Report on a AI Firefighting Drone - FireWatch

William Wu[1], Yangxuezhe (Harry) Sun[1], and Jiarui (Jerry) Hu[1]

[1]Team USA

November 28, 2024

Contents

1 Introducing the Team

Our team, Driver US, composed of passionate high school students from California, undertook the challenge of developing and creating an unique, innovative autonomous vehicle solution for the 2024 World Robot Olympiad (WRO) Future Engineers competition.

Coach:

Fei Guo

Team Members:

William Wu

Yangxuezhe

Sun Jiarui Hu

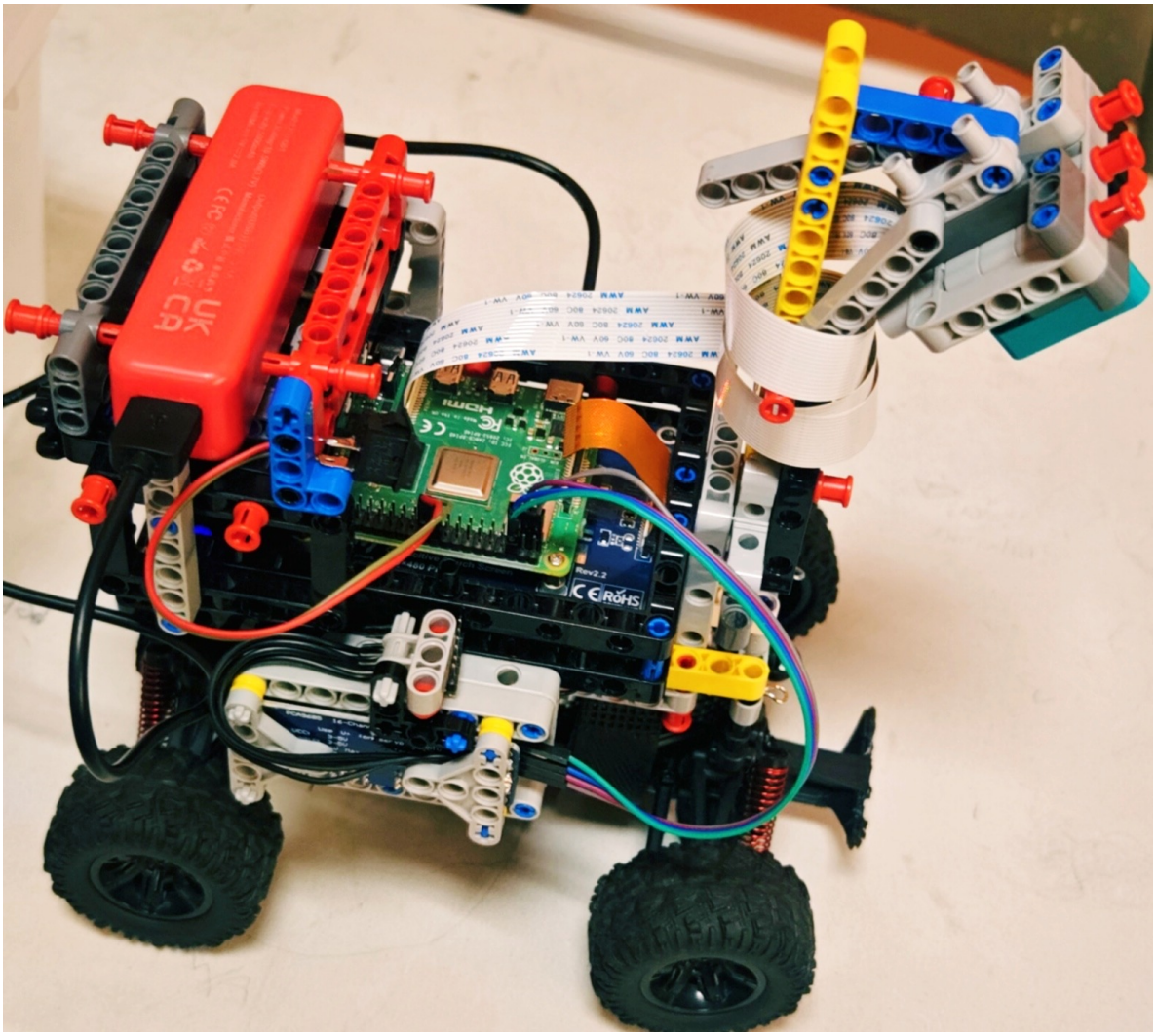
William Wu: 10th-grade student skilled in programming and vehicle design, handling assembly and PWM for precise speed and steering.

