DuckieTown: GlitchieDuck

Bachelor Practical Course (INHN0021, INHN4052) – Duckietown

Esmir Kićo, Supratik Patel, Sahil Virani

Agenda

Overview of key project milestones and collaborative tasks

Assembly of the Duckiebot

- The physical construction of the Duckiebot
- Preparing required components

Installing the DTS Shell software

- Configuring Ubuntu Virtual Machine
- Setting up Duckietown Shell environment
- Enabling control of the Duckiebot

Calibration

- Precise calibration of the Duckiebot's camera and wheels
- Ensuring accurate perception and movement during operation

Implement DT-Core functionalities

- Integrating the core Duckietown software modules that
- Control navigation and sensor management for the robot.

Creation of the VLM algorithm

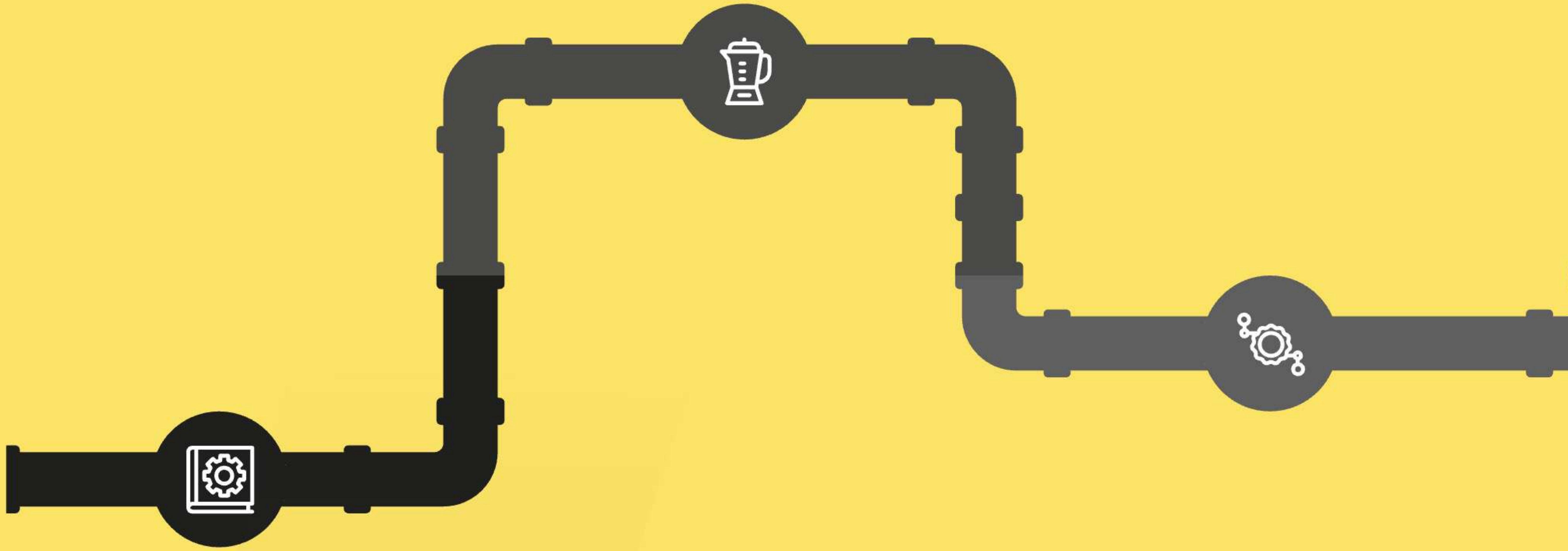
- Creating sufficient Visual Language Model
- Enhance autonomous decision-making in complex environments.

Integration of the VLM algorithm

- Integrating VLM with the current Duckietown software modules

Assembly of Duckiebot

Step-by-step hardware setup and challenges in building the Duckiebot



Duckietown manual guide

Used the official Duckietown manual for detailed hardware setup instructions to ensure accuracy.

Connections and power

Checked all electrical connections and power supply before installing software to prevent issues.

