Final Project II 2147417:

Collaborative Transport using Swarm Robotics

6438079621 Tinapat Limsila

6438187721 Mehul Sharma

6438075021 Ting-Yi Lin

6438067021 Nattadon Tangsasom

6438118421 Noppawan Srikhirin

Advisor: Asst.Prof.Paulo Fernando Rocha Garcia, Ph.D.

6438079621
Tinapat Limsila

6438067021
Nattadon
Tangsasom

6438187721
Mehul Sharma

6438075021
Ting-Yi Lin

6438118421
Noppawan
Srikhirin

Advisor: Asst.Prof. Paulo Fernando Rocha Garcia, Ph.D.

Agenda

01 Recap and Demo

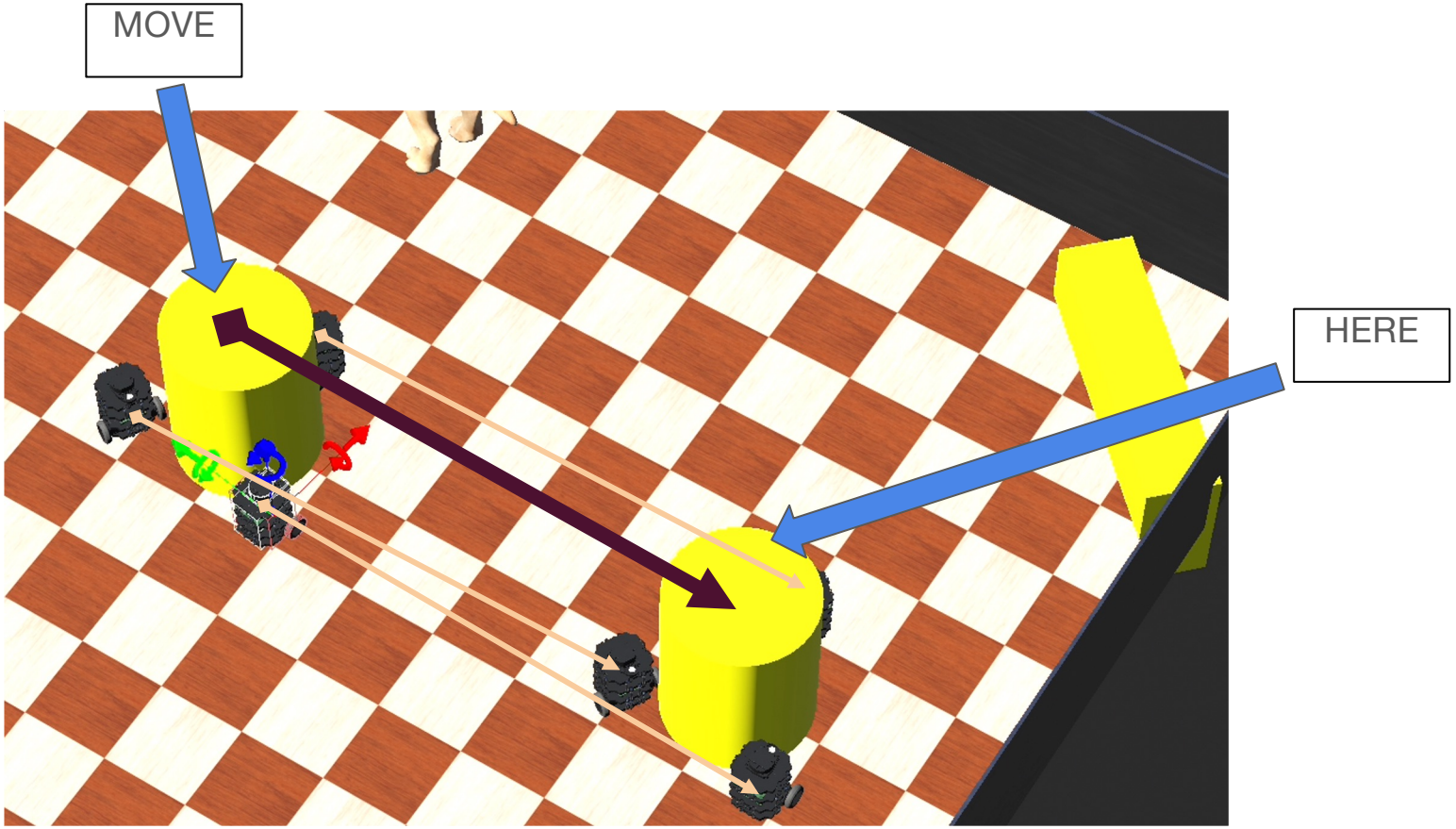
02 High-level Workflow

03 Modules Implementation

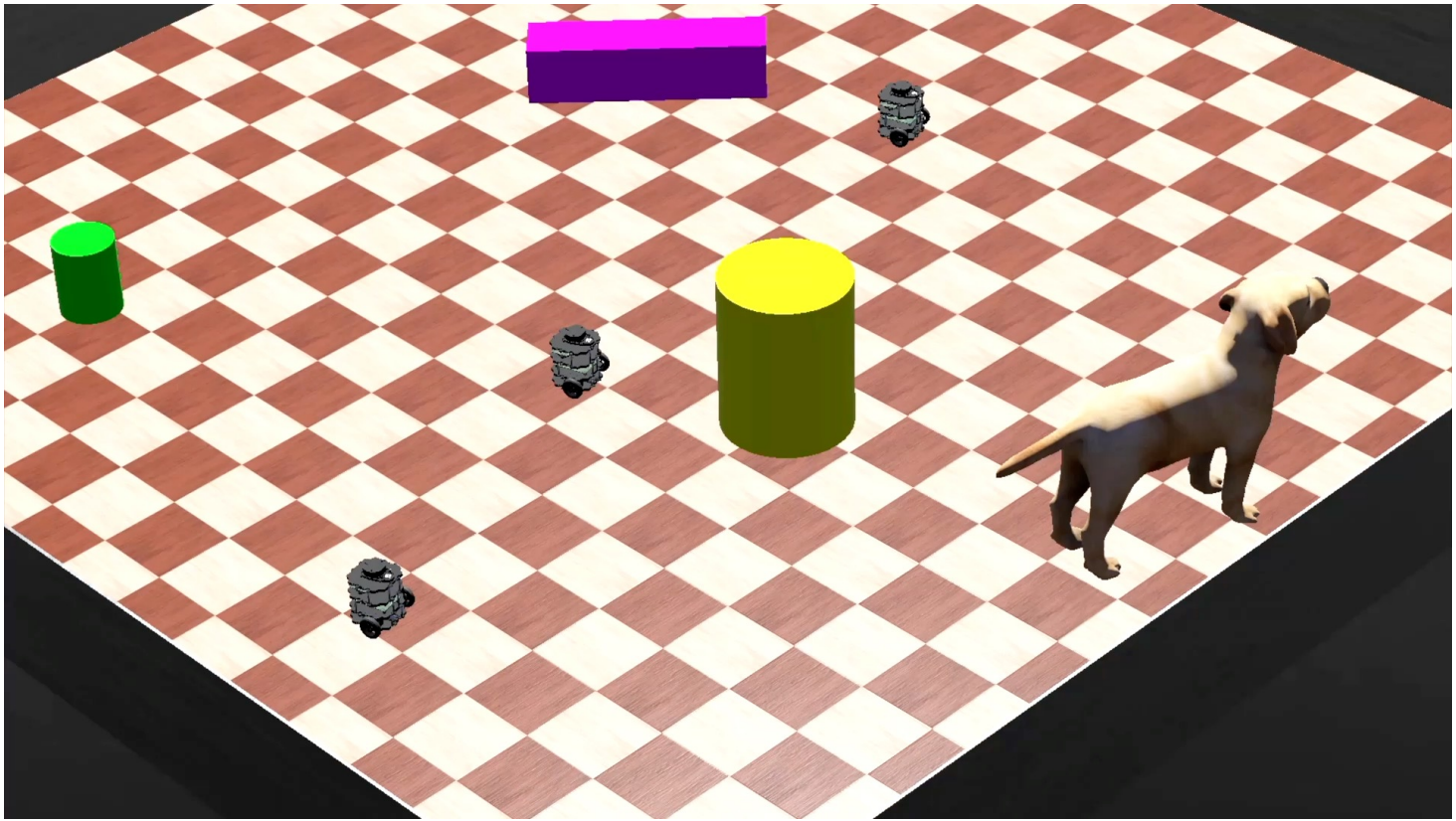
04 Challenges & Lesson Learned

05 Q&A

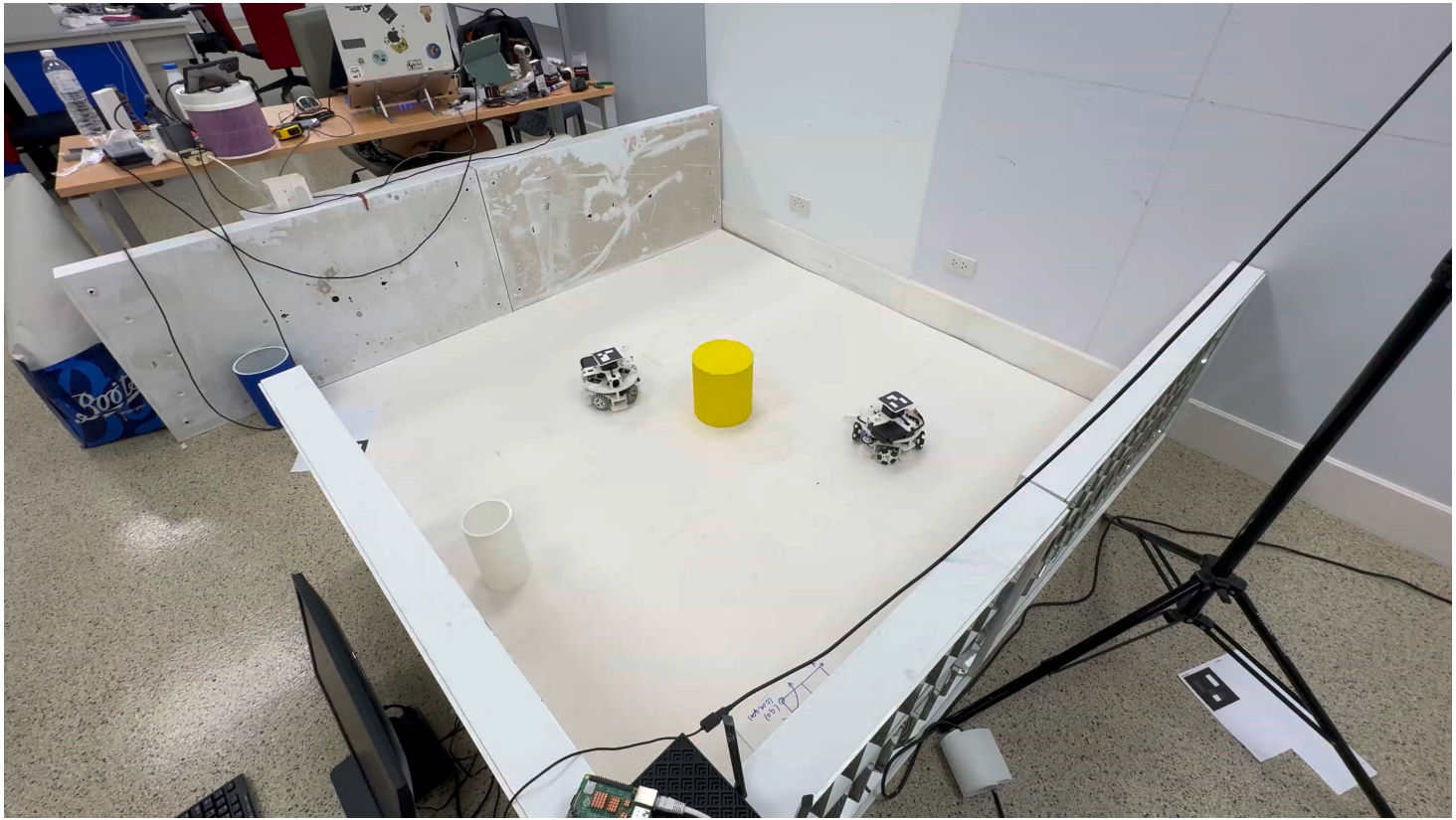
Goal



Recap last semester: Simulation Demo

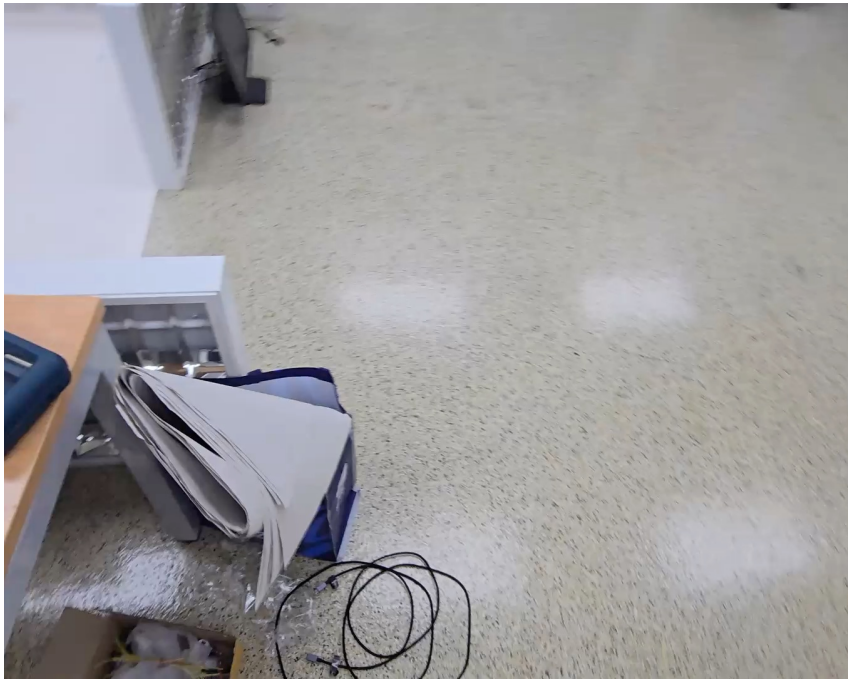


Real Demo

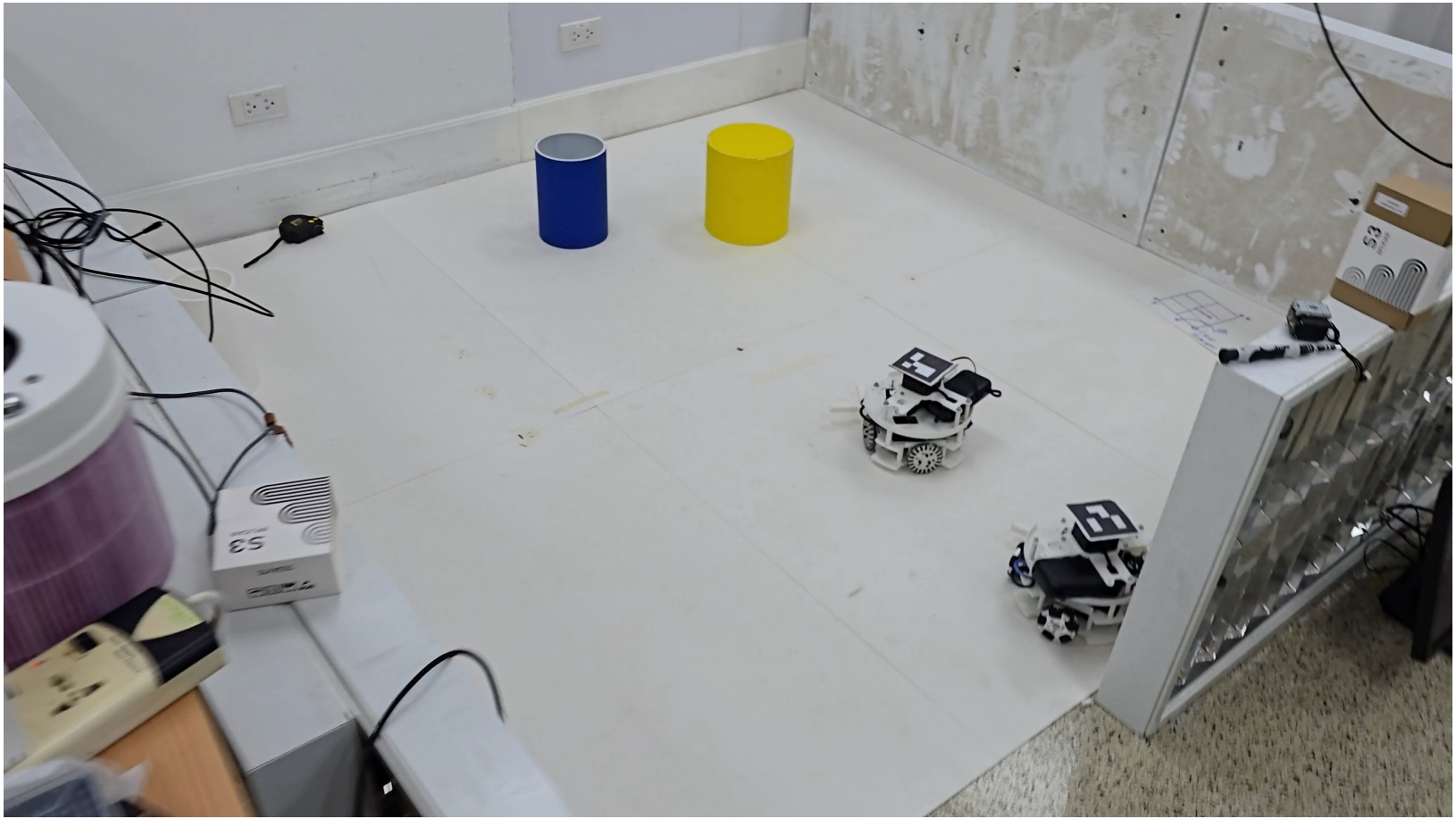


Demo Milestone : Successful Forward Kinematic

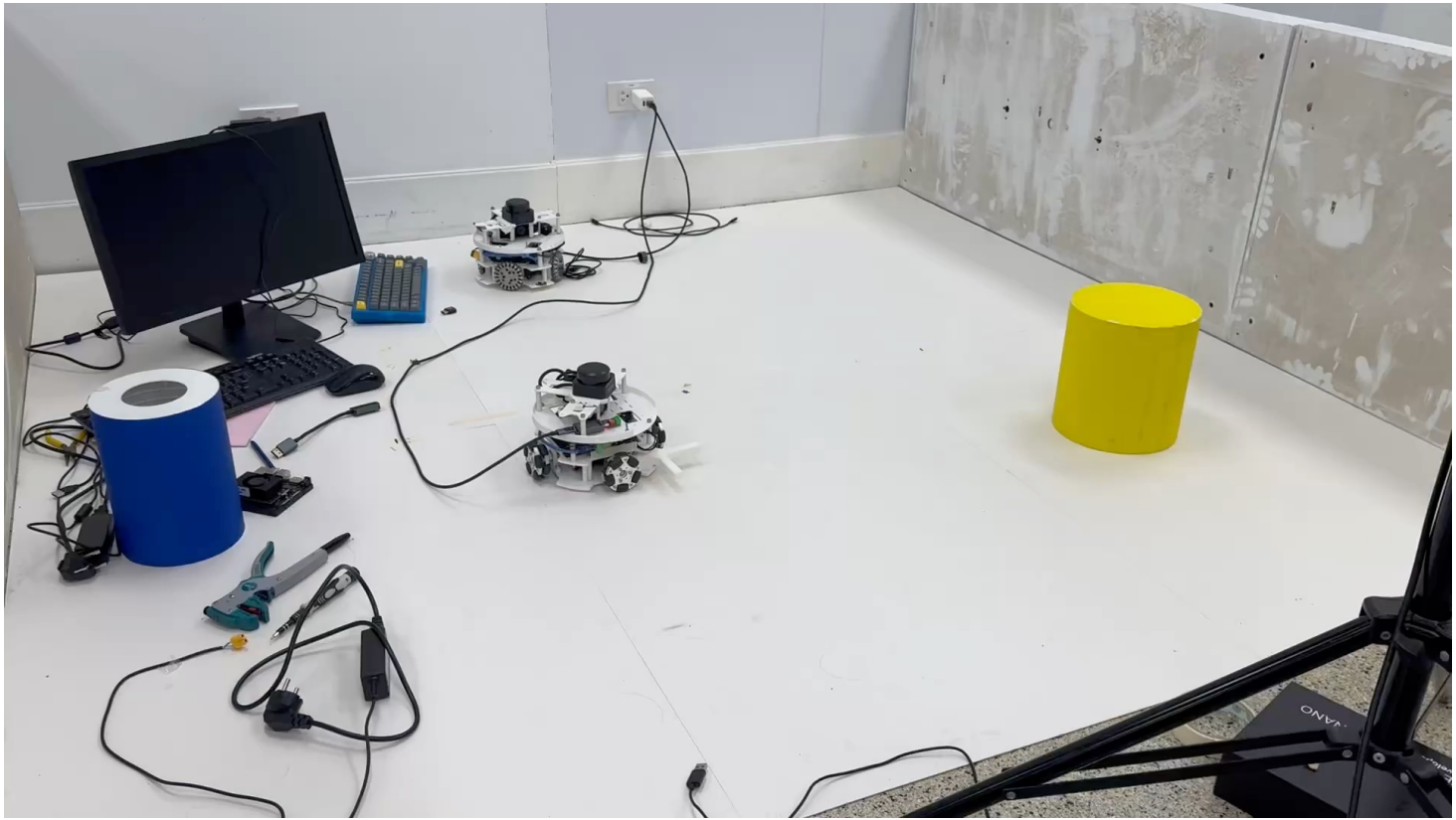
Demo Milestone : Autonomous Centering PID Inverse Kinematic