Yangxuezhe Sun: 11th-grade student specializing in AI algorithm integration, ensuring real-time AI performance in complex track conditions.

Jiarui Hu: 12th-grade student focused on AI optimization and 3D modeling, simulating stable AI adaptability before real-world testing

Executive Summary

The 2024 WRO Future Engineers competition challenges participants to design and develop fully autonomous vehicles capable of dynamic obstacle avoidance, precise navigation, and accurate parking in simulated real-world scenarios. Our team, Driver US, employs a cutting-edge approach that combines AI-driven technologies with robust hardware engineering. By utilizing convolutional neural networks (CNNs) and integrating these models into Raspberry Pi-based systems, we achieved a seamless blend of artificial intelligence and robotics. This solution ensures real-time decision- making, precise maneuvering, and reliable performance on complex tracks.



2 Introduction

2.1 The Inspiration

This project stems from the concept of leveraging AI to enable seamless on-the-spot adaptation and advanced computational capabilities. Instead of relying on traditional analytical software requiring extensive design and modeling, our approach focuses on integrating an AI model directly into the system, streamlining functionality and enhancing performance.

3 Project Description

3.1 General Aspects

Our project focuses on developing an intelligent, autonomous robotics system that combines advanced AI capabilities with robust hardware design. By leveraging machine learning models, state-of-the-art sensors, and efficient control systems, we aim to create a dynamic platform capable of real-time decision-making and adaptation to complex environments. This work emphasizes seamless integration of software and hardware, enabling high-performance computing, accurate obstacle detection, and precision control for tasks such as navigation, object recognition, and automated parking. Our approach showcases the potential of AI-driven robotics in solving real-world challenges with innovation and efficiency.

3.2 Technical Aspects

3.2.1 Mechincal Construction of a Solution

Our vehicle's design represents a careful combination of mechanical engineering, AI development, and control system integration, with a strong emphasis on robust, modular, and Self-Designed hardware.

RC Car Base: The platform for the vehicle is an RC car (Bezgar Remote Control Car [3]), chosen for its durability, speed, and reliability in high-performance environments. This base provided a sturdy foundation for handling the demanding tasks of the WRO competition.

LEGO: We designed a modular LEGO structure [12] mounted atop the RC car to securely house the essential hardware components. The LEGO construction was chosen for its adaptability and standardization of parts, allowing us to make iterative adjustments during testing phases. This modularity proved critical when fine-tuning the positioning of the camera, gyroscope, and Raspberry Pi, ensuring optimal performance across varied track conditions and lighting environments.

Gyroscope: The WT901 gyroscope [14] was integrated into the system to monitor the vehicle's orientation and angular velocity. This sensor played a key role in determining when to stop the vehicle after 3 laps and monitored orientation data throughout the run.

Raspberry Pi: At the heart of the vehicle lies the Raspberry Pi 4 Model B [7], which served as the central processing unit (CPU). It processes data from the camera and gyroscope in real time, running AI models to make instantaneous decisions. The Raspberry Pi was selected for its computational power, compact size, and support for advanced machine learning frameworks such as TensorFlow. Most of the code is located on the raspberry pi inside a hard drive (SD card) and is able to accessed easily

Picamera: The vehicle is equipped with a Raspberry Pi Camera Module v2 [8], which provides a live video feed of the track. This feed is critical for the AI system to recognize track boundaries, detect obstacles, and make informed navigational decisions. The camera was mounted on an adjustable LEGO arm, enabling precise positioning.

PWM: To achieve precise control over the vehicle's movements, we implemented Pulse Width Modulation (PWM with ic2 interface [1]) for motor management. PWM allowed fine-tuned adjustments to the vehicle's speed and steering. This control system was essential for the raspberry pi to assimilate with the RC car and control it. Additionally, PWM optimized power consumption, ensuring reliable performance throughout extended competition runs.

Tires: To improve performance, the stock RC car tires were replaced with high-traction, off-road tires. These upgraded tires provided enhanced stability and grip on track surface and allowed it to make sharp turns without accidentally drifting off-course.

