International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 10 | Oct – 2025 DOI: 10.55041/ISJEM05115
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Design and Fabrication of Internet Controlled Transport Car

Dr. K. Ankamma Rao, Professor, Department of Mechanical Engineering, Mahatma Gandhi Institute of Technology, Hyderabad. kankamma_mct@mgit.ac.in

D. Kareemulla, Student of Mechanical (MCT) Engineering Dept., MGIT, Hyd. Dudekulakareemulla271@gmail.com

G. Abhinay, Student of Mechanical (MCT) Engineering Dept., MGIT, Hyd. abhinaysachin.d7b@gmail.com

K. Jathin, Student of Mechanical (MCT) Engineering Dept., MGIT, Hyd. jathink0292@gmail.com

P. Tapan sai, Student of Mechanical (MCT) Engineering Dept., MGIT, Hyd. Tapansai4299@gmail.com

Abstract:

The emergence of the Internet of Things (IoT) has paved the way for innovative technologies, including internet-controlled transport cars. This is a transport car to deploy and transport important materials such as food to flood affected citizens or arms to military personnel or transport medicine during times like pandemics, where people cannot easily go. Here we can use internet controlled robot vehicles to transport the required materials or goods to the destination from anywhere in the world as long as there is internet connection to the robot and the controller which maybe a computer or a mobile phone. The transport car houses a cabinet which is servo controlled that opens and closes on command and has a sufficient storage area. The car is a robust mini van which can go over small obstacles with ease and has a camera feedback system through which we can see what the robot is seeing and control the robot through it.

The basic components required for building the robot are, a Robust Robot chassis, Raspberry PI 4, Wi-Fi module, Raspberry PI Camera module, Motor Driver Board, 4 wheels, 4 DC Servo motors and a smart phone or a computer to control the car. In the first step we design and simulate the working of the components in Tinker CAD, Blynk or Sketch. Then build a prototype to test the working of the concept.

Keywords: Remote controlled Camera Car, Remote controlled Mini Transport Car, Internet controlled Transport Robot, Unmanned Transport vehicle

I. Introduction

1.1 History of Radio Control (RC) Cars

The remote-controlled (RC) cars date back to the mid-20th century and have evolved significantly over the years. Here's a brief overview of the history of RC cars:

Fig 1.1 RC Car

1. Development of Radio Control (1960s):

- The 1960s saw a significant advancement in RC technology with the introduction of radio control systems.
- These systems allowed for more freedom of movement as compared to the tethered models. Hobbyists began building and racing these radio-controlled cars for recreational purposes.

© 2025, ISJEM (All Rights Reserved)

2. Advancements in Technology (2000s-Present):

- The 2000s and beyond have witnessed continuous advancements in RC technology. Brushless electric motors, LiPo batteries, and sophisticated radio control systems have significantly improved the speed, efficiency, and handling of RC cars.
- The introduction of ready-to-fly (RTF) and bind-and-drive (BND) models has made the hobby more accessible to beginners.

1.2 Working

The working of RC cars involves several components and technologies that work together to control the car's movement. When the operator manipulates the controls on the transmitter, the signals are transmitted to the receiver, which then instructs the ESC and servo to adjust the speed and direction of the RC car accordingly. This coordination of components allows for precise control and maneuverability.

1.3 TYPES OF RC CONTROLLING WAYS

- **Wifi Controlled Car**

To enable wireless communication, a Wi-Fi module, such as the ESP8266 or ESP32, is connected to the Arduino. We used Raspberry pi.

- **Bluetooth Controlled Car**

Bluetooth-controlled RC cars provide a modern and convenient way to operate remote-controlled vehicles. These cars utilize Bluetooth technology to establish a wireless connection between a mobile device, such as a smartphone or tablet, and the RC car itself.

- **Radio Controlled Car**

These are conventional Radio control cars that use radio frequency to control the car.

II. Current Challenges

The current challenges with the existing RC cars is their range and integretability with IoT devices. They have a short range compared to the internet and cannot be remotely controlled from different cities or countries like IoT devices can be.

In transitioning a manual internet-controlled car to semi-automatic with a storage box, several key considerations must be addressed to ensure seamless functionality and user convenience.

Firstly, the manual control system of the car needs to be modified to incorporate semi-automatic features. This involves integrating sensors and actuators that allow for automated control of certain functions, such as acceleration, braking, and steering, while still retaining manual override capabilities. Additionally, connectivity features must be enhanced to enable seamless communication between the car and external control systems via the internet.

