**PiDriver**

Final Project Report

Author: Abdumalik Shamurov



Subject: Technical Project - TPJ655NAB

Program: Computer Engineering Technology

Instructor: Benjamin Shefler

Date: 4/7/2023

SENECA COLLEGE OF APPLIED ARTS AND TECHNOLOGY

School of Electronics and Mechanical Engineering Technology

1750 Finch Ave East, Toronto, Ontario, M2J 2X5.

Tel. 416-491-5050 Fax 416-490-9839

[www.senecacollege.ca](http://www.senecacollege.ca)

# Table of Contents

[Table of Contents 1](#_Toc131802708)

[Table of Figures 2](#_Toc131802709)

[Table of Tables 2](#_Toc131802710)

[Executive Summary 3](#_Toc131802711)

[Introduction 3](#_Toc131802712)

[Functional Features 4](#_Toc131802713)

[Product Specifications 5](#_Toc131802714)

[Operating Instructions 6](#_Toc131802715)

[Product Design, Implementation, and Operation of the System 7](#_Toc131802716)

[Components Images and Description 7](#_Toc131802717)

[Raspberry Pi 3 B+ 7](#_Toc131802718)

[Raspberry Pi Camera 8](#_Toc131802719)

[Motor Controller 8](#_Toc131802720)

[RC Car 9](#_Toc131802721)

[Gaming Wheel and Pedals 9](#_Toc131802722)

[Power Bank 10](#_Toc131802723)

[Assembling 11](#_Toc131802724)

[Replacing Regular Motor Controller with L298n Motor Driver 11](#_Toc131802725)

[Connecting L298N to Rapspberry Pi GPIOS 13](#_Toc131802726)

[Connecting Raspberry Pi Camera to Rapsberry Pi 14](#_Toc131802727)

[Theory of Operation of the Entire System 16](#_Toc131802728)

[Initialization 16](#_Toc131802729)

[Hardware functionality 17](#_Toc131802730)

[Software functionality 18](#_Toc131802731)

[Code Description 19](#_Toc131802732)

[process() Function 20](#_Toc131802733)

[gen() Function 21](#_Toc131802734)

[PiDriver GUI 22](#_Toc131802735)

[Block-diagram 22](#_Toc131802736)

[Software Diagram 23](#_Toc131802737)

[Maintenance Requirements 24](#_Toc131802738)

[Further Developments 25](#_Toc131802739)

[Project Building Approaches 26](#_Toc131802740)

[Conclusion 27](#_Toc131802741)

[Appendix A – Source Code 28](#_Toc131802742)

[Appendix B – Electrical Schematics 33](#_Toc131802743)

[Raspberry Pi 3 B+ Schematic Diagram 33](#_Toc131802744)

[L298N Motor Controller Schematic Diagram 34](#_Toc131802745)

[Appendix C – Parts List 35](#_Toc131802746)

[Appendix D – References 36](#_Toc131802747)

[Appendix E - Contact Information 37](#_Toc131802748)

# Table of Figures

[Figure 1 - Raspberry Pi 3B+ 7](#_Toc131800890)

[Figure 2 - Raspberry Pi Camera 8](#_Toc131800891)

[Figure 3 – L298N Motor Controller 8](#_Toc131800892)

[Figure 4 - RC Car 9](#_Toc131800893)

[Figure 5 - Gaming Wheel and Pedals 9](#_Toc131800894)

[Figure 6- Power Bank 10](#_Toc131800895)

[Figure 7 - Motor Controller Connection 12](#_Toc131800896)

[Figure 8 - Modified RC Car 15](#_Toc131800897)

[Figure 9 - User Interface 22](#_Toc131800898)

[Figure 10 - Block-Diagram 22](#_Toc131800899)

[Figure 11 - Software Diagram 23](#_Toc131800900)

[Figure 12 - Raspberry Pi 3B+ Schematic Diagram 32](#_Toc131800901)

[Figure 13 - L298N Motor Controller Schematic Diagram 33](#_Toc131800902)

# Table of Tables

[Table 1- Raspberry Pi 3B+ Specifications 7](#_Toc131800806)

[Table 2 - Raspberry Pi Camera Specifications 8](#_Toc131800807)

[Table 3 - L298N Moto Controller Specifications 8](#_Toc131800808)

[Table 4 - RC Car Specifications 9](#_Toc131800809)

[Table 5 - Gaming Wheel and Pedals Specifications 9](#_Toc131800810)

[Table 6 - Power Bank Specifications 10](#_Toc131800811)

# Executive Summary

For a long time, remote control cars have been a well-liked hobby due to the fun and interactive experience they offer. As technology has advanced, people have become increasingly interested in adding more sophisticated features to these cars, like integrating a Raspberry Pi and gaming accessories.

The Raspberry Pi has demonstrated its versatility and capability in various projects, including controlling remote control cars. By incorporating a gaming wheel and pedals, which are commonly used in racing games, controlling a remote control car can be more thrilling and authentic.

Moreover, the real-time video streaming feature of the remote control car can offer users a distinctive viewpoint and improve their overall experience. The project can serve as a safe and educational platform for individuals who are not yet legally permitted to drive, allowing them to experience controlling a car within a secure environment. Additionally, by showcasing the Raspberry Pi's versatility and potential, this project can inspire and motivate others to explore technology's possibilities in various applications.

# Introduction

The integration of a Raspberry Pi, gaming accessories, and real-time video streaming can enhance the experience of controlling a remote control car, but there may be potential challenges such as compatibility issues and programming complexities. However, the knowledge and skills gained from this project could lead to the development of new technology and innovations, potentially improving industries such as engineering, computer science, and technology. Additionally, the impact of this project could extend into the remote control car industry itself, inspiring manufacturers to create more advanced products and creating a new market for enthusiasts. Overall, this project has the potential to be an exciting and worthwhile endeavor that extends beyond entertainment and education.

# Functional Features

1. Remote Control: The car will be controlled using a gaming wheel and pedals, connected to a Raspberry Pi. The user will be able to control the car's movements, such as accelerating, braking, and steering.
2. Real-Time Video Streaming: A camera system will be integrated into the car to capture real-time video, which will be streamed to a webserver hosted on the Raspberry Pi. The video stream will be accessible over the internet and viewable by users.
3. User-Friendly Interface: The interface for controlling the car and accessing the video stream will be user-friendly and easy to use. The gaming wheel and pedals will be configured to provide a seamless experience for the user.
4. Versatile Raspberry Pi Control: The project will demonstrate the versatility and capability of the Raspberry Pi as a controller for various applications, including controlling remote control cars.
5. Robust Hardware Integration: The car components, Raspberry Pi, gaming wheel and pedals, and camera system will be integrated in a way that provides a reliable and stable performance.