By integrating these carefully selected hardware components into a cohesive design, our vehicle was able to handle the rigorous challenges of the WRO competition, demonstrating exceptional adaptability and performance in real-world scenarios.

3.2.2 Software Construction of a Solution

The Main Plan: We run the Linux operating system on our Raspberry Pi. Within this system, we use Python as our programming language and TensorFlow [10] as the tool for our AI modeling and training tasks.

Python Programming: Using Python, we control the Raspberry Pi's camera to capture images and collect data. Additionally, we connect a joystick via Bluetooth, and both the joystick data and the images from the camera are stored together which will later be processed by the code.

The Training Process: The AI model for our autonomous vehicle was developed using real-world data collected during manual driving sessions on the competition track, facilitated by the script manage.py. These sessions enabled us to capture critical data, including camera images and vehicle control inputs, which served as the foundation for training the model.

AI Analysis: Next, we package all of this data and hand it over to TensorFlow using a system called FileZilla [11]. Inside TensorFlow, we've built a Convolutional Neural Network (CNN) model for training. The inputs to the model are a video matrix, steering control input, and throttle data input. These three types of inputs pass through eleven layers of the network, and in the end, two types of outputs are produced: the predicted steering and throttle values.

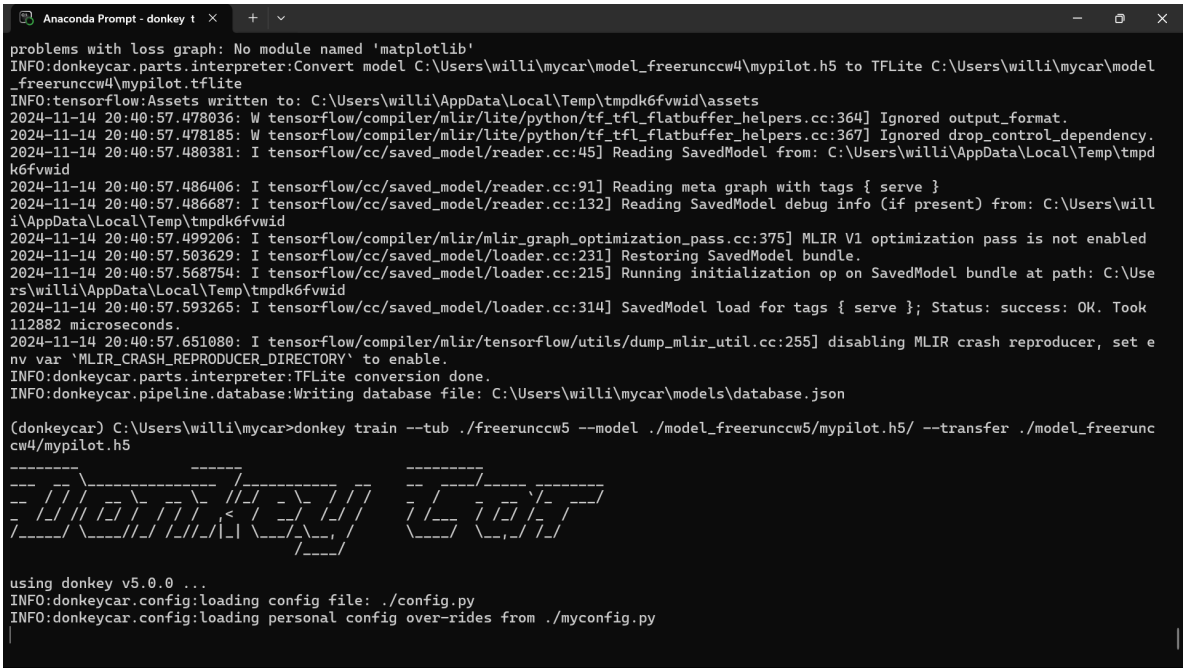
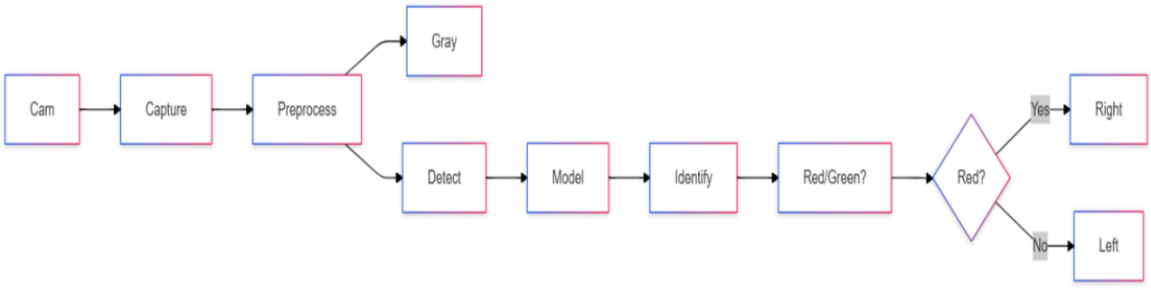