Demo Milestone : Activation Object Detection

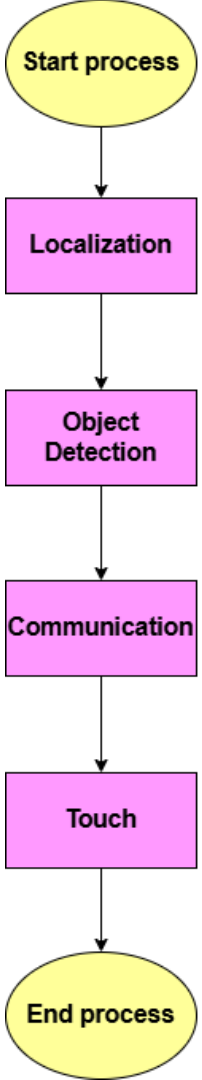


Demo Milestone : Autonomous Object Recognition and Navigation



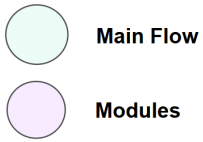
High-level Workflow

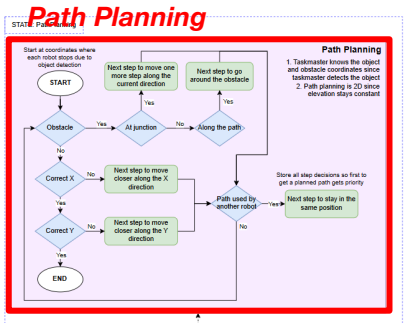
Process Overview

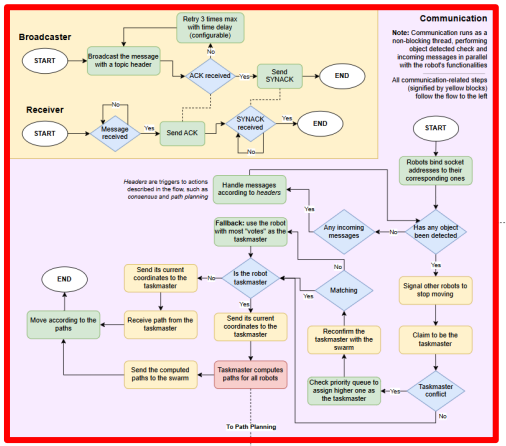


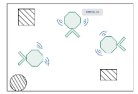
High-level Workflow

5 Modules

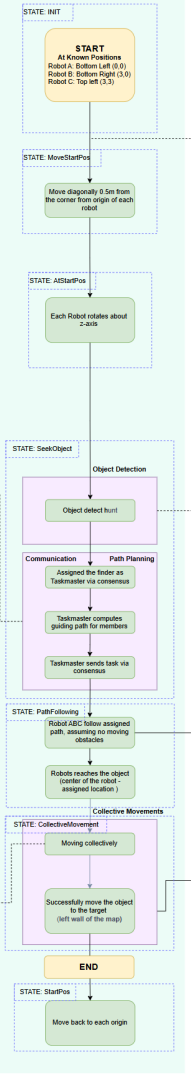


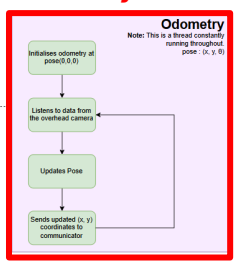


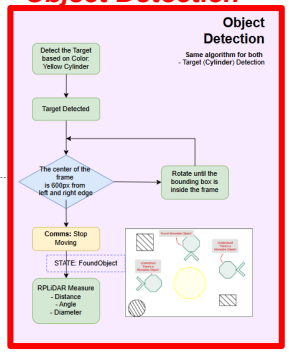


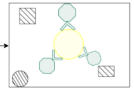


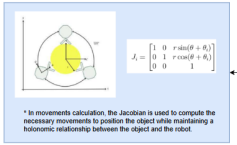
Move to 50cm towards the center
Robot A: Bottom Left (0.35,0.35)
Robot B: Bottom Right (2.65,0.35)
Robot C: Top left (2.65,2.65)

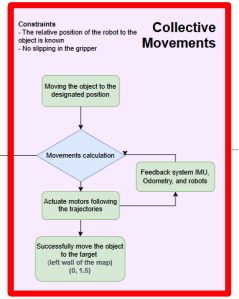


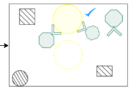












Modules Milestones

 *Meet the goal and Done*

 *Either doesn't meet the goal or not done*

 *Neither*

**According to our goals*

Milestone	Description	Status
Hardware	Built 3 robots fully functioning	
Object Detection and Measurement	Able to measure **width, angle, and distance** accurately regardless of the object size, color, and lighting.	
Communication	Able to communicate P2P, no centralized server	
Path Planning	From scratch move in global x, y, theta	
Localisation	Using SLAM	

Module 1

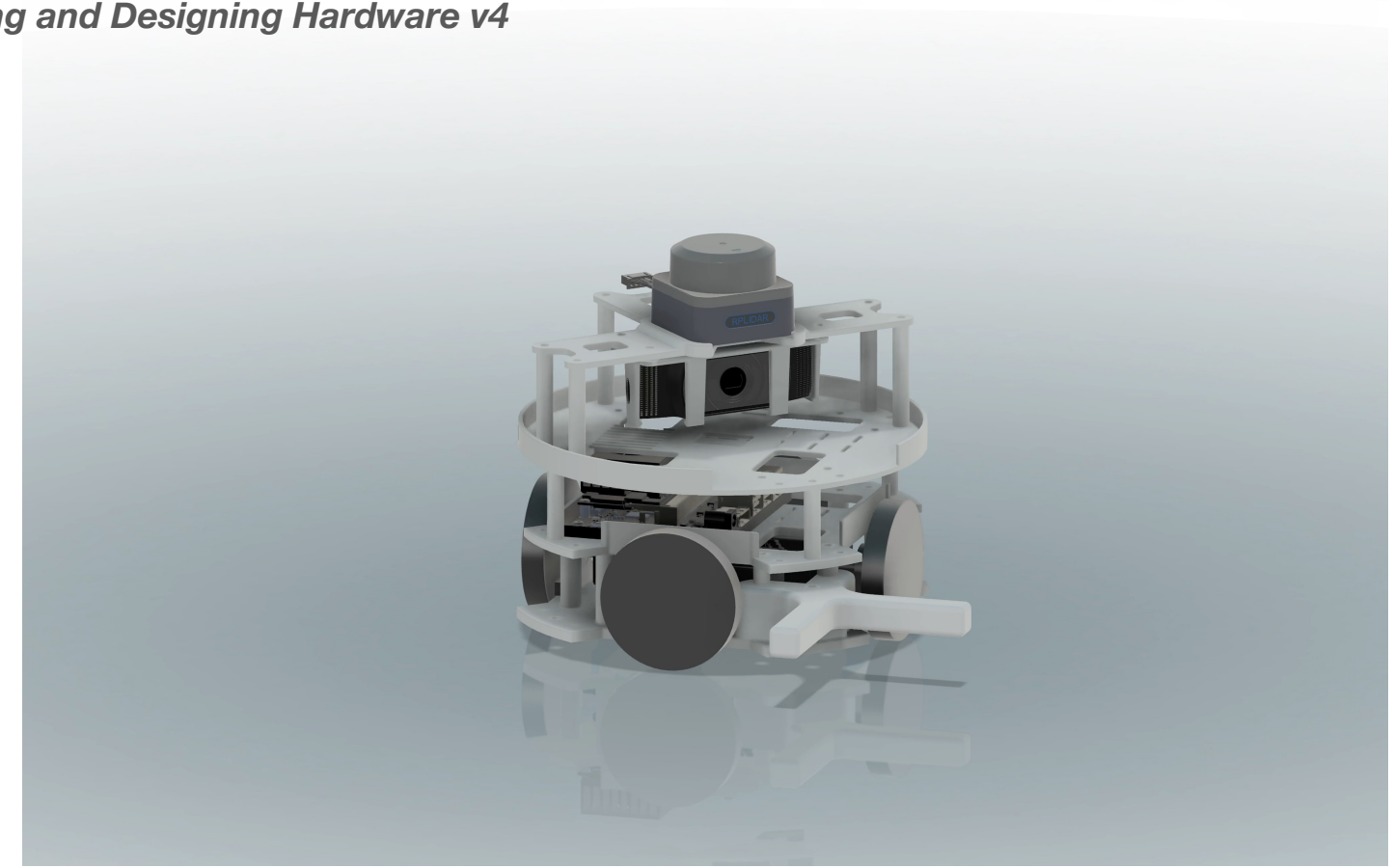
Hardware

Enviromental Requirements

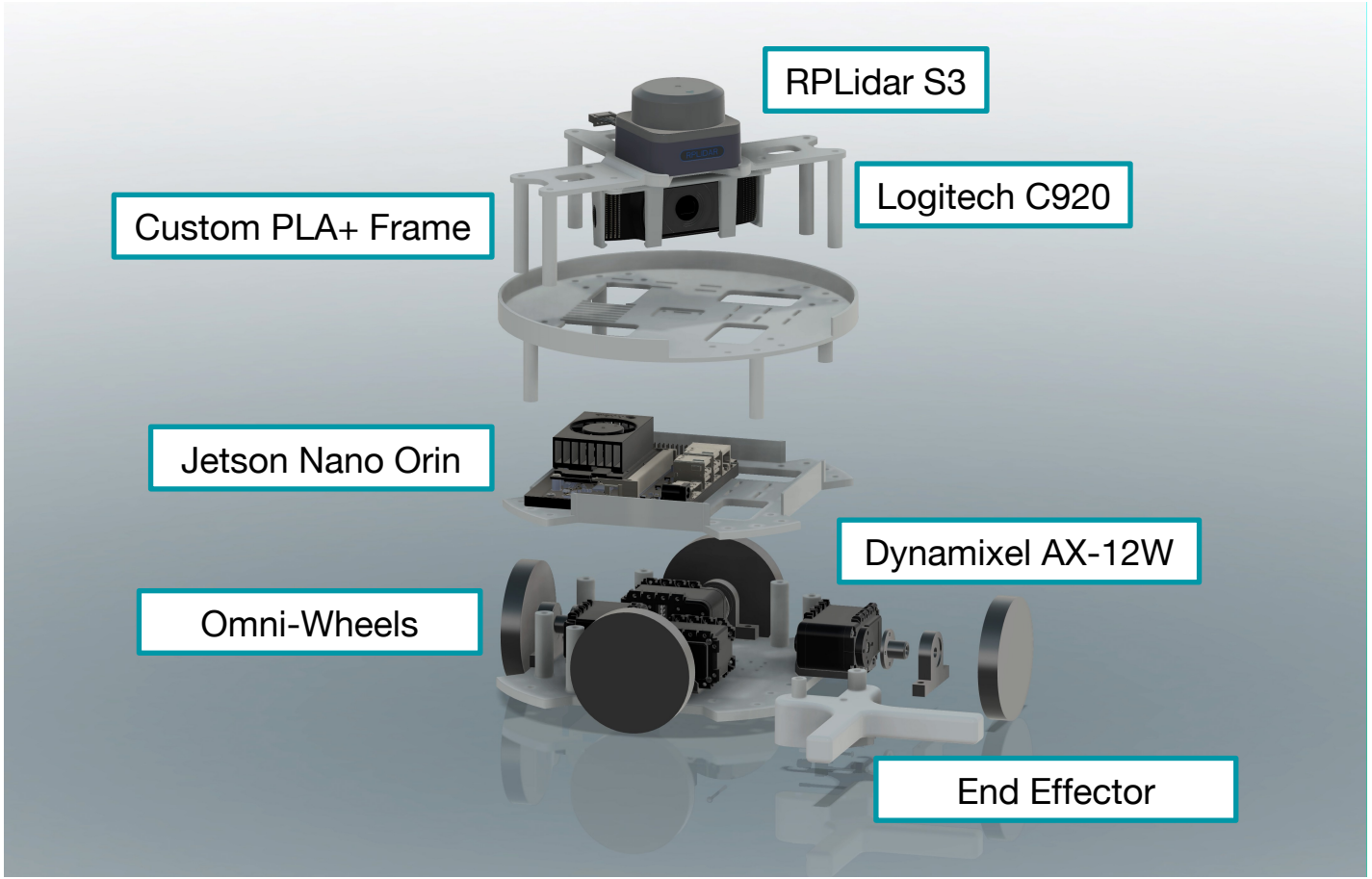
- Lab condition
  - Flat surface to operate on -> Chulapat Floor are not flat
  - Area Size: 4 square meters
  - Non-reflective floor

Hardware Breakdown

Planning and Designing Hardware v4

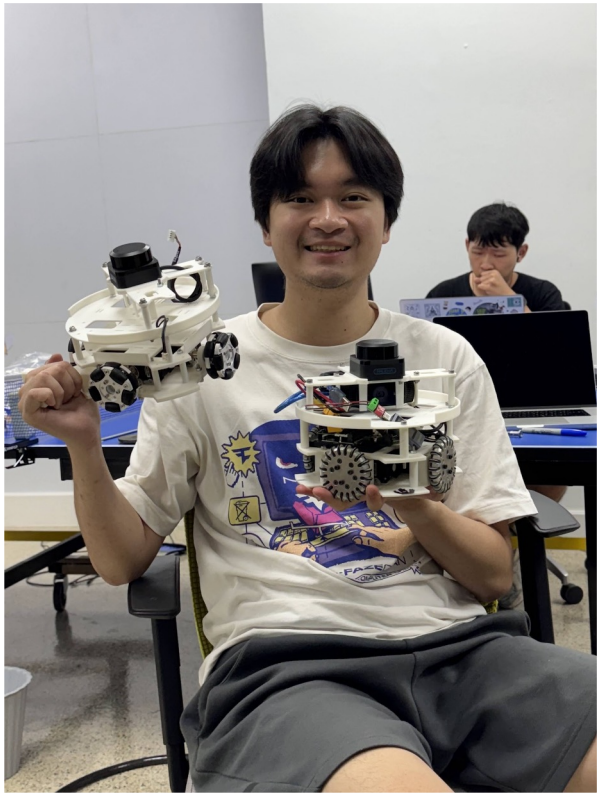


Hardware Breakdown



Hardware

What Happened with Hardware?

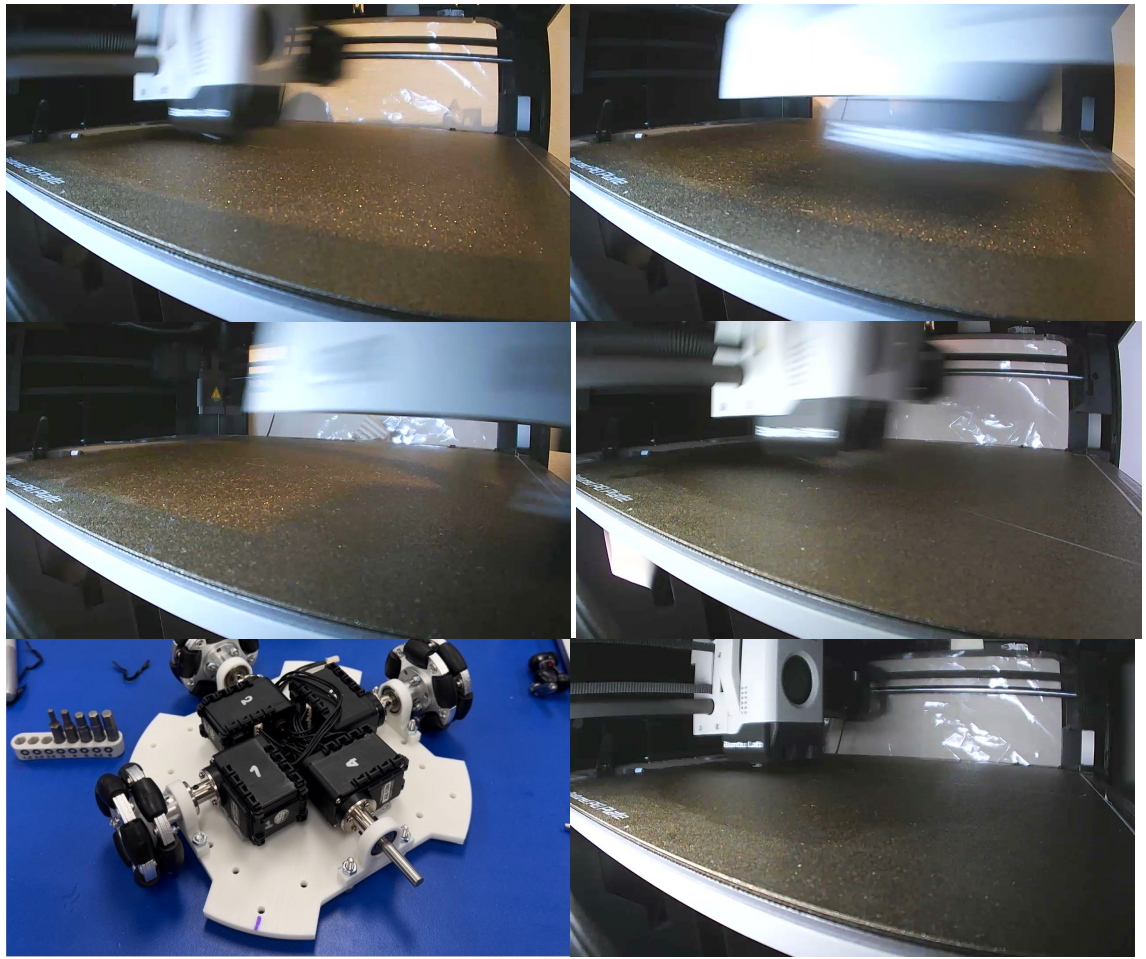


Hardware

How it's made? BOQ

3D Printed Frame using PLA+

4x AX-12W Dynamixel Motors
4x Omni Wheels
4x Bearing 626ZZ 06x19x06mm
1x RPLidar S3 Lidar
1x Logitech C920 RGB Camera
1x 6mm Shaft 304
1x Jetson Orin Nano
1x USB – TTL comms board

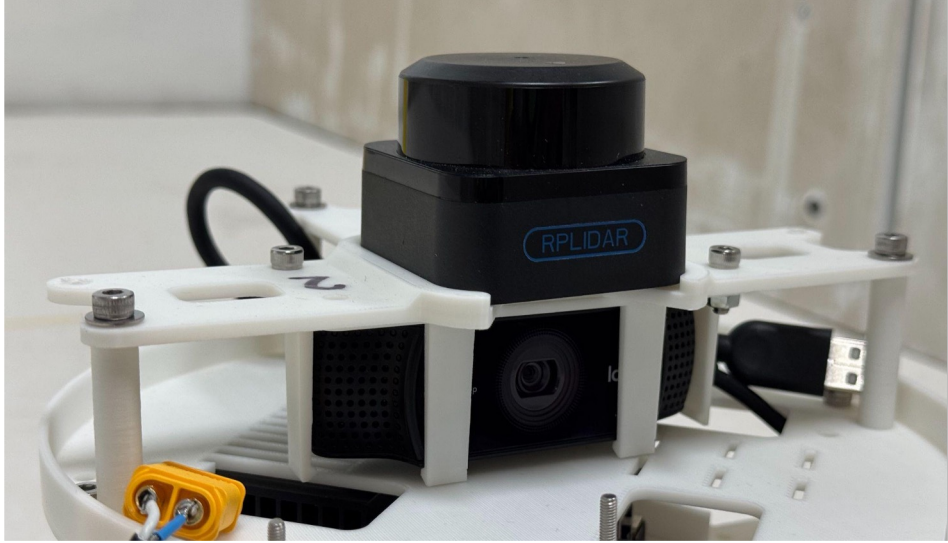


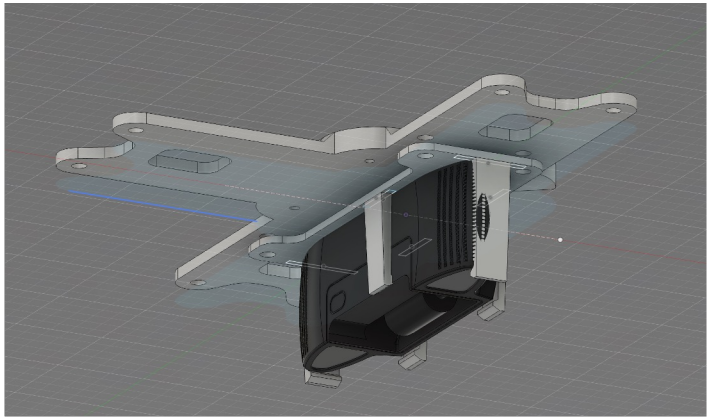
Hardware

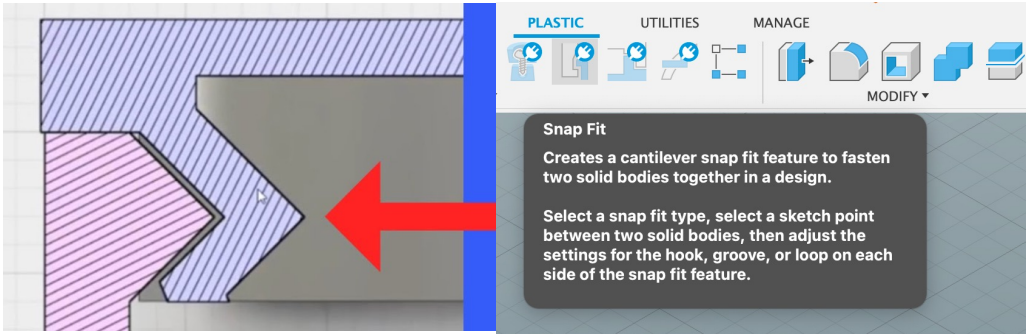
How does it see?