The project will aim to provide a fun and interactive experience for users while showcasing the versatility and capability of the Raspberry Pi and advanced hardware integration.

# Product Specifications

1. Remote Control Car: The remote control car should have a maximum speed of around 15-20 mph and be able to handle rough terrain. The car should have a battery life of around 30 minutes to 1 hour.
2. Raspberry Pi: A Raspberry Pi 2 or later model should be used as the controller for the car. The Raspberry Pi should have at least 1GB of RAM and 32GB of storage.
3. Gaming Wheel and Pedals: The gaming wheel and pedals should be compatible with laptop or PC operation system and have adjustable settings for sensitivity and responsiveness.
4. Camera System: The camera system should have a resolution of at least 320p and a field of view of around 120-140 degrees. The camera should be mounted on the car and have the ability to stream video in real-time.
5. Webserver: The webserver should be hosted on the Raspberry Pi and accessible over the Wi-Fi. The webserver should be able to stream the video from the camera in real-time and provide a user-friendly interface for accessing the video and receiving gaming wheel and pedals inputs.

The product specifications are subject to change based on the availability of components and budget constraints. The project will aim to provide a functional remote control car with a user-friendly interface and real-time video streaming capabilities, while incorporating the latest hardware and technology available.

# Operating Instructions

1. Connect a power bank (portable power source) to the Raspberry Pi.
2. Turn on the RC car using the switch on the bottom of the car.
3. Connect your gaming wheel and pedals to a PC, which will be used as the control setup for the remote control car. This PC will also be used for sending and receiving inputs, as well as for streaming video.
4. Ensure that both the Raspberry Pi and the PC are connected to the same wireless network.
5. Open a web browser, preferably Chrome, and type in "raspberypi.local" in the address bar, and then press "Enter".
6. The web server will check if your gaming wheel is compatible. If it is not compatible, a warning message will appear, stating that your device is not supported for this application. If it is compatible, the setup will proceed.
7. Press any button on your gaming controller to start.
8. The camera stream will appear, and you can now control the car:

* Rotating the wheel to turn left or right.
* Pushing the right pedal to move forward, and pushing the left pedal to move backward.
* You can stop the car by pressing the "STOP" button.

# Product Design, Implementation, and Operation of the System

## Components Images and Description

### Raspberry Pi 3 B+

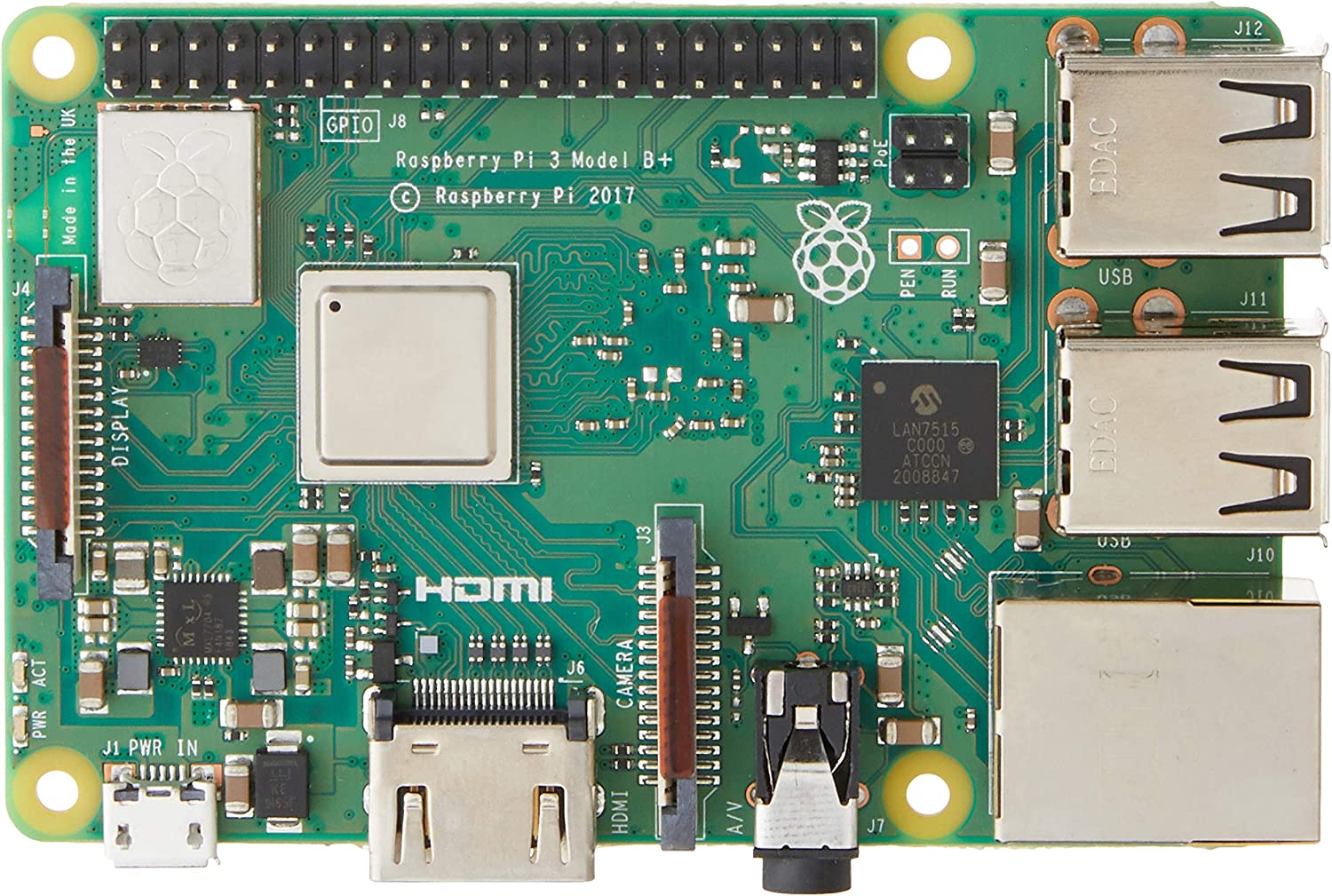


Figure 1 - Raspberry Pi 3B+

Table 1- Raspberry Pi 3B+ Specifications

|  |  |
| --- | --- |
| GPU | Broadcom Videocore-IV |
| Networking | Gigabit Ethernet, 2.4GHz and 5GHz 802.11b/g/n/ac Wi-Fi |
| Storage | Micro-SD |
| GPIO | 40 pin GPIO header |
| Ports | HDMI, 4 x USB 2.0, Ethernet, Camera Serial Interface, Display Serial Interface |
| Item Weight | 50g |
| Item Dimensions | 6cm x 4cm x 2.5cm |
| Platform | Linux |