Secondly, the installation of a storage box adds another layer of complexity to the design. The storage box should be securely integrated into the car's chassis to ensure stability and safety during operation. Furthermore, mechanisms for opening and closing the storage box need to be implemented, possibly through motorized actuators controlled by the car's onboard systems or remotely via the internet.

To achieve semi-automatic functionality, the car's control system must be programmed to manage various tasks autonomously, such as maintaining a constant speed, following predefined routes, and detecting obstacles to avoid collisions. Advanced algorithms and machine learning techniques may be employed to enhance the car's decision-making capabilities and adapt to different driving conditions effectively.

Moreover, user interfaces need to be developed to facilitate interaction with the semi-automatic car and its storage box. This includes designing intuitive control panels or mobile applications that allow users to input commands, monitor the car's status, and access the contents of the storage box remotely.

III. Methodology

Developing a semi-automatic internet-controlled car using a Raspberry Pi involves a structured methodology that encompasses several stages of planning, implementation, and testing to ensure a successful project outcome. Firstly, the project requires a thorough understanding of the hardware components needed. This includes selecting the appropriate motors for propulsion and steering, sensors for detecting obstacles and other environmental cues, and actuators for controlling these components. The Raspberry Pi serves as the central control unit, responsible for processing sensor data, making decisions, and sending commands to the actuators. The next step involves setting up the hardware components. This includes wiring the motors, sensors, and other peripherals to the GPIO pins of the Raspberry Pi according to their specifications. Additionally, power management systems must be implemented to ensure reliable operation of the car's components.

With the hardware in place, the focus shifts to software development. This involves writing code to interface with the various sensors and actuators, interpret sensor data, and implement control algorithms. Python is a popular programming language for Raspberry Pi projects due to its versatility and ease of use, and libraries such as RPi.GPIO and picamera are commonly used for GPIO control and camera functionality, respectively. For internet connectivity, the Raspberry Pi can be configured to connect to a local Wi-Fi network or utilize a USB dongle for mobile data access. Once connected, the car can communicate with external control systems or user

interfaces via TCP/IP or other networking protocols. Secure communication protocols such as HTTPS or MQTT with TLS encryption may be employed to protect sensitive data and ensure privacy.

In semi-automatic mode, the car's software must be programmed to perform certain tasks autonomously while still allowing manual intervention when necessary. This may include implementing algorithms for obstacle avoidance, lane following, or basic navigation using input from sensors and feedback from the vehicle's onboard systems. User interfaces play a crucial role in interacting with the internet-controlled car. This may involve developing a web-based dashboard or a mobile application that allows users to send commands, receive real-time telemetry data, and monitor the car's status remotely. User-friendly interfaces and intuitive controls enhance the overall user experience and make the car more accessible to a wider audience.

Finally, thorough testing and validation are essential to ensure the reliability, safety, and performance of the internet-controlled car. This involves conducting extensive testing under various conditions to identify and address any potential issues or shortcomings in the hardware or software implementation. Iterative refinement may be necessary to optimize the car's performance and enhance its capabilities.

By following this methodology, a semi-automatic internet-controlled car using a Raspberry Pi can be successfully developed as a compelling project that integrates hardware, software, and networking technologies to create an engaging and interactive experience.

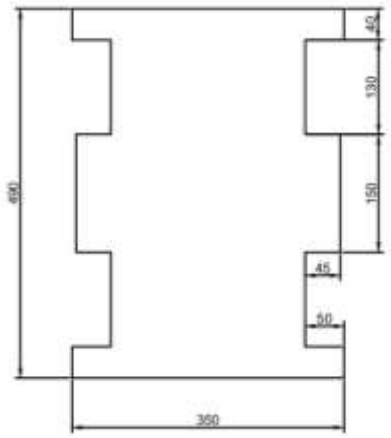
IV. Design Process, Fabrication and Code

4.1 Material and Components required

1. Chassis
2. Microcontroller – Raspberry pi 3 B+, 5V transformer and 12V, 12ah battery.
3. L298 H bridge Motor dirver module boards
4. Wheels
5. Geared DC motors
6. RaspiCam
7. High Torque Servo Motor
8. Ultrasonic Sensor
9. Storage Box
10. Head Lights
11. Bumpers for safety
12. Buzzer

4.2 Chassis and build

A chassis refers to the framework or structure that provides the underlying support for the vehicle's components and body. Made with Polyamide nylon board, a high-performance synthetic polymer, boasts exceptional versatility and durability across various industries. Composed of repeating amide linkages, this thermoplastic material exhibits remarkable strength, resilience, and resistance to wear and tear. Its molecular structure contributes to outstanding chemical and thermal stability, making it suitable for a spectrum of applications.



