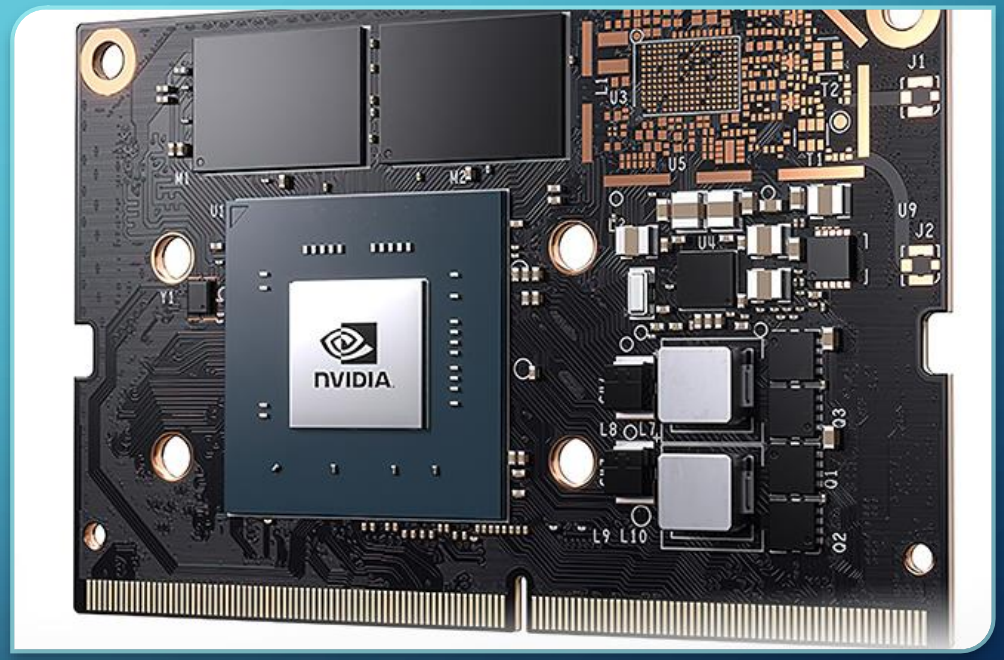
SELF DRIVING & OBSTACLE AVOIDING CAR

BY GROUP 3: ILIA PANAYOTOV, ESRA&ESIN MEHMEDOVI, NOUR HADJ FREDJ, DIMITAR GAVOV, JING YAO SEOW

INTRODUCTION

WHAT IS JETSON NANO?

- A compact AI computer designed by NVIDIA.
- Capable of running multiple neural networks in parallel.
- Ideal for AI applications such as image classification, object detection, segmentation, and speech processing.



WHY JETRACER AI ROS KIT?

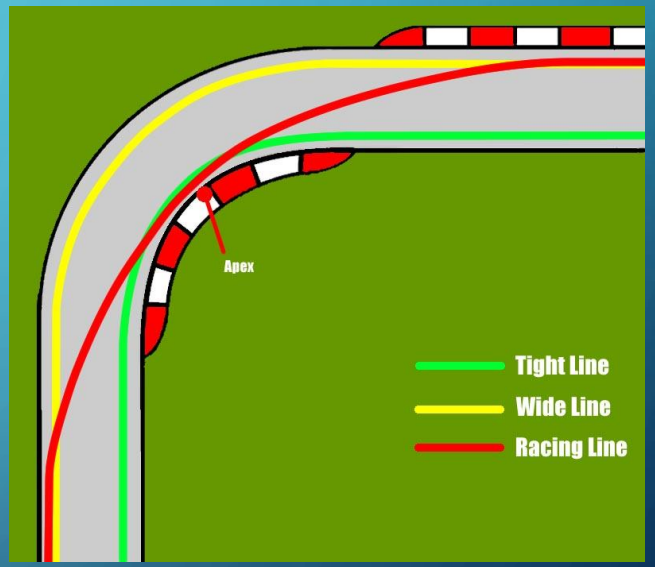
KEY FEATURES

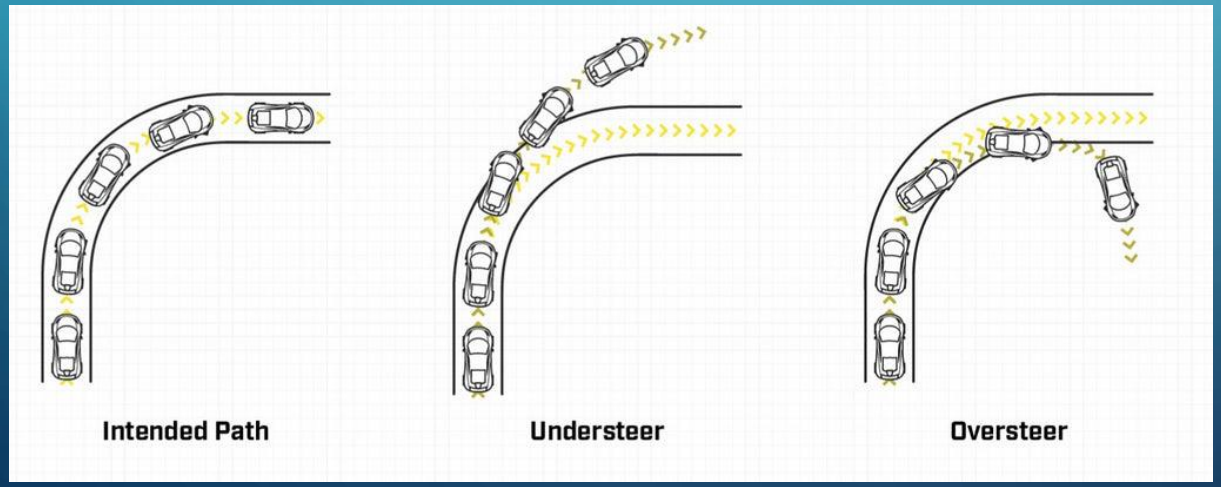
- **Camera**: 160-degree field of view, manufactured by Sony, ensuring high-quality image capture.
- **Main Board**: Jetson Nano – offers powerful AI capabilities, surpassing Raspberry Pi due to integrated NVIDIA graphics.
- **Powerful motors:** The jet racer isn't called a racer without reason. It has two brushed motors with reduced gears for more torque and a lower response rate.