SD Card Flashing

Preparing the installation of Ubuntu 18.04 image to be used as Duckiebot's operating system

Installation of DTS Shell

Step-by-step setup to enable Duckietown software interaction on Duckiebot

Virtual Machine Setup

Installation and configuration of the Ubuntu 22.04 Virtual Machine

DTS Shell Installation

DTS Shell enabling seamless communication with the Duckietown software components

Preparing environment

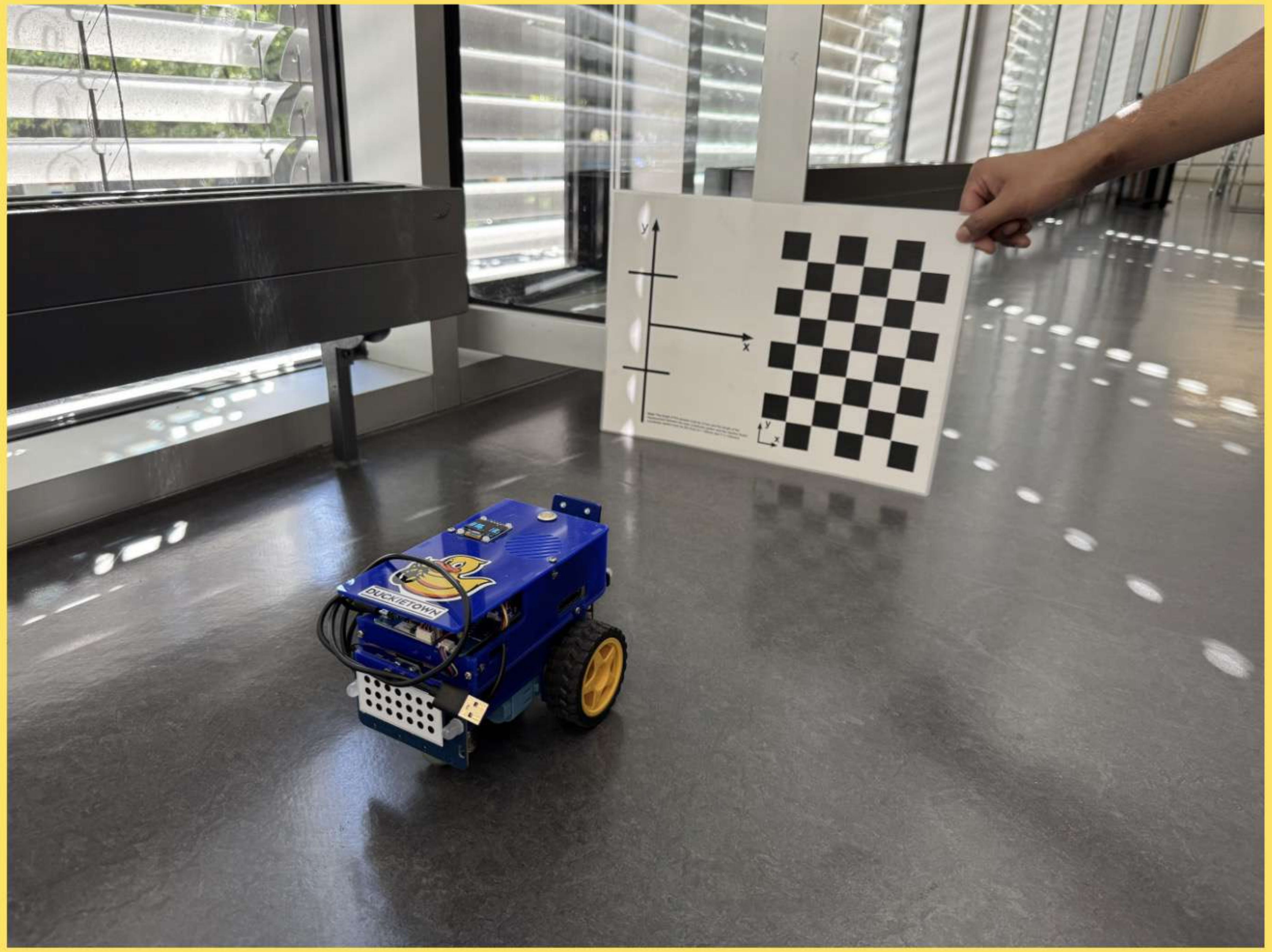
Installing necessary dependencies, and configuring the system

