013
  frame_id: "base_laser"
angle_min: 0.0
angle_max: 6.2831854820251465
angle_increment: 0.013809198513627052
time_increment: 0.00022104113304521888
scan_time: 0.10057371854782104
range_min: 0.019999999552965164
range_max: 12.0
ranges: [0.0, 0.003000000026077032, 0.0, 0.0, 0
 0.0, 0.003000000026077032, 0.00300000002607703
7032, 0.003000000026077032, 0.00300000002607703
00000026077032, 0.004000000189989805, 0.0060000
8079071, 0.02199999988079071, 0.023000000044703
06499999761581421, 0.09099999815225601, 0.12300
.21199999749660492, 0.21400000154972076, 0.2189
0.26899999380111694, 0.28200000524520874, 0.293
460327, 0.3840000033378601, 0.40400001406669617
```

Algorithm (scan2cloud) - The 2D scan









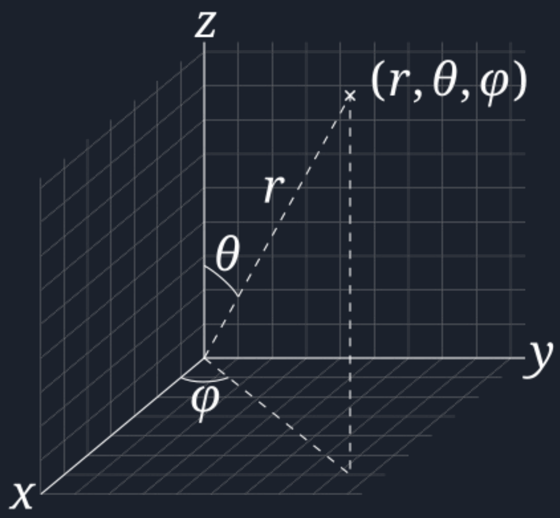
2D Scan: Array of 455 distances, each roughly 0.79 degrees apart

Left-handed angle

A 2D Lidar Scan - a series of polar coordinates

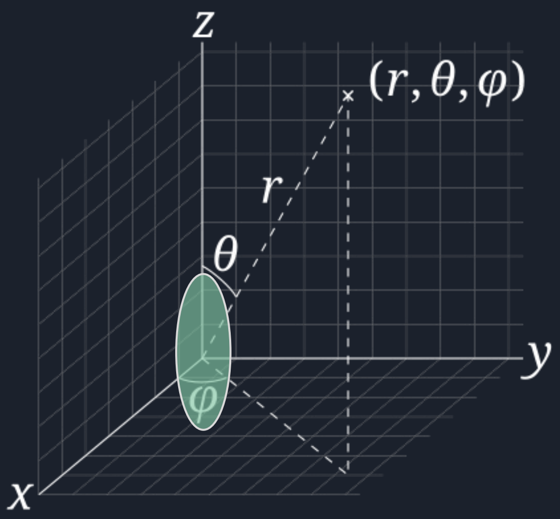
Projected polar coordinates into 2D Cartesian space

Spherical coordinates - project to 3D



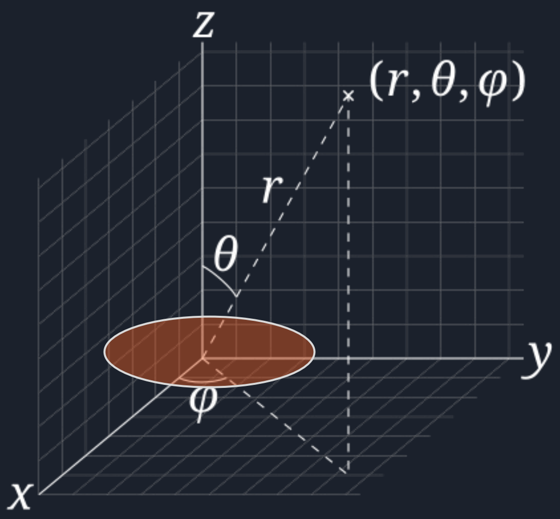