Table 2 - Raspberry Pi Camera Specifications

|  |  |
| --- | --- |
| Brand | KEYESTUDIO |
| Image Capture Speed | 30 fps |
| Video capture resolution | 1080 |
| Item Weight | 18.14g |
| Item Dimensions | 13.21 x 8.41 x 3.61 cm |

### Raspberry Pi Camera

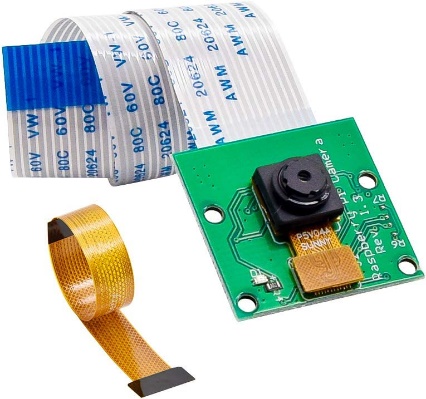


Figure 2 - Raspberry Pi Camera

### Motor Controller

Table 3 - L298N Moto Controller Specifications

|  |  |
| --- | --- |
| Name | EU-L298N MotoDriver |
| Brand | ‎Neuftech |
| Item Weight | 20g |
| Item Dimensions | ‎8.64 x 4.06 x 4.06 cm |

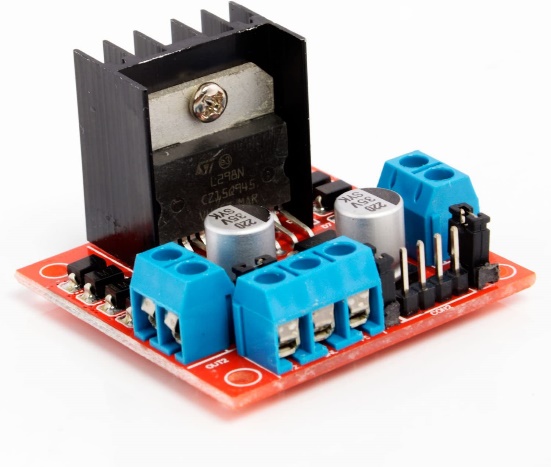


Figure 3 – L298N Motor Controller

### RC Car

Table 4 - RC Car Specifications

|  |  |
| --- | --- |
| Name | RC Buggy Truck |
| Brand | RCROKS |
| Material | PVC |
| Colour | Green |
| Speed | 28km/h |
| Scale | 1/12 scale |
| Item Weight | 1.22 Kilograms |
| Item Dimensions | ‎32 x 20.83 x 14.73 cm |



Figure 4 - RC Car

### Gaming Wheel and Pedals



Figure 5 - Gaming Wheel and Pedals

Table 5 - Gaming Wheel and Pedals Specifications

|  |  |
| --- | --- |
| Name | Ferrari 458 Spider Racing Wheel |
| Manufacturer | Thrustmaster |
| Connectivity Type | USB |
| Item Weight | 4.31 Kilograms |
| Item Dimensions | 39.37 x 34.04 x 32.77 cm |

### Power Bank

Table 6 - Power Bank Specifications

|  |  |
| --- | --- |
| Name | AWTOK Portable Charger |
| Brand | AWTOK |
| Connector type | USB |
| Battery capacity | 10000 Milliamp Hours |
| Item Weight | ‎181 g |
| Item Dimensions | 9.53 x 6.35 x 2.41 cm |



Figure 6- Power Bank

## Assembling

### Replacing Regular Motor Controller with L298n Motor Driver

The L298N is a dual H-bridge motor driver integrated circuit (IC) that is commonly used to control the speed and direction of DC motors and stepper motors. It is a popular choice for robotics projects and other applications that require precise motor control.

1. Remove the existing motor controller: Depending on the type of RC car, the motor controller may be soldered onto the circuit board or connected with wires. You will need to disconnect it from the circuit board or remove the wires from the motor controller.
2. Prepare the L298N motor driver: The L298N motor driver is a small electronic board that can be purchased online or from electronics stores. You will need to prepare the board by soldering the header pins onto it, so it can be connected to the RC car's circuit board.
3. Connect the L298N motor driver to the RC car's circuit board: Once you have prepared the L298N motor driver, you will need to connect it to the RC car's circuit board. This can be done by soldering the header pins onto the appropriate points on the circuit board, or by connecting wires between the two boards.
4. Connect the motors to the L298N motor driver: The L298N motor driver can control two motors. You will need to connect the wires from the motors to the appropriate terminals on the L298N motor driver.
5. Connect power to the L298N motor driver: The L298N motor driver requires a power supply to operate. You can use the same battery pack that powers the RC car or a separate power supply. You will need to connect the positive and negative terminals of the power supply to the appropriate points on the L298N motor driver.
6. Program the L298N motor driver: The L298N motor driver can be controlled using a microcontroller or a computer. You will need to program the driver to control the speed and direction of the motors.
7. Test the RC car: Once you have completed all the above steps, you can test the RC car to ensure that the new motor controller is working properly. If everything is working as expected, your RC car should be ready to run with the new L298N motor driver.