Running basic functionalities

Utilising keyboard control, GUI tools and duckiebot web interface

Calibration of Camera and Wheels

Ensuring accurate sensor data and precise robot navigation through calibration



DT-Core Implementation Overview

Adapting and integrating DT-Core modules for lane following

DT-Core software stack

Provides key modules for vision, localization, control, and communication in Duckiebots

Customizing DT-Core modules

Adapted core modules specifically for our Duckiebot setup to enhance compatibility and performance.

Integrated ROS nodes

Implemented ROS nodes to handle sensor data processing and execute control commands effectively

DT-Core Control Module

Key components and dynamic control processes for autonomous driving

01

Anti-instagram

Receives real-time video feed and performs color correction to isolate colours

02

Lane Following

Processes perception data to generate precise movement commands for the vehicle.

03

Stop-line FIlter

Recognizes red line on the lane and stops for 3 seconds before proceeding

04

Apriltags

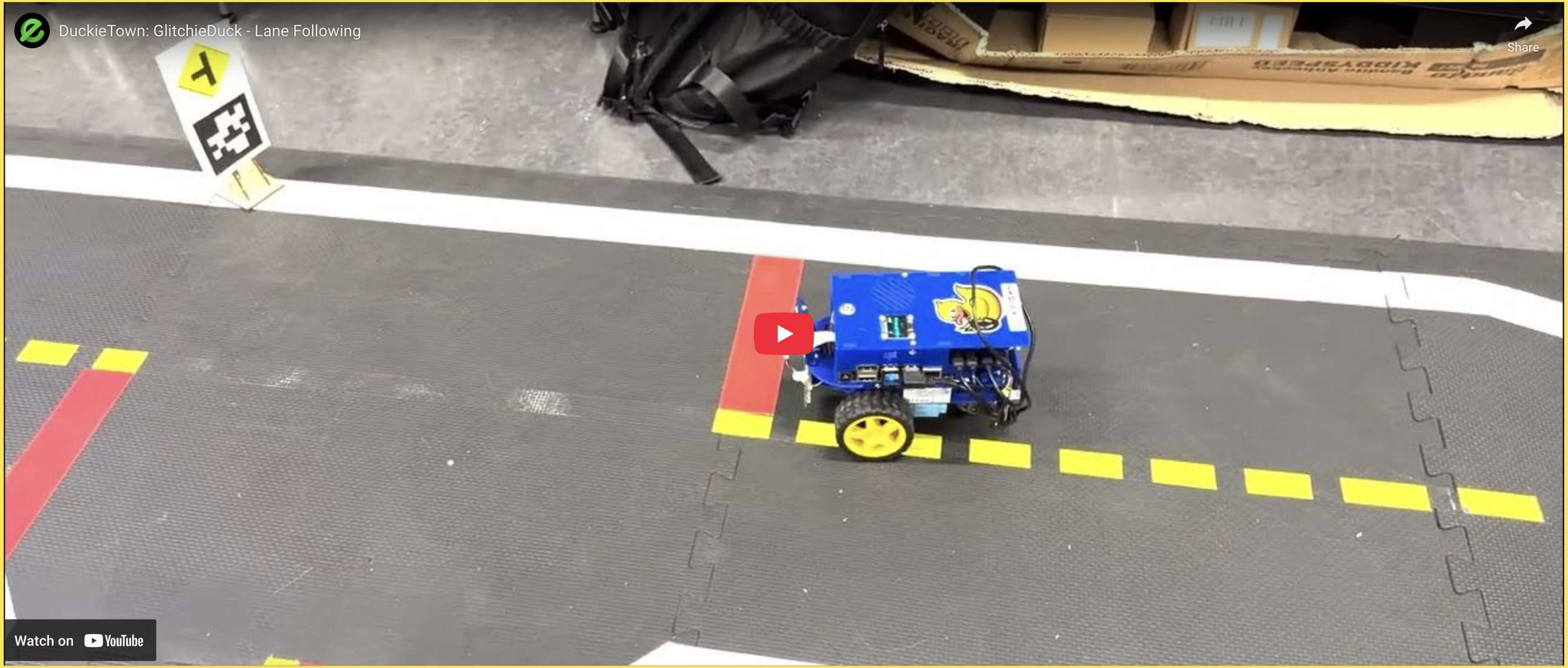
Recognizes street signs using special codes and behaves accordingly