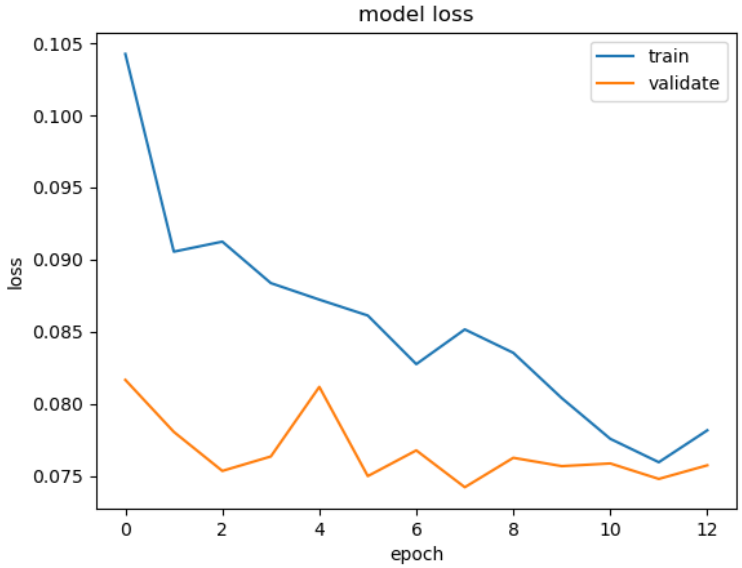
Figure 1: Training Using Tensorflow

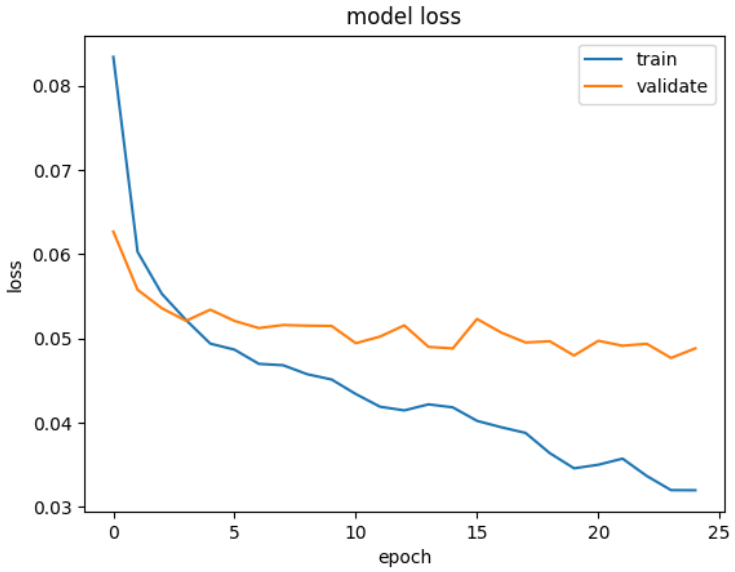
The training process was iterative, involving extensive testing and fine-tuning to optimize the model's accuracy and performance. After each iteration, we evaluated the AI's ability to handle various track scenarios, including straight paths, sharp turns, and obstacle navigation. With each improvement, the AI system became increasingly adept at interpreting visual inputs, making real-time decisions, and autonomously navigating the complex track conditions with precision.



Reporting Fires: The trained AI model is then deployed back onto the Raspberry Pi. The AI, using real-time images, performs wildfire detection and sends analysis reports along with GPS coordinates to the control center. This allows for early warnings, quick responses, and real-time monitoring of wildfires.