Designing a special mount for the Logitech C920 camera + Lidar

- **Fusion360: Snap Fit**





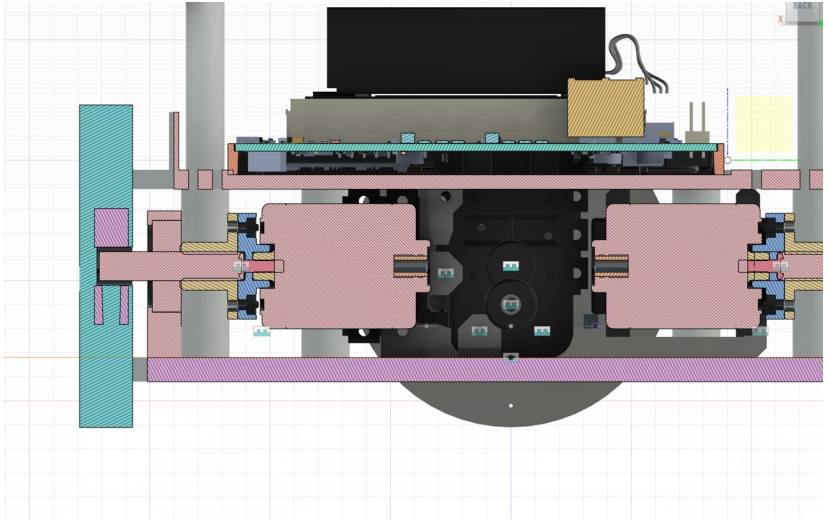


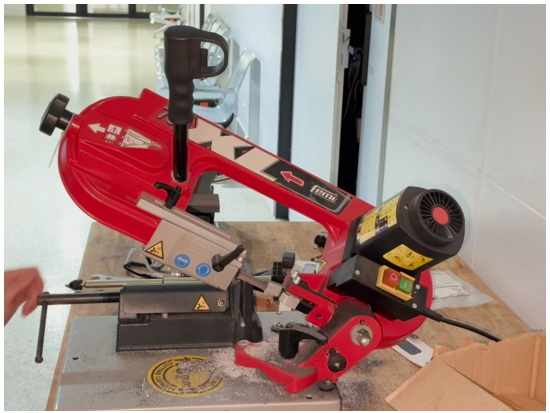
Snap Fit learnt in Robotic Project IV

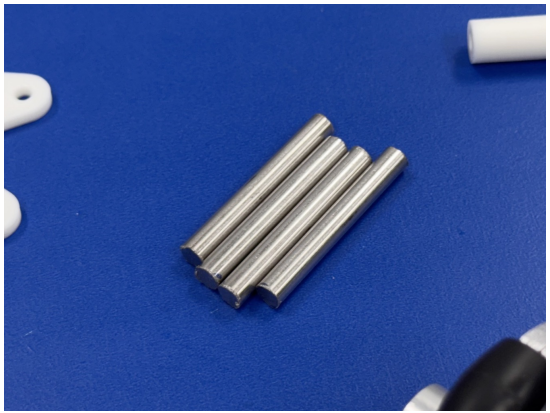
Hardware

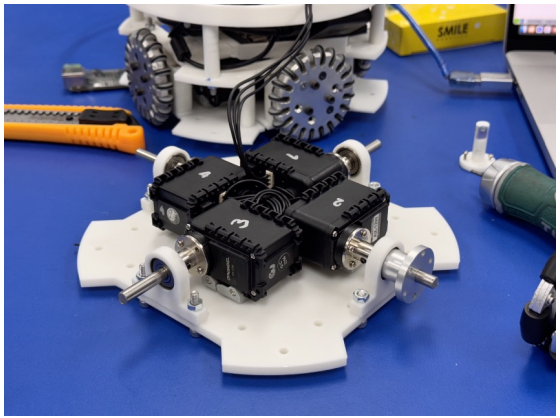
Version before v1 v2, Press fit of the bearing

- **Plastic Shaft -> Metal Shaft**
- **Cable Barrier for better routing**
- **Experiment with flexible base**
- **Problem** Difficult to find a Pillow for Bearing 626ZZ 06x19x06mm

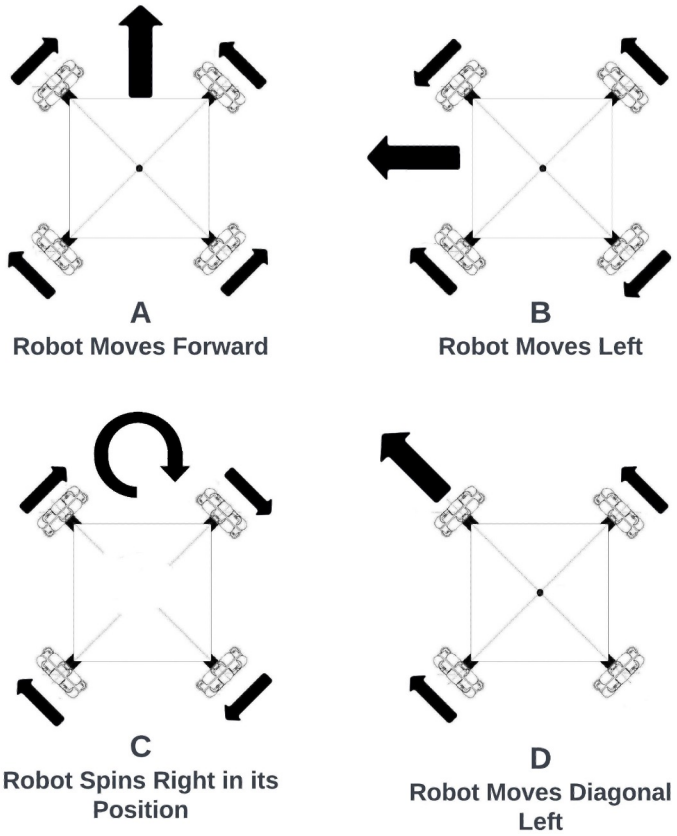








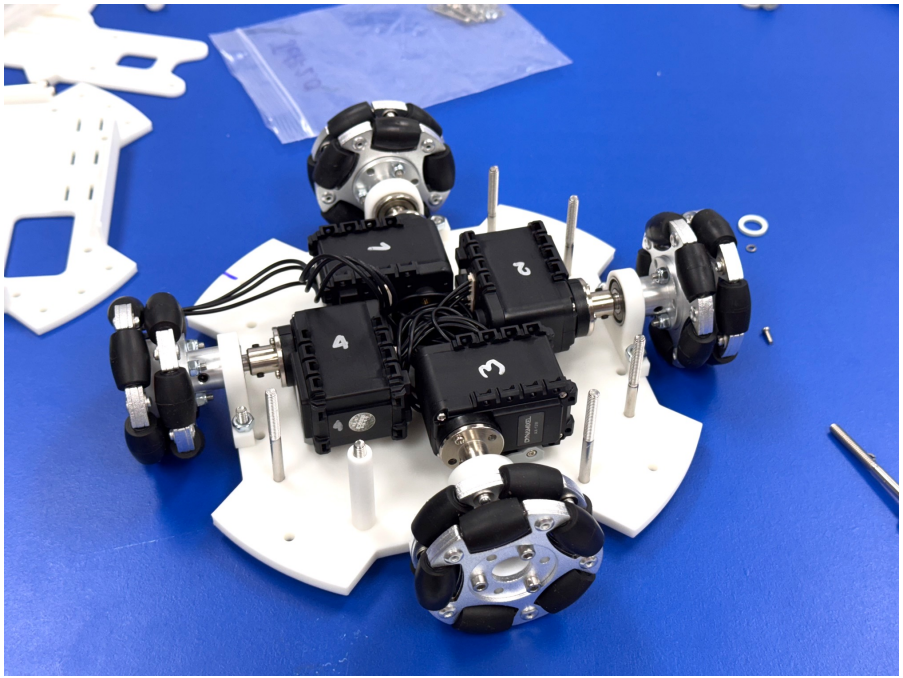
Hardware Robot Movement



Ref: https://www.theroboticsspace.com/blog/Kinematic-Modelling-for-Mobile-Robots/

Hardware

Basic Kinematics, Industrial Robot



Forward Kinematics

$$V_r = \begin{bmatrix} V_x \\ V_y \\ \dot{\theta} \end{bmatrix} = J \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \quad (1)$$

$$J = \frac{r}{2} \begin{bmatrix} -sin\left(\theta + \frac{\pi}{4}\right) & cos\left(\theta + \frac{\pi}{4}\right) & \frac{1}{2R} \\ -sin\left(\theta + \frac{3\pi}{4}\right) & cos\left(\theta + \frac{3\pi}{4}\right) & \frac{1}{2R} \\ -sin\left(\theta + \frac{5\pi}{4}\right) & cos\left(\theta + \frac{5\pi}{4}\right) & \frac{1}{2R} \\ -sin\left(\theta + \frac{7\pi}{4}\right) & cos\left(\theta + \frac{7\pi}{4}\right) & \frac{1}{2R} \end{bmatrix}^T \quad (2)$$

Inverse Kinematics

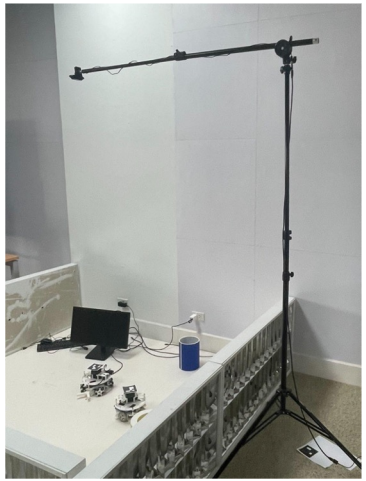
$$V_w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} -sin\left(\theta + \frac{\pi}{4}\right) & cos\left(\theta + \frac{\pi}{4}\right) & R \\ -sin\left(\theta + \frac{3\pi}{4}\right) & cos\left(\theta + \frac{3\pi}{4}\right) & R \\ -sin\left(\theta + \frac{5\pi}{4}\right) & cos\left(\theta + \frac{5\pi}{4}\right) & R \\ -sin\left(\theta + \frac{7\pi}{4}\right) & cos\left(\theta + \frac{7\pi}{4}\right) & R \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ \dot{\theta} \end{bmatrix}$$

Citation: Phunopas, A., & Inoue, S. (2018). Motion improvement of four-wheeled omnidirectional mobile robots for indoor terrain. *Journal of Robotics, Networking and Artificial Life*, *4*(4), 275-282.


ubuntu


>_
SSH


WebSocket




Wi Fi

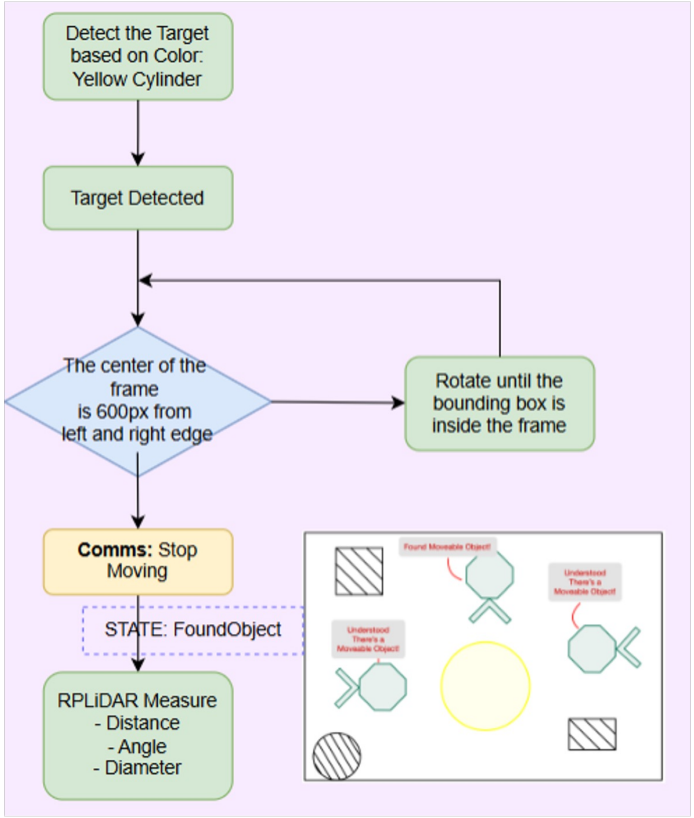

ubuntu

Module 2

Object Detection

Object detection

Obtaining the output



Hardware

S3 RPLiDAR

Measuring distance and angle using rclpy

Logitech C922 Webcam

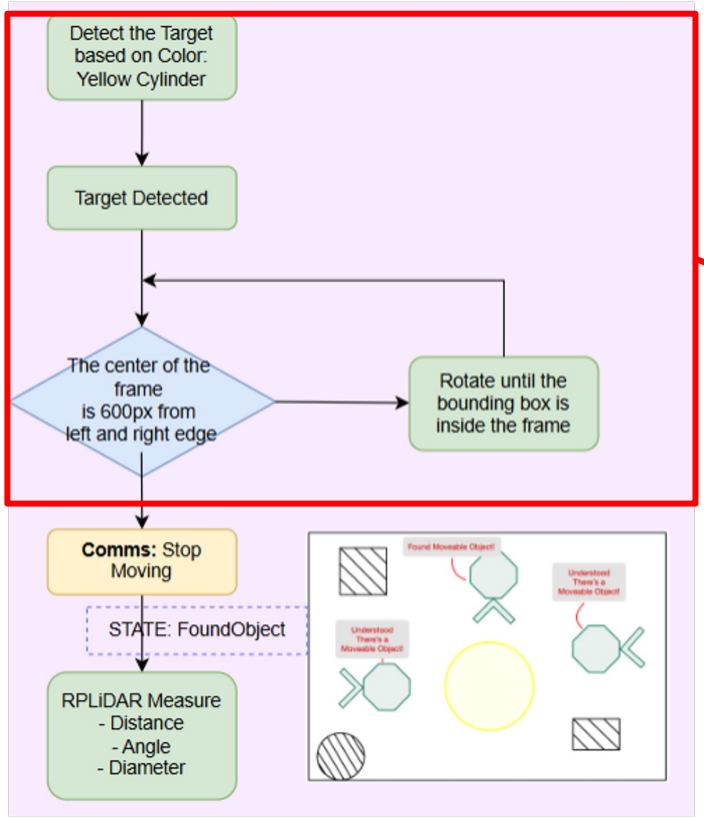
Detecting the target based on color with YCbCr color profile

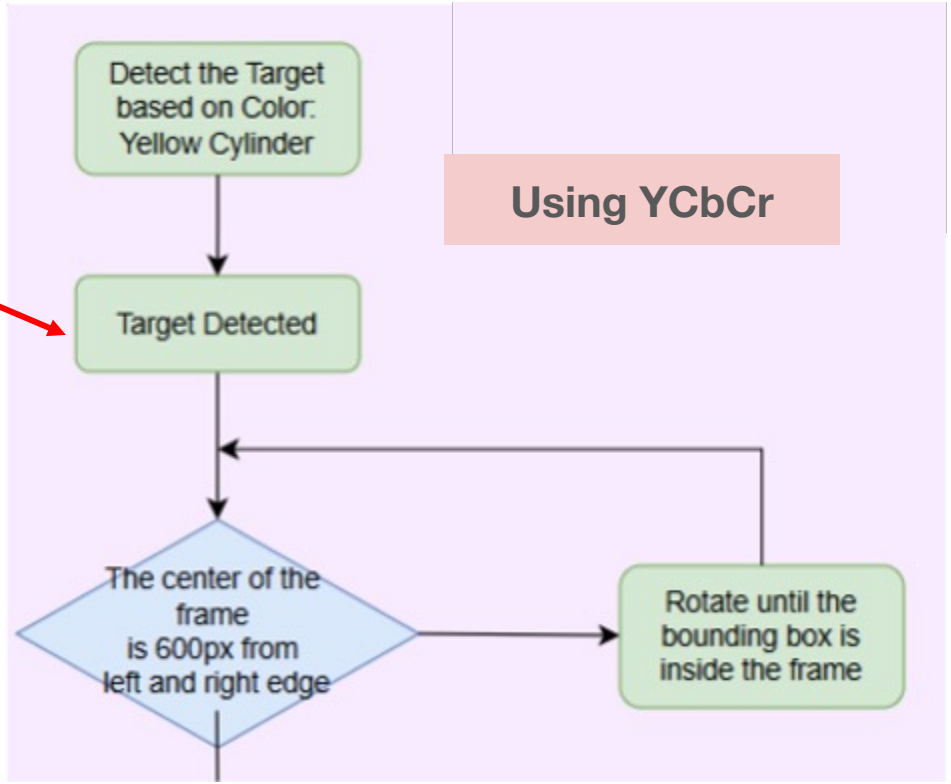
Data Output

- **Distance (m)** from the camera to the target surface
- **Width (m)** of the detected target (diameter)
- **Angle (degree)** made between the centre of the frame and the centre of the bounding box around detected object