QuickDuck: Issues

Issues with default dt-core implementation

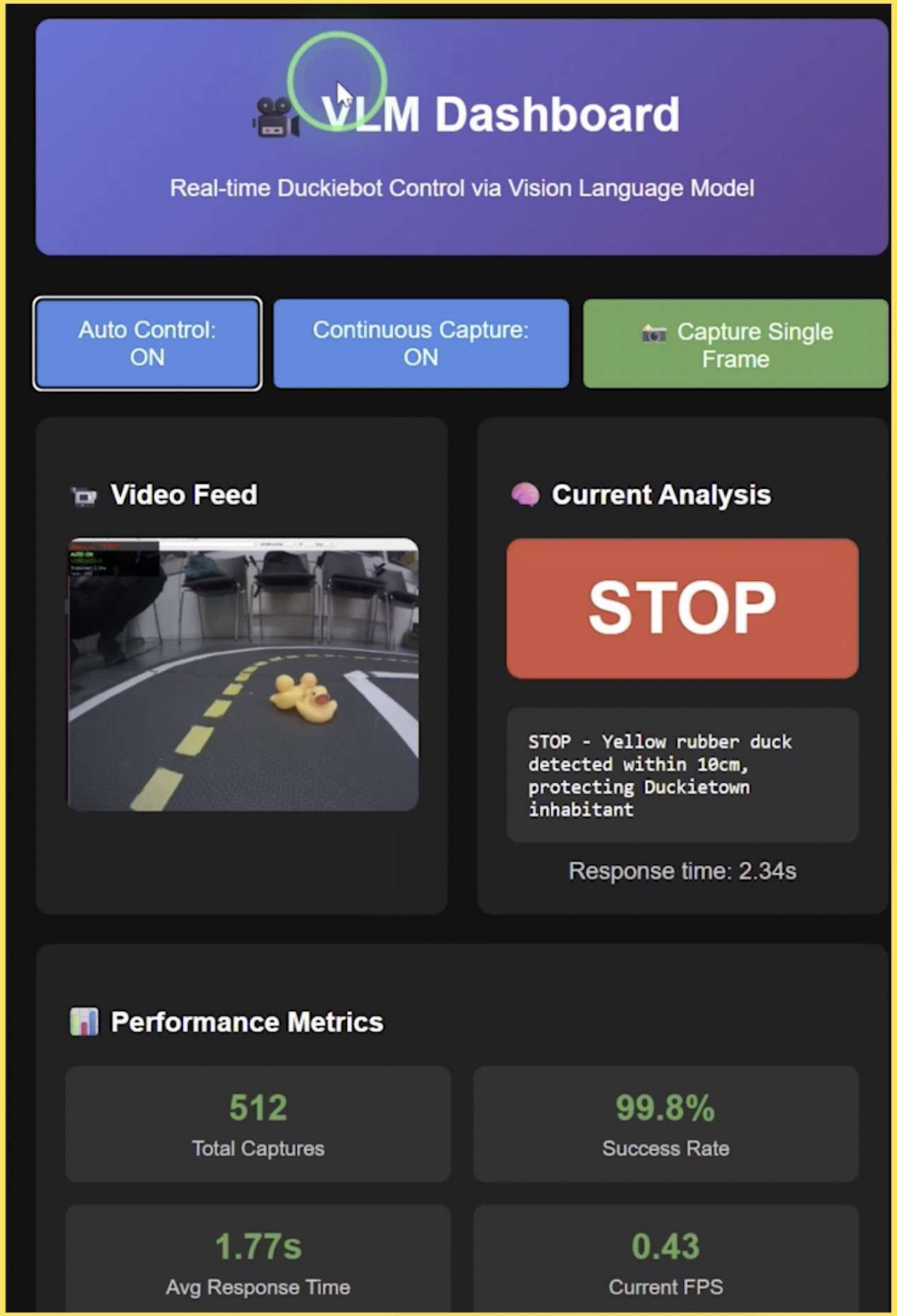
QuickDuck: Lane Following

Autonomous driving implementation using dt-core packages



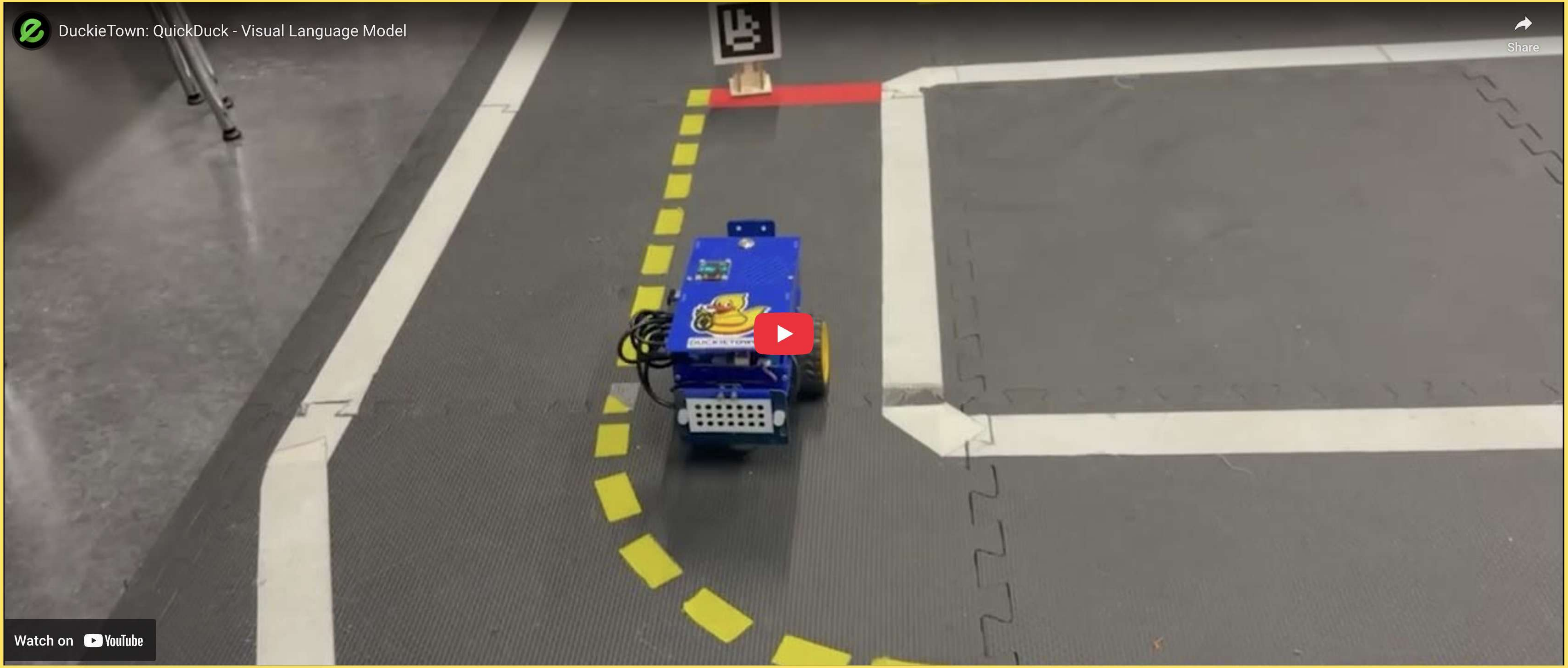
Introduction to VLM Algorithm

