|  |
| --- |
|  |
| AI Toy Car Driving System  Presented by sahal muhammed k k |
| |  |  |  | | --- | --- | --- | |  |  |  | |

1. **Introduction**

This document provides a comprehensive guide for building and operating an AI-driven toy car. The AI system enables the toy car to navigate autonomously using computer vision and machine learning algorithms. This project is ideal for educational purposes, hobbyists, and developers interested in robotics and AI.

1. **System Requirements**

**Hardware**

* Toy car chassis with motor and steering mechanism
* Raspberry Pi or Arduino board
* Camera module (compatible with Raspberry Pi/Arduino)
* Motor driver (H-Bridge)
* Ultrasonic sensors
* Battery pack
* MicroSD card (for Raspberry Pi)
* Breadboard and jumper wires

**Software**

* Operating System: Raspbian (for Raspberry Pi) or Arduino IDE
* Programming Languages: Python (for Raspberry Pi) or C++ (for Arduino)
* Libraries: OpenCV, TensorFlow, Keras (for Raspberry Pi)
* Other tools: SSH client, VNC viewer

**3. Hardware Components**

**Toy Car Chassis**

The base of the toy car includes the frame, wheels, and motors. Ensure that the chassis can accommodate additional components like the Raspberry Pi/Arduino and sensors.

**Raspberry Pi/Arduino Board**

This serves as the brain of the toy car, processing inputs from sensors and controlling the motors.

**Camera Module**

A camera module captures the surroundings of the car, enabling computer vision capabilities for the AI system.

**Motor Driver (H-Bridge)**

The motor driver controls the speed and direction of the car's motors based on signals from the Raspberry Pi/Arduino.

**Ultrasonic Sensors**

Ultrasonic sensors detect obstacles and help the car navigate safely.

**Power Supply**

A battery pack powers the entire system. Ensure it has enough capacity to run all components for an extended period.

**4. Software Components**

**Operating System and Development Environment**

* Install Raspbian OS on the Raspberry Pi.
* For Arduino, install the Arduino IDE on your computer.

**Libraries and Dependencies**

For Raspberry Pi:

* OpenCV: Used for image processing and computer vision tasks.
* TensorFlow and Keras: Libraries for building and training machine learning models.

For Arduino:

* Arduino libraries for motor control and sensor interfacing.

**5. Setup Instructions**

**Hardware Assembly**

1. **Chassis Assembly**: Assemble the toy car chassis, attach the motors, and ensure wheels are securely fitted.
2. **Mount the Camera**: Attach the camera module to the front of the car, ensuring it has a clear view of the surroundings.
3. **Connect Raspberry Pi/Arduino**: Secure the Raspberry Pi/Arduino board onto the chassis.
4. **Wiring**: Connect the motors to the motor driver, and then connect the motor driver to the Raspberry Pi/Arduino. Attach the ultrasonic sensors to the front and sides of the car.
5. **Power**: Connect the battery pack to power the system. Ensure correct voltage levels to avoid damage.

**Software Setup**

1. **Install OS and Libraries**: Install Raspbian on the Raspberry Pi, and the necessary libraries (OpenCV, TensorFlow, Keras). For Arduino, set up the IDE and required libraries.
2. **Code Deployment**: Write or upload the AI control code to the Raspberry Pi/Arduino. Ensure the code includes functions for image processing, obstacle detection, and motor control.

**6. Training the AI Model**

**Data Collection**

* Collect images and sensor data while manually driving the car in various environments.
* Label the collected data to indicate the correct steering angles and speeds.

**Model Training**

* Use the labeled data to train a neural network model. Utilize libraries such as TensorFlow and Keras.
* Train the model on a separate computer with a powerful GPU if necessary, and then transfer the trained model to the Raspberry Pi.

**7. Testing and Validation**

1. **Initial Tests**: Run initial tests in a controlled environment to ensure the car responds correctly to commands.
2. **Field Tests**: Test the car in various real-world scenarios, adjusting the model and code as needed based on performance.

**8. Troubleshooting**

* **Connectivity Issues**: Ensure all wires are securely connected. Check for loose connections.
* **Power Issues**: Verify the battery pack is fully charged and supplying correct voltage.
* **Software Bugs**: Debug the code by checking logs and using print statements to track execution.

**9. Future Enhancements**

* **Advanced Sensors**: Integrate more advanced sensors like LIDAR for improved obstacle detection.
* **Improved Algorithms**: Develop more sophisticated algorithms for better path planning and decision making.
* **Autonomous Navigation**: Implement GPS-based navigation for outdoor environments.