Object detection

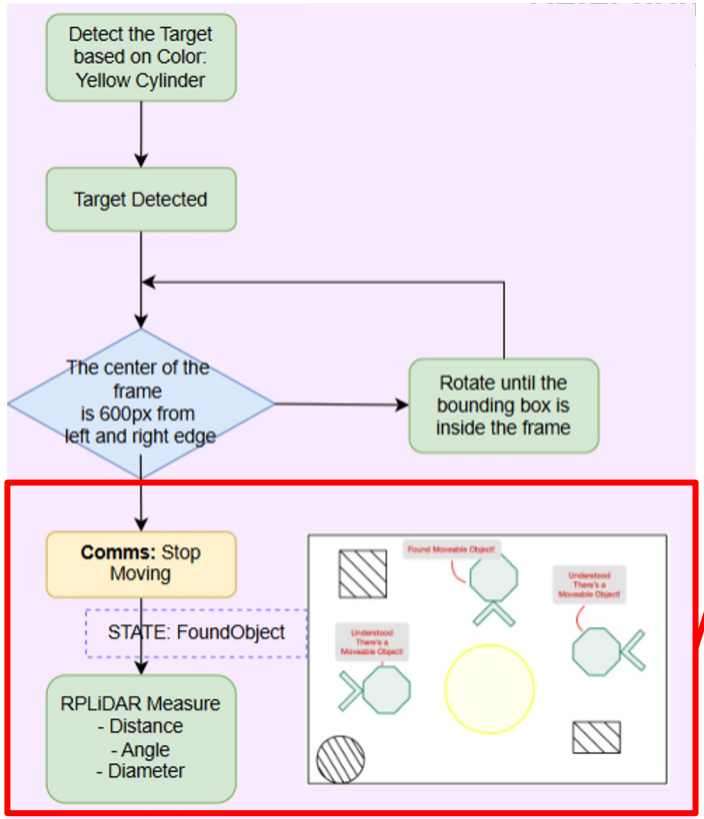
Workflow

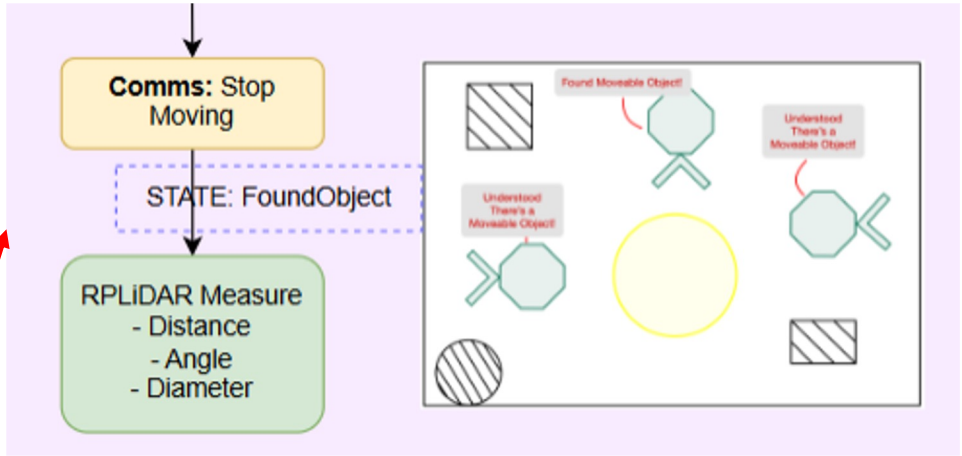




Object detection

Workflow

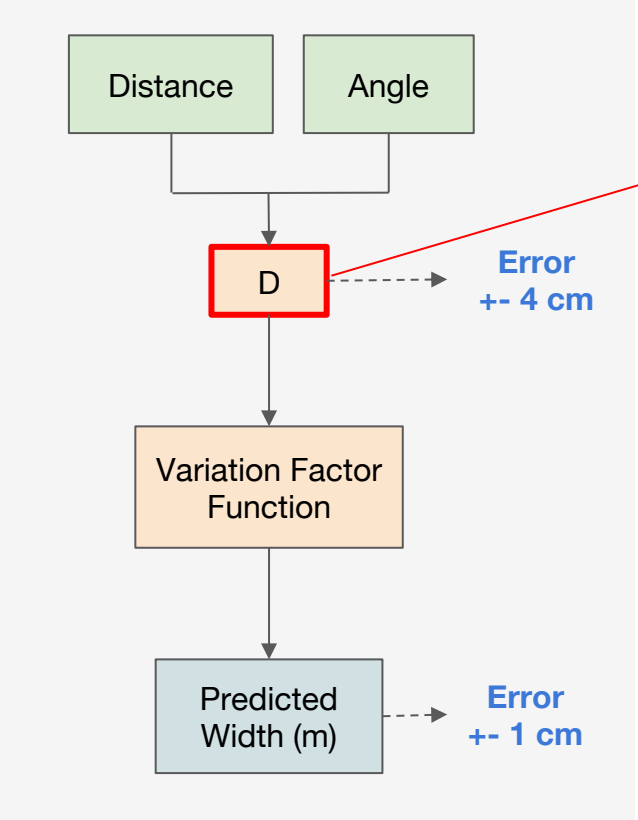


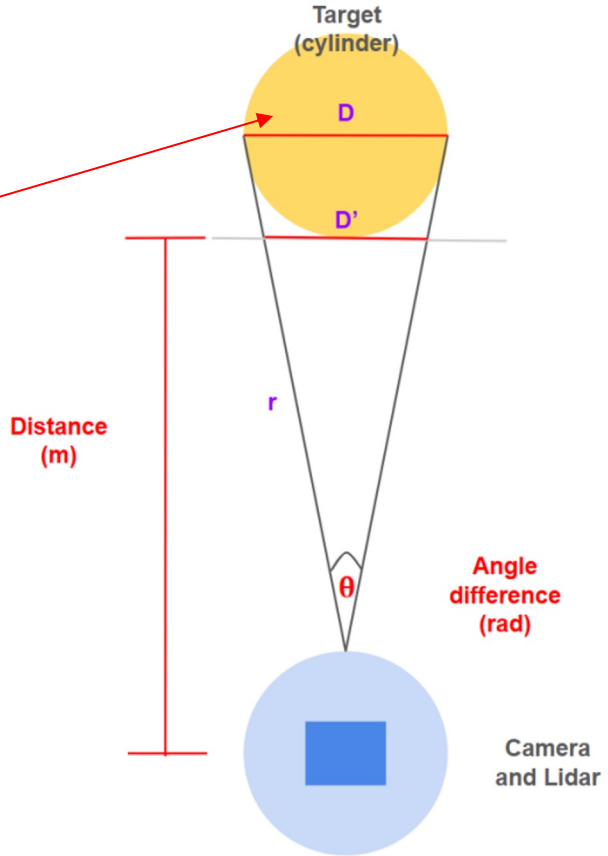


Obtaining the output from the LiDAR by using **camera's FOV** to map location from camera to LiDAR space.

Object detection

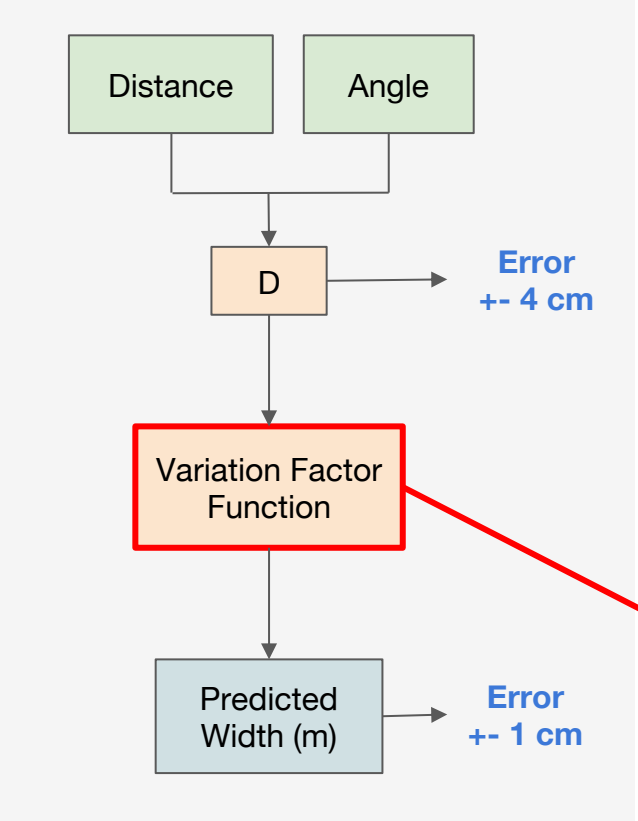
Obtaining the output

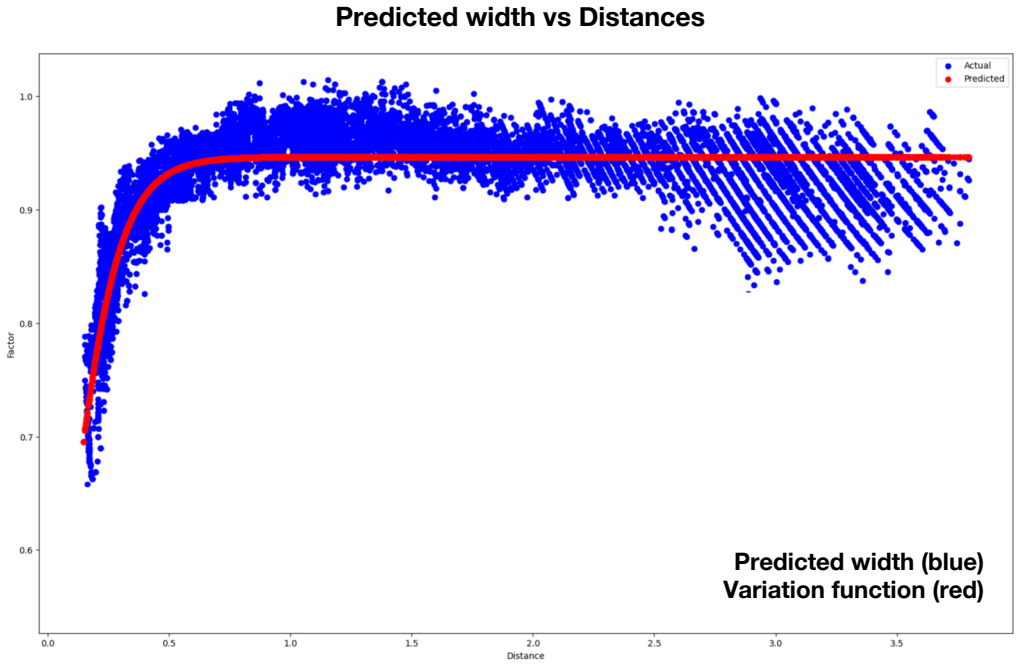




Object detection

Obtaining the output



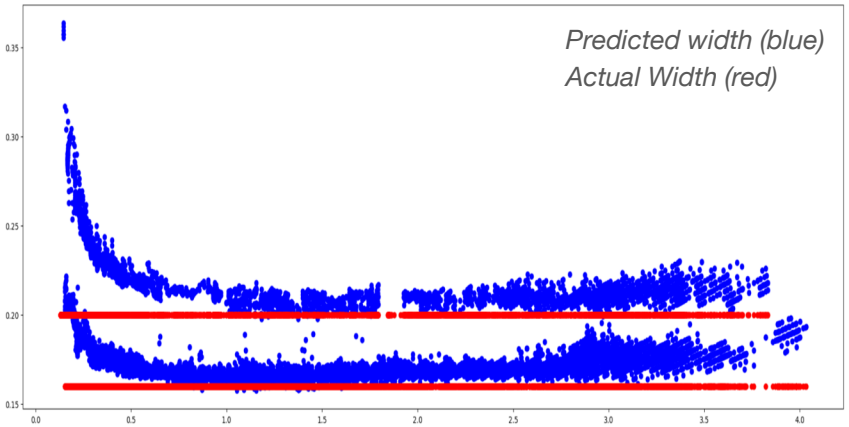


$$\text{factor} = \frac{0.9461218686}{1 + e^{-9.0309891171(x - 0.0335094048)}}$$

Object detection

Comparison on Predicted Width by Applying Variation Factor Function

Before



After

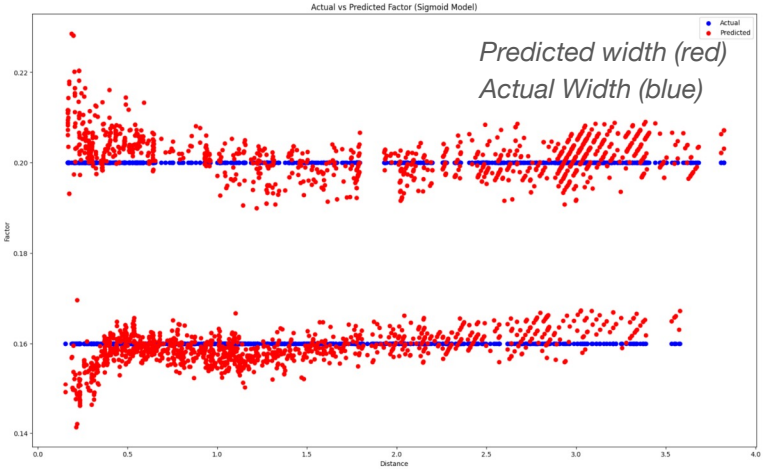
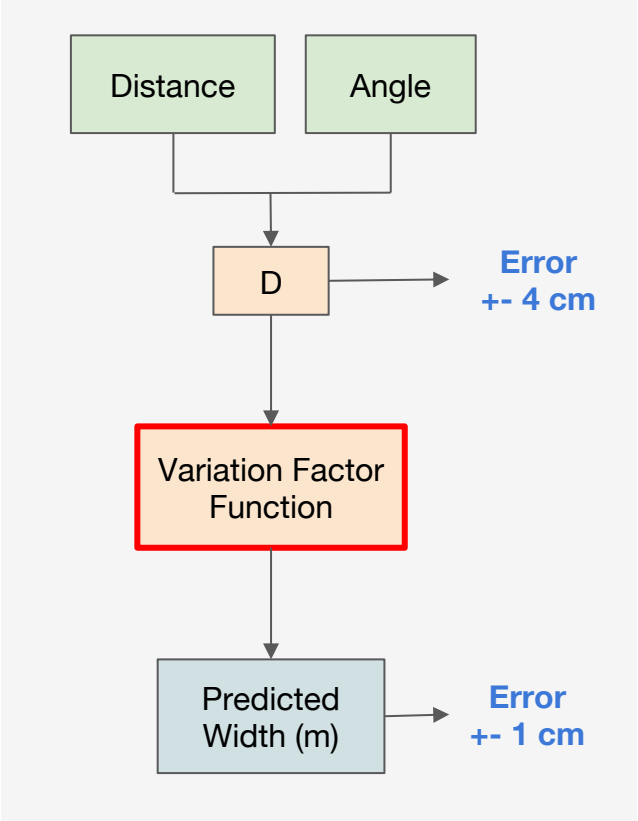
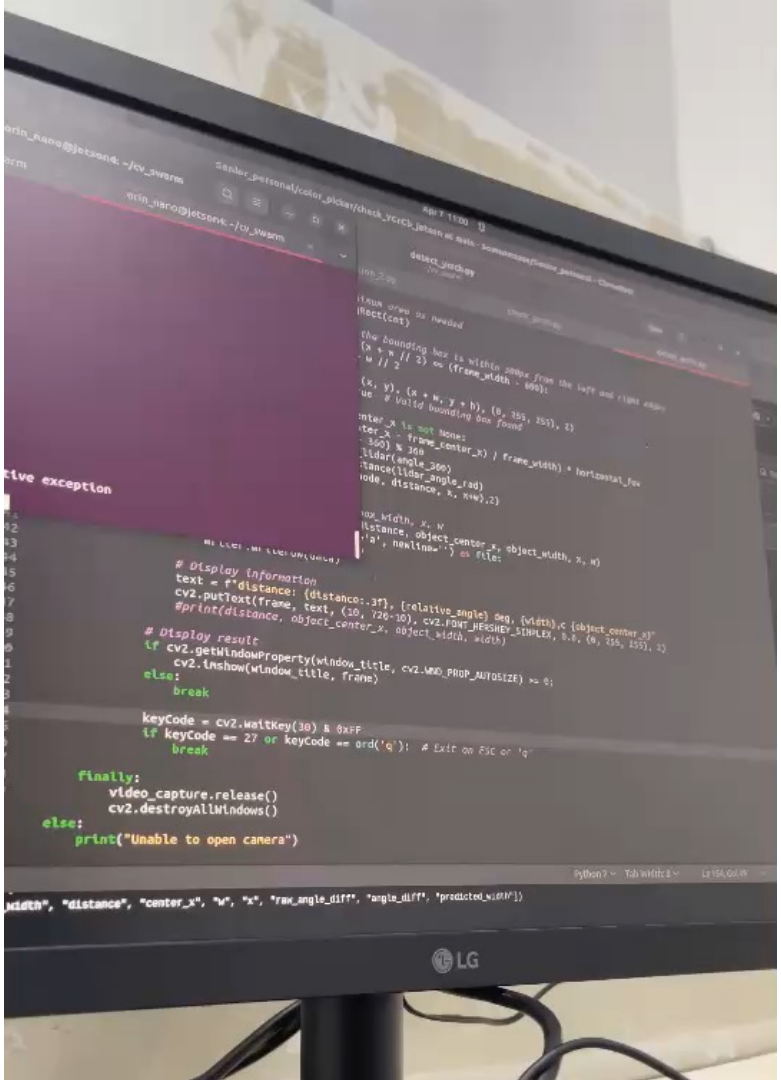


Table	Mean Absolute Width Error (m)	Mean Absolute Angle Error (°)
16 cm	0.0022	0.2389
20 cm	0.0017	0.2219
12 cm	0.0061	1.0519

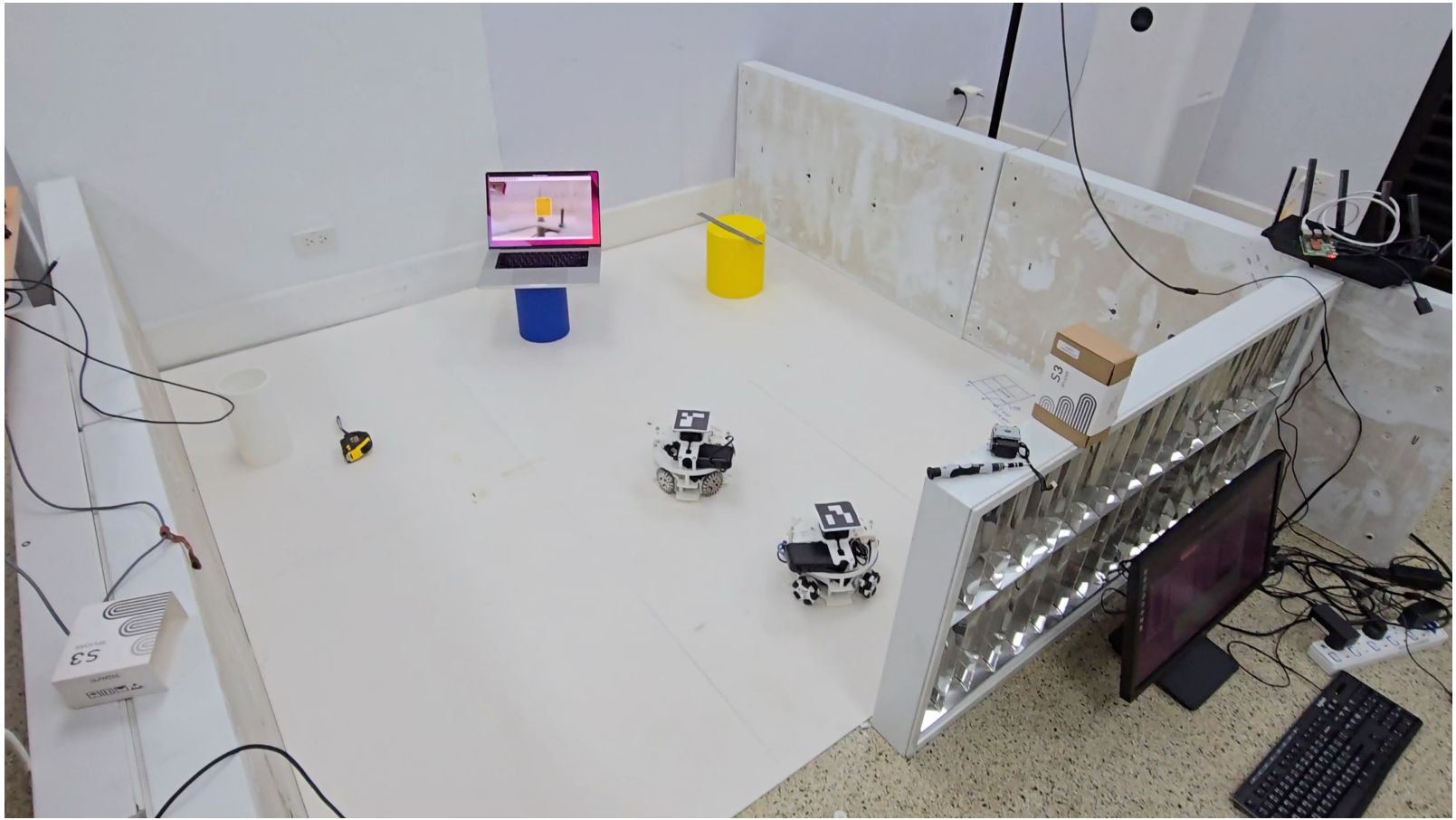
Object detection

Measurement and Width Prediction Result





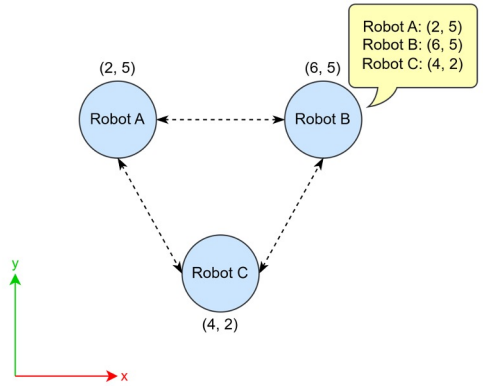
Demo Milestone : Activation Object Detection