Visual Language Model used as a brain to decide how the Duckiebot operates



QuickDuck: Visual Language Model

Using QWEN 2.5 7B to control the Duckiebot



Integration Challenges and Solutions

Overcoming compatibility issues with testing and best practices

Installation of DTS Shell

Issues with installing DTS Shell on Ubunut VM running on **MacOS with ARM**

Calibration

Software Freezing

Cross-platform Compatibility

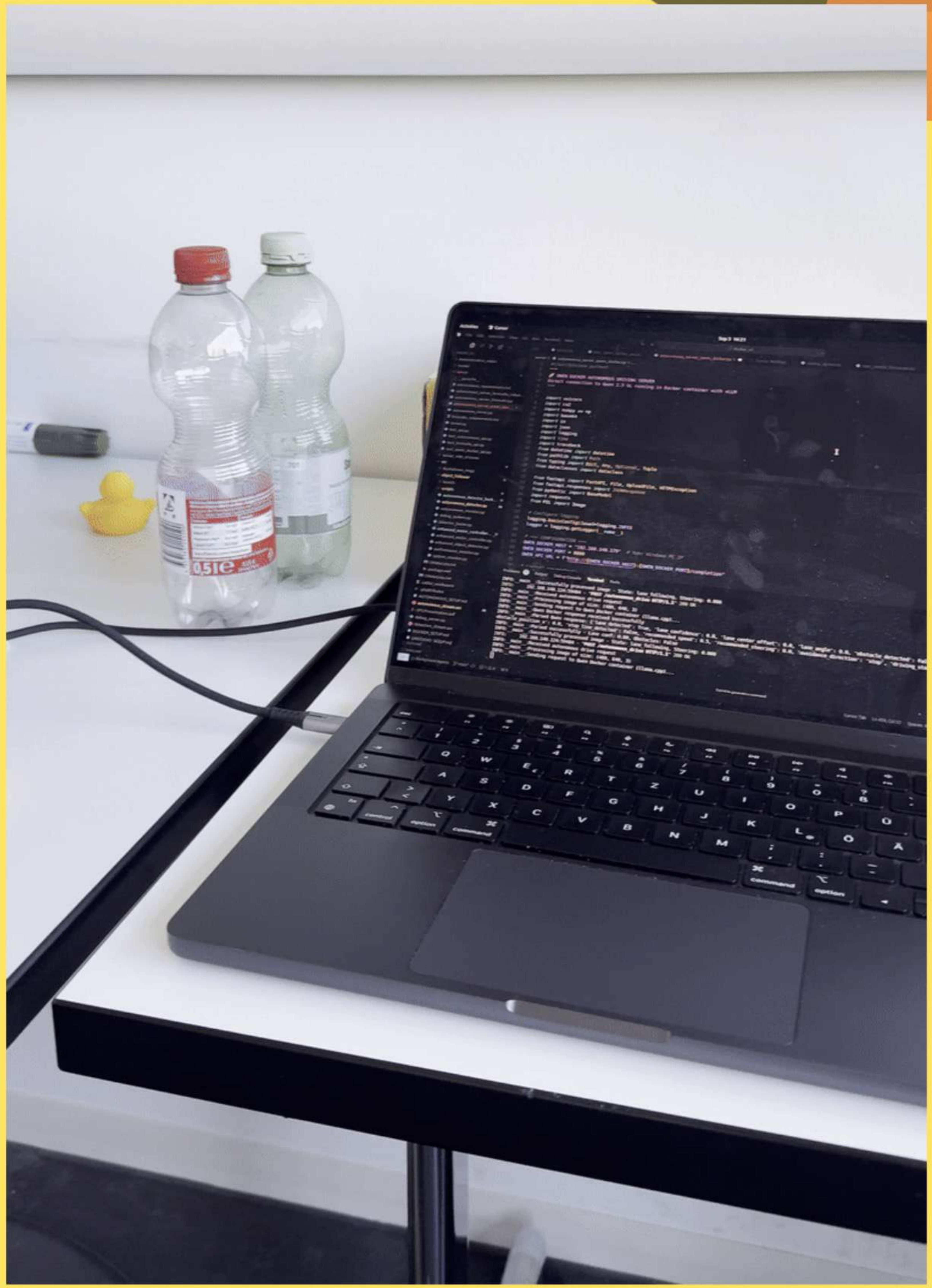
Windows, MacOS and different version of Ubuntu

