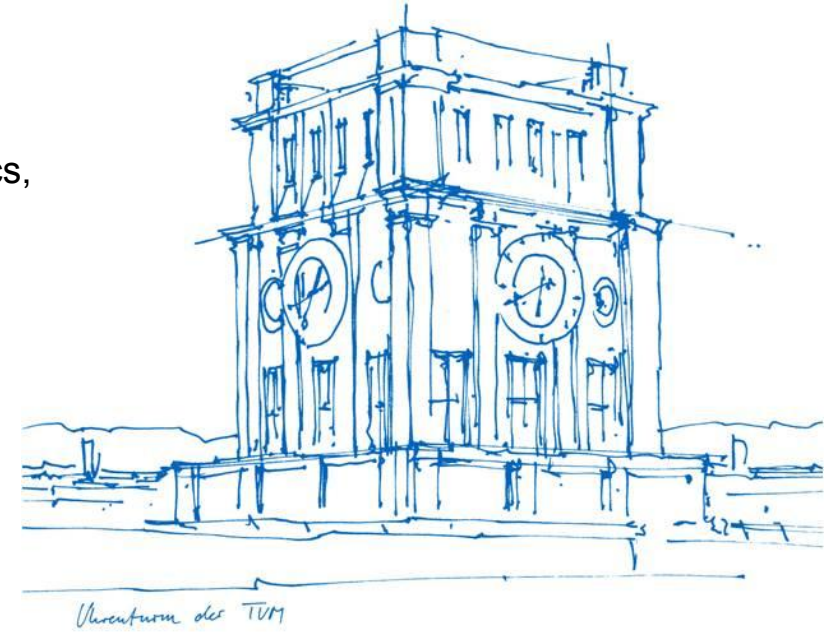
Embedded Systems, Cyber-Physical Systems and Robotics
School of Computer, Information and Technology
Technical University of Munich

# Autonomous Vision-Based Object Tracking Robot

Embedded Systems, Cyber-Physical Systems and Robotics,
CIT, Technical University of Munich

September 2nd, 2025

Rajkumar Pambhar, Sumeet Mourya, Dharmang Pambhar, Aayush Gupta, Romit Kheni, Saumil Savani, Sahil Virani



# Overview

- Project Overview & Motivation
- Team Roles & Responsibilities
- Challenges
- System Architecture
- Hardware Architecture
- Software Design
- Integration of Software & Hardware
- Live Demonstration and Results

# Project Overview & Motivation

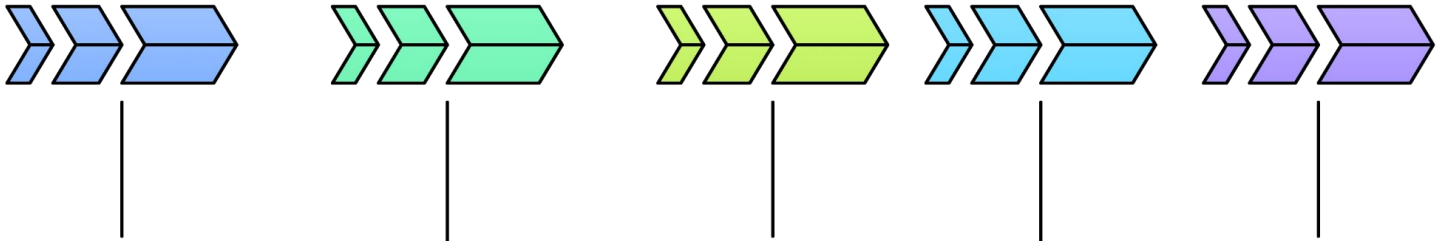
- **Objective**

  Develop a resource-efficient, vision-based robot capable of perceiving and tracking dynamic objects autonomously in real-world scenarios .

- **Motivation**

  Object following is a foundational skill for autonomous systems in navigation, service robotics, surveillance, and human robot interaction .