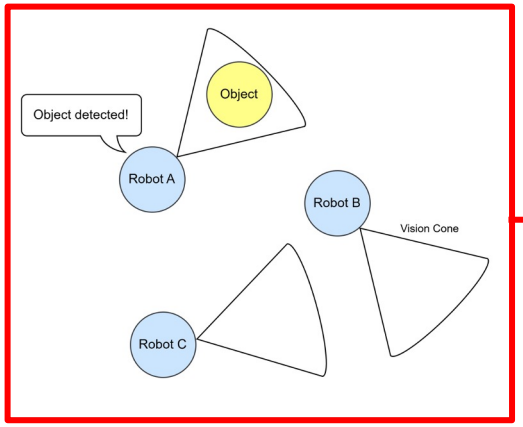


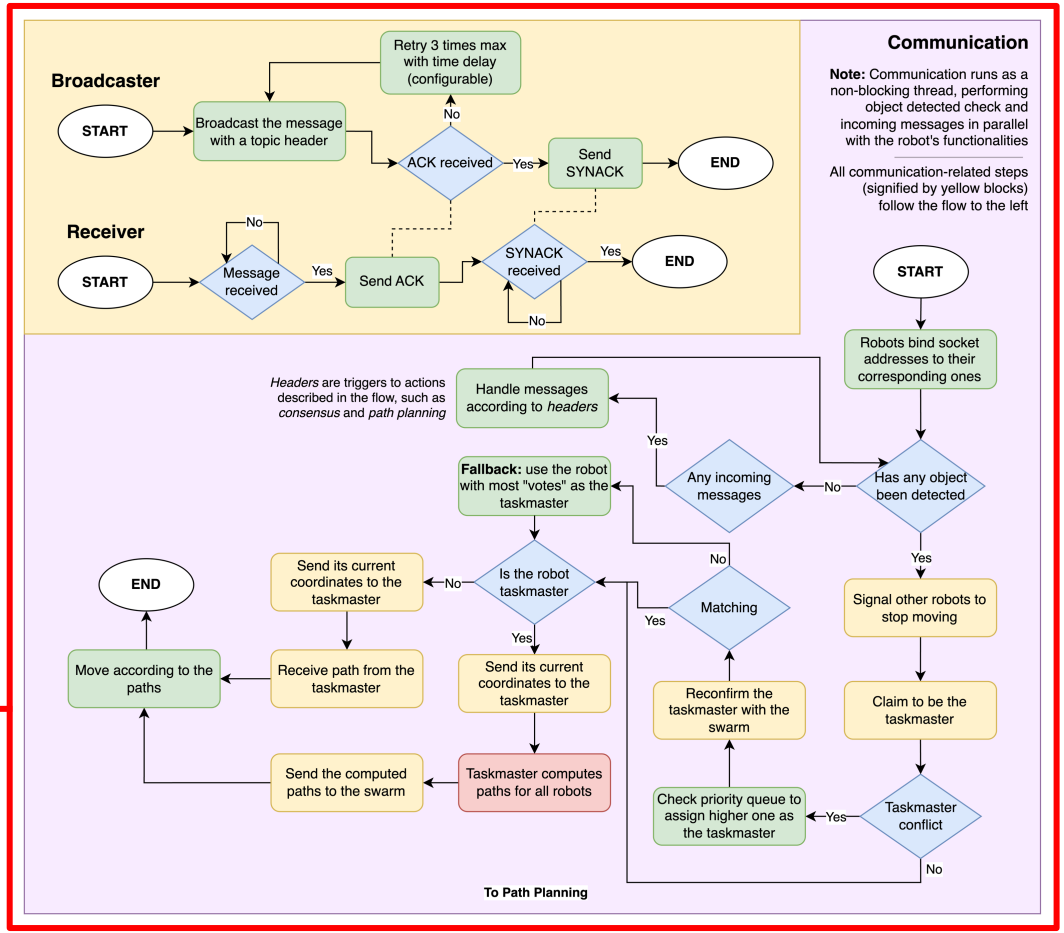
Module 3

Communication

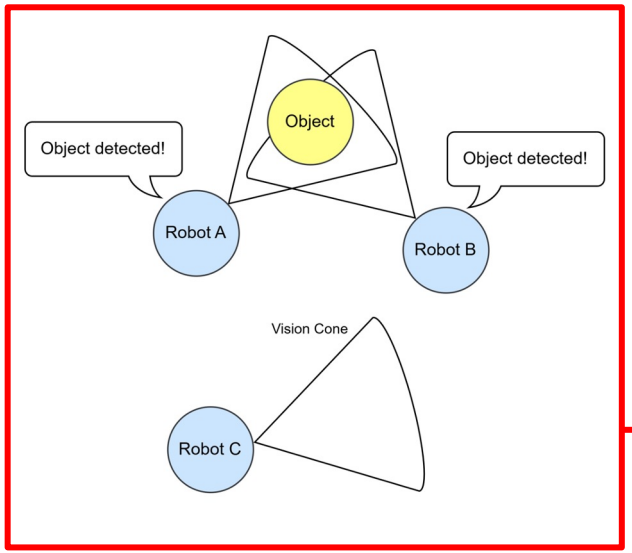
Communication - *Overview*

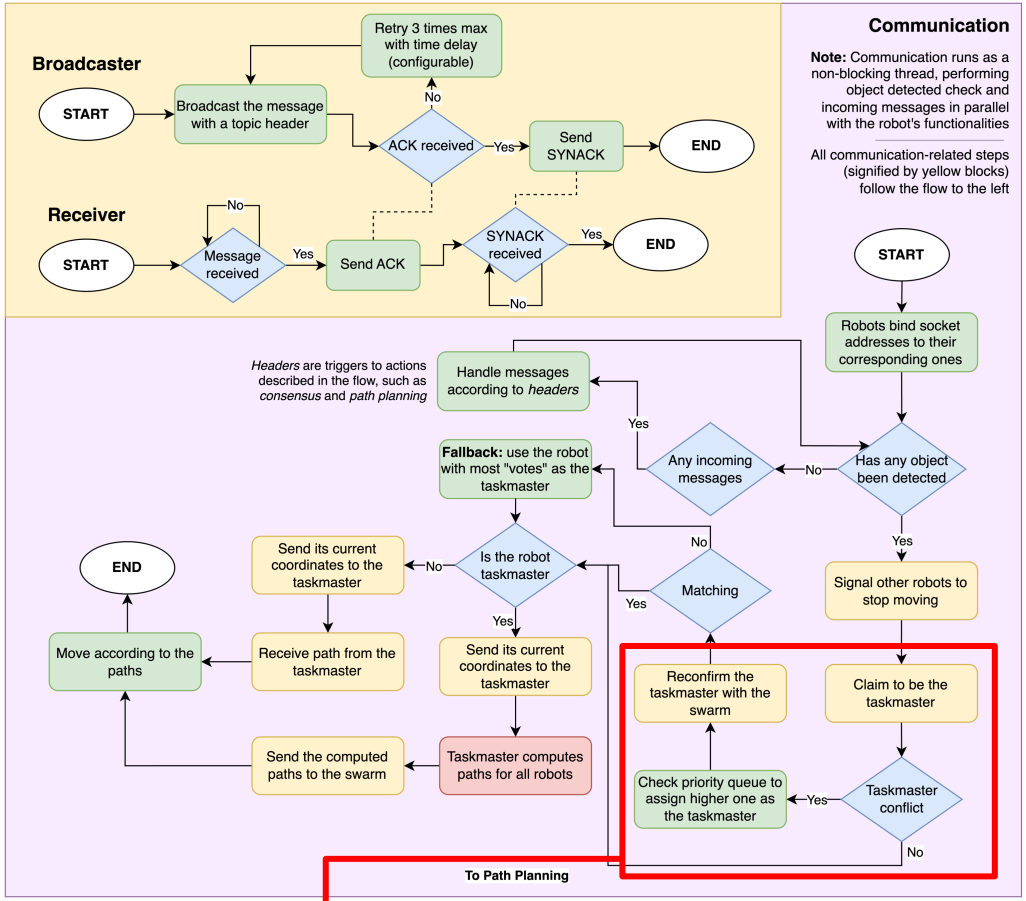






Communication - *Consensus*

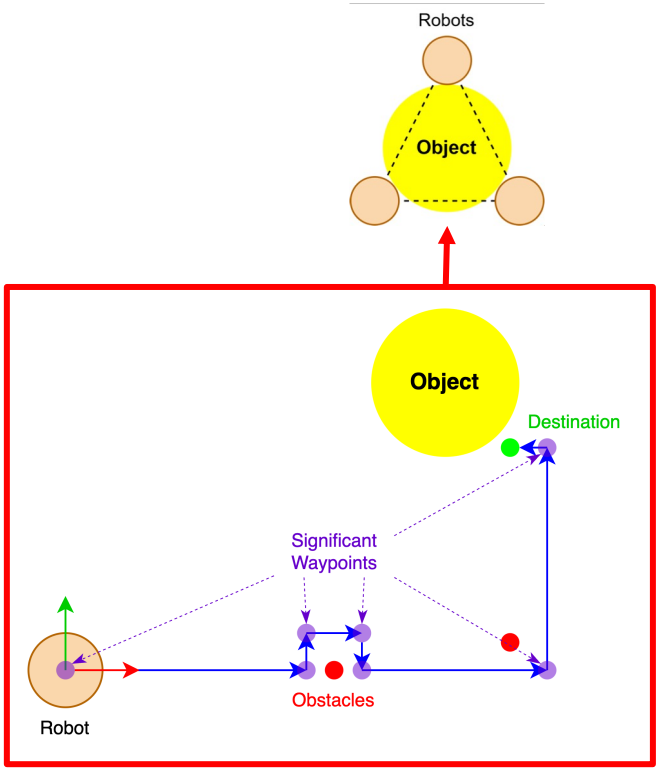


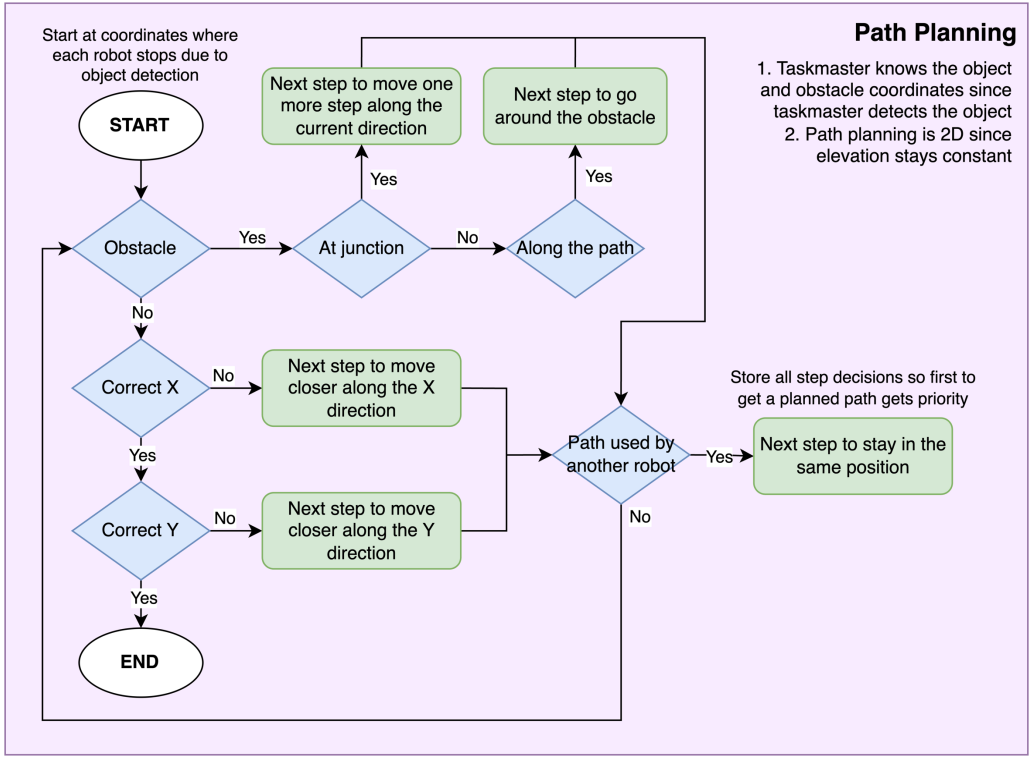


Module 4

Path Planning

Path Planning





Module 5

Localization

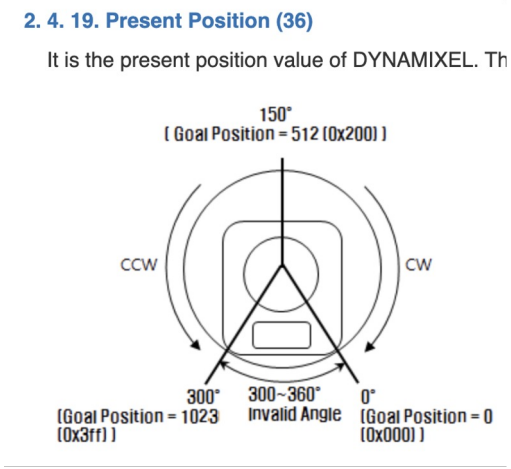
SLAM

SLAM issues

- **Objective:** Use SLAM to build a map, correct odometry, and enable autonomous navigation.
- **Problem 1 – Limited Arena Size:**
  - Arena dimensions (~2.1 m × 1.4 m) are too small for effective SLAM operation.
- **Problem 2 – Sparse Environmental Features:**
  - SLAM requires distinctive landmarks for reliable scan matching.
  - The test area lacks sufficient geometric features, reducing map accuracy.
- **Problem 3 – Poor Odometry Quality:**
  - Wheel encoder odometry (Dynamixel) has a blindspot.
  - Lidar-based odometry (rf2o) also shows drift and instability.
  - Inaccurate motion estimates degrade SLAM performance and map consistency.

Dynamixel Wheel Odometry

- AX-12W in wheel mode, angle readings become invalid and unreliable.
- **Position register still updates, but:**
  - Values wrap around every 300° (~1023 → 0).
  - No way to track full revolutions or absolute angle.
  - Encoder output is an ambiguous unit.



Lidar Odometry - Overview

Lidar Odometry

- Uses range flow to estimate motion from sequential 2D lidar scans.
- Calculates odometry by analyzing how point distances shift over time.
- Package RF20

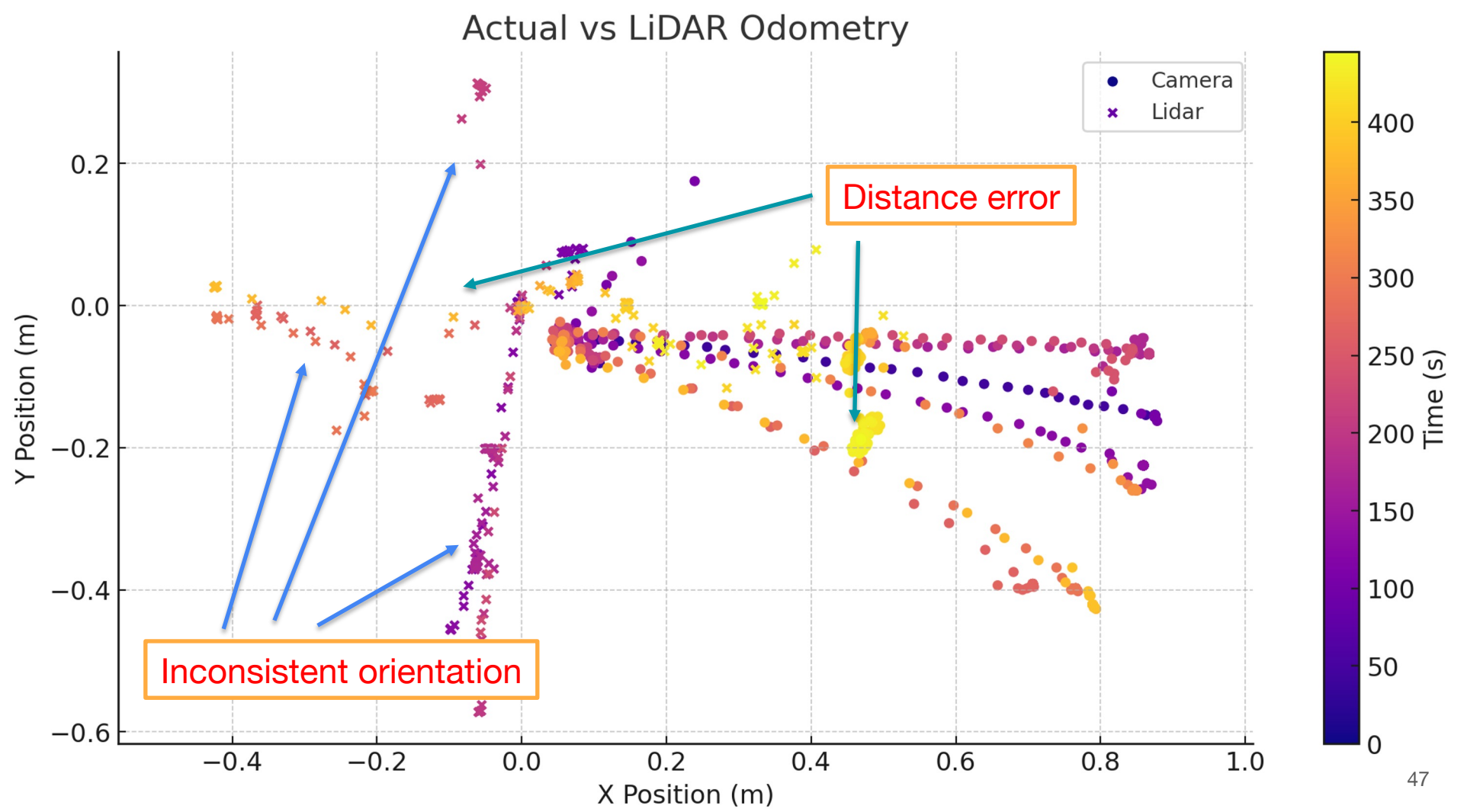
Lidar Odometry - Problem

Problem Observed

- Odometry shows continuous drift even when the robot is stationary.
- Estimated pose accumulates movement without real displacement.
- Inconsistent orientation

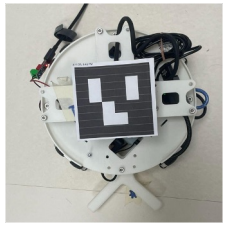
Likely Sources of the Problem

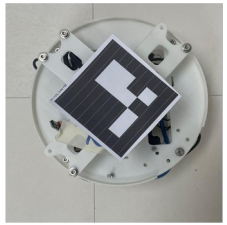
- Sensor noise from the RPLidar S3 interpreted as motion.
- Time synchronization errors causing scan mismatches.
- Incorrect rf2o parameter tuning (e.g., scan noise stddev too low).


Actual vs LiDAR Odometry
Camera
Lidar
Distance error
Inconsistent orientation
Y Position (m)
X Position (m)
Time (s)
0.2
0.0
−0.2
−0.4
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
400
350
300
250
200
150
100
50
0

Implementation of ARUCO [QR-Code]

“Position data is streamed from the server to the clients in real time.”





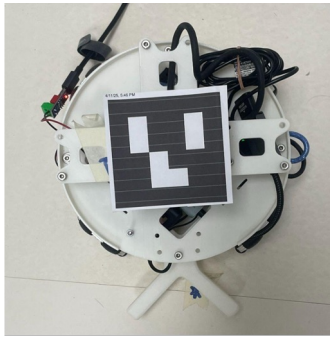
Robot Localization using ARUCO [QR-Code]

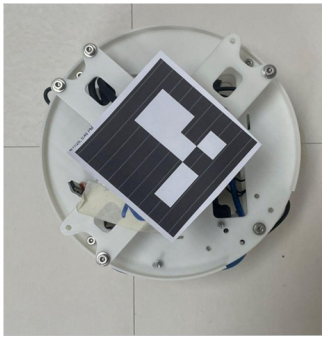
"A substitution for a robot's precise localization in a confined environment."