All dimensions are in mm

Fig 4.1 AUTO CAD Diagram

Fig 4.2 Chemical formula of polyamides

Fusion 360 (3D):

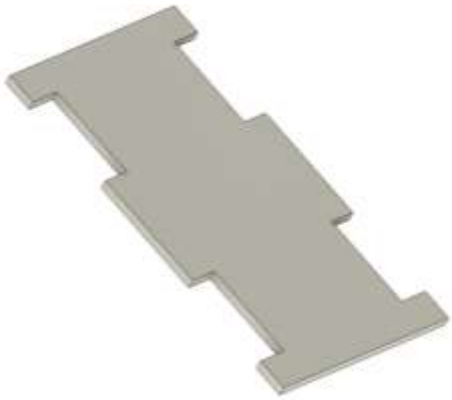


Fig 4.3 Fusion 360 diagram

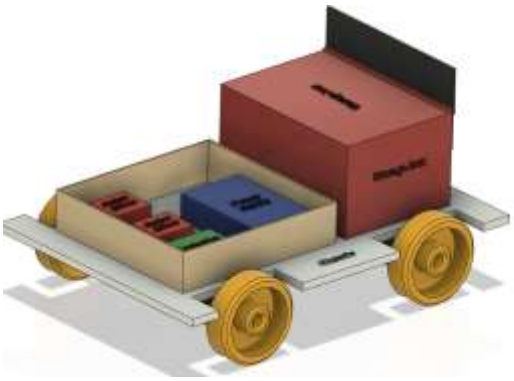


Fig 4.4 Expected output 3D model

4.3 Circuit Diagrams

The pin diagram/description of raspberry pi is really important to make all the right connections with the components for the device to work properly because of the way the code is writte.

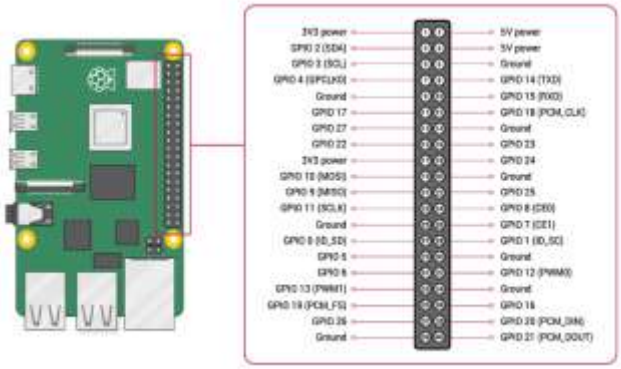


Fig 4.5 Raspberry Pi Pins description

Circuit Diagram of Motors:

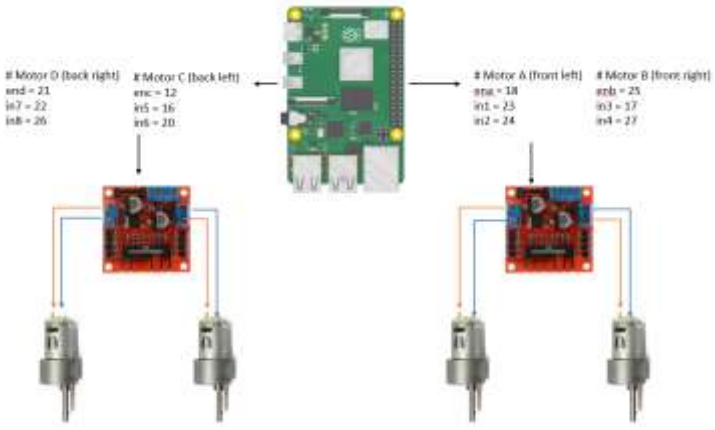
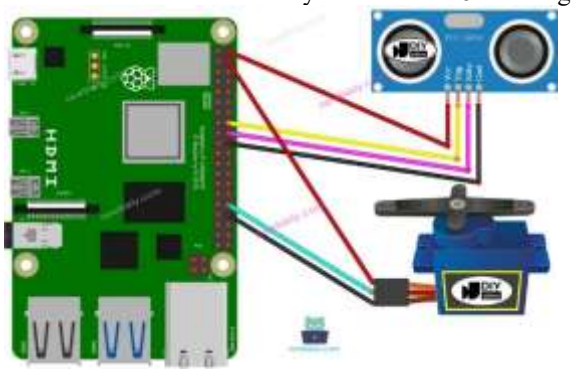


Fig 4.6 Circuit diagram of motors

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 10 | Oct – 2025 DOI: 10.55041/ISJEM05115
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

All the necessary connections to be made and their pins:

- For ultra-sonic sensor, Pins are, TRIG = 4, ECHO = 3.
- For servo motor, pins are, Servo pin = 6.
- For buzzer, pins are, buzzer pin = 5.
- For led headlights, pin = 2.
- The motor layout and pins are in the diagram above.
- The ultrasonic sensor and the servo motor sensor connections are in the figure below.
- All the ground connections and the power input connections are taken from the raspberry pi GPIO.
- To not overload the controller with components, we can also make separate connections to the components positive terminal, directly from the battery with a transformer in between for right voltage and amp.
- Power to raspberry pi is given from a 12V from the 12ah battery converted to 5V through transformer.