Polar angle θ
Azimuthal angle φ

Perpendicular mount



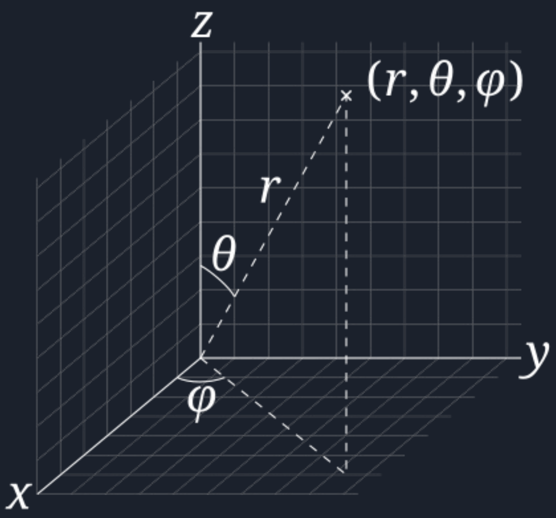
θ = LIDAR angle
φ = Servo angle (increasing CCW)

Parallel mount



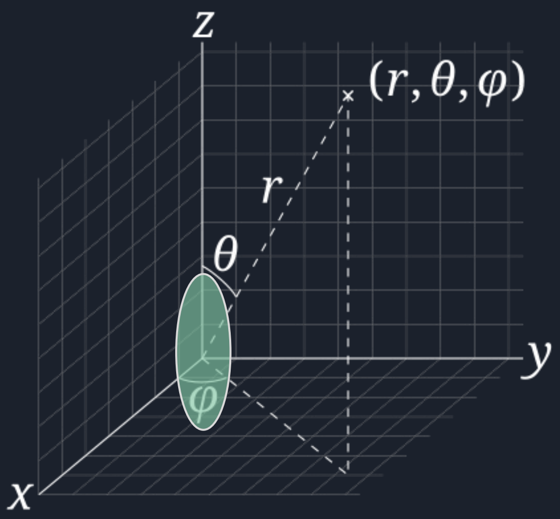
θ = Servo angle??
φ = LIDAR angle (CCW+)

Parallel mount - but correct



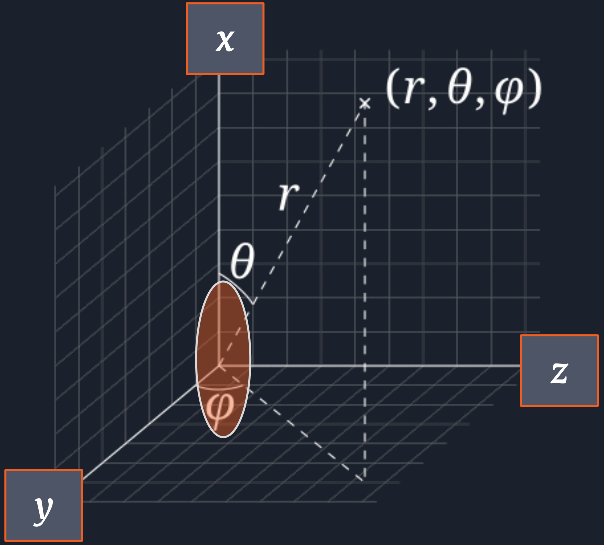
Polar angle θ
Azimuthal angle φ

Perpendicular mount



θ = LIDAR angle
φ = Servo angle (increasing CCW)

Parallel mount



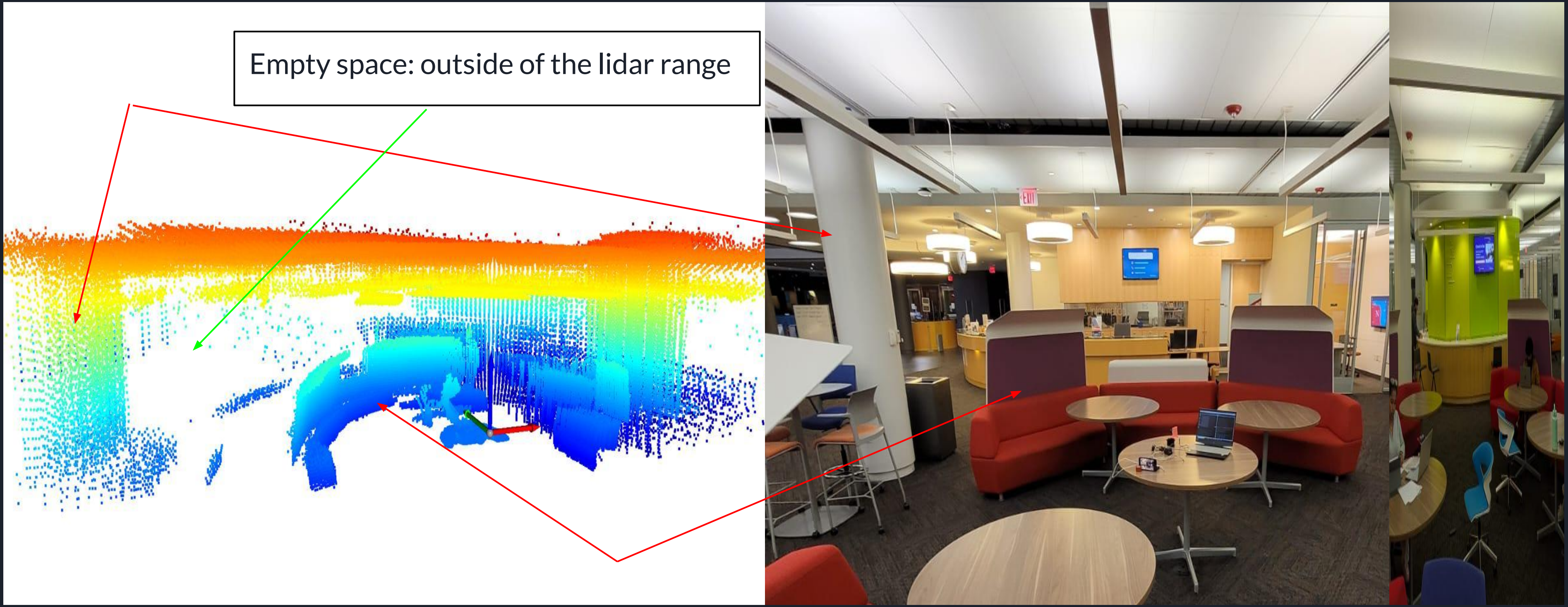
θ = LIDAR angle