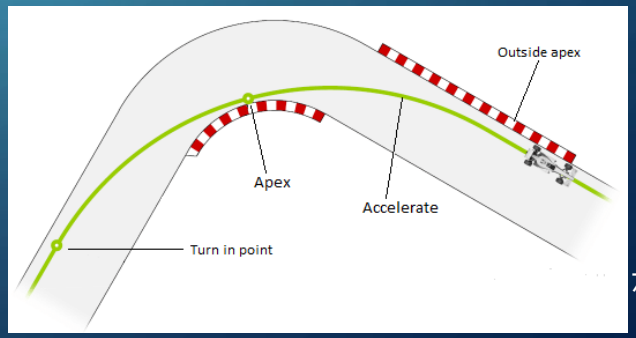
INITIAL GOAL

DEVELOP A PILOTLESS RACER ROBOT

- Objectives:
  - Follow a line on a track, simulating a Formula One race.
  - Optimize the robot's speed and path to prevent understeer and oversteer.



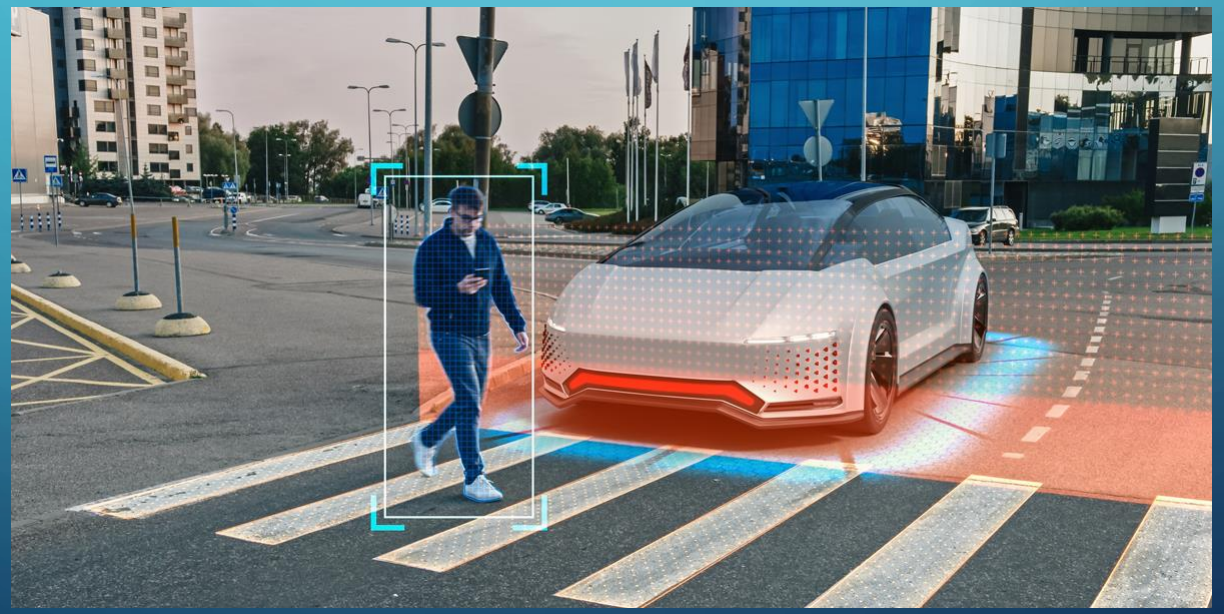




PROJECT EVOLUTION

AUTONOMOUS URBAN NAVIGATION AND PEDESTRIAN AVOIDANCE

- Navigate a town-like road setup.
- Detect and stop for pedestrians (represented by yellow rubber ducks).



SENSORS USED

- Camera for both line following and pedestrian detection.
- Motors with position detection for accurate movement control.

BACKGROUND INFORMATION

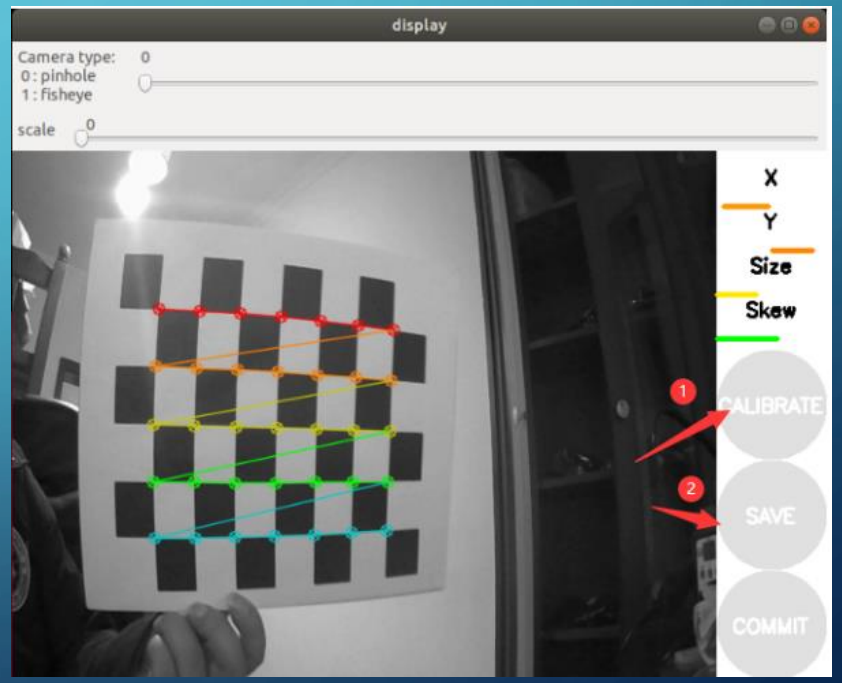
CAMERA

- High-Quality Image Capture: Equipped with a Sony sensor, the camera provides a 160-degree wide-angle view, crucial for comprehensive environmental monitoring.
- Applications: Utilized for both navigation (line following) and obstacle detection (pedestrian recognition).
- Features: Capable of capturing high-resolution images and video, ensuring accurate data for processing by the AI algorithms.

CAMERA

- Because of the high view angle, we had to use the ROS software for image distortion correction.

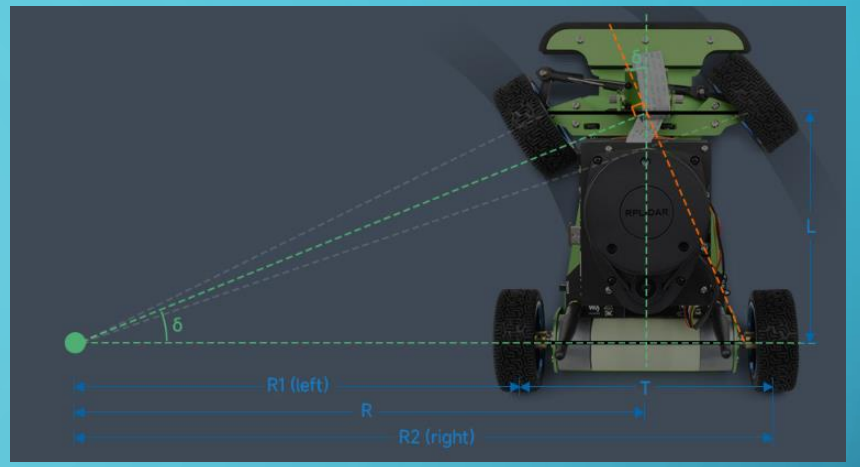
Calibration was done with the provided chess playboard paper on the right side. The squares should be moved in all corners so the car can understand and correct the distortion accordingly.