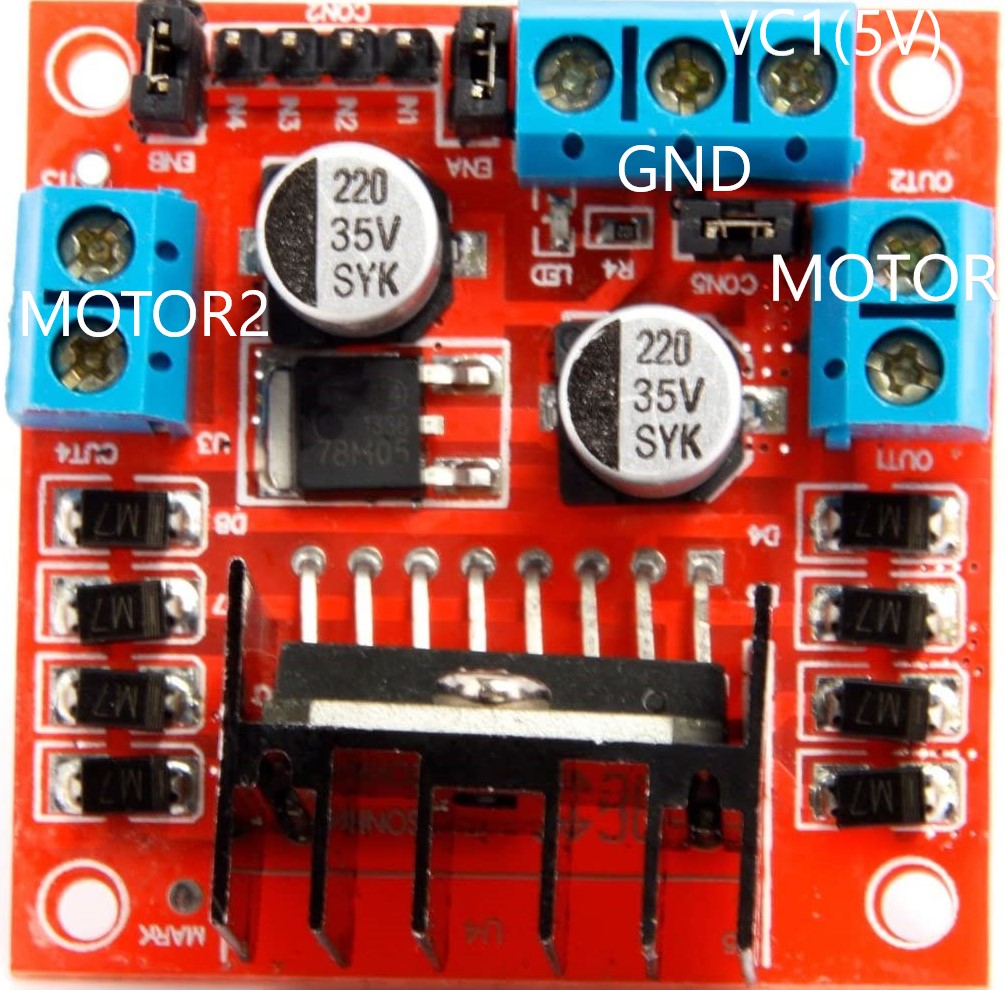


Figure 7 - Motor Controller Connection

### Connecting L298N to Rapspberry Pi GPIOS

1. Identify the GPIO pins: On a Raspberry Pi, the GPIO pins are numbered from 1 to 40. You will need to identify which pins you will be using to connect the L298N motor driver.
2. Connect the power supply: Connect the power supply to the L298N motor driver. You can use the 5V and GND pins on the Raspberry Pi's GPIO header to power the motor driver, or you can use a separate power supply.
3. Connect the enable pins: The L298N motor driver has two enable pins that control the speed of the motors. Connect the ENA pin to a GPIO pin on the Raspberry Pi (e.g. GPIO18), and connect the ENB pin to another GPIO pin (e.g. GPIO19).
4. Connect the direction pins: The L298N motor driver has four direction pins, two for each motor. Connect the IN1 and IN2 pins to one motor and the IN3 and IN4 pins to the other motor. You can use any available GPIO pins for these connections, but you will need to specify which pins you used in your code.
5. Write the code: You will need to write some code to control the L298N motor driver. There are many programming languages you can use, but Python is a popular choice for Raspberry Pi projects. You can use the RPi.GPIO library in Python to control the GPIO pins.

### Connecting Raspberry Pi Camera to Rapsberry Pi

1. Turn off the Raspberry Pi: Before connecting the camera, it is recommended to turn off the Raspberry Pi to avoid damaging the camera or the Pi.
2. Enable the camera interface: By default, the camera interface is disabled on the Raspberry Pi. To enable it, open the terminal and type sudo raspi-config. Select Interfacing Options, then select Camera and follow the prompts to enable the camera interface.
3. Connect the camera ribbon cable: The camera module connects to the Raspberry Pi using a ribbon cable. Gently lift up the tabs on the camera connector on the Raspberry Pi and insert the ribbon cable with the metal contacts facing towards the HDMI port. Push down the tabs to secure the ribbon cable in place.
4. Power on the Raspberry Pi: Once the ribbon cable is connected, power on the Raspberry Pi. The camera should be detected automatically.
5. Test the camera: To test the camera, open the terminal and enter the command raspistill -o test.jpg. This command will capture an image and save it as test.jpg. If the camera is working properly, you should see the image displayed on the screen momentarily.
6. Use the camera in your projects: Now that the camera is connected and working, you can use it in your projects using Python or other programming languages. There are several libraries available to control the camera, including picamera and opencv. You can refer to the official Raspberry Pi documentation for more information on using the camera module in your projects.