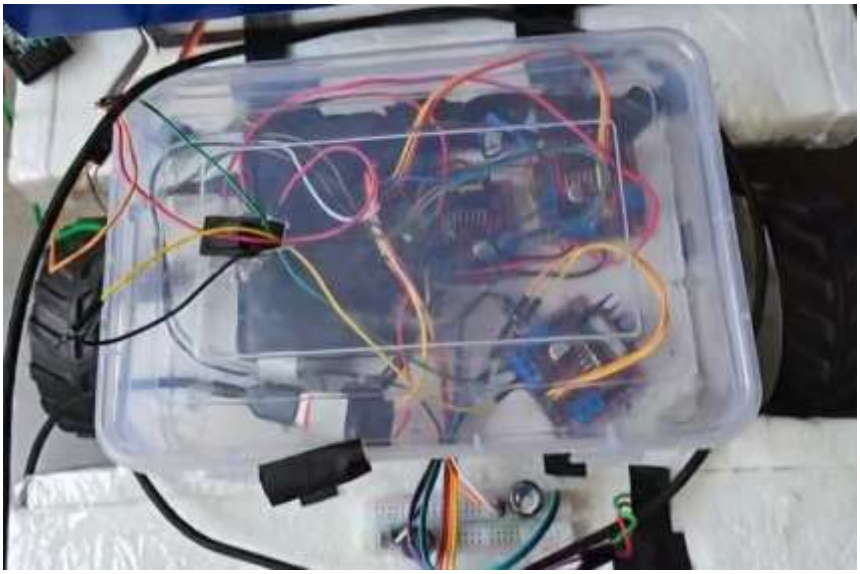


4.7 Pin diagram of lid servo motor and ultrasonic sensor

Fig 4.8 WIP Front view of the car

4.9 Front view of the car with ultrasonic sensor and headlights

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 10 | Oct – 2025 DOI: 10.55041/ISJEM05115
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

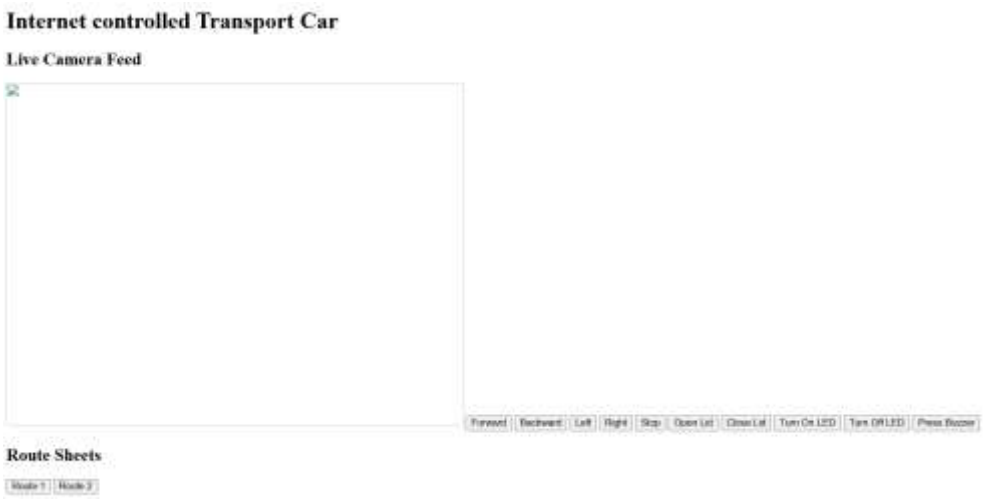


4.10 Control box containing Raspberry pi, camera module, motor controllers, buzzer, led indicator

4.11 Servo controlled storage box containing batter and charger

4.5 Control and Code

An internet-controlled car integrates a combination of hardware and software components to enable remote control and monitoring through the internet. The car is equipped with sensors, microcontrollers, and communication modules such as Wi-Fi or cellular connectivity. These components work together to transmit real-time data, including sensor readings and control commands, between the car and a centralized server or a user interface. Users can interact with the car through a web or mobile application, sending commands such as steering, acceleration, and braking. The server processes these commands and communicates them to the car's onboard system, allowing for immediate and responsive control. Additionally, the system facilitates feedback by streaming live data from the car, providing users with information on the vehicle's speed, location, and environmental conditions. This internet-controlled car project demonstrates the potential of remote vehicle management and monitoring, showcasing the intersection of embedded systems, communication technologies, and user interfaces in the realm of smart and connected vehicles.