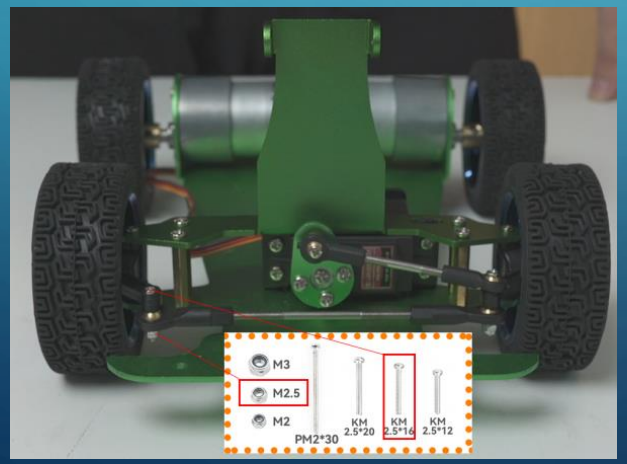
MOTORS WITH POSITION DETECTION

- **Accurate Tracking of Wheel Rotation:** Each brushless motor includes a built-in sensor to monitor the exact rotational position.
- **Benefits:** Enables precise control over the robot's movement, essential for maintaining the correct speed and path during autonomous driving.
- **Additional Uses:** Helps in calculating the distance traveled and assists in dynamic adjustments to steering and acceleration.

STEERING MECHANISM

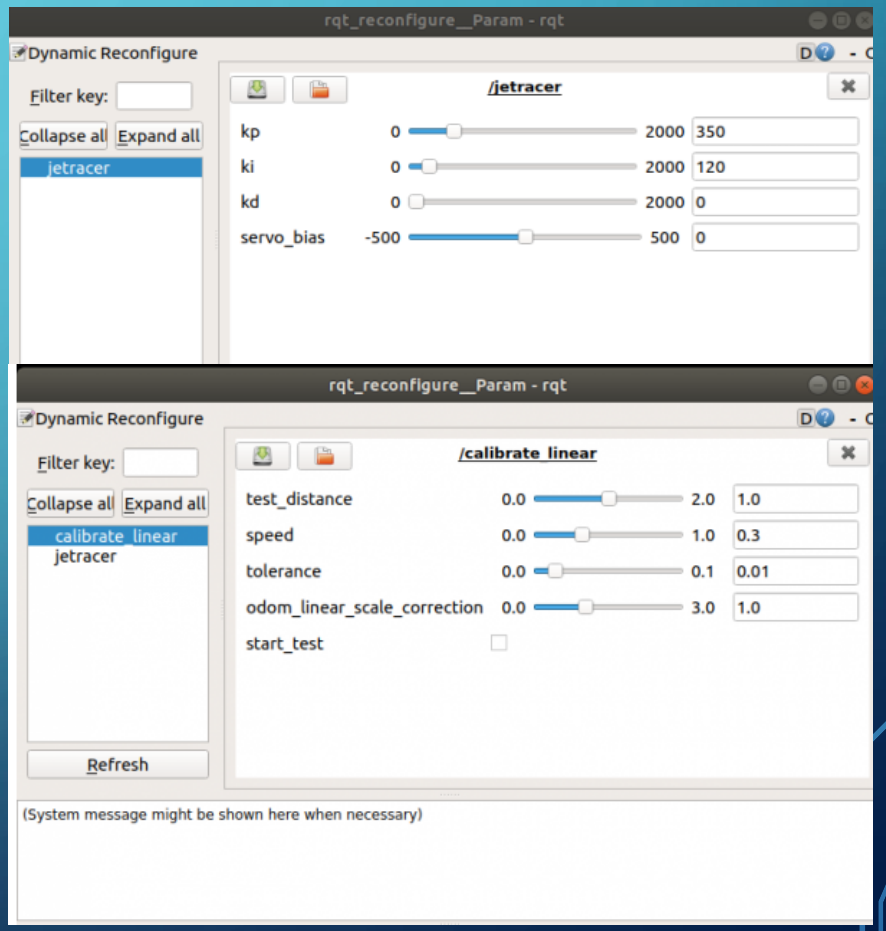


- **Ackermann Steering Geometry:** This mechanism ensures that all wheels are correctly aligned during turns, reducing tire wear and enhancing maneuverability.
- **Servo and Tie Rods:** The steering is controlled by a servo motor connected to the wheels via tie rods and ball joints, providing responsive and accurate steering.