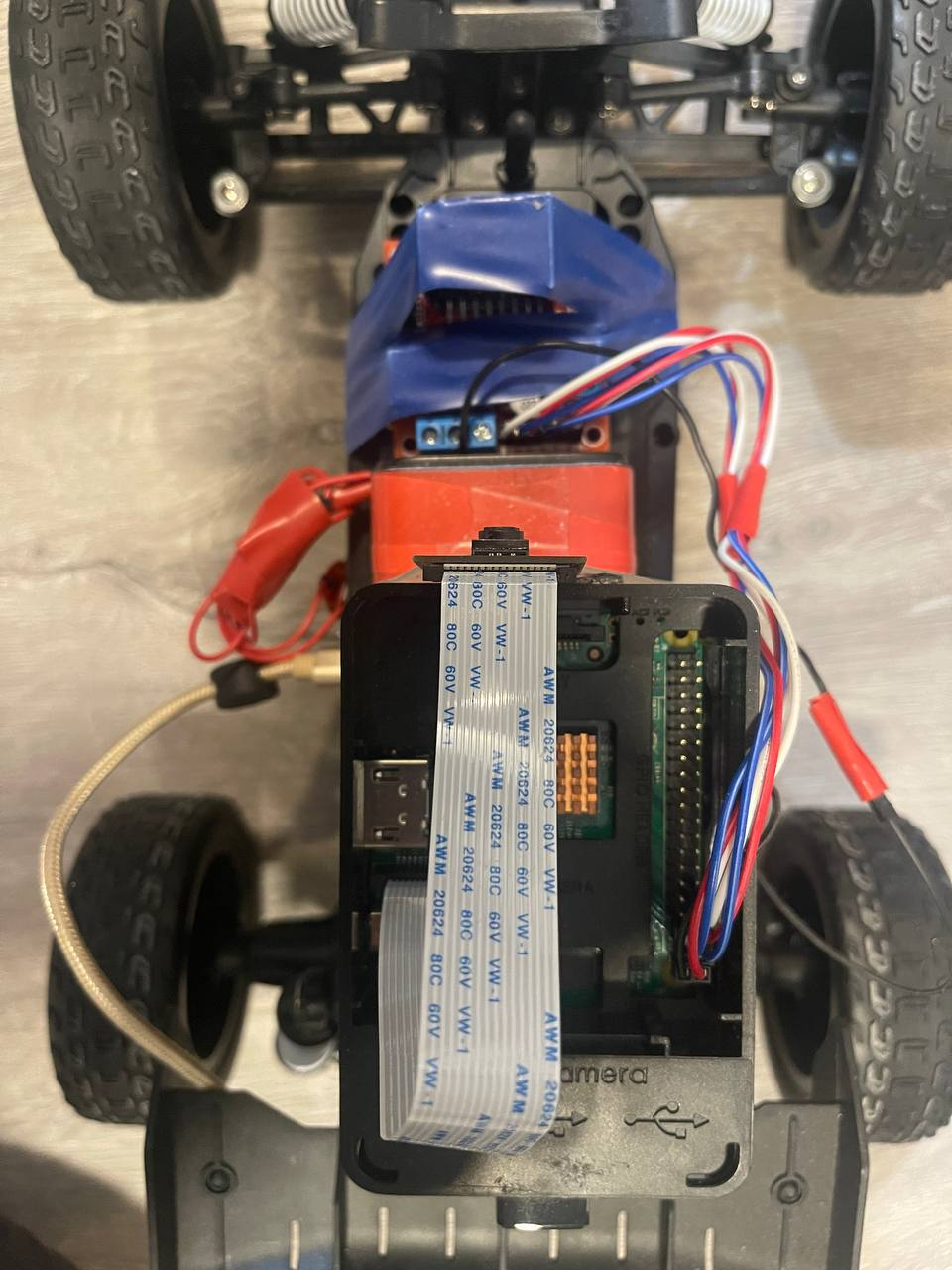


Figure 8 - Modified RC Car

## Theory of Operation of the Entire System

### Initialization

The Remote Control Car project aims to create a user-friendly and versatile system for remotely controlling a car using a Raspberry Pi. The project will involve integrating a gaming wheel and pedals with the Raspberry Pi to provide precise control of the car's movements. Additionally, a camera system will be installed in the car to capture real-time video, which will be streamed over the internet to a webserver hosted on the Raspberry Pi.

To ensure reliable and stable performance, the car components, Raspberry Pi, gaming wheel and pedals, and camera system will be integrated using robust hardware integration techniques. This will involve carefully designing and assembling the various components to ensure compatibility and minimize potential issues.

The project will demonstrate the versatility and capability of the Raspberry Pi as a controller for various applications, including remote control cars. To achieve this, the system will be designed to be easily modifiable and adaptable, allowing for future upgrades and improvements.

Finally, the user interface will be designed to be intuitive and easy to use. The gaming wheel and pedals will be configured to provide a seamless experience for the user, and the video stream will be accessible over the wi-fi, making it easy for users to access the car's real-time video feed from large distance.

### Hardware functionality

Hardware functionality for the Remote Control Car project can be broken down into several components:

Raspberry Pi: The Raspberry Pi will act as the brain of the system, receiving input from the gaming wheel and pedals and sending commands to the car's motors. It will also handle video processing and streaming.

Gaming wheel and pedals: The gaming wheel and pedals will allow the user to control the car's movements, such as accelerating, braking, and steering. These inputs will be sent to the Raspberry Pi, which will translate them into commands for the car's motors.

Car components: The car's components, including the motors, wheels, and chassis, will be controlled by the Raspberry Pi. The Raspberry Pi will send signals to the motors to control the car's movements based on the user's inputs.

Camera system: The camera system will capture real-time video of the car's surroundings and stream it over the internet to a webserver hosted on the Raspberry Pi. The camera system will be integrated with the Raspberry Pi to enable real-time video processing and streaming.

Power supply: The system will require a power supply to operate, which may include a battery pack or a power supply unit (PSU).

Cables and connectors: Various cables and connectors will be required to connect the various components, including the gaming wheel and pedals, Raspberry Pi, camera system, and power supply. These will need to be carefully selected and configured to ensure compatibility and reliable performance.

Overall, the hardware functionality of the Remote Control Car project will involve careful integration and configuration of these components to create a reliable and stable system that provides a seamless user experience.

### Software functionality

The Remote Control Car project requires software functionality to enable communication between the hardware components and to facilitate control of the car and video streaming. Some of the key software functions required for the project are:

Operating system: The Raspberry Pi will require an operating system (OS) to manage the system's resources and to provide a platform for running applications. Popular OS options for the Raspberry Pi include Raspbian, Ubuntu, and Arch Linux.

Input processing: The software will need to receive and process input from the gaming wheel and pedals. This will involve setting up a communication protocol between the gaming wheel and pedals and the Raspberry Pi, as well as software to translate the input signals into commands for the car's motors.

Motor control: The software will need to control the car's motors to enable movements such as acceleration, braking, and steering. This will involve configuring the GPIO (General Purpose Input/Output) pins on the Raspberry Pi to communicate with the motor driver and to send signals to the motors.

Video processing and streaming: The software will need to process the video feed captured by the camera system and stream it over the internet to a webserver hosted on the Raspberry Pi. This will involve configuring the camera system and setting up a video streaming protocol, such as RTSP (Real-Time Streaming Protocol), to enable real-time video streaming.

User interface: The software will require a user interface to allow the user to control the car and view the real-time video stream. This may involve developing a web application that can be accessed over the internet or a graphical user interface (GUI) that runs on the Raspberry Pi.

Debugging and error handling: The software will need to include debugging and error handling functionality to diagnose and fix any issues that arise during operation.

Overall, the software functionality of the Remote Control Car project is critical to enable communication between the hardware components and to provide a user-friendly and seamless experience for the user.

### Code Description

This code is a Python script that runs a web server using Flask on a Raspberry Pi. It imports Flask, picamera, and RPi.GPIO libraries. The RPi.GPIO library is used to control the GPIO pins of the Raspberry Pi. The picamera library is used to capture and stream video from a camera attached to the Raspberry Pi.

The code sets up the GPIO pins for controlling a motor driver, specifically an L298N motor driver, which is connected to a DC motor. The motor driver is connected to the Raspberry Pi GPIO pins 21, 26, 19, 20, 16, and 12. The code sets the GPIO pins for the motor driver as outputs and initializes the PWM channels for controlling the speed of the motor.