# Team Roles & Responsibilities



**Assembly and Running**

Assembling and testing the Duckiebot

**Data Collection and Labelling for Model Training**

Collecting and Labelling the Data for Training the Object Detection Model

**Main Implementation**

Executing the core project tasks

**Testing**

Conducting tests to ensure functionality

**Documentation**

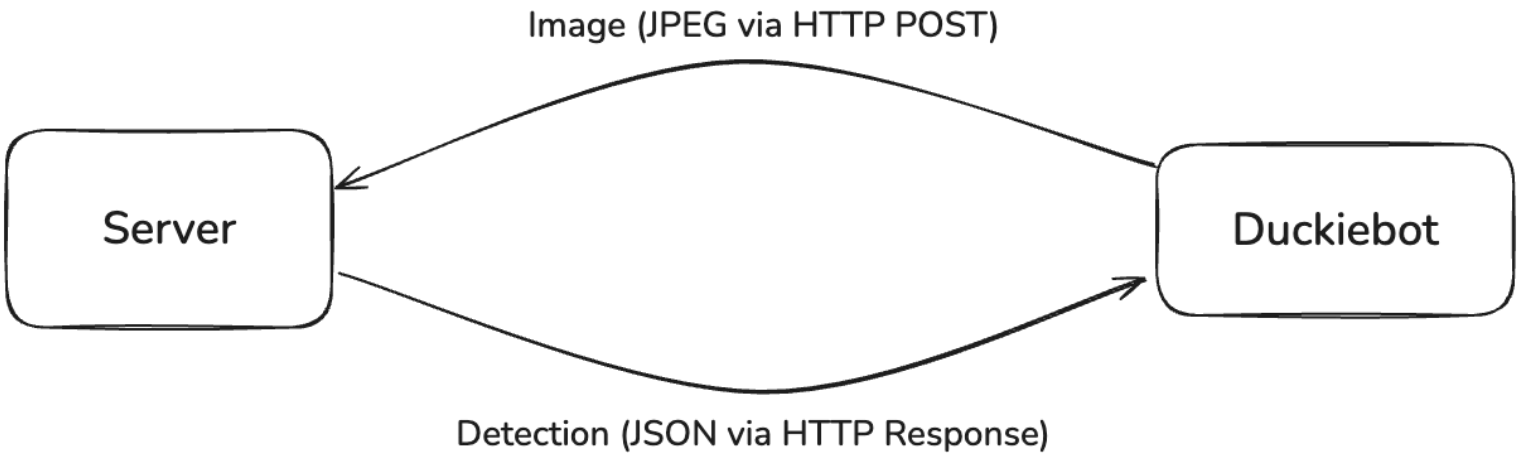
Preparing and finalizing project documentation

# Challenges

1. **Real-Time Processing Constraints:** Ensuring fast image capture, transmission, and inference without exceeding robot CPU resources
2. **Robust Object Detection:** Handling diverse operating conditions, variable lighting, and motion blur for consistent detection.
3. **Integrating hardware with software** and coordinating ROS nodes, and motor control without delays or failures.
4. **Tracking system health** in real time while ensuring monitoring does not interfere with main control tasks.