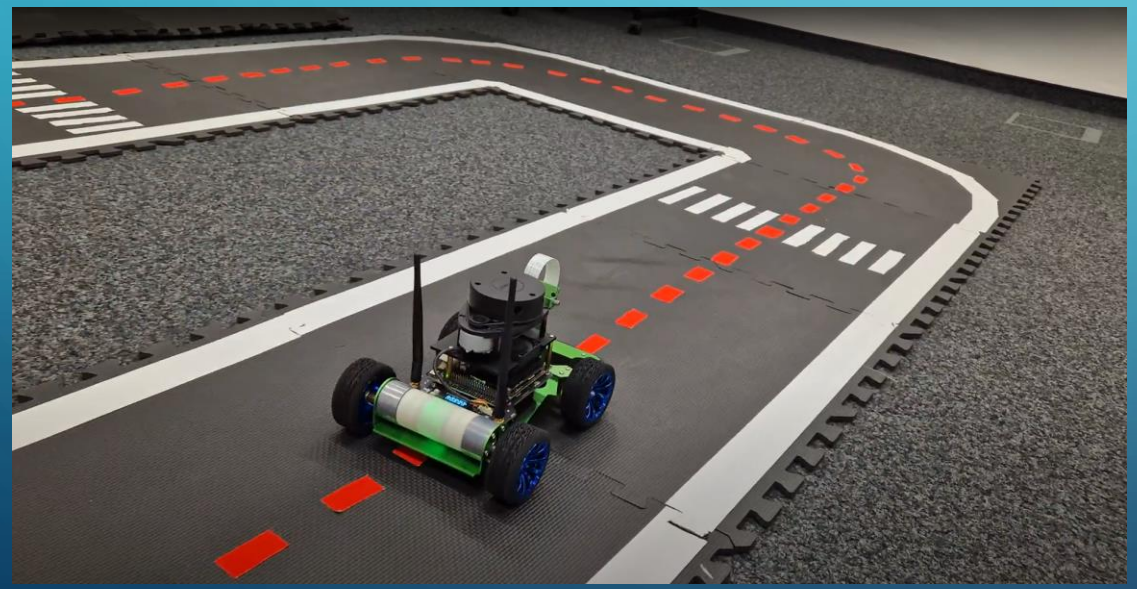
MOTORS WITH POSITION DETECTION & SERVO MOTOR

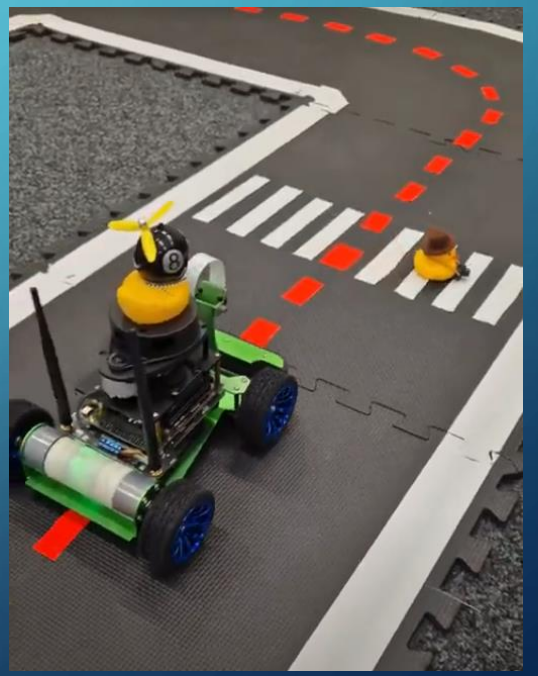
- The servo motor operates the steering, and it had to be centered, again using ROS.
- The drive motors had to be calibrated also. There is a ready test for them which makes the robot move for a meter. Fidgeting with test_distance we set the distance.



TRACK AND PEDESTRIAN SETUP

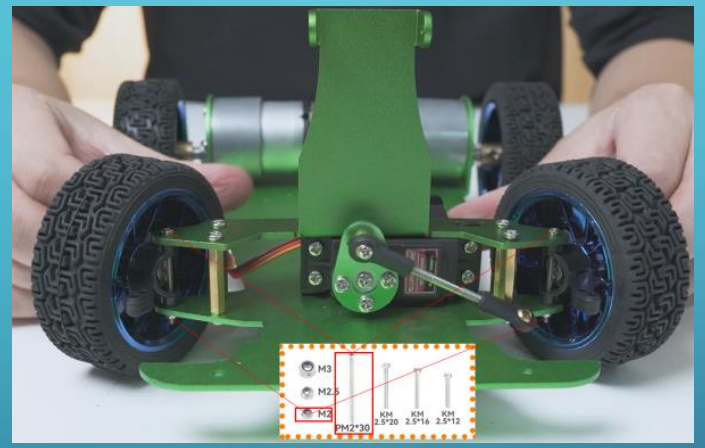
- Custom Track: A minimized town road setup within the university lab designed to simulate real-world driving conditions.
- Pedestrians: Yellow rubber ducks are used as proxies for pedestrians and placed to test the robot's detection and response capabilities.

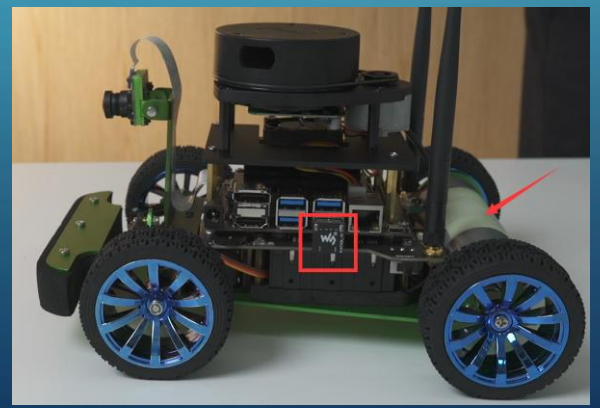


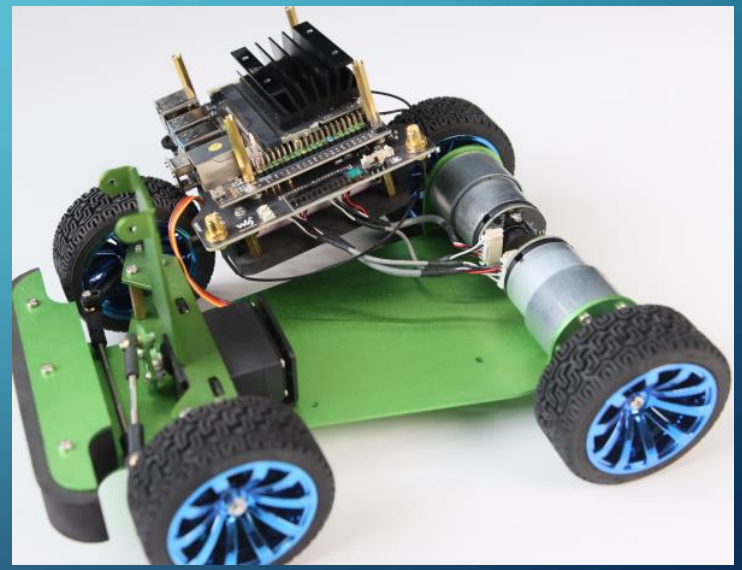


ASSEMBLY OF THE ROBOT

- Involved in Combined mechanical and electronic components, such as the chassis, wheels, motors, sensors, and the Jetson Nano board. Following steps from the manufacturer.
- Software Setup: Installing and configuring ROS on both the Jetson Nano and a virtual machine to enable seamless communication and control.