φ = Servo angle (increasing CCW)

scan2cloud - data pipeline

rosbag → numpy arrays → preprocessing → computation → Open3D point cloud → visualization

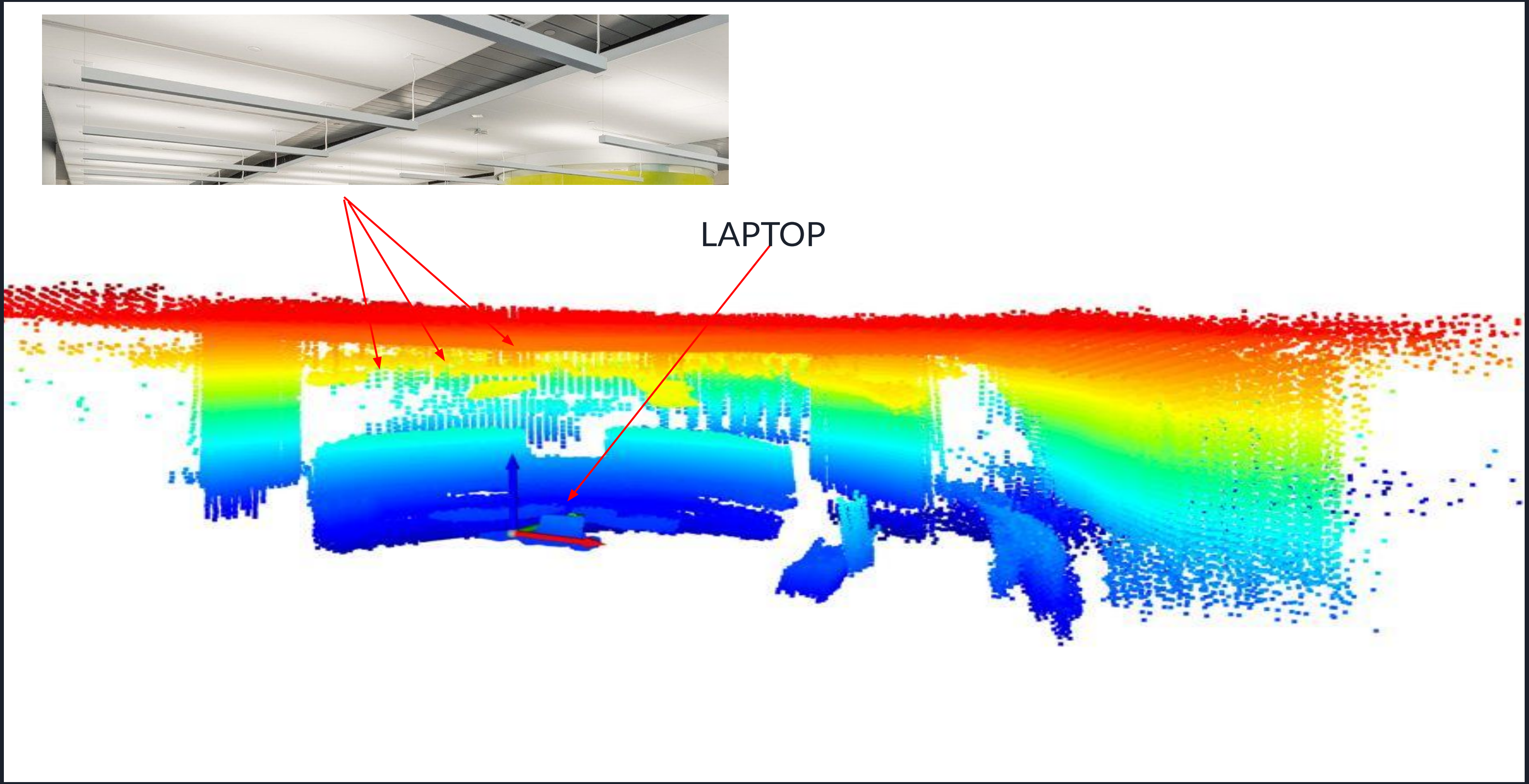
Results

LiDAR Mapping in Open Areas Using a Perpendicular Sensor Setup (for front 0 to 180 degree)



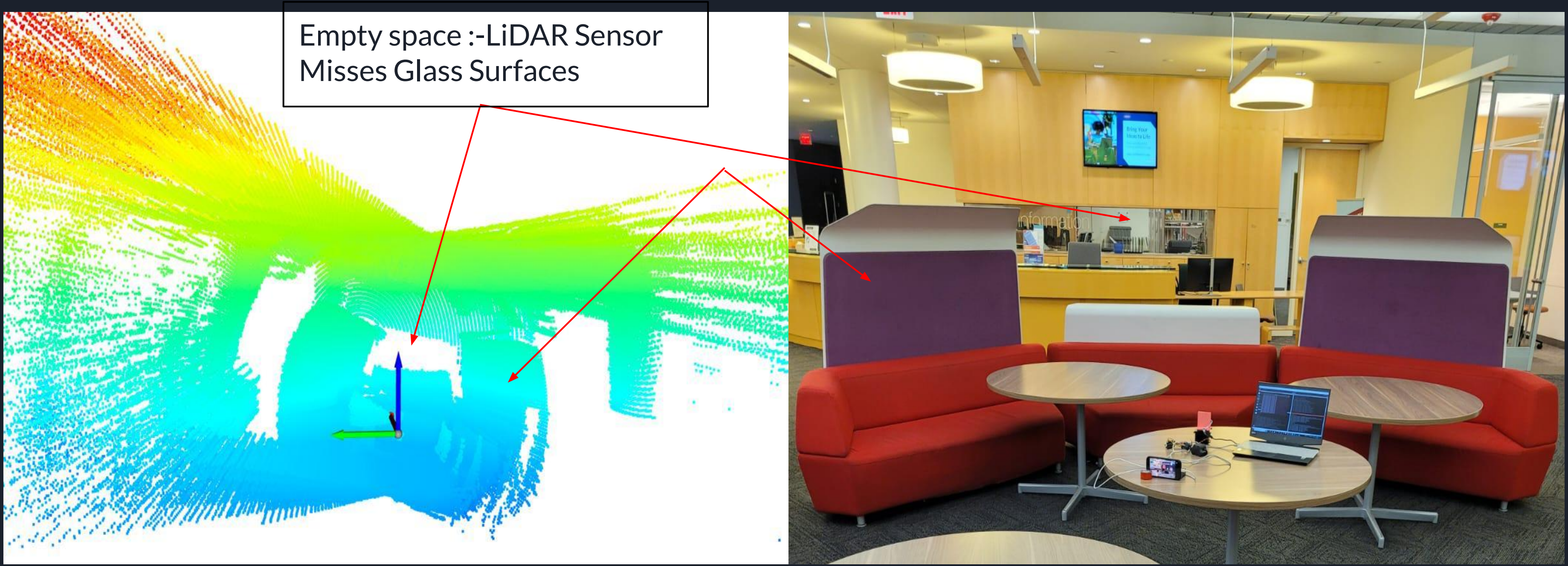
Results

LiDAR Mapping in Open Areas Using a Perpendicular Sensor Setup (for front 180 to 360 degree)



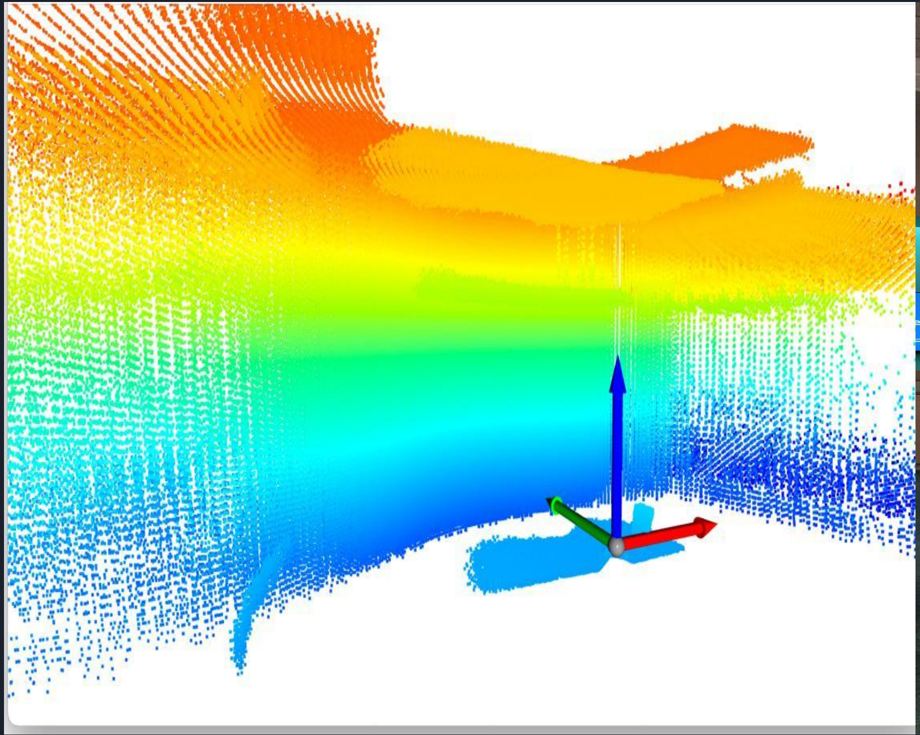
Results

LiDAR Mapping in Open Areas Using a Parallel Sensor Setup