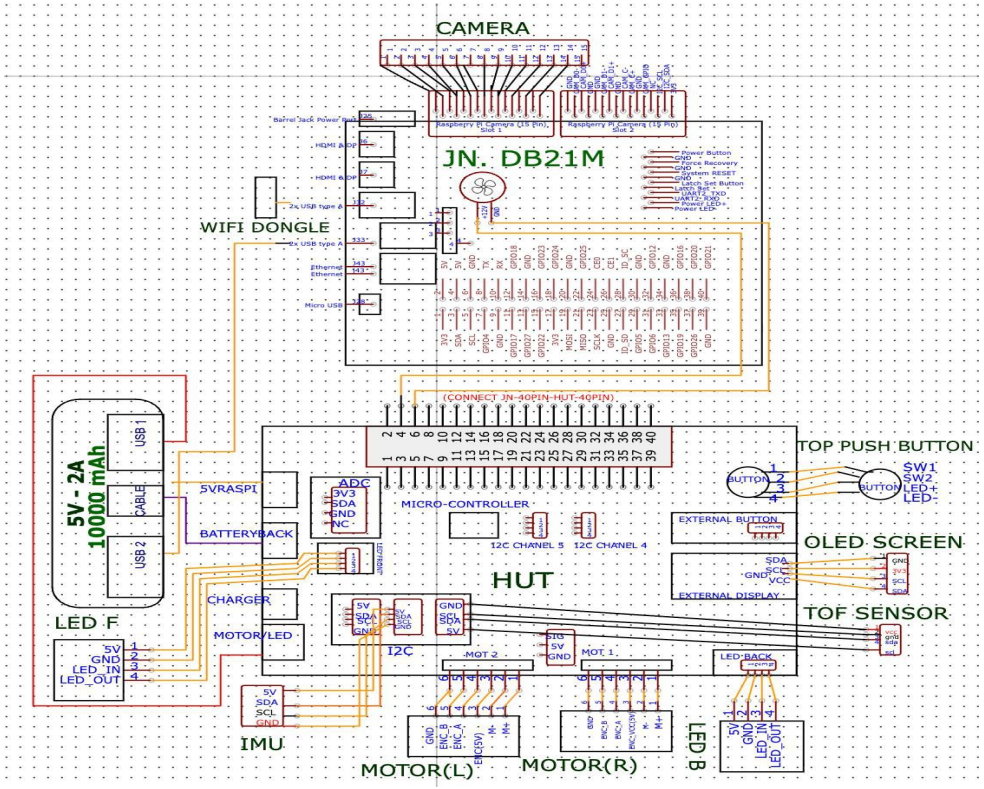
# System Architecture & Workflow

- Duckiebot streams camera images to the remote server for deep learning inference
- **YOLO v8** model on server returns detection results via JSON
- Detection results trigger control actions onboard Duckiebot in real time
- Client-server separation allows **high-speed detection without overloading** robot CPU