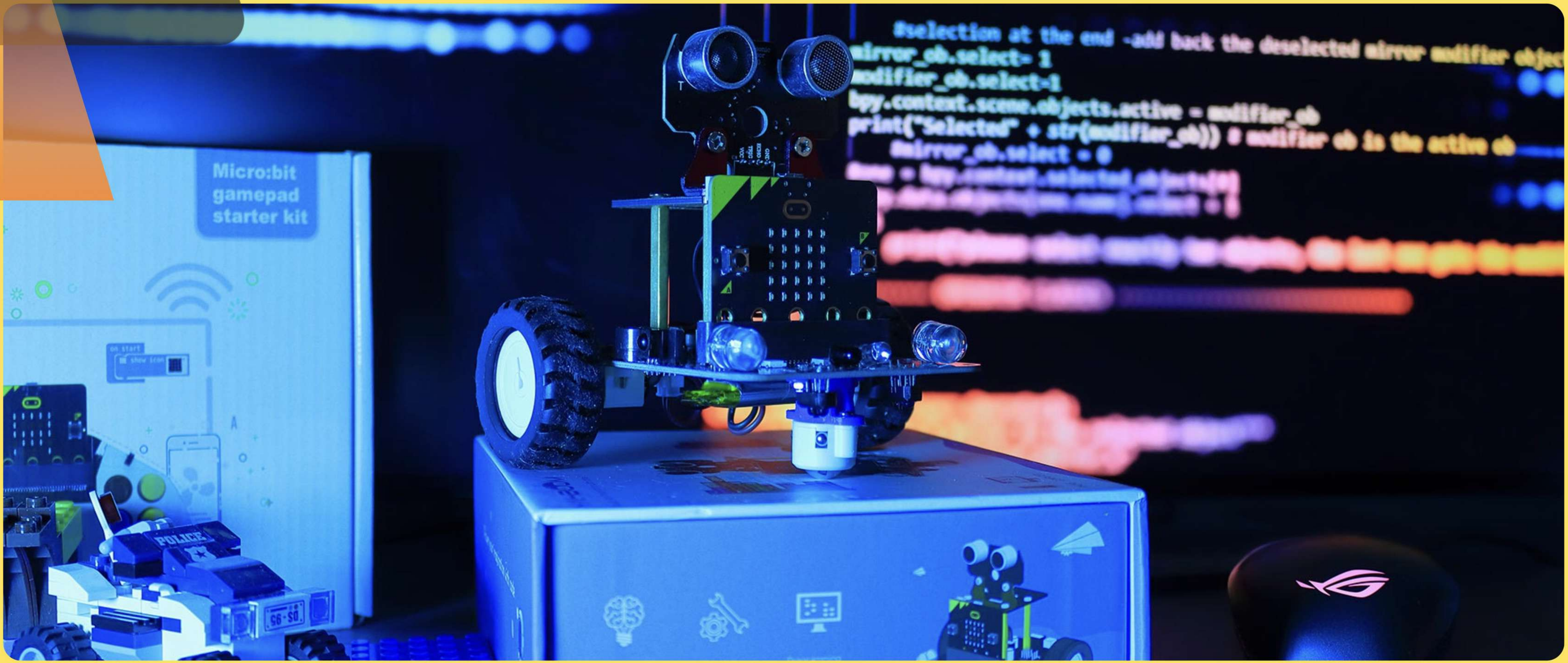
Dt-core

Setting up preconfigured ROS nodes

Integration of VLM into dt-core

Issues with sending commands to DuckieBot, limited resources





Thank you for your attention