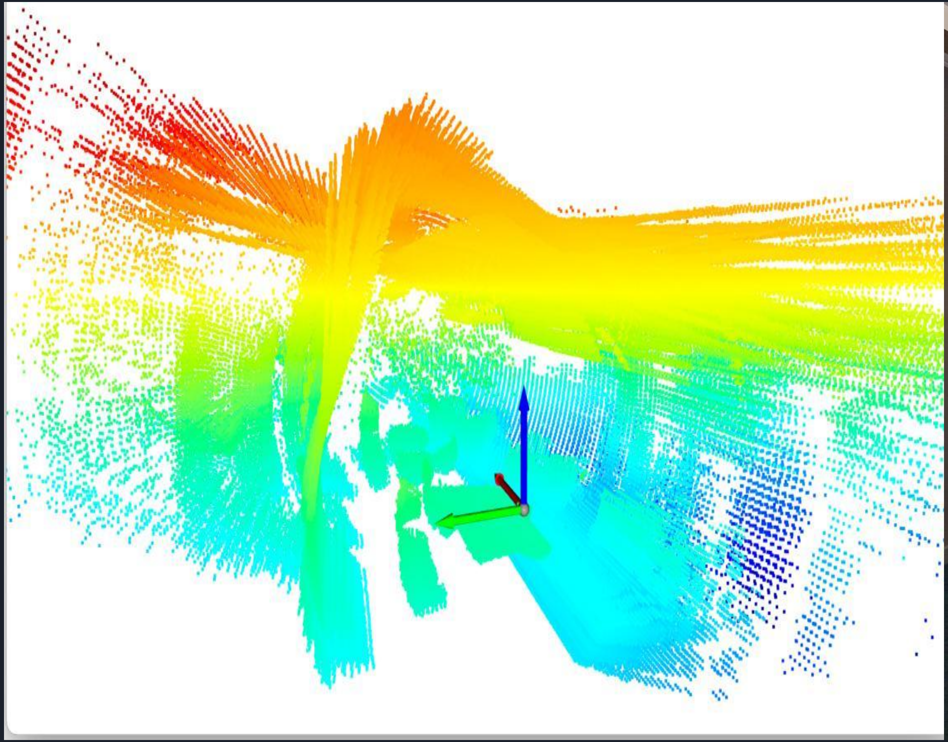
Results

LiDAR Mapping in Front of Wall Areas Using a Perpendicular Sensor Setup



Results

LiDAR Mapping in Front of Wall Areas Using a Parallel Sensor Setup



Conclusion

- Overview
  - Developed a cost effective system for generating pseudo - 3D point clouds using a 2D LiDAR
  - Implemented 2 configurations:
    - Perpendicular setup
    - Parallel setup
- Limitations
  - **LiDAR and Servo Constraints:** Limited Range and Resolution
  - **Dynamic obstacles:** Challenges in real time detection and mapping
- Future Scope
  - **Sensor Fusion:** Integrate cameras and IMUs for Robust mapping
  - **Real time Processing:** GPU Acceleration and Parallel Processing
  - **Dynamic Obstacle Detection:** Usage of more than 1 LiDAR for simultaneous coverage
  - **Fine Grained servo angles:** Improved resolution
  - **Mobile Platform Integration:** Adapt for use on Drones, Robots and vehicles
  - **Outdoor Environment Adaptation:** Weather Resistant and Large Scale Mapping

Thank you! Questions?