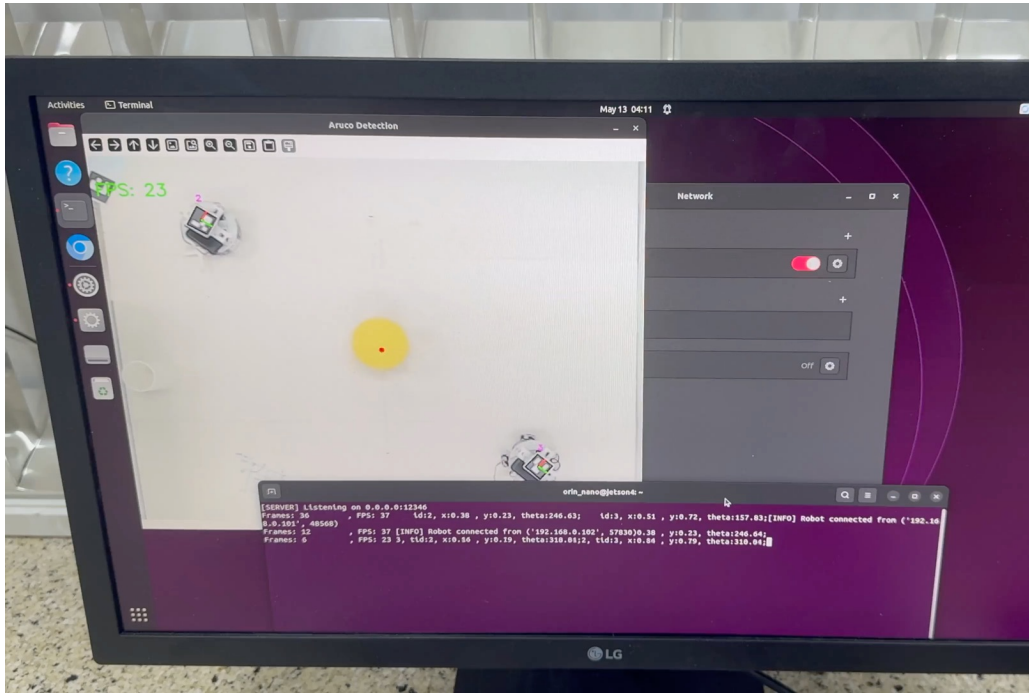
Use Cases:

1. Odometry Validation
2. Robot Localization

Precision: +- 2 cm, +-3 degree



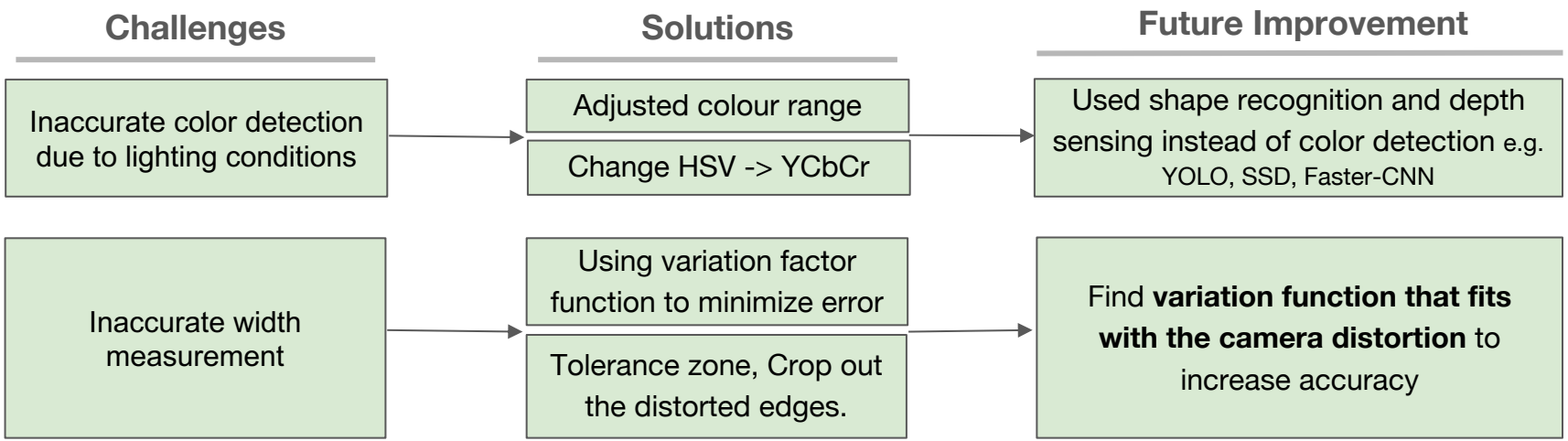




Challenges and Lesson Learned

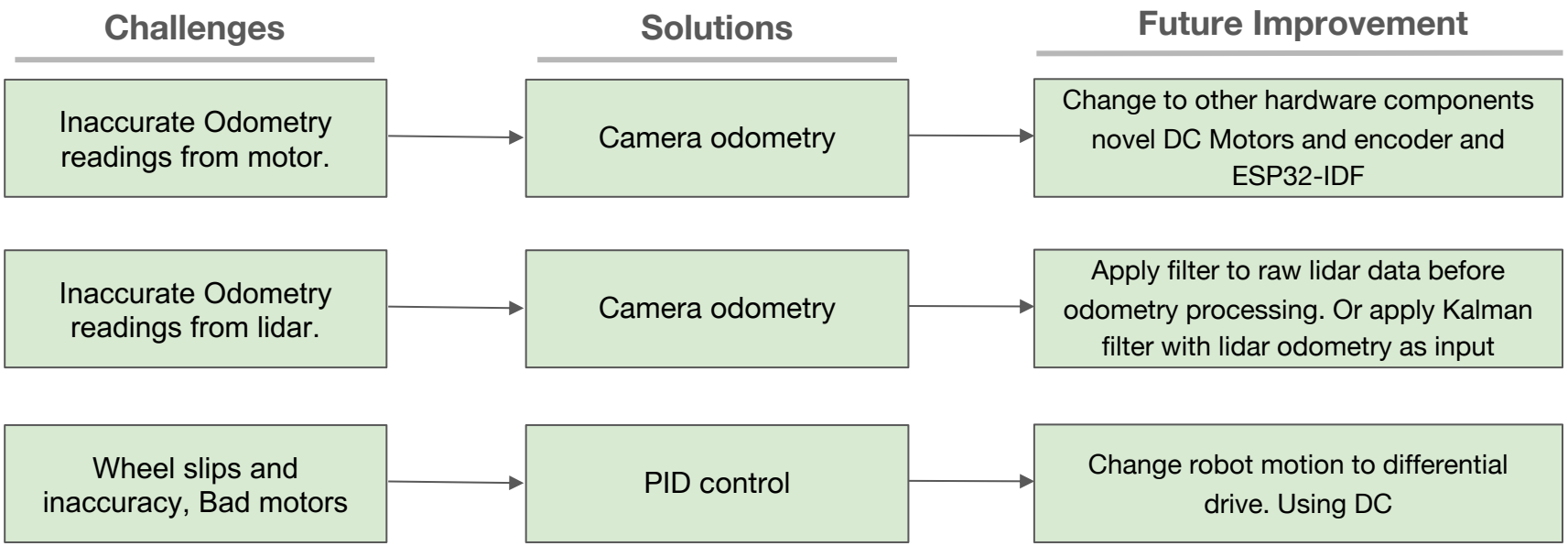
Object detection

Challenges and Solution



Hardware

Challenges and Solution



Lesson Learned

3 Key Learnings

Read the documentation and specification & Better Hardware Requirements

High level flow of the project, Better planning

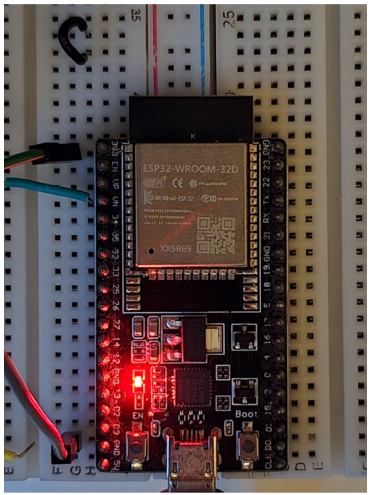
Fewer, more focused objectives

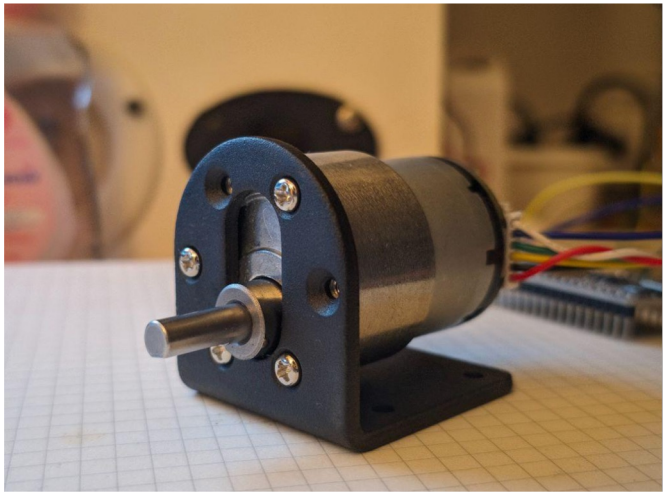
Hardware – Implementing

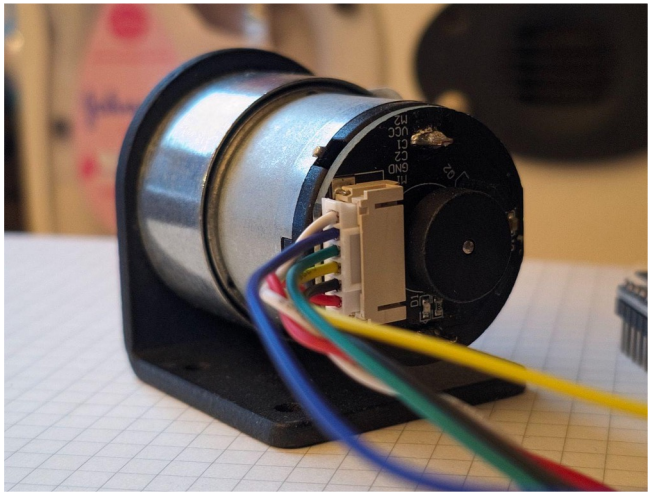
With the mentioned Dynamixel motor issues. We are designing a new motor base using cheap dc motor with encoder.

- ESP-IDF(Embedded C, not like slow Arduino)
- Metal and Thick Acrylic Base

ESP-IDF





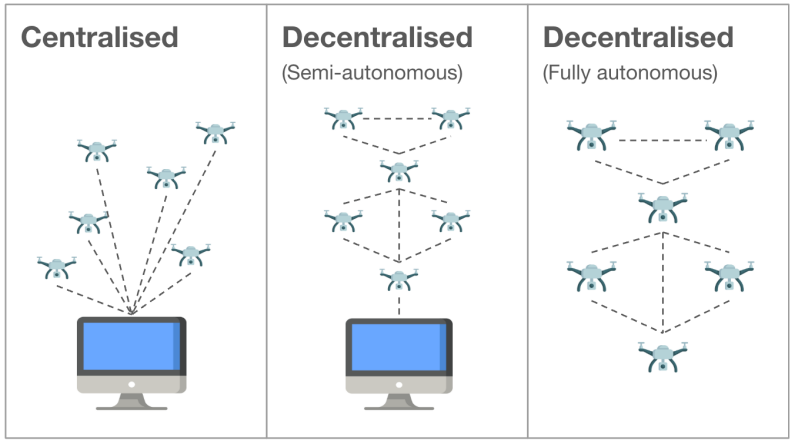


Q&A

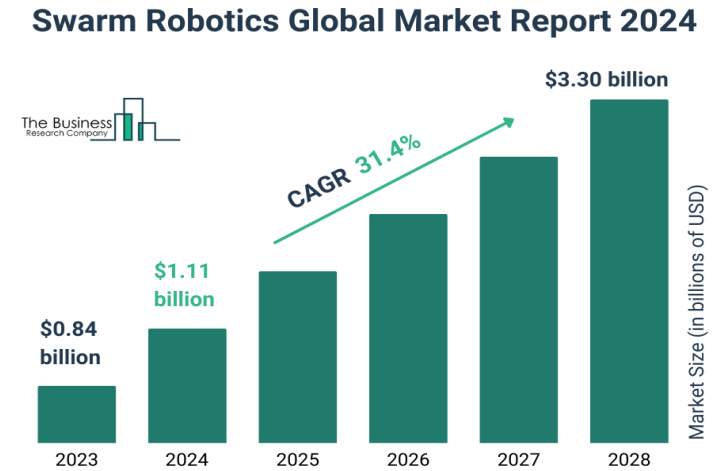
Appendix II

Growth & Problem

One server/controller, one single point of failure



The market is growing…



Turtlebot like Differential Drive

- The map is given.
- Localisation Process:
  - Differential drive kinematics
  - Position update using encoder data
  - Calculated forward movement and change in orientation which serves as the odometry

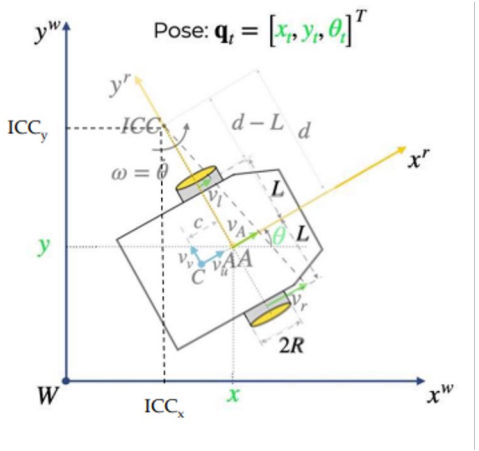
$$\Delta_\theta = \frac{\Delta_{\text{right}} - \Delta_{\text{left}}}{\text{wheel base}}$$

$$\Delta_{\text{center}} = \frac{\Delta_{\text{left}} + \Delta_{\text{right}}}{2}$$

$$x_{\text{new}} = x_{\text{old}} + \Delta_{\text{center}} \cdot \cos(\theta_{\text{old}})$$

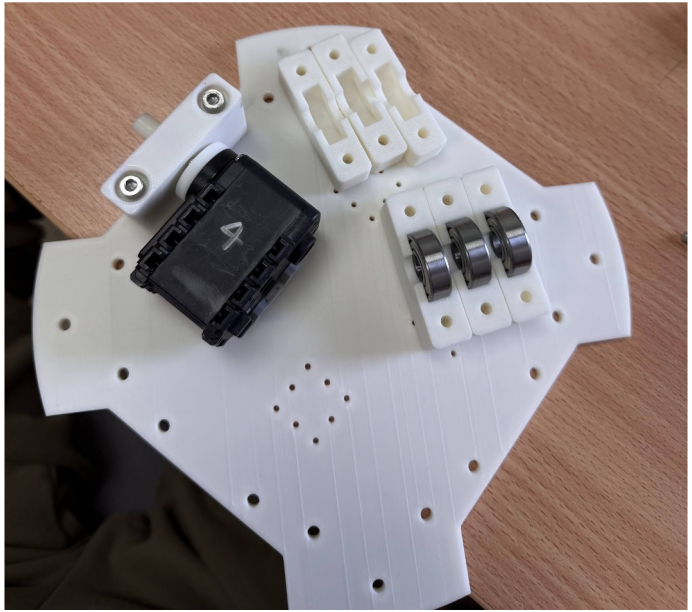
$$y_{\text{new}} = y_{\text{old}} + \Delta_{\text{center}} \cdot \sin(\theta_{\text{old}})$$

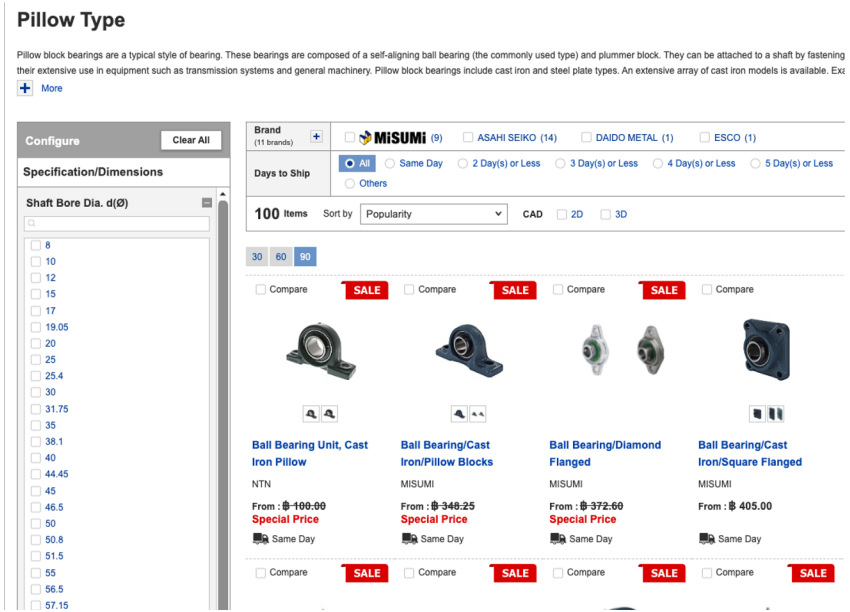
$$\theta_{\text{new}} = \theta_{\text{old}} + \Delta_\theta$$



Hardware – Appendix II

Bearing 626ZZ 06x19x06mm, Difficult to find a pillow block to accommodate.

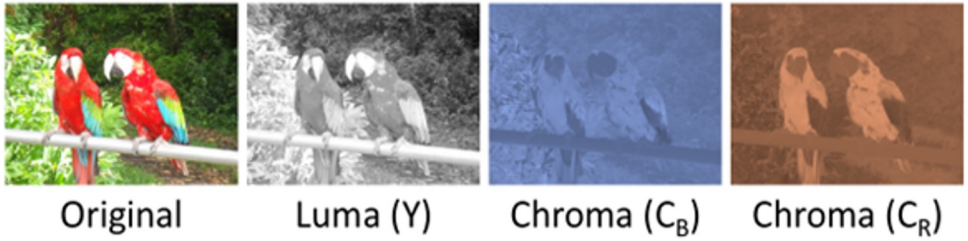




Object Detection APPENDIX - Color Detection

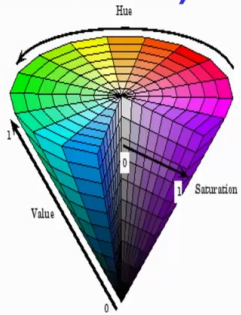
YCbCr Color Range

separates an image's brightness (luma) from its color information (chroma)



HSV (Hue, Saturation and Value)

- Hue corresponds to the color components (base pigment), hence just by selecting a range of Hue you can select any color. (0-360).
- Saturation is the amount of color (depth of the pigment)(dominance of Hue) (0-100%)
- Value is basically the brightness of the color. (0-100%).



Yellow: [100, 140, 50] to [255, 255, 100]

Advantages

- **Better separation of luminance and chrominance, making it less sensitive to lighting variations.**
- More stable under different lighting conditions compared to HSV.

Blue: [0, 100, 140] to [255, 255, 255]

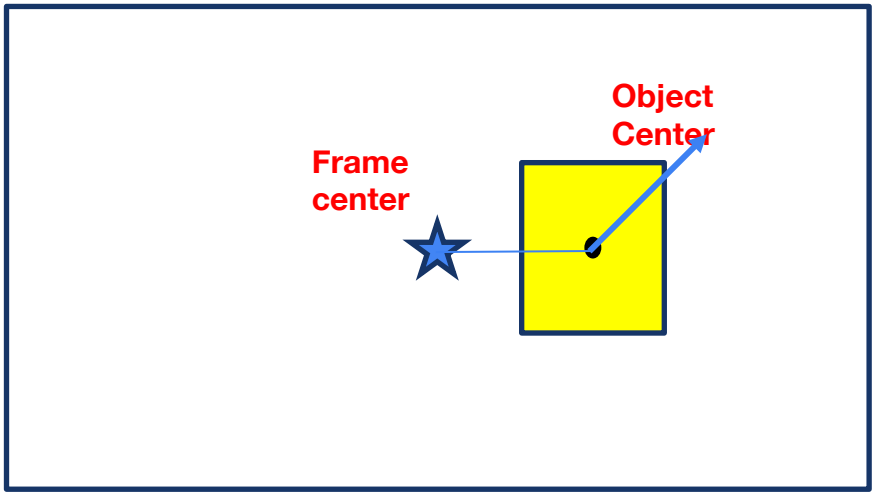
Limitation

- Potential loss of information when converting back to RGB.
- **Sensitivity to noise in the chrominance channels in low-light conditions.**

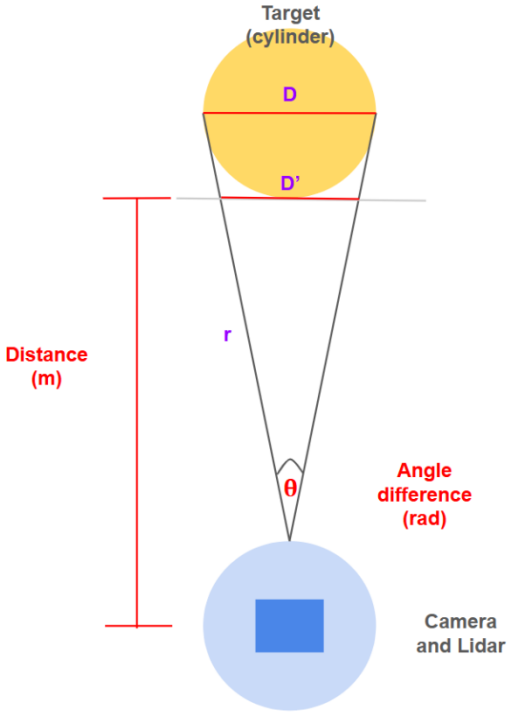
Object Detection APPENDIX - Width measurement

$$\text{relative\_angle} = \left(\frac{\text{object\_center\_x} - \text{frame\_center\_x}}{\text{frame\_width}} \right) \times \text{horizontal\_fov} \qquad (1.1)$$