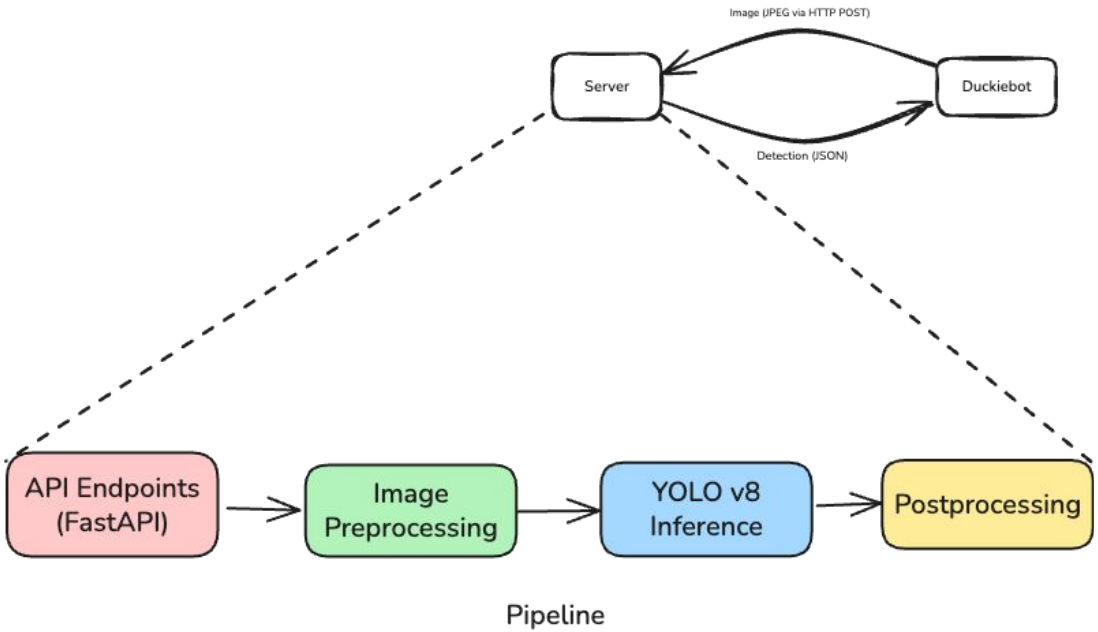
# Hardware Architecture

1. **Camera:** Captures images for object detection (connected to CPU).
2. **Jetson Nano:** Runs ROS, communicates with server, processes all sensor/motor I/O.
3. **WiFi Connectivity:** Enables communication with external YOLO v8 server.
4. **Battery & Power Regulation:** Supplies power to all components.
5. **Motors:** Enable differential drive for movement and turning.



# Software Design and Implementation

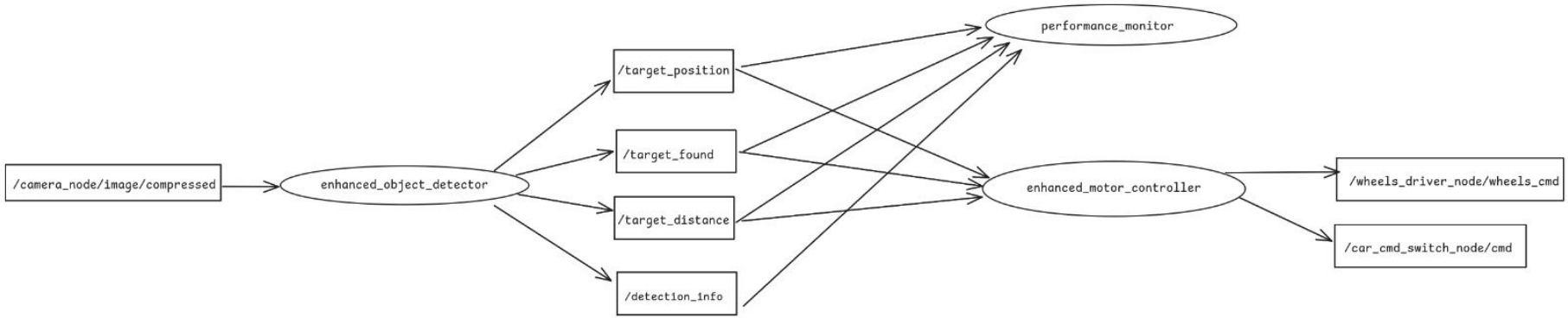
1. **API Endpoints:** Used FastAPI for developing API endpoints for POST request and response.
2. **Image Preprocessing:** Decodes and resizes the incoming image to be suitable for AI inference.
3. **YOLOv8 Inference:** Runs neural network model for object detection (tennis ball).
4. **Postprocessing:** Converts raw detection output to JSON format for the robot client.




Pipeline

# Integration of Hardware & Software

1. **Distributed Processing Pipeline:** Object Detection → Control → Actuation: Clear separation of concerns with enhanced_object_detector processing camera input and enhanced_motor_controller handling motion control. Real-time data flow through /target_position, /target_found, and /target_distance topics
2. **Multi-Level Motor Control:** Publishing to both low-level wheel commands (/wheels_driver_node/wheels_cmd) and high-level DuckieBot coordination (/car_cmd_switch_node/cmd)
3. **Performance Monitoring Integration:** performance_monitor node subscribes to all key topics for system health tracking. Monitoring doesn't interfere with main control loop, just observes the data flow



# Live Demonstration & Results

The final system achieves:

1. **Detection rates** of 15-20 Hz
2. **Position errors** of ±0.05-0.10 normalized units
3. **Response time** under 200ms

demonstrating reliable real-time object following capabilities with robust performance across diverse operating conditions.

**Github:** https://github.com/Sumeet-2023/duckie_v2