Visualizing Data: Additionally, we have developed a web interface that displays the live footage of the vehicle from the computer, allowing semi-remote control. In addition to this we also created a feature that was included when we were training our data using TensorFlow. The data loss for the AI training model is collected and displayed in graphs 3.2.2 and 3.2.2. In the beginning we were using around 8000 pieces of data per model. We can see that the training loses around 8.5% of data throughout the training process. We improved this later on by training only around 5000 data pieces per model. The training loss graph now only shows around a 5% loss of data, a 41% increase in efficiency. Thus in the end we decided to use around 5000 pieces of data which maximized the training efficiency of our data.





3.2.3 Overview of the Solution

In summary, the software aspect of our solution integrates machine learning, sensor processing, and real-time control to enable autonomous functionality. Using Python as the core programming language, we employ TensorFlow for training and deploying AI models that handle tasks like object recognition and decision-making. Additional tools, such as realVNC [2] and FileZilla [11], streamline remote access and file management for development efficiency. The control software processes inputs from cameras, gyroscopes, and other sensors to adaptively steer and throttle the system based on environmental conditions, ensuring seamless operation and robust obstacle avoidance.

4 Software and Coding

Here is a list and description of our important python scripts are listed below. Note that the full codes can be accessed on our GitHub [6].

4.1 Code Descriptions

- manage.py is the main code that collects training data for the AI model
- config.py defines the default Pulse Width Modulation (PWM) values, which are critical for controlling the vehicle's motors and steer-

ing. These parameters ensure consistent and precise vehicle movement.

- calibrate.py is used to synchronize the AI model with the vehicle's hardware, this script ensures that sensor data and system responses are accurately aligned during operation.
- airc_drive10.py serves as a testing tool for the AI model. It allows us to evaluate and debug the vehicle's performance in a controlled environment before deploying it in competition.
- myconfig.py provides the foundational framework upon which we built our customized AI solutions which is imported from the donkey car forums [5].
- main_freerun.py is the final AI program for the freerun segment of the competition. This script enables the vehicle to autonomously complete three laps around the track and stop precisely based on gyroscope readings.
- main_obstacle.py is the final AI program for the obstacle run segment of the competition. It drives the car through two laps while dynamically avoiding obstacles according to the competition rules. The program then performs an additional lap (clockwise or counterclockwise as specified by the rules) and concludes with a final lap, parking the vehicle accurately in the designated slot.

4.2 Example Code Segments

Listed here are segments of code from our most important scripts.

Code Blame 125 lines (94 loc) · 3.07 KB Raw

```
'''
This is a very basic program that uses donkeycar camera images to feed a keras model to make predictions.
'''

import os
import time
import sys
current_path = os.getcwd()
sys.path.append(current_path)
from donkeycar.parts.camera import PiCamera
import cv2
from donkeycar.parts.interpreter import KerasInterpreter
from donkeycar.parts.keras import KerasLinear
import Adafruit_PCA9685

"""
init pwm9685
"""
# pwm = Adafruit_PCA9685.PCA9685()
# Alternatively specify a different address and/or bus:
pwm = Adafruit_PCA9685.PCA9685(address=0x40, busnum=1)
# Configure min and max servo pulse lengths
# Helper function to make setting a servo pulse width simpler.
def set_servo_pulse(channel, pulse):
    pulse_length = 1000000    # 1,000,000 us per second
    pulse_length //= 60       # 60 Hz
    print('{0}us per period'.format(pulse_length))
    pulse_length //= 4096     # 12 bits of resolution
    print('{0}us per bit'.format(pulse_length))
```

Figure 2: airc_drive10.py

```
'''
Extremely simple code that detects whether the on/off switch is connected to raspberry pi
'''

# Imports the GPIO library
import RPi.GPIO as GPIO
import time

# Set the GPIO mode
GPIO.setmode(GPIO.BCM)  # Use BCM GPIO numbering

# GPIO connection slot on the raspberry pi
gpio_Num = 25
GPIO.setup(gpio_Num, GPIO.IN,GPIO.PUD_UP)

# Prints the GPIO connection signal
while True:
    print(GPIO.input(gpio_Num))
    time.sleep(0.1)
```