4.12 HTML Live control and feed UI page

4.5.1 Code for controlling the Internet Controlled Car

```
from flask import Flask, render_template, request, Response
from flask_socketio import SocketIO
from threading import Thread
import RPi.GPIO as GPIO
import time
from picamera import PiCamera
from io import BytesIO

app = Flask(__name__)
socketio = SocketIO(app)
camera = PiCamera()

# Set GPIO mode
GPIO.setmode(GPIO.BCM)

# Motor A (front left)
ena = 18
in1 = 23
in2 = 24

# Motor B (front right)
enb = 25
in3 = 17
in4 = 27

# Motor C (back left)
enc = 12
in5 = 16
in6 = 20

# Motor D (back right)
end = 21
in7 = 22
in8 = 26

# Set motor pins as output
motor_pins = [ena, in1, in2, enb, in3, in4, enc, in5, in6, end, in7, in8]
GPIO.setup(motor_pins, GPIO.OUT, initial=GPIO.LOW)

# PWM setup
pwm_a = GPIO.PWM(ena, 1000)
pwm_b = GPIO.PWM(enb, 1000)
pwm_c = GPIO.PWM(enc, 1000)
```

```
pwm_d = GPIO.PWM(end, 1000)
pwm_a.start(0)
pwm_b.start(0)
pwm_c.start(0)
pwm_d.start(0)

# Ultrasonic sensor setup
TRIG = 4
ECHO = 3
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

# Servo motor setup
servo_pin = 6
GPIO.setup(servo_pin, GPIO.OUT)
servo = GPIO.PWM(servo_pin, 50)  # PWM frequency: 50 Hz

# Buzzer setup
buzzer_pin = 5
GPIO.setup(buzzer_pin, GPIO.OUT)

# LED setup
led_pin = 2
GPIO.setup(led_pin, GPIO.OUT)
led_state = False

# Motor control functions
def motor_control(pwm, in1_pin, in2_pin):
    GPIO.output(in1_pin, GPIO.HIGH)
    GPIO.output(in2_pin, GPIO.LOW)
    pwm.ChangeDutyCycle(100)

def forward():
    motor_control(pwm_a, in1, in2)
    motor_control(pwm_b, in3, in4)
    motor_control(pwm_c, in5, in6)
    motor_control(pwm_d, in7, in8)

def backward():
    GPIO.output(in1, GPIO.LOW)
    GPIO.output(in2, GPIO.HIGH)
    GPIO.output(in3, GPIO.LOW)
    GPIO.output(in4, GPIO.HIGH)
    GPIO.output(in5, GPIO.LOW)
    GPIO.output(in6, GPIO.HIGH)
    GPIO.output(in7, GPIO.LOW)
    GPIO.output(in8, GPIO.HIGH)
    pwm_a.ChangeDutyCycle(100)
    pwm_b.ChangeDutyCycle(100)
    pwm_c.ChangeDutyCycle(100)
    pwm_d.ChangeDutyCycle(100)

def left():
    GPIO.output(in1, GPIO.LOW)
    GPIO.output(in2, GPIO.HIGH)
    GPIO.output(in3, GPIO.HIGH)
    GPIO.output(in4, GPIO.LOW)
    GPIO.output(in5, GPIO.LOW)
    GPIO.output(in6, GPIO.HIGH)
    GPIO.output(in7, GPIO.HIGH)
    GPIO.output(in8, GPIO.LOW)
    pwm_a.ChangeDutyCycle(100)
    pwm_b.ChangeDutyCycle(100)
    pwm_c.ChangeDutyCycle(100)
    pwm_d.ChangeDutyCycle(100)
```

```
def right():
    GPIO.output(in1, GPIO.HIGH)
    GPIO.output(in2, GPIO.LOW)
    GPIO.output(in3, GPIO.LOW)
    GPIO.output(in4, GPIO.HIGH)
    GPIO.output(in5, GPIO.HIGH)
    GPIO.output(in6, GPIO.LOW)
    GPIO.output(in7, GPIO.LOW)
    GPIO.output(in8, GPIO.HIGH)
    pwm_a.ChangeDutyCycle(100)
    pwm_b.ChangeDutyCycle(100)
    pwm_c.ChangeDutyCycle(100)
    pwm_d.ChangeDutyCycle(100)

def stop():
    GPIO.output(motor_pins, GPIO.LOW)
    pwm_a.ChangeDutyCycle(0)
    pwm_b.ChangeDutyCycle(0)
    pwm_c.ChangeDutyCycle(0)
    pwm_d.ChangeDutyCycle(0)

def open_lid():
    servo.start(7.5)  # Duty cycle for 0 degree position
    time.sleep(1)

def close_lid():
    servo.start(2.5)  # Duty cycle for 90 degree position
    time.sleep(1)

def beep():
    GPIO.output(buzzer_pin, GPIO.HIGH)
    time.sleep(0.5)
    GPIO.output(buzzer_pin, GPIO.LOW)

def ultrasonic_distance():
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)
    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()
    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()
    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150
    distance = round(distance, 2)
    return distance

# Function to check obstacle in a separate thread
def check_obstacle():
    while True:
        distance = ultrasonic_distance()