The code also sets up two routes for the web server:

The first route ('/') renders the HTML template 'index.html' when a user visits the website.

The second route ('/process') handles POST requests with data received from the website. The code reads the data from the request and interprets it as input values for controlling the motor.

The code checks the input data and based on the values received, it sets the direction and speed of the motor. Additionally, the code sets up a video streaming endpoint ('/video\_feed') using the camera attached to the Raspberry Pi.

Finally, the code starts the Flask application, listening for requests on the Raspberry Pi's IP address at port 8080. When running in debug mode, any errors in the application will be displayed in the console.

### process() Function

The process() function is a Flask route function that handles HTTP POST requests from the client-side. It retrieves JSON data from the POST request using the request.get\_json() method and then extracts two values from the JSON data using data[list(data.keys())[0]] and data[list(data.keys())[1]].

The function then checks the values of num and value to determine the user's input. If num is 1 and value is less than 1, the function will check if value is less than -0.3 or greater than 0.3. If this is true, it will set the motor to turn left or right by setting the appropriate GPIO pins to high or low. If value is between -0.3 and 0.3, it will stop the motor by setting both GPIO pins to low.

If num is 8 and value is greater than 0, the function will set the motor to move forward at a speed proportional to value. It will set the GPIO pins for forward and backward movement, set the speed using PWM, and print "f" to the console.

If num is 8 and value is 0, the function will stop the motor by setting the GPIO pin for forward movement to low and print "motor stop" to the console.

If num is 7 and value is greater than 0, the function will set the motor to move backward at a speed proportional to value. It will set the GPIO pins for forward and backward movement, set the speed using PWM, and print "back" to the console.

If num is 7 and value is 0, the function will stop the motor by setting the GPIO pin for backward movement to low and print "motor stop" to the console.

If num is 6, the function will check the previous direction of the motor and stop it by setting both GPIO pins to low. It will print "stop motor forward" or "stop motor back" to the console accordingly.

Finally, the function returns a JSON response indicating a success message and the data received from the client.

### gen() Function

The gen() function is a generator function that uses the picamera library to capture video frames from the Raspberry Pi camera module and convert them into JPEG format. It continuously captures frames and yields them as a byte string in multipart format. The yielded frame is read by the Response object in the video\_feed() function and sent to the browser as a video stream in a format that can be displayed on a web page. This is used to display live video feed on the web page when the user accesses the video\_feed route.