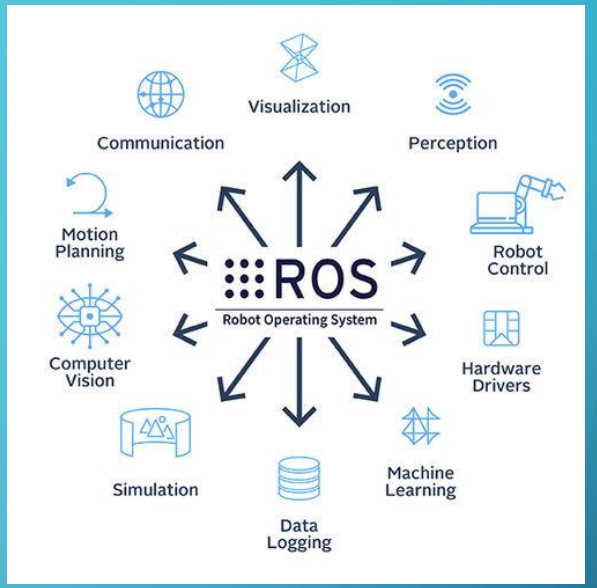




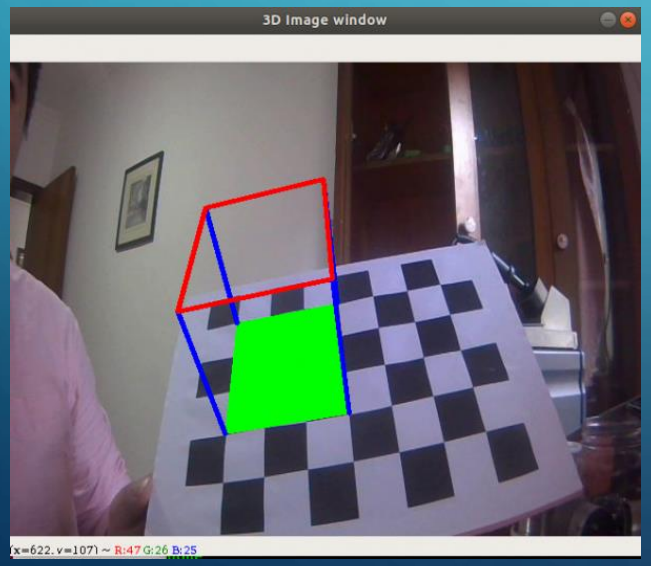


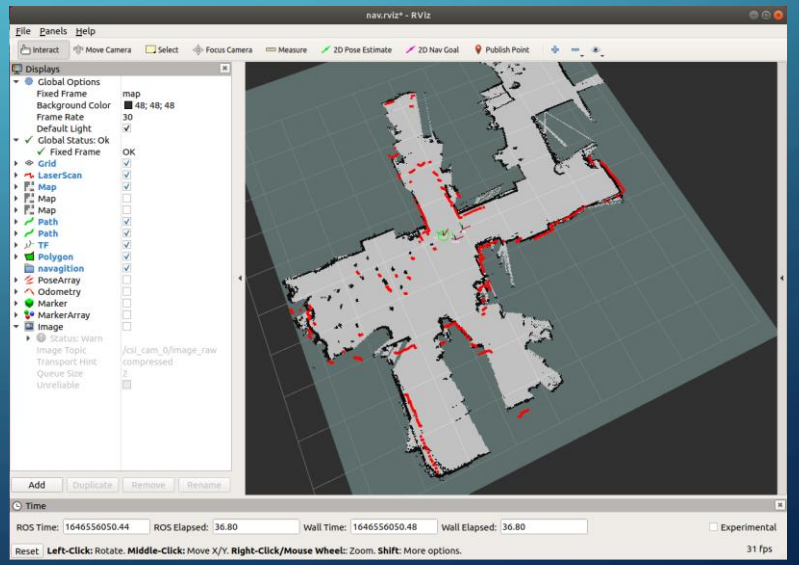
INTRODUCTION TO ROS

ROBOT OPERATING SYSTEM (ROS)



- A flexible framework for writing robot software.
- Provides tools and libraries to help build robot applications.
- Facilitates communication between the robot and a virtual machine (VM).





SETTING UP ROS
AND MULTI-MACHINE COMMUNICATION

MAIN SYSTEMS

- Virtual Machine (VM): ROS installed on a VM to control the robot.
- Jetson Nano Robot: Ubuntu system that adopts jetson nano as the main control

CONFIGURE MULTI-MACHINE COMMUNICATION

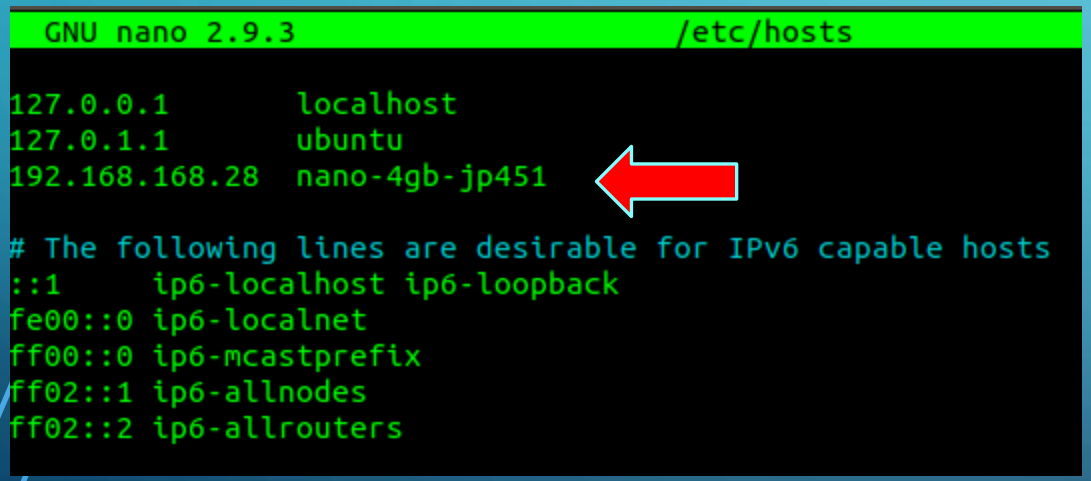
1. Add environment variables to the robot and virtual machine to set the robot as the host and the virtual machine as the slave.