72 degrees



Object Detection APPENDIX - Width measurement



The estimation begins by calculating the distance r from the LiDAR to the object's side using:

$$r = \frac{\text{distance}}{\cos\left(\frac{\theta}{2}\right)} \tag{1.2}$$

Here, θ is the angle subtended by the object in the camera frame. Then, the projected diameter D' is computed using the law of cosines:

$$D' = \sqrt{2r^2 - 2r^2\cos(\theta)} \tag{1.3}$$

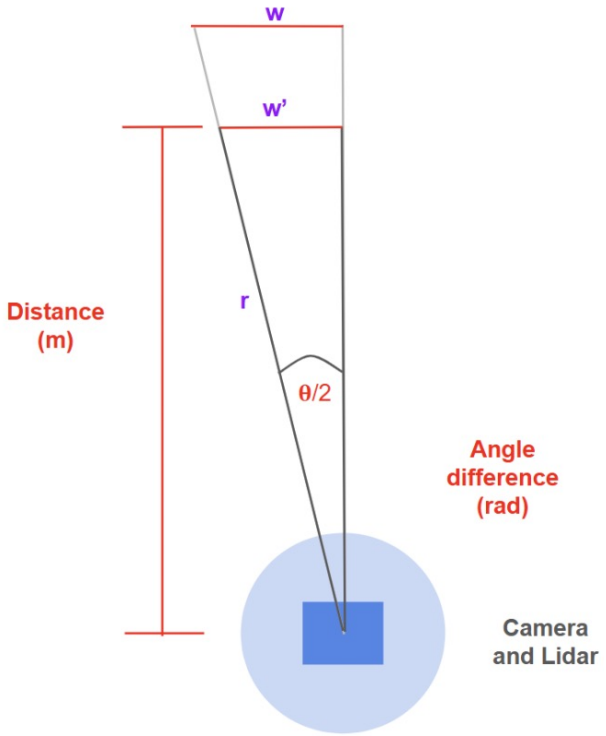
The projected radius is:

$$w' = \frac{D'}{2} \tag{1.4}$$

Using geometric similarity, the actual radius w is calculated as:

$$w = \frac{\text{distance} \cdot \tan\left(\frac{\theta}{2}\right)}{1 - \tan\left(\frac{\theta}{2}\right)} \tag{1.5}$$

Object Detection APPENDIX - Width measurement



The estimation begins by calculating the distance r from the LiDAR to the object's side using:

$$r = \frac{\text{distance}}{\cos\left(\frac{\theta}{2}\right)} \tag{1.2}$$

Here, θ is the angle subtended by the object in the camera frame. Then, the projected diameter D' is computed using the law of cosines:

$$D' = \sqrt{2r^2 - 2r^2\cos(\theta)} \tag{1.3}$$

The projected radius is:

$$w' = \frac{D'}{2} \tag{1.4}$$

Using geometric similarity, the actual radius w is calculated as:

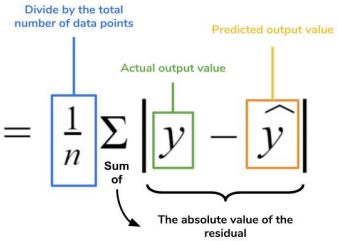
$$w = \frac{\text{distance} \cdot \tan\left(\frac{\theta}{2}\right)}{1 - \tan\left(\frac{\theta}{2}\right)} \tag{1.5}$$

Object Detection APPENDIX – Variation Factor Model Performance

Mean Error Squared

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- average of the squares of the errors
- useful for identifying when predictions are significantly off.
- **Interpretation:** The **closer the MSE is to 0, the more accurate the model.**



- Average of the absolute differences between predicted values and actual values.
- Useful for measuring the average magnitude of errors without considering their direction.
- **Interpretation:** The **closer the MAE is to 0, the more accurate the model.**

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

- Measures the proportion of variance in the dependent variable that is predictable from the independent variables.
- Useful for assessing the overall fit of the model, not individual prediction accuracy.
- **Interpretation:** Values **closer to 1 indicate better model performance**, while values near 0 suggest poor predictive power.

Object Detection APPENDIX – Variation Factor Model Performance

Model performance on the test set is evaluated using the following metrics:

Table 1.1: Model Accuracy Metrics

Metric	Value
Mean Squared Error (MSE)	2.802×10^{-5}
Mean Absolute Error (MAE)	0.004
R-squared (R^2)	0.916

Object Detection APPENDIX – Model Testing, 20 cm diameter

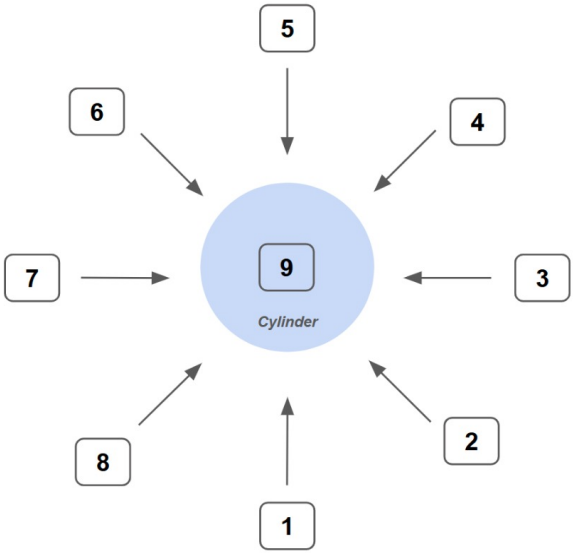


Table	Mean Absolute Width Error (m)	Mean Absolute Angle Error (°)
16 cm	0.0022	0.2389
20 cm	0.0017	0.2219
12 cm	0.0061	1.0519

Table A.1: Predicted diameter under different lighting conditions of 20 cm diameter cylinder, placed at actual relative angle of 0 degree.

Lighting Condition	Distance (m)	Angle (°)	Predicted Width (m)
1	0.5	0.21	0.20
2	0.5	0.21	0.20
3	0.5	0.21	0.20
4	0.5	0.21	0.20
5	0.5	0.28	0.20
6	0.5	0.28	0.20
7	0.5	0.13	0.20
8	0.5	0.21	0.20
9	0.5	0.21	0.21
1	1.00	0.37	0.20
2	1.01	0.43	0.20
3	1.00	0.43	0.20
4	1.00	0.37	0.20
5	1.00	0.43	0.20
6	1.00	0.31	0.20
7	1.00	0.29	0.20
8	1.00	0.33	0.20
9	1.00	0.33	0.20
1	2.00	0.18	0.20
2	1.99	0.18	0.20
3	2.00	0.18	0.20
4	2.00	0.18	0.20
5	2.00	0.18	0.20
6	1.99	0.18	0.20
7	2.00	0.18	0.20
8	2.00	0.18	0.20
9	2.00	0.18	0.20
1	3.00	0.35	0.20
2	3.00	0.29	0.20
3	3.00	0.29	0.20
4	2.99	0.29	0.20
5	3.00	0.29	0.20
6	3.00	0.29	0.20
7	3.01	0.35	0.19
8	3.00	0.29	0.20
9	3.00	0.29	0.20

Object Detection APPENDIX – Model Testing, 16 cm diameter

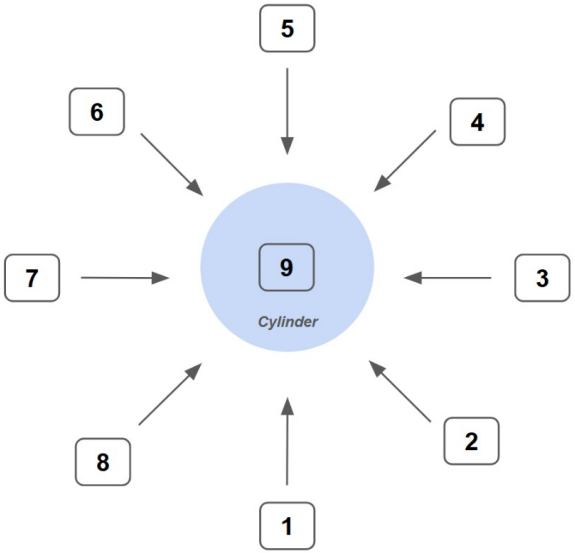


Table	Mean Absolute Width Error (m)	Mean Absolute Angle Error (°)
16 cm	0.0022	0.2389
20 cm	0.0017	0.2219
12 cm	0.0061	1.0519

Table A.2: Predicted diameter under different lighting conditions of 16 cm diameter cylinder, placed at actual relative angle of 0 degree.

Lighting Condition	Distance (m)	Angle (°)	Predicted Width (m)
1	0.5	358.11	0.16
2	0.5	358.11	0.16
3	0.5	358.11	0.16
4	0.5	358.05	0.16
5	0.5	358.11	0.16
6	0.5	358.11	0.16
7	0.5	358.11	0.16
8	0.5	358.11	0.16
9	0.5	358.11	0.16
1	1.00	0.18	0.16
2	1.00	0.18	0.16
3	1.00	0.18	0.16
4	1.00	0.18	0.16
5	1.00	0.12	0.16
6	1.00	0.18	0.16
7	1.00	0.12	0.16
8	1.00	0.18	0.16
9	1.00	0.18	0.16
1	2.00	0.55	0.16
2	2.00	0.61	0.16
3	2.00	0.61	0.16
4	2.00	0.61	0.16
5	2.00	0.55	0.16
6	2.00	0.55	0.16
7	2.00	0.55	0.16
8	2.00	0.55	0.16
9	2.00	0.55	0.16
1	3.00	0.22	0.16
2	3.01	0.28	0.16
3	3.00	0.28	0.16
4	3.00	0.22	0.17
5	3.00	0.28	0.16
6	3.00	0.28	0.16
7	3.00	0.28	0.17
8	3.00	0.28	0.16
9	3.00	0.28	0.16

Object Detection APPENDIX – Model Testing, 16 cm diameter

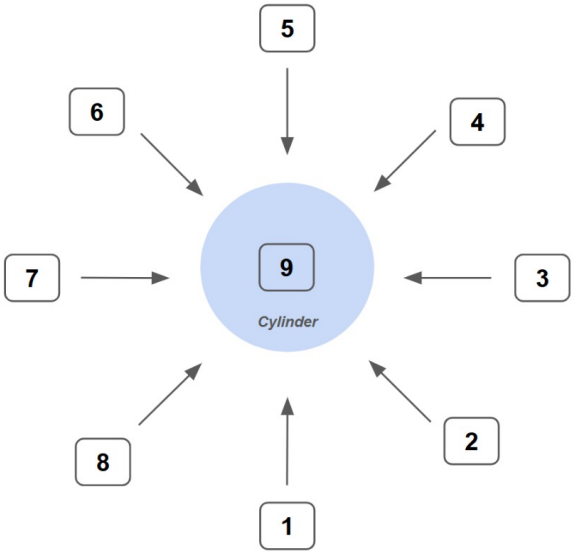
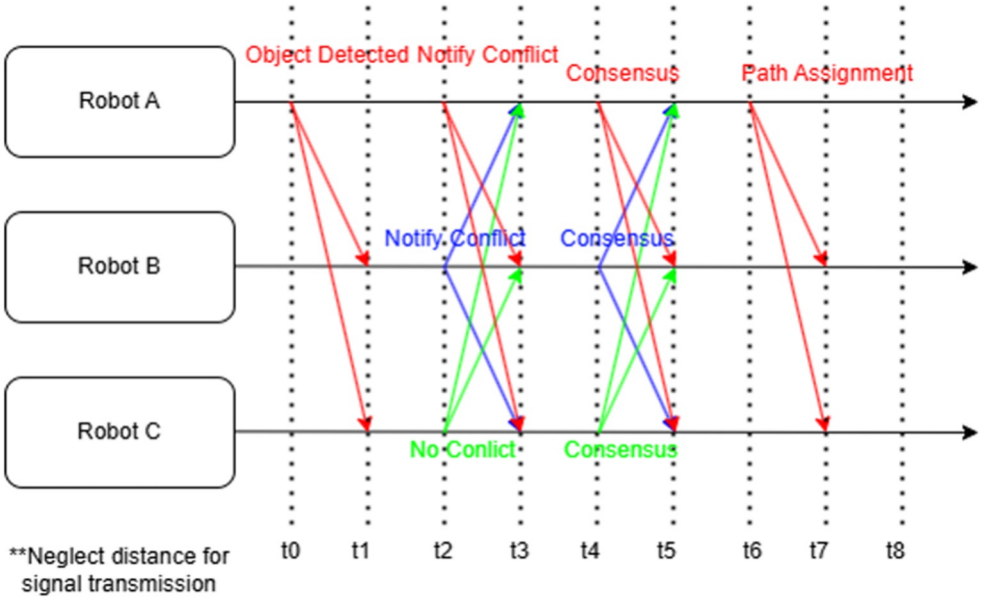


Table	Mean Absolute Width Error (m)	Mean Absolute Angle Error (°)
16 cm	0.0022	0.2389
20 cm	0.0017	0.2219
12 cm	0.0061	1.0519

Table A.3: Predicted diameter under different lighting conditions of 12 cm diameter cylinder, placed at actual relative angles of 2, 0, 1.5, and 0 degrees.

Lighting Condition	Distance (m)	Angle (°)	Predicted Width (m)
1	0.5	2.01	0.13
2	0.5	2.01	0.13
3	0.5	2.01	0.12
4	0.5	2.00	0.14
5	0.5	2.01	0.13
6	0.5	2.01	0.12
7	0.5	2.02	0.13
8	0.5	2.01	0.14
9	0.5	2.01	0.13
1	1.00	0.21	0.12
2	1.00	0.21	0.13
3	1.00	0.20	0.13
4	1.00	0.21	0.12
5	1.00	0.21	0.12
6	1.01	0.23	0.14
7	1.00	0.21	0.12
8	1.00	0.23	0.13
9	1.00	0.21	0.13
1	2.00	1.65	0.12
2	2.00	1.65	0.12
3	2.01	1.68	0.12
4	2.00	1.65	0.13
5	2.00	1.68	0.13
6	2.00	1.65	0.12
7	2.00	1.65	0.12
8	1.99	1.68	0.14
9	2.00	1.68	0.14
1	3.00	0.05	0.12
2	3.00	0.05	0.12
3	3.00	0.05	0.14
4	3.01	0.05	0.12
5	3.00	0.05	0.14
6	3.00	0.05	0.13
7	3.00	0.05	0.12
8	3.00	0.05	0.13
9	3.00	0.05	0.12

Communication



Consensus

Lidar Odom

$$\left(\cos\theta + \frac{R_\alpha k_\alpha \sin\theta}{r}\right) v_{x,s} + \left(\sin\theta - \frac{R_\alpha k_\alpha \cos\theta}{r}\right) v_{y,s} + (x\sin\theta - y\cos\theta - R_\alpha k_\alpha)\omega_s + R_t = 0$$

Odometry Overhead Camera APPENDIX - Equation

ArUco Marker Localization

Given the four detected corner points of an ArUco marker in image coordinates:

$$\text{Corners} = \{\mathrm{p}_1, \mathrm{p}_2, \mathrm{p}_3, \mathrm{p}_4\} = \{\text{topLeft}, \text{topRight}, \text{bottomRight}, \text{bottomLeft}\}$$

1. Center of the Marker

$$c_x = \frac{x_{\text{topLeft}} + x_{\text{bottomRight}}}{2}, \quad c_y = \frac{y_{\text{topLeft}} + y_{\text{bottomRight}}}{2}$$

2. Direction Vector of the Top Edge

$$\Delta x = x_{\text{topRight}} - x_{\text{topLeft}}, \quad \Delta y = y_{\text{topRight}} - y_{\text{topLeft}}$$

3. Orientation Estimation Display orientation in degrees:

$$\theta_{\text{display}} = \arctan 2(\Delta y, -\Delta x)$$

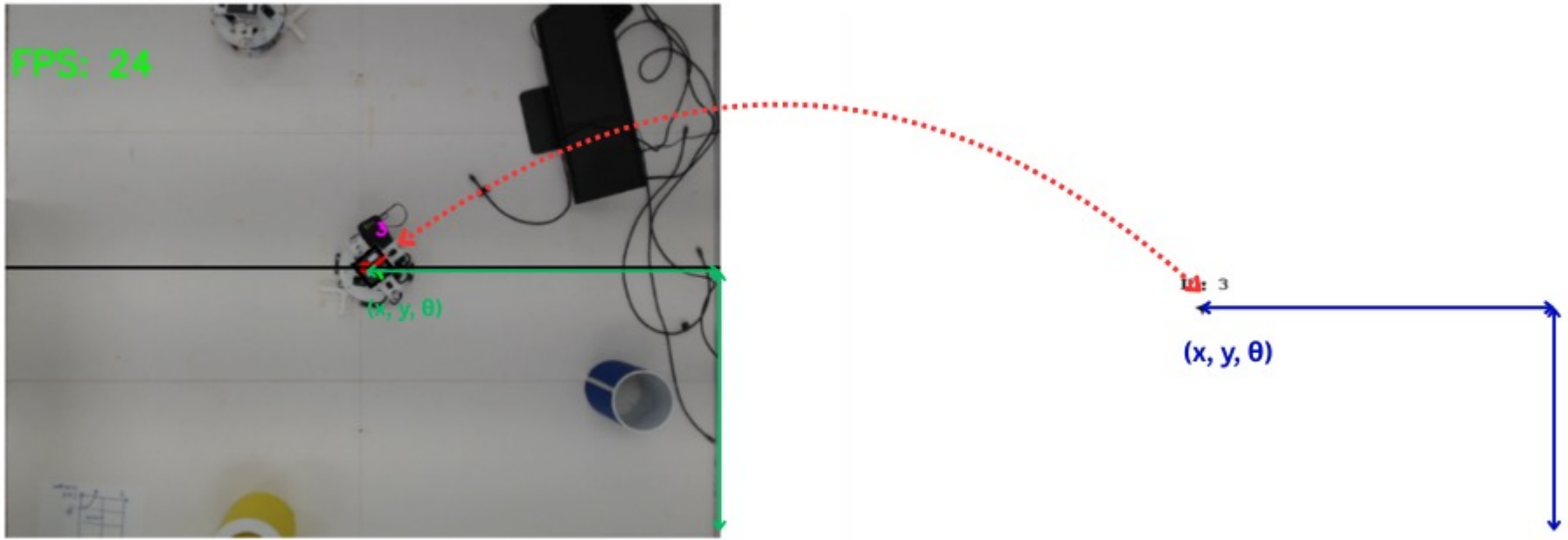
4. Normalized Image Coordinates Assuming image width W and height H:

$$x_{\text{norm}} = \frac{c_x}{W}, \quad y_{\text{norm}} = \frac{c_y}{H}$$

5. Real-World Position Mapping Given real-world map dimensions S_x and S_y:

$$x_{\text{map}} = (x_{\text{norm}} - 0.5) \cdot S_x, \quad y_{\text{map}} = -(y_{\text{norm}} - 0.5) \cdot S_y$$

Odometry Overhead Camera APPENDIX - Map



Appendix I

Robot States Table

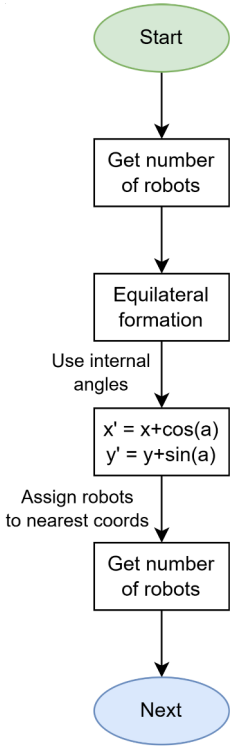
idle (Default)	State where the robot halt and wait for instructions
consensus	State where an agreement of the task master is decided.
path_finding	State where the task master generates navigation paths for all robots.
path_following	State where the robot follows it's path.
task	States that handles task management, particularly conflicts.
reassign	States that reassigns the task master when conflicts arise.