## PiDriver GUI

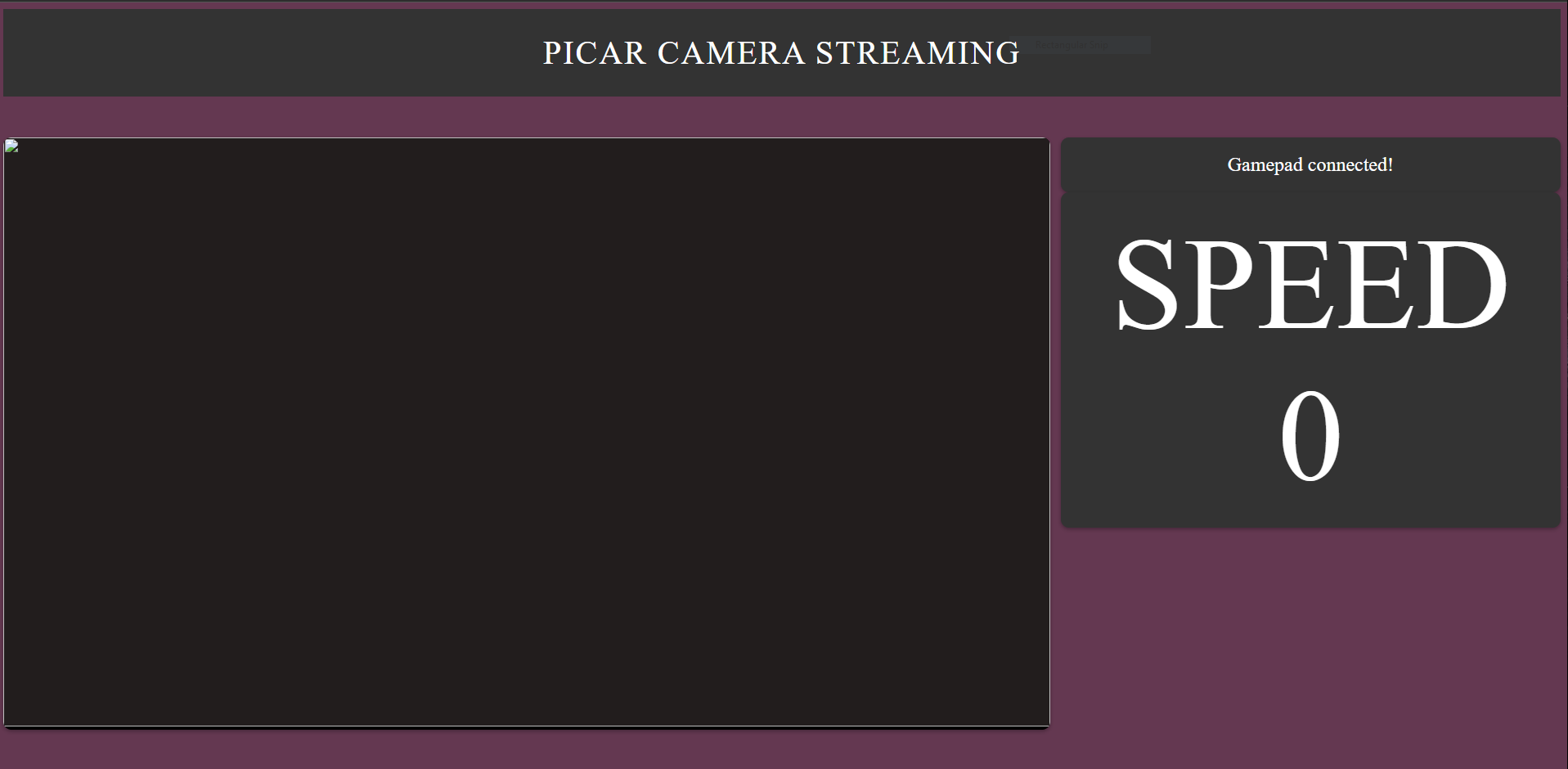


Figure 9 - User Interface

## Block-diagram

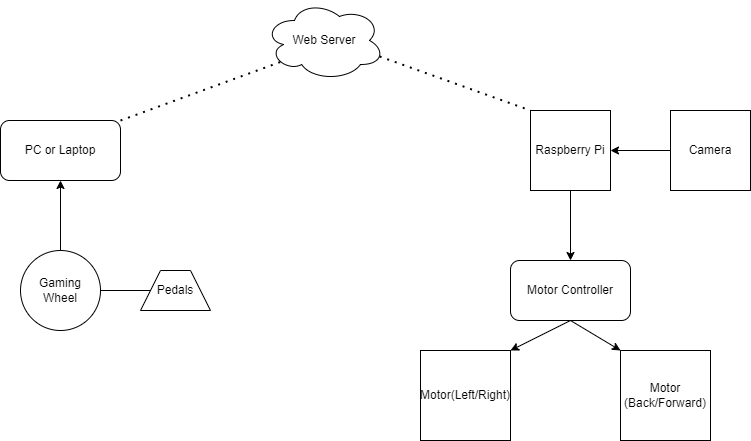


Figure 10 - Block-Diagram

## Software Diagram

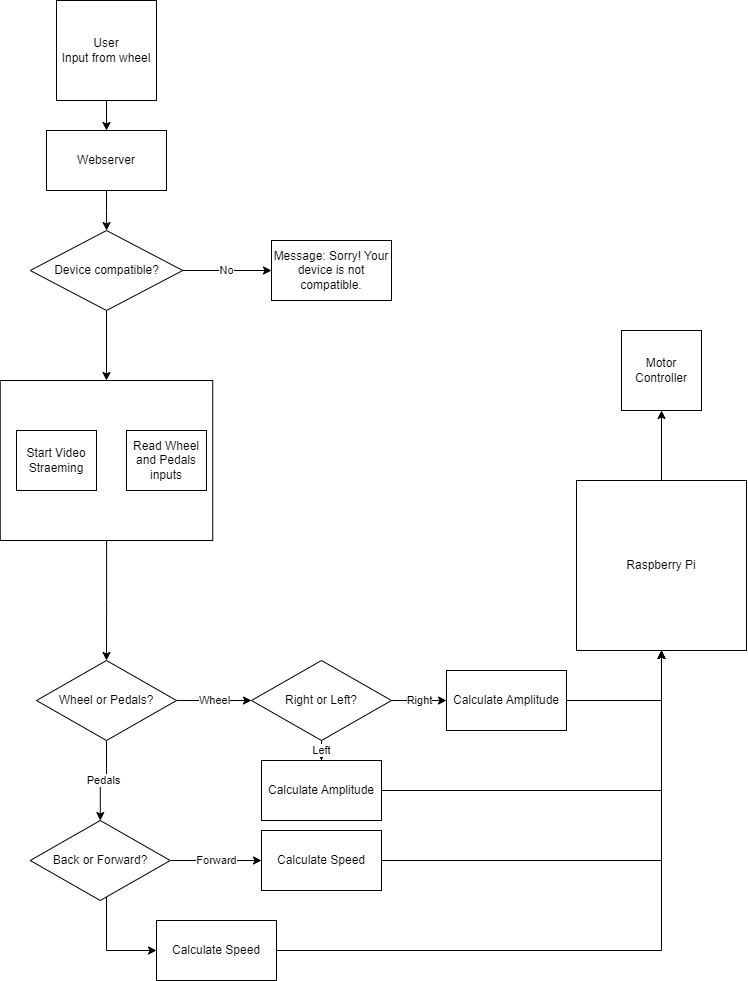


Figure 11 - Software Diagram

# Maintenance Requirements

The Remote Control Car project will require regular maintenance to ensure reliable and optimal performance. Some of the key maintenance requirements for the project include:

Regular hardware checks: The hardware components, including the car's motors, wheels, chassis, and the Raspberry Pi, will need to be regularly checked for any signs of wear and tear. Any damaged or faulty components should be repaired or replaced as soon as possible to avoid further damage.

Software updates: The software components of the system, including the operating system and any applications, will require regular updates to ensure that they are up to date with the latest security patches and bug fixes. Regular software updates will also help to improve the system's performance and stability.

Battery maintenance: If the system is powered by a battery pack, the batteries will need to be regularly checked and maintained. This may involve checking the battery voltage, ensuring that the batteries are properly charged, and replacing any damaged or worn batteries.

Camera cleaning: The camera system will need to be regularly cleaned to ensure that the video feed is clear and free from any debris or dust. Cleaning the camera lens and housing can help to maintain the video quality and prevent damage to the camera system.

Cable management: The cables and connectors used to connect the various hardware components will need to be regularly checked to ensure that they are properly connected and in good condition. Any damaged or worn cables should be replaced to avoid connectivity issues.

Debugging and troubleshooting: Inevitably, issues and errors may arise during operation. As such, the system will require regular debugging and troubleshooting to diagnose and resolve any issues that arise. This may involve reviewing logs, running diagnostics, and troubleshooting hardware and software components.

By regularly maintaining the system and addressing any issues as soon as they arise, the Remote Control Car project can provide a reliable and optimal performance, ensuring an enjoyable and seamless user experience.

# Further Developments

There are several potential future developments for the PiDriver project that could enhance its functionality and improve the user experience.

One possible future development is to incorporate additional sensors into the car, such as ultrasonic sensors or infrared sensors. These sensors could be used to detect obstacles or other objects in the car's path, which would allow for more accurate and safer remote control.

Another future development could involve implementing machine learning algorithms to enable autonomous control of the car. This would require the integration of advanced sensors and machine learning software, but it could provide a significant upgrade in the car's capabilities.

Additionally, improving the video streaming quality and reducing latency could also enhance the user experience. This could be achieved by upgrading the camera system and optimizing the streaming software used.

Another potential development is to create a mobile app that could be used to control the car remotely. The app could be designed to work with the gaming wheel and pedals or other input devices and would allow users to control the car from their mobile devices.

Finally, incorporating additional features, such as GPS tracking or voice control, could further enhance the car's capabilities and user experience.

Overall, there are several potential future developments for the Remote Control Car project, and the possibilities are endless. The project's versatility and adaptability will allow for continuous upgrades and improvements, making it an exciting and dynamic project to work on.

# Project Building Approaches

The project building approaches will follow an iterative process, where the components will be integrated and tested, and improvements will be made based on the results of the testing. The project will aim to provide a high-quality and reliable remote control car with real-time video streaming capabilities, while also ensuring a positive user experience.