Figure 3: GPIO_Detection.py

```
from donkeycar.parts.behavior import BehaviorPart
from donkeycar.parts.file_watcher import FileWatcher
from donkeycar.parts.launch import AiLaunch
from donkeycar.parts.kinematics import NormalizeSteeringAngle, UnnormalizeSteeringAngle, TwoWheelSteeringThrottle
from donkeycar.parts.kinematics import Unicycle, InverseUnicycle, UnicycleUnnormalizeAngularVelocity
from donkeycar.parts.kinematics import Bicycle, InverseBicycle, BicycleUnnormalizeAngularVelocity
from donkeycar.parts.explode import ExplodeDict
from donkeycar.parts.transform import Lambda
from donkeycar.parts.pipe import Pipe
from donkeycar.utils import *

logger = logging.getLogger(__name__)
logging.basicConfig(level=logging.INFO)


def drive(cfg, model_path=None, use_joystick=False, model_type=None,
          camera_type='single', meta=[]):
    """
    Construct a working robotic vehicle from many parts. Each part runs as a
    job in the Vehicle loop, calling either it's run or run_threaded method
    depending on the constructor flag `threaded`. All parts are updated one
    after another at the framerate given in cfg.DRIVE_LOOP_HZ assuming each
    part finishes processing in a timely manner. Parts may have named outputs
    and inputs. The framework handles passing named outputs to parts
    requesting the same named input.
    """
    logger.info(f'PID: {os.getpid()}')
```

Figure 4: manage.py

These scripts seamlessly integrate with the electromechanical systems, including the Raspberry Pi, gyroscope, camera, and motors. Together, the software and hardware ensure a smooth and efficient training process, real-time decision-making, and flawless execution during competition. This harmonious interaction between software and hardware underscores the sophistication of our autonomous vehicle's design.

5 Power and Sense Management

Effective power management was essential to ensure reliable performance throughout the competition. To achieve this, Pulse Width Modulation (PWM) was employed to optimize the vehicle's power consumption, allowing us to extend battery life while maintaining consistent operation and precision control.

Our vehicle was powered by two sources:

- Li-Polymer Model 603048 (7.4V, 850mAh [4]): This battery powered the RC car's motors, ensuring sufficient energy for movement and steering.

- HYD001 Portable Charger [13]: This device powered the Raspberry Pi, which in turn supplied energy to the gyroscope and camera.

This power setup was carefully designed to ensure stability and reliability during operation. The modular structure allowed for efficient energy distribution across all components, providing sufficient power for extended competition runs and maintaining the system's overall functionality.

6 Development Process and Challenges

6.1 Hardware Integration

Since we are using off-the-shelf components, there are times when we couldn't find suitable connectors to assemble the parts together. To address this issue, we used a LEGO parts that provided a concrete basis for our design structure and welding technology to remove/add any compositions. This allowed us to ensure that all components fit together properly and functioned as intended.

6.2 System Weight and Power Constraints

Due to time and resource limitations, we installed our software systems and motor controls on two different systems (Raspberry Pi and RC Car). This resulted in the combined weight being dangerously close to the maximum weight which we had to solve. Moving forward we also combined the startup buttons that drew power from 2 different sources into just one to conform to the international rules.

6.3 Camera Burnout

The Raspberry Pi camera is easy to burn out and break during long periods of testing, likely due to overheating and excessive power draw. Out solution was having multiple backup cameras that we are able to easily swap out the case we notice something is amiss.

6.4 Training Data Output Algorithm

The initial output from the CNN model produced inconsistent predictions, especially in areas with dynamic lighting that differs from the usual conditions that we train in. We improved this by training under multiple different conditions and improving algorithmic efficiency.

6.5 One Button Start

A major issue that we faced during the finalization of the project was the WRO rule that stated only one button can be used to boot up the program. As of the current project we technically had 3 buttons: 2 to start up the Raspberry Pi and the base car and one to start the code. We had to fix this by welding the two power lines for the Raspberry Pi and the car together so that we would only have one "on button".

Figure 5: Welding Power Buttons Together

7 Conclusion

Our project represents the culmination of an intense, multi-month endeavor characterized by relentless dedication, innovative thinking, and synergistic teamwork in designing and developing a fully autonomous vehicle. By combining cutting-edge artificial intelligence methodologies,

robust hardware architecture, and seamless integration of mechanical and electrical components, we successfully engineered a system capable of navigating complex environments, adapting to unforeseen challenges, and completing the rigorous requirements of the 2024 WRO Future Engineers competition—entirely without human intervention.

This transformative journey deepened our understanding of foundational and advanced concepts in machine learning, control systems, and robotics, while simultaneously fostering critical problem-solving, creative thinking, and collaboration skills. From the earliest stages of prototyping and structural development to the intricate processes of sensor integration, AI model training, and system optimization, each phase of the project brought its own distinct challenges and opportunities. These experiences taught us to approach complex problems with ingenuity and persistence, constantly iterating to refine our approach and achieve our ambitious goals.

Our work exemplifies the confluence of theoretical knowledge and practical application. We integrated sensors to perceive the environment accurately, trained advanced AI algorithms to interpret real-time data, and deployed control systems to enable precise and reliable navigation. This multidisciplinary approach not only equipped us to tackle the challenges of the competition but also allowed us to glimpse the transformative potential of autonomous systems in addressing real-world issues such as transportation efficiency, environmental sustainability, and industrial automation.

Participating in this project has not only prepared us for the technical and intellectual demands of a highly competitive environment but also ignited our passion for exploring the broader applications of autonomous vehicles and artificial intelligence. It has inspired us to think beyond competition and envision the profound societal impact these technologies can achieve.

We are proud to share the results of our efforts, including the models and designs hosted on our GitHub repository and our YouTube channel [9] where we share our designs and process. By making these resources publicly accessible, we aim to contribute to the collective advancement of robotics and autonomous systems, empowering others to build upon our work, innovate further, and succeed in their own ventures. This project

is not just a milestone for our team; it is a step toward driving progress in the rapidly evolving field of AI-driven autonomous vehicles, and we look forward to tackling even more ambitious challenges in the future.

References

[1] Adafruit. 16-channel 12-bit pwm servo driver with i2c interface. https://www.amazon.com/16-Channel-12-bit-Servo-Driver-Interface/dp/B00EIB0U7A.

[2] ATT. Realvnc. https://www.realvnc.com/en/.

[3] Bezgar. Bezgar remote control car. https://bezgar.com/products/hp161s-brushless-rc-monster-truck?srsltid=AfmBOooFGoxjODbHSdft2b3kG1eCqt0nHd6ybDr41GkETmGHhnId5QGx8ZQ.

[4] Blomiky. 7.4v 2s 30c 1000mah 7.4wh lipo battery t plug and usb charger cable suitable for hbx 16889 16890a and xlh q903 q901 q902 1/16 scale 2845 brushless racing rc truck / q903 battery 2. https://www.amazon.com/Blomiky-1000mAh-Battery-Charger-Brushless/dp/B086PFSSD9.

[5] DonkeyCar. Donkeycar forums. https://docs.donkeycar.com.

[6] DriverUS. Usa-future-engineers—driverus. https://github.com/Utcassyxz/USA-Future-Engineers---DriverUS/tree/main, 2024.

[7] Raspberry Pi Foundation. Raspberry pi 4 model b. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/.

[8] Raspberry Pi Foundation. Raspberry pi camera module v2 150 degrees wild angle. https://www.raspberrypi.com/documentation/accessories/camera.html.

[9] @DriverUS future engineers. Driverus. urlhttps://www.youtube.com/@DriverUS-future-engineers, 2024.

[10] Google. Tensorflow. https://www.tensorflow.org.

[11] Tim Kosse. Filezilla the free ftp solution. https://filezilla-project.org.

[12] LEGO. Lego materials. https://simple.wikipedia.org/wiki/Lego#:~:text=LEGO%20bricks%20are%20colorful%20plastic,building%20toy%20in%20the%20world.

[13] Miady. Portable charger 5000mah, 3.45oz lightweight power bank, 5v/2.4a output 5v/2a input battery pack charger. https://www.amazon.com/Miady-Portable-Charger-5000mAh-Lightweight/dp/B083VRD7CX?th=1.

[14] WitMotion. Wt901 high-accuracy 3-axis acceleration+gyroscope+tilt angle +magnetometer with kalman filtering, 9-axis mpu9250 usb 200hz imu raspberry pi ahrs module. https://www.amazon.com/Accelerometer-Acceleration-Gyroscope-Electronic-Magnetometer/dp/B07GBRTB5K.