        if distance < 30:  # Adjust threshold as needed
            stop()
            time.sleep(0.5)  # Adjust sleep time as needed
            # You may add additional actions here if an obstacle is detected
        time.sleep(0.1)  # Adjust sleep time as needed

# Camera streaming function
def generate_video():
    while True:
        # Create a byte buffer for storing the stream
        stream = BytesIO()
        # Capture an image from the camera
        camera.capture(stream, format='jpeg')
```

```
        # Reset the stream position to the beginning
        stream.seek(0)
        # Yield the stream as bytes
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + stream.read() + b'\r\n')

# Flask route for video streaming
@app.route('/video_feed')
def video_feed():
    return Response(generate_video(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

# Flask route for control page
@app.route('/')
def index():
    return render_template('control_page.html')

# WebSocket event for controlling the car and additional functionalities
@socketio.on('control_event')
def control(direction):
    if direction == 'forward':
        forward()
    elif direction == 'backward':
        backward()
    elif direction == 'left':
        left()
    elif direction == 'right':
        right()
    elif direction == 'stop':
        stop()
    elif direction == 'open_lid':
        open_lid()
    elif direction == 'close_lid':
        close_lid()
    elif direction == 'beep':
        beep()
    elif direction == 'route1':
        route1()
    elif direction == 'route2':
        route2()
    elif direction == 'toggle_led':
        toggle_led()
    elif direction == 'turn_on_led':
        GPIO.output(led_pin, GPIO.HIGH)  # Turn LED on
    elif direction == 'turn_off_led':
        GPIO.output(led_pin, GPIO.LOW)   # Turn LED off

# Function to implement route 1
def route1():
    # Start the thread for obstacle detection
    obstacle_thread = Thread(target=check_obstacle)
    obstacle_thread.start()

    # Go straight for 5 seconds
    forward()
    time.sleep(5)
    stop()
    time.sleep(1)

    # Turn right for 2 seconds
    right()
    time.sleep(2)
    stop()
    time.sleep(1)
```

```
    # Go straight for 5 seconds
    forward()
    time.sleep(5)
    stop()
    time.sleep(1)

    # Turn left for 2 seconds
    left()
    time.sleep(2)
    stop()
    time.sleep(1)

    # Go straight for 5 seconds
    forward()
    time.sleep(5)
    stop()

    # Wait for the obstacle detection thread to finish
    obstacle_thread.join()

# Function to implement route 2
def route2():
    # Start the thread for obstacle detection
    obstacle_thread = Thread(target=check_obstacle)
    obstacle_thread.start()

    # Go straight for 5 seconds
    forward()
    time.sleep(5)
    stop()
    time.sleep(1)

    # Turn left for 2 seconds
    left()
    time.sleep(2)
    stop()
    time.sleep(1)

    # Go backward for 3 seconds
    backward()
    time.sleep(3)
    stop()
    time.sleep(1)

    # Turn right for 2 seconds
    right()
    time.sleep(2)
    stop()
    time.sleep(1)

    # Go straight for 5 seconds
    forward()
    time.sleep(5)
    stop()

    # Wait for the obstacle detection thread to finish
    obstacle_thread.join()


# Function to toggle LED
def toggle_led():
    global led_state
    led_state = not led_state
    GPIO.output(led_pin, led_state)
```

```
if __name__ == '__main__':
    # Run the Flask app with SocketIO
    socketio.run(app, debug=True, host='0.0.0.0', use_reloader=False)
```

4.5.2 HTML Code for UI and Camera Feed

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Robot Control</title>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.0.1/socket.io.js"></script>
</head>
<body>
    <h1>Internet controlled Transport Car</h1>