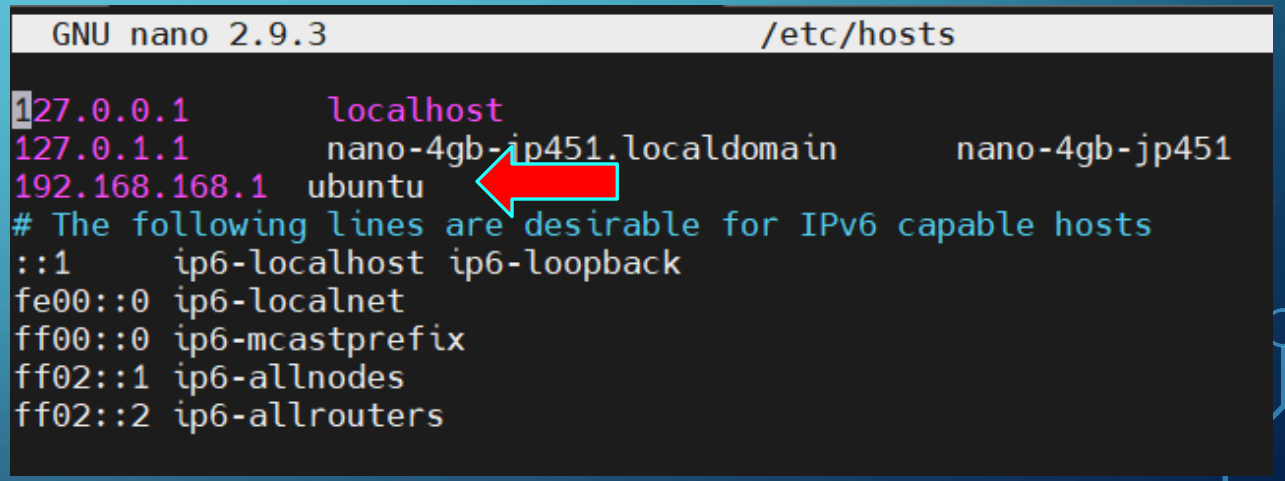
```
 GNU nano 2.9.3                  /home/jetson/.bashrc

export ROS_MASTER_URI=http://nano-4gb-jp451:11311
export ROS_HOSTNAME=ubuntu

# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/home/jetson/miniconda3/bin/conda' 'shell.bash' 'hook' 2> /d$
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/home/jetson/miniconda3/etc/profile.d/conda.sh" ]; then
        . "/home/jetson/miniconda3/etc/profile.d/conda.sh"
    else
        export PATH="/home/jetson/miniconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<

export ROS_MASTER_URI=http://nano-4gb-jp451:11311
export ROS_HOSTNAME=ubuntu
```

```
 GNU nano 2.9.3                  /home/jetson/.bashrc

alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Add an "alert" alias for long running commands.  Use like so:
#   sleep 10; alert
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || ech$

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
source /opt/ros/melodic/setup.bash
source ~/catkin_ws/devel/setup.bash
export ROS_MASTER_URI=http://nano-4gb-jp451:11311
export ROS_HOSTNAME=nano-4gb-jp451
```

2. Add the IPv4 addresses and hostnames to the VM and Robot.

- Connection Challenges: Issues with IPv4 addresses on the VM side resolved through troubleshooting.



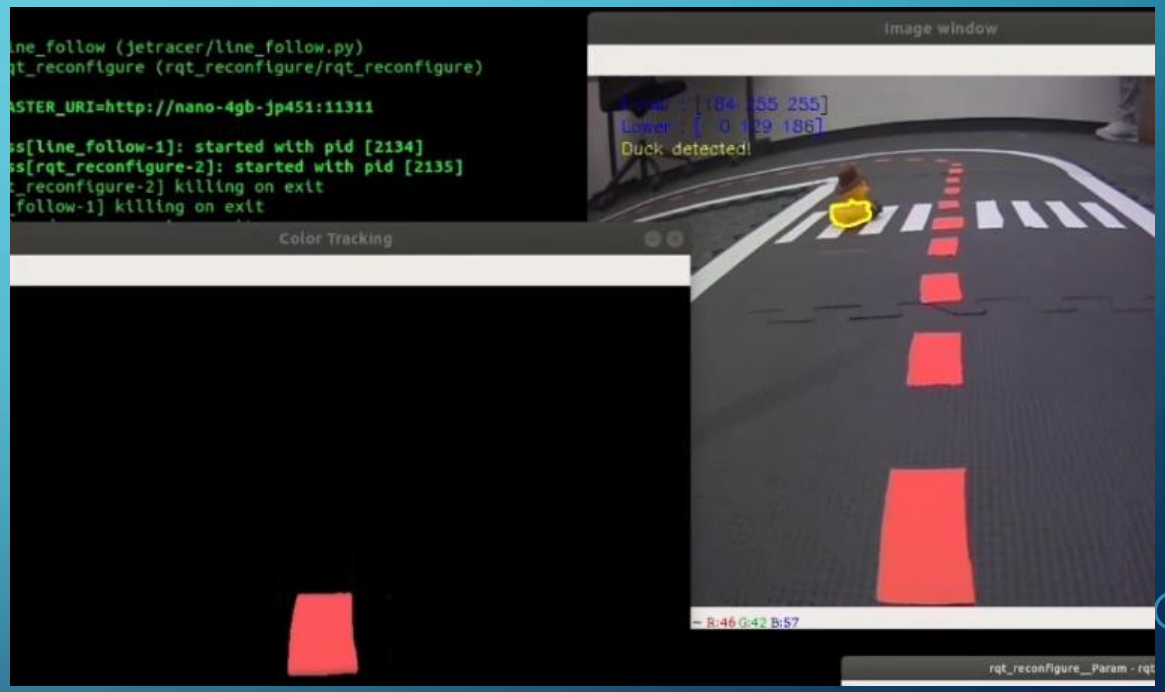


PROBLEM DEFINITION

- **Primary Challenge:** Enable the robot to follow a line while detecting and stopping for pedestrians.

- **Line Following:**
  - Monitor wheel rotations for speed control.
  - Adjust steering based on camera input.

- **Pedestrian Detection:**
  - Integrate camera data for real-time obstacle detection.



SOLUTION APPROACH

AUTONOMOUS DRIVING IMPLEMENTATION

Predefined ROS packages used for basic operations:

- Start the robot node.

```
jetson@nano-4gb-jp451:~$ roslaunch jetracer jetracer.launch
... logging to /home/jetson/.ros/log/910f6106-33a1-11ef-bf88-7404f1beae47/roslau
nch-nano-4gb-jp451-28573.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://nano-4gb-jp451:39637/
```

- Initiate camera data transmission to the VM.

```
jetson@nano-4gb-jp451:~$ roslaunch jetracer csi_camera.launch
... logging to /home/jetson/.ros/log/910f6106-33a1-11ef-bf88-7404f1beae47/roslau
nch-nano-4gb-jp451-29053.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://nano-4gb-jp451:41361/
```