* Hardware Integration: The first step in building the remote control car will be to integrate the hardware components, such as the car, Raspberry Pi, gaming wheel and pedals, and camera system. This will involve connecting the components to the Raspberry Pi, configuring the gaming wheel and pedals, and mounting the camera system on the car.
* Software Configuration: The next step will be to configure the software on the Raspberry Pi. This will involve setting up the webserver, installing the necessary libraries and dependencies, and writing the code to control the car and stream the video.
* Testing and Debugging: Once the hardware and software components have been integrated and configured, the project will undergo extensive testing and debugging to ensure a reliable and stable performance. This will involve testing the car's movements, the video streaming capabilities, and the user interface.
* User Experience Optimization: Finally, the project will focus on optimizing the user experience, such as improving the user interface, enhancing the video quality, and making the car more responsive to user input.

# Conclusion

This project aims to design and build a remote control car that is controlled by a Raspberry Pi using a gaming wheel and pedals. The car will also have a camera that will capture real-time video and stream it to a webserver hosted on the Raspberry Pi. The project will involve assembling the car components, connecting the Raspberry Pi, integrating the gaming wheel and pedals, and setting up the video streaming system. The expected outcomes include a functional remote control car, real-time video streaming, and a user-friendly interface. The project is estimated to take 4-6 weeks to complete and has an estimated budget of CA$325.66.

# Appendix A – Source Code

from flask import Flask

from flask import (

    jsonify,

    render\_template,

    request,

    Response

)

import picamera

import RPi.GPIO as GPIO

import time

import io

# Set up GPIO pins

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

forward =21

back = 26

speed = 19

left = 20

right = 16

amplitude = 12

GPIO.setup(forward, GPIO.OUT)

GPIO.setup(back, GPIO.OUT)

GPIO.setup(speed, GPIO.OUT)

GPIO.setup(right, GPIO.OUT)

GPIO.setup(left, GPIO.OUT)

GPIO.setup(amplitude, GPIO.OUT)

# Set up PWM pins for motor control

pi\_pwm1 = GPIO.PWM(speed, 100)

pi\_pwm1.start(100)

pi\_pwm2 = GPIO.PWM(amplitude, 100)

pi\_pwm2.start(100)

# Create Flask app

app = Flask(\_\_name\_\_)

# Set up route for index page

@app.route('/')

def index():

    return render\_template('index.html')

# Set up route for processing incoming data from joystick

@app.route('/process', methods=['POST'])

def process():

    data = request.get\_json()  # Get JSON data from request

    num = (data[list(data.keys())[0]])  # Extract number value from JSON data

    value = (data[list(data.keys())[1]])  # Extract value from JSON data

    global last  # Declare last variable as a global variable to track previous motor direction

    # If joystick is pushed to the left or right

    if value < 1 and num == 1:

        if (value < -0.3 or value > 0.3):

            # If joystick is pushed to the left

            if value < 0:

                GPIO.output(right, GPIO.LOW)  # Turn right motor off

                GPIO.output(left, GPIO.HIGH)  # Turn left motor on

            # If joystick is pushed to the right

            if value > 0.14:

                GPIO.output(right, GPIO.HIGH)  # Turn right motor on

                GPIO.output(left, GPIO.LOW)  # Turn left motor off

        # If joystick is centered

        else:

            GPIO.output(right, GPIO.LOW)  # Turn right motor off

            GPIO.output(left, GPIO.LOW)  # Turn left motor off

    # If joystick is pushed forward

    if num == 8 and value > 0:

        last = "forward"  # Update last variable to track previous motor direction

        speed = value  \* 100  # Convert joystick value to duty cycle percentage for motor control

        pi\_pwm1.ChangeDutyCycle(speed)  # Set motor speed using PWM

        GPIO.output(forward, GPIO.HIGH)  # Turn forward motor on

        GPIO.output(back, GPIO.LOW)  # Turn backward motor off

    # If joystick is released

    if num == 8 and value ==0:

        GPIO.output(forward, GPIO.LOW)  # Turn forward motor off

    # If joystick is pushed backward

    if num == 7 and value > 0:

        last = "back"  # Update last variable to track previous motor direction

        speed = value \* 100  # Convert joystick value to duty cycle percentage for motor control

        pi\_pwm1.ChangeDutyCycle(speed) # Set the speed of the motor using PWM

        GPIO.output(forward, GPIO.LOW) # Turn forward motor off

        GPIO.output(back, GPIO.HIGH) # Turn back motor on

    # If joystick is in neutral position

    if num == 7 and value == 0:

        GPIO.output(back, GPIO.LOW) # Turn back motor off

    # If joystick is pushed forward

    if num == 7 and value < 0:

        last = "forward" # Update last variable to track previous motor direction

        speed = abs(value) \* 100 # Convert joystick value to speed percentage

        pi\_pwm1.ChangeDutyCycle(speed) # Set the speed of the motor using PWM

        GPIO.output(back, GPIO.LOW) # Turn back motor off

        GPIO.output(forward, GPIO.HIGH) # Turn forward motor on

# If joystick is pushed left or right

    if num == 6:

        if last == "forward": # If the last direction was forward

            pi\_pwm1.ChangeDutyCycle(100) # Set the speed to maximum

            GPIO.output(forward, GPIO.LOW) # Turn forward motor off

            GPIO.output(back, GPIO.HIGH) # Turn back motor on

            time.sleep(0.5) # Wait for 0.5 seconds

            GPIO.output(back, GPIO.LOW) # Turn back motor off

        if last == "back": # If the last direction was back

        pi\_pwm1.ChangeDutyCycle(100) # Set the speed to maximum

        GPIO.output(back, GPIO.LOW) # Turn back motor off

        GPIO.output(forward, GPIO.HIGH) # Turn forward motor on

        time.sleep(0.5) # Wait for 0.5 seconds

        GPIO.output(forward, GPIO.LOW) # Turn forward motor off

    return jsonify(message='Success', stickdata=data)

def gen():

    with picamera.PiCamera() as camera:

        camera.resolution = (320, 240)

        camera.framerate = 24

        camera.rotation = 180

        # Set the video format to be JPEG

        stream = io.BytesIO()

        for \_ in camera.capture\_continuous(stream, format='jpeg', use\_video\_port=True):

            stream.seek(0)

            yield (b'--frame\r\n'

            b'Content-Type: image/jpeg\r\n\r\n' + stream.read() + b'\r\n\r\n')

        stream.seek(0)

        stream.truncate()

@app.route('/video\_feed')

def video\_feed():

    return Response(gen(),

mimetype='multipart/x-mixed-replace; boundary=frame')

if name == 'main':

    app.run(debug=True, port=8080, host='raspberrypi.local')

# Appendix B – Electrical Schematics

## Raspberry Pi 3 B+ Schematic Diagram