    <!-- Camera feed -->
    <h2>Live Camera Feed</h2>
    <img id="video_feed" src="/video_feed" style="width: 640px; height: 480px;">

    <!-- Control Buttons -->
    <button onclick="control('forward')">Forward</button>
    <button onclick="control('backward')">Backward</button>
    <button onclick="control('left')">Left</button>
    <button onclick="control('right')">Right</button>
    <button onclick="control('stop')">Stop</button>
    <button onclick="control('open_lid')">Open Lid</button>
    <button onclick="control('close_lid')">Close Lid</button>
    <button onclick="control('turn_on_led')">Turn On LED</button>
    <button onclick="control('turn_off_led')">Turn Off LED</button>
    <button onmousedown="pressBuzzer()" onmouseup="releaseBuzzer()">Press Buzzer</button>

    <!-- Route Sheets -->
    <h2>Route Sheets</h2>
    <button onclick="control('route1')">Route 1</button>
    <button onclick="control('route2')">Route 2</button>

    <script>
        var socket = io.connect('http://' + document.domain + ':' + location.port);

        // Function to send control command to the server
        function control(direction) {
            socket.emit('control_event', direction);
        }

        // Function to send command to start the buzzer
        function pressBuzzer() {
            socket.emit('control_event', 'turn_on_buzzer');
        }