Robot Message Formats

probe	Broadcasts real-time position data for efficient task coordination. **Sender**: Sends robot's position data. **Receiver**: Updates information the robot_entries dictionary. **Format**: {[probe], robot_id, message_id, (robot_position["x"], robot_position["y"], robot_position["theta"])}
object_detected	Broadcasts the encounter of the target object. **Sender**: Sends object's position data. **Receiver**: Changes it's state to *idle* and waits for further action. **Format**: {[object_detected], robot_id, found cylinder @ {cylinder_position}}
task	Broadcasts the assigning of a new taskmaster for the path planning. **Sender**: Sends an initiative to assign itself as the taskmaster. **Receiver**: Changes it's state to *task* and report conflicting status. **Format**: {[task], robot_id, message_id, cylinder_position}
task_conflict	Broadcasts the conflict occurs in task master assignment. **Sender**: Sends conflict information to other robots. **Receiver**: Change it's state to *reassign*. **Format**: {[task_conflict], robot_id, message_id, priority_queue}
task_successful	Broadcasts a success operation in task master assignment. **Sender**: Confirms the success of the operation and the task master. **Receiver**: Ensures all data and information complies to itself. **Format**: {[task_successful], robot_id, message_id, task_master}
path_following	Broadcasts a notification for robots to follow the given paths. **Sender**: Sends planned path to each robots. **Receiver**: Changes it's state to *path_following*, and updates it's path. **Format**: {[path_following], robot_id, message_id, paths_json}

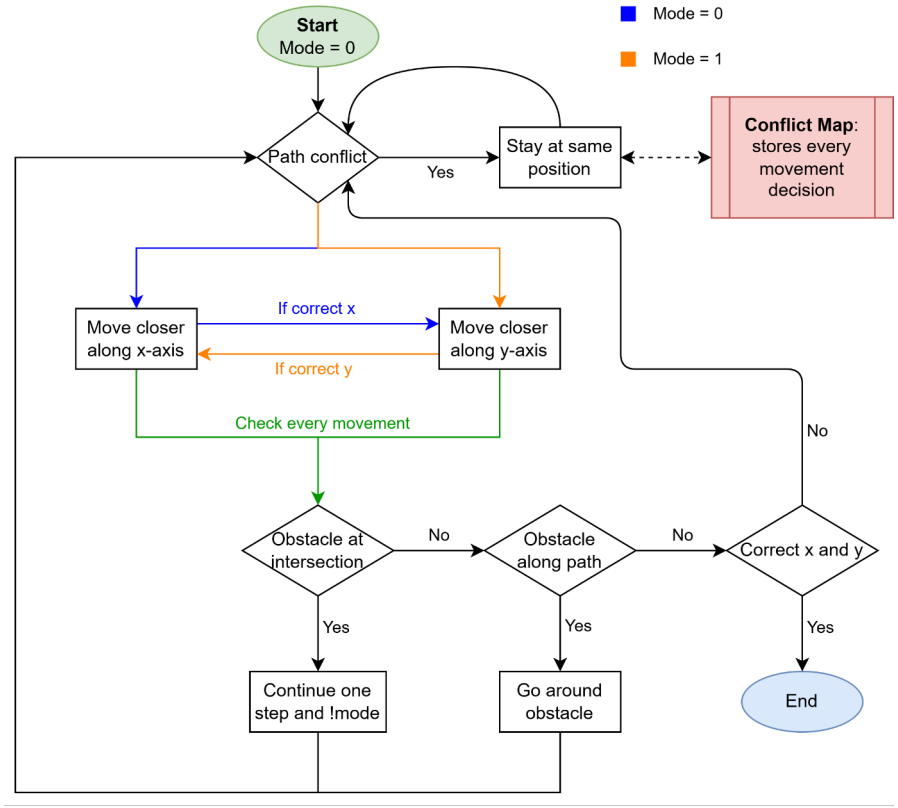
Path Planning - Formation

Formation as an equilateral shape according to number of robots

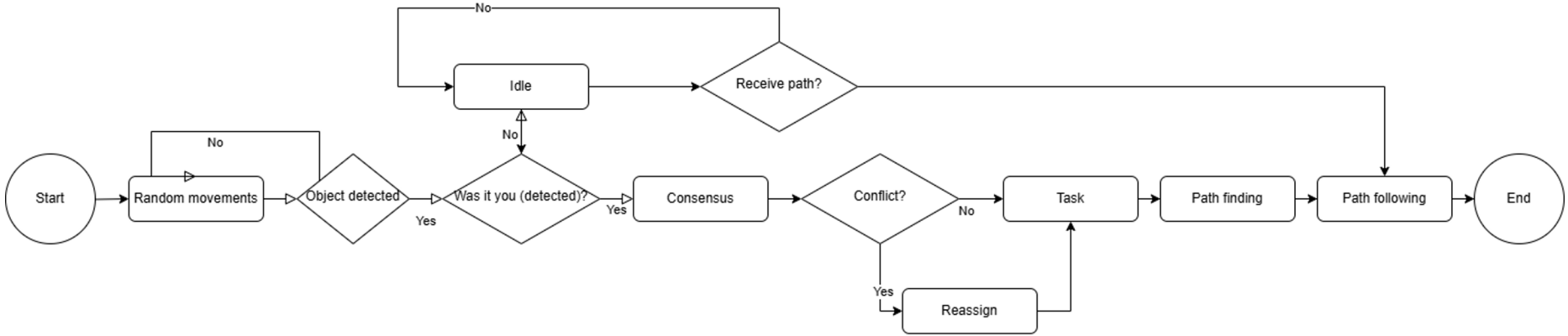


Path Planning - Movement

Flow of movement considering obstacles

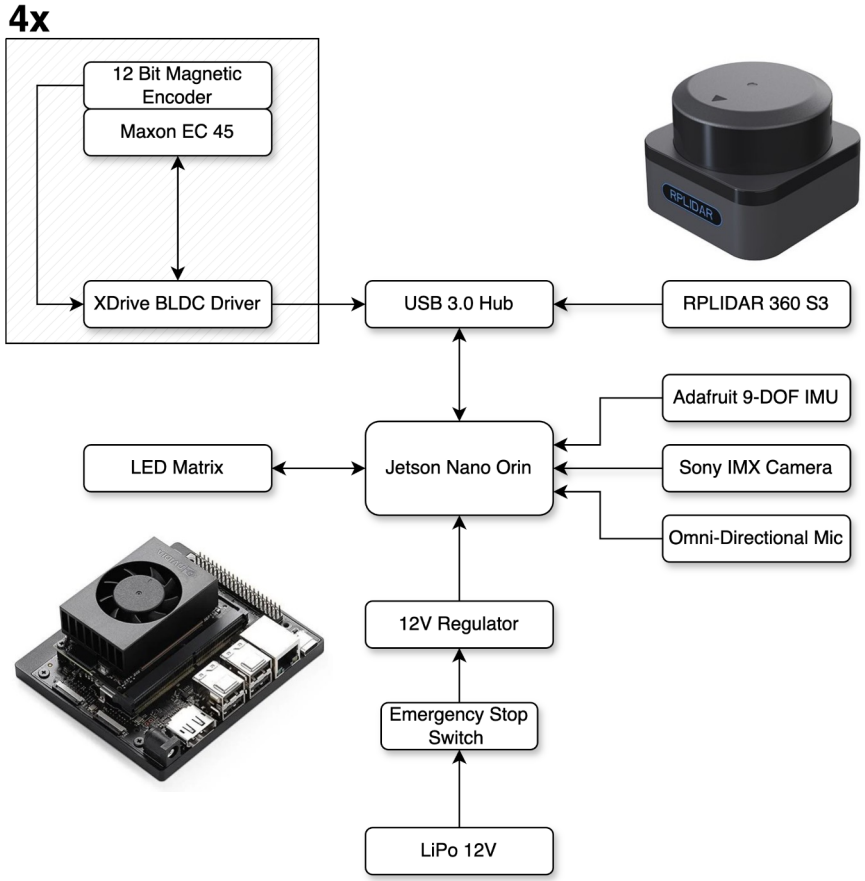


Robot Communication Flow Chart

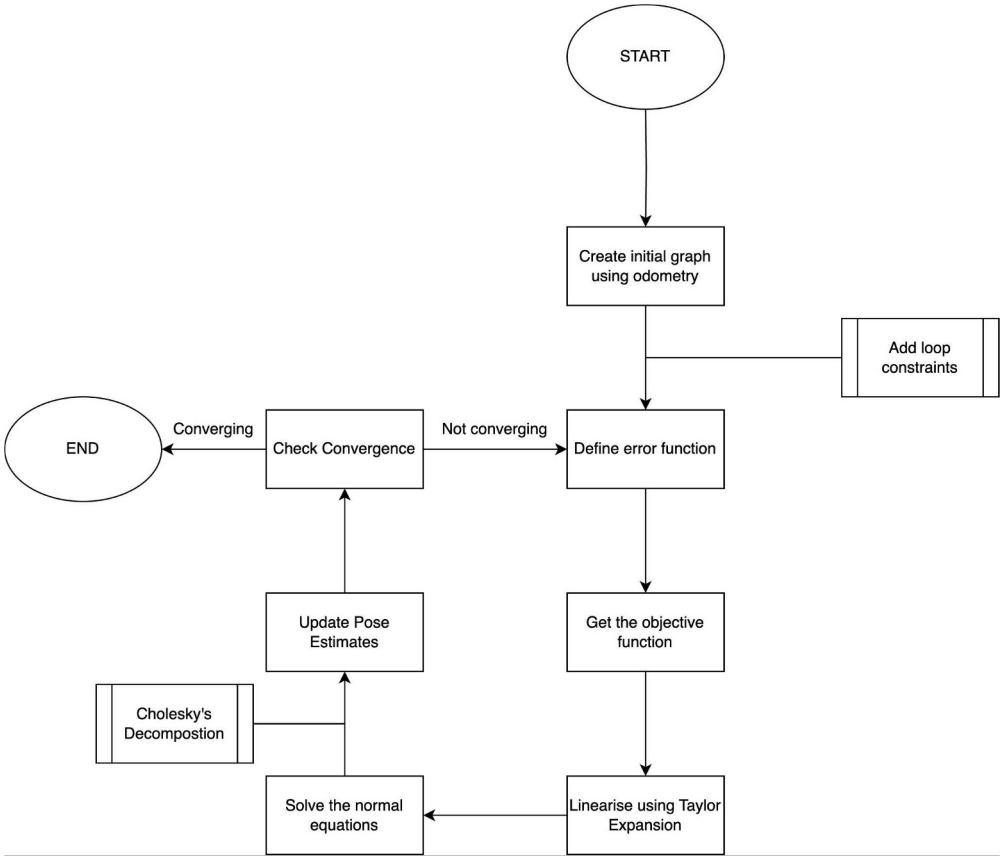


Hardware Future Plans APPENDIX

- Focus on the software
- Use of it is under the budget 250K
- Jetson:
  - Uses M.2 instead of Micro SD
  - Experienced using it, reducing dev time
  - We are not mass producing



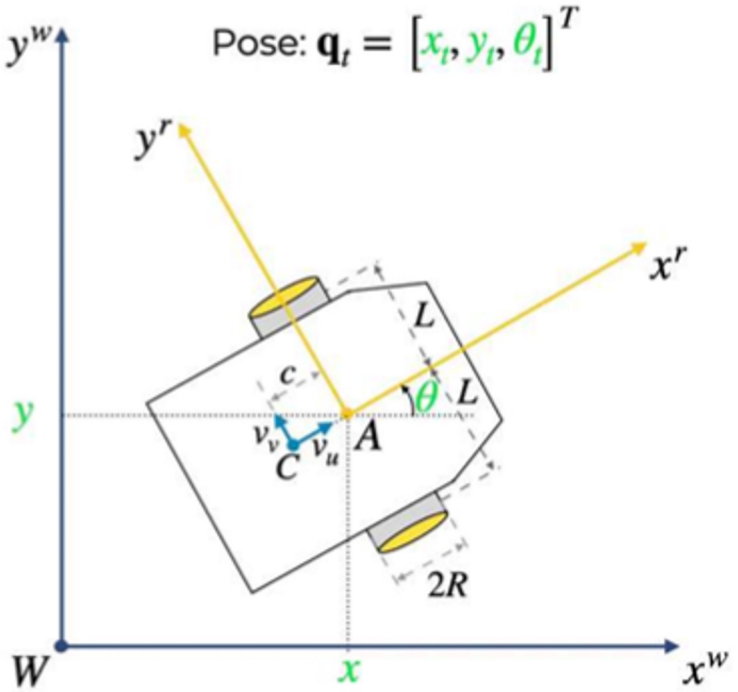
Graph SLAM APPENDIX



Kinematics APPENDIX - Wheeled Locomotion Equation

$$\dot{x}_t = v_{u,t}\cos\theta_t - \left(v_{v,t} - c\dot{\theta}_t\right)\sin\theta_t$$
$$\dot{y}_t = v_{u,t}\sin\theta_t + \left(v_{v,t} - c\dot{\theta}_t\right)\cos\theta_t$$
$$\dot{\theta}_t = \omega_t$$

$$\dot{q}_t = \begin{bmatrix}\dot{x}_t \\ \dot{y}_t \\ \dot{\theta}_t\end{bmatrix} = \begin{bmatrix}cos\theta_t & 0 \\ sin\theta_t & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}v_{A,t} \\ \omega_t\end{bmatrix}$$



Ref: Mobile Robot 2147326, Asst. Prof. Surat Kwanmuang

Kinematics APPENDIX - Wheeled Locomotion Forward Kinematics

$$ICC = [x - d \cdot sin\theta, y + d \cdot cos\theta]$$

$$d = L\frac{v_r + v_l}{v_r - v_l} \qquad \omega = \frac{v_r - v_l}{2L}$$

$$v_l = \omega(d - L)$$
$$v_A = \omega d$$
$$v_r = \omega(d + L)$$

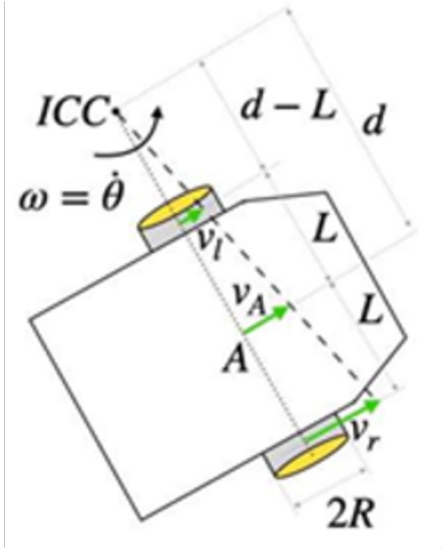
$$\begin{bmatrix} v_A \\ \omega \end{bmatrix} = \frac{R}{2}\begin{bmatrix} 1 & 1 \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix}\begin{bmatrix} \dot{\phi}_r \\ \dot{\phi}_l \end{bmatrix}$$

$$\dot{q}_t = \begin{bmatrix} \dot{x}_t \\ \dot{y}_t \\ \dot{\theta}_t \end{bmatrix} = \frac{R}{2}\begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix}\begin{bmatrix} \dot{\phi}_r \\ \dot{\phi}_l \end{bmatrix}$$

3 outputs

2 inputs

Non-holonomic $\longleftarrow \dot{y}\cos(\theta) - \dot{x}\sin(\theta) = 0$



Ref: Mobile Robot 2147326, Asst. Prof. Surat Kwanmuang

Kinematics APPENDIX - Wheeled Locomotion Robot Position

$$\dot{q}_t = \begin{bmatrix} \dot{x}_t \\ \dot{y}_t \\ \dot{\theta}_t \end{bmatrix} = \frac{R}{2} \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix} \begin{bmatrix} \dot{\phi}_r \\ \dot{\phi}_l \end{bmatrix}$$

$$\longrightarrow$$

$$x_t = \frac{R}{2} \int_0^t (\dot{\phi}_r + \dot{\phi}_l) cos\theta dt$$
$$y_t = \frac{R}{2} \int_0^t (\dot{\phi}_r + \dot{\phi}_l) sin\theta dt$$
$$\theta_t = \frac{R}{2L} \int_0^t (\dot{\phi}_r - \dot{\phi}_l) dt$$

Position of the ICC point

$$ICC = [x - d \cdot sin\theta, y + d \cdot cos\theta]$$

At time $t + \delta t$, the robot's pose will be

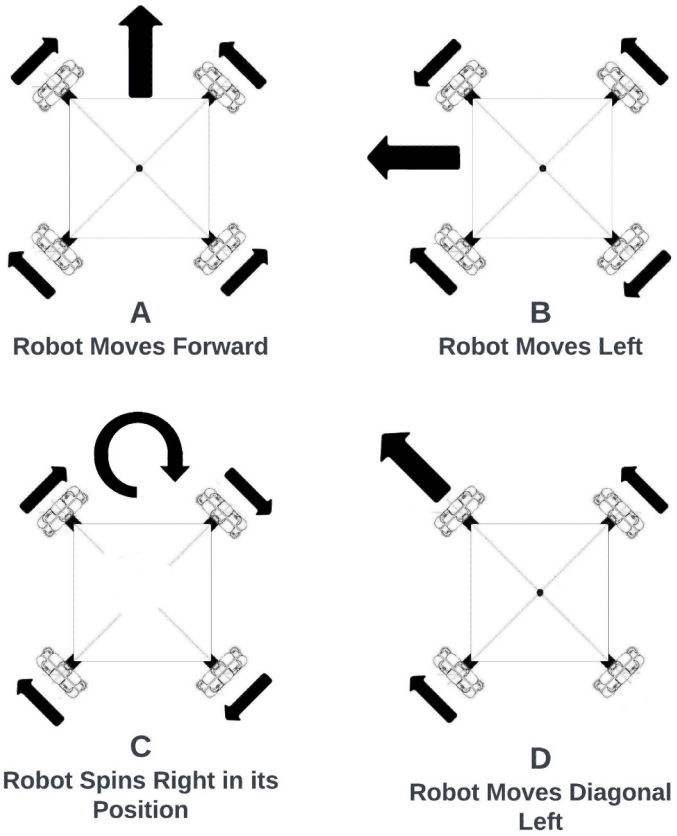
$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{t+\delta t} = \begin{bmatrix} \cos(\omega\delta t) & -sin(\omega\delta t) & 0 \\ sin(\omega\delta t) & cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t - ICC_x \\ y_t - ICC_y \\ \theta_t \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\delta t \end{bmatrix}$$

Special case, $v_r = v_l = v_A$, $\omega = 0$, $d = \infty$

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{t+\delta t} = \begin{bmatrix} x_t + v_A \cos(\theta_t)\delta t \\ y_t + v_A \sin(\theta_t)\delta t \\ \theta_t \end{bmatrix}$$

Ref: Mobile Robot 2147326, Asst. Prof. Surat Kwanmuang

Kinematics APPENDIX - Omnidirectional Wheeled Robot



Ref: https://www.theroboticsspace.com/blog/Kinematic-Modelling-for-Mobile-Robots/

Kinematics APPENDIX - Differential Drive vs Omnidirectional

Property	Differential Drive	Omnidirectional
Non-Holonomic	Yes (velocity constraint)	No (fully holonomic)
Lateral Movement	Not possible (forward-only)	Possible
Integration	Respects constraints	Direct integration

Graph SLAM APPENDIX - Create Initial Graph using odometry

$$x_t = x_i + \Delta s \cos(\theta_i),$$

$$y_t = y_i + \Delta s \sin(\theta_i),$$

$$\theta_t = \theta_i + \Delta\theta$$

where Δs is the distance traveled.

The edge e_{ij} is represented as:

$$e_{ij} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}.$$

Graph SLAM APPENDIX - Loop Closure

2. **Loop closure**

   When a robot recognizes a previous landmark, an additional edge is added between non-consecutive poses x_i and x_j where $j \neq i + 1$. This is a sensor based measurement, not odometry.

$$e_{ij} = \begin{bmatrix} \Delta x_{ij} \\ \Delta y_{ij} \\ \Delta \theta_{ij} \end{bmatrix}$$

Graph SLAM APPENDIX - Defining Error Function

$$e_{ij} = \begin{bmatrix} x_j - (x_i + \Delta x_{ij}) \\ y_j - (y_i + \Delta y_{ij}) \\ \theta_j - (\theta_i + \Delta \theta_{ij}) \end{bmatrix}.$$

Here, x_j, y_j, θ_j are the current noisy estimates, and $x_i + \Delta x_{ij}, y_i + \Delta y_{ij}, \theta_i + \Delta \theta_{ij}$ are the expected values for pose j based on pose i (from $[i]$) and the expected transformation $\Delta x_{ij}, \Delta y_{ij}, \Delta \theta_{ij}$.

Graph SLAM APPENDIX - Gauss-Newton

(a) Linearize using Taylor expansion:

$$e_{ij}(x_i, x_j) \approx e_{ij}(x_{i0}, x_{j0}) + J_{ij}(x_i - x_{i0}),$$

where $J_{ij} = \left.\frac{\partial e_{ij}}{\partial x}\right|_{x=x_0}$.

(b) Substitute the linearized term into the objective function:

$$f(x) = \sum_{(i,j)\in \text{edges}} \left[e_{ij}(x_{i0}, x_{j0}) + J_{ij}(x_i - x_{i0})\right]^T \Omega_{ij} \left[e_{ij}(x_{i0}, x_{j0}) + J_{ij}(x_i - x_{i0})\right].$$

Graph SLAM APPENDIX - Gauss-Newton Normal Equations

$$H\Delta x = -g,$$

where

$$H = \sum_{(i,j)\in \text{edges}} J_{ij}^T \Omega_{ij} J_{ij} \quad \text{(Hessian approximation)},$$

$$g = \sum_{(i,j)\in \text{edges}} J_{ij}^T \Omega_{ij} e_{ij}(x_{i0}, x_{j0}) \quad \text{(gradient vector)}.$$

Graph SLAM APPENDIX - Gauss-Newton Normal Equations

(a) Decompose H using Cholesky decomposition:

$$H = LL^T,$$

where L is the lower triangular matrix.

(b) Solve for y using:

$$Ly = -g.$$

(c) Solve for Δx using:

$$L^T \Delta x = y.$$

After obtaining Δx, update:

$$x \leftarrow x + \Delta x.$$

Check for convergence by ensuring $f(x)$ is below a predefined threshold. If so the algorithm has converged and it can stop. Else, once again compute the residuals and the Jacobian.