AUTONOMOUS DRIVING IMPLEMENTATION

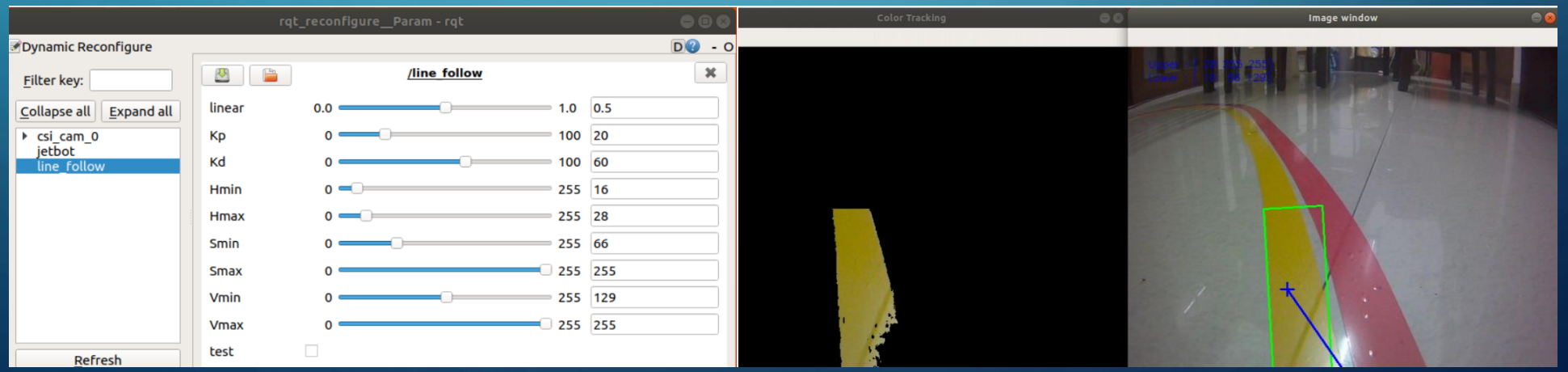
Predefined ROS packages used for basic operations:

- Start the visual line follow control node

```
jetson@ubuntu:~$ roslaunch jetracer line_follow.launch
... logging to /home/jetson/.ros/log/689c01f8-33e6-11ef-bdc1-000c298973cf/roslaunch-ubuntu-2206.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ubuntu:44227/
```

AUTONOMOUS DRIVING IMPLEMENTATION

- Use ROS to monitor and adjust the car's path based on camera input.
- Define the line color for the robot to follow and commence autonomous driving.
- Adjust the motor's speed parameters to ensure correct line following and path error correction



PEDESTRIAN DETECTION INTEGRATION

The robot recognizes rubber ducks as pedestrians:

1. Define an upper and lower bound for the HSV (hue, saturation, value) color of the rubber ducks

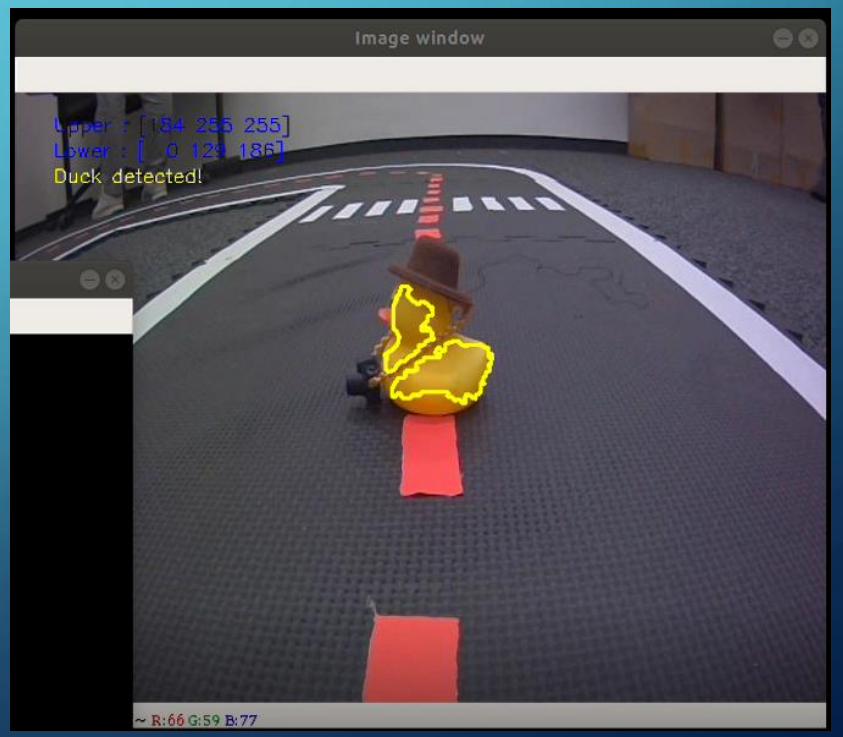
```
self.lower_yellow = np.array([22, 93, 0])
self.upper_yellow = np.array([45, 255, 255])
self.yellow_detected = False
```

PEDESTRIAN DETECTION INTEGRATION

The robot recognizes rubber ducks as pedestrians:

2. Draw contours around yellow objects if the area of the object exceeds the min_area_threshold

```
# Set the minimum area threshold for yellow detection
min_area_threshold = 500  # Adjust this value as needed
self.yellow_detected = False

for contour in contours:
    area = cv.contourArea(contour)
    if area > min_area_threshold:
        self.yellow_detected = True
        cv.drawContours(cv_image, [contour], -1, (0, 255, 255), 2)
```



PEDESTRIAN DETECTION INTEGRATION

The robot recognizes rubber ducks as pedestrians:

3. Stop the car when a duck is detected by publishing a Twish message

```
self.cmd_pub.publish(Twist())
cv.putText(cv_image, "Duck detected!", (30, 70), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 1)
```

4. Continue driving when the duck is no longer detected, with the same parameters as before the stop.

PROJECT CHALLENGES AND MILESTONES

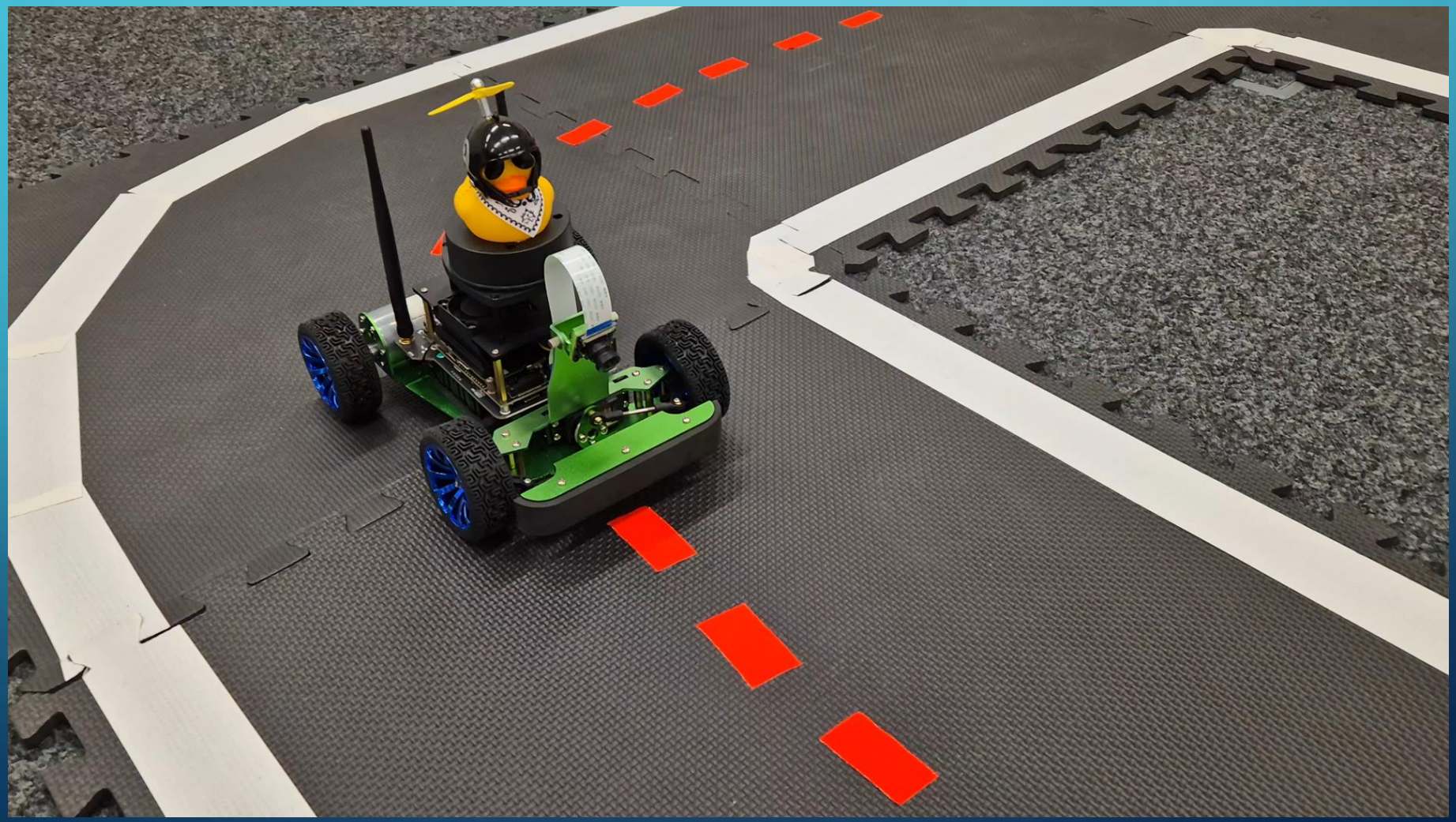
CHALLENGES

- Initial non-functional camera sensor and pink image-tinting.
- Debugging and troubleshooting connectivity issues.
- Learning and implementing complex ROS functionalities.

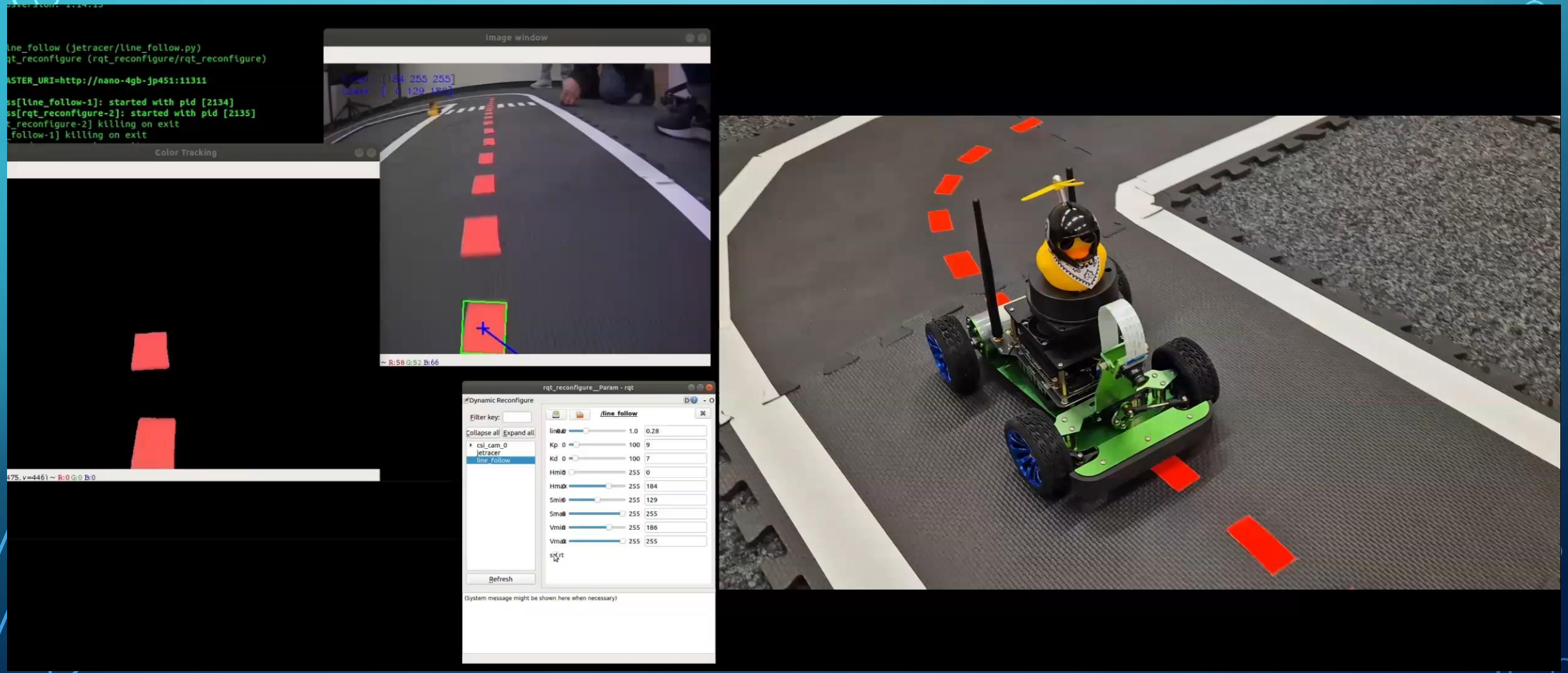
MILESTONES

- Assembling the robot.
- Successful manual control.
- Functional camera integration.
- First autonomous drive.
- Successful pedestrian detection and stopping.

MAIN MILESTONE – WORKING CAR



BACKEND INCLUDED



CONCLUSION

PROJECT SUMMARY

- Transitioned from a simple line-following racer to a sophisticated self-driving car.
- Overcame various technical challenges to achieve reliable autonomous navigation.
- Demonstrated the effectiveness of the Jetracer AI ROS Kit in real-world applications

FUTURE WORK

- Further refine pedestrian detection algorithms.
- Explore additional sensors and features to enhance autonomy.
- Integrate LIDAR for second layer of obstacle avoidance.

QUESTIONS?