        // Function to send command to stop the buzzer
        function releaseBuzzer() {
            socket.emit('control_event', 'turn_off_buzzer');
        }
    </script>
</body>
</html>
```

It was not possible to completely automate or semi automate the car, so there are 2 route sheets coded into the code, that have fixed movement lengths and the car stops when there is an obstacle detected in that path to not veer off the course. But the future plan is to implement both together and use AI and ML to go on correct path, if the robot veers of the path when it avoids an obstacle. But for now, that function is manual.

V. Machine future developments, advantages and disadvantages

Internet-controlled transport mini cars are evolving toward smarter, safer urban mobility. Integration with IoT and 5G will enable real-time remote control and fleet coordination. AI-driven automation will reduce human intervention, boosting efficiency and accessibility. Future use includes smart deliveries, shared micro-mobility, and eco-friendly urban transport. This can not only be used for transport of material, but for human travel as well.

5.1 Advantages

1. Efficiency: By automating certain driving tasks, semi-autonomous storage cars can optimize route planning and navigation, reducing delivery times and increasing overall efficiency. They can operate continuously without the need for driver breaks, resulting in faster and more reliable deliveries.
2. Cost-Effectiveness: With reduced reliance on human drivers, semi-autonomous storage cars can potentially lower labor costs associated with transportation. Additionally, their efficient route planning helps minimize fuel consumption and vehicle wear and tear, leading to cost savings for operators.
3. Improved Safety: Semi-autonomous features such as collision detection and adaptive cruise control enhance the safety of storage cars by reducing the risk of accidents caused by human error. These systems can detect obstacles, pedestrians, and other vehicles, helping to prevent collisions and ensure safer transportation of goods.
4. Flexibility: Semi-autonomous storage cars are versatile and can be adapted to various transportation needs. They can accommodate different types of cargo, from small packages to large items, making them suitable for a wide range of industries and applications.
5. Real-Time Monitoring and Management: Internet connectivity enables operators to remotely monitor the location, status, and performance of semi-autonomous storage cars in real-time. This allows for proactive management, route optimization, and timely interventions in case of any issues or delays.
6. Scalability: Semi-autonomous storage cars can easily scale to meet changing demand levels. Operators can deploy additional vehicles or adjust routes and schedules as needed to accommodate fluctuations in delivery volumes or seasonal variations.
Overall, semi-autonomous storage cars offer numerous advantages, including improved efficiency, cost-effectiveness, safety, and flexibility, making them valuable assets in modern logistics and transportation operations.

5.2 Disadvantages

1. Cybersecurity Risks: Internet-connected cars are susceptible to hacking and cyber-attacks. Malicious actors could potentially gain unauthorized access to a vehicle's systems, compromising safety and privacy. Ensuring robust cybersecurity measures is crucial to mitigate these risks.
2. Privacy Concerns: The continuous connectivity of cars raises privacy concerns. The collection and transmission of data, including location, driving habits, and personal preferences, could be exploited or misused without adequate safeguards in place.
3. Dependency on Network Infrastructure: Internet-controlled cars rely heavily on network infrastructure. In areas with poor or no network coverage, the functionality of connected features may be compromised, affecting navigation, communication, and other connected services.
4. Software Bugs and Glitches: Like any software-driven technology, internet-controlled cars can experience bugs or glitches. These issues may lead to malfunctions, unexpected behavior, or system failures, posing safety risks if not promptly addressed through software updates.
5. Limited Standardization: There is currently a lack of standardized protocols and regulations for internet-controlled cars. This lack of uniformity can result in interoperability issues between different vehicle brands and may slow down the widespread adoption of connected car technologies.

VI. Conclusion

In conclusion, this internet-controlled car is a portable and excellent way to transport things with stealth and precision without the need of human intervention or human driver which saves many lives during pandemics and in military sectors. It is also a very convenient way to transport materials over smaller distances in warehouses and hospitals which can be controlled from anywhere in the world and can be made autonomous like an AGV as long as it is connected to internet. A semi-automated internet-controlled car represents a groundbreaking fusion of automotive technology and remote connectivity, offering a glimpse into the future of transportation. This innovative vehicle integrates traditional automotive systems with advanced automation features and internet connectivity to enable remote control and monitoring capabilities. At its core, the semi-automated internet-controlled car leverages a combination of onboard sensors, actuators, and computing power to facilitate autonomous driving functions. These systems, including cameras, LiDAR, radar, and ultrasonic sensors, enable the vehicle to perceive its environment and make real-time decisions to navigate safely on roads. It is an excellent project which uses all the mechanical, electronic and software sectors to make a very important future use.

6.1 Credit authorship contribution statement

Gottiparthi Abhinay: Team leader, research coordination, project conceptualization, designing, fabrication writing and editing. **Dudekula Kareemulla:** Data collection, literature review, documentation integrity, fabrication. **Kandala Jathin:** Intermediate fabrication, process optimization, quality control. **P. Tapan sai:** Materials sourcing and procurement, inventory management, documentation assistance. **Dr. K. Ankamma Rao** Guidance, mentorship, Automation.

6.2 Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Akshay Bhati, Balendu Teterbay, Ayush Srivastava, “Smartphone Controlled Multipurpose Robot Car”, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 9, Issue 05, May-2020.

[2] H. Rissanen, J. Mahonen, K. Haataja, M. Johansson, J. Mielikainen and P. Toivanen, "Designing and implementing an intelligent Bluetooth enabled robot car," 2009 IFIP International Conference on Wireless and Optical Communications Networks, Cairo, 2009.

[3] S. Mandal, S. K. Saw, S. Maji, V. Das, S. K. Ramakuri and S. Kumar, "Low cost arduino wifi bluetooth integrated path following robotic vehicle with wireless GUI remote control," 2016 International Conference on Information Communication and Embedded Systems (ICICES), Chennai, 2016.

[4] Ritika Pahuja, Narender Kumar, “Android controlled Bluetooth robot using 8051 Microcontroller” ISSN (Online), 2015.

[5] Mrumal.K. Pathak, Javed Khan, “Robot control design using android smartphone”, 2 Feb 2015, ISSN :2347- 5471.

[6] S R Madkar (Assistant Professor), Vipul Mehta, Nitin Bhuwania, Maitri Parida, “Robot controlled car using Wi-Fi module”, ISSN: 2277.

[7] Aniket R. Yeole, Sapana M. Bramhankar, Monali D. Wani, “Smartphone controlled robot using ATMEGA328 Microcontroller”, ISO 3297: 2007.

[8] T. L. Chien, H. Guo, K. L. Su and S. V. Shiau, "Develop a Multiple Interface Based Fire Fighting Robot," 2007 IEEE International Conference on Mechatronics, Changchun, Jilin, 2007, doi:10.1109/ICMECH.2007.4280040.

[9] Vito M. Guardi “Design of a Bluetooth enabled android application for a microcontroller driven robot”, IEEE International Conference on Mechatronics, Changchun, Jilin, May 2014.

[10] M. Kumari, A. Kumar and R. Singhal, "Design and Analysis of IoT-Based Intelligent Robot for Real-Time Monitoring and Control", 2020 International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC), 2020.

[11] S. R. Madkar et al., "Robot Controlled Car Using Wi-Fi Module", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 6, no. 5, 2016. [9]Y. Xiao, N. Sakib, Z. Yue, Y. Wang, J. You, and J. Militky, “Study on the relationship between structure parameters and filtration performance of polypropylene meltblown nonwovens,” AUTEX Res. J., vol. 20, no. 4, pp. 1–6, 2020, doi: 10.2478/aut-2019-0029.

[10] V. K. Kothari, A. Das, and A. Sarkar, “Effect of processing parameters on properties of layered composite needle-punched nonwoven air filters,” Indian J. Fibre Text. Res., vol. 32, no. June, pp. 196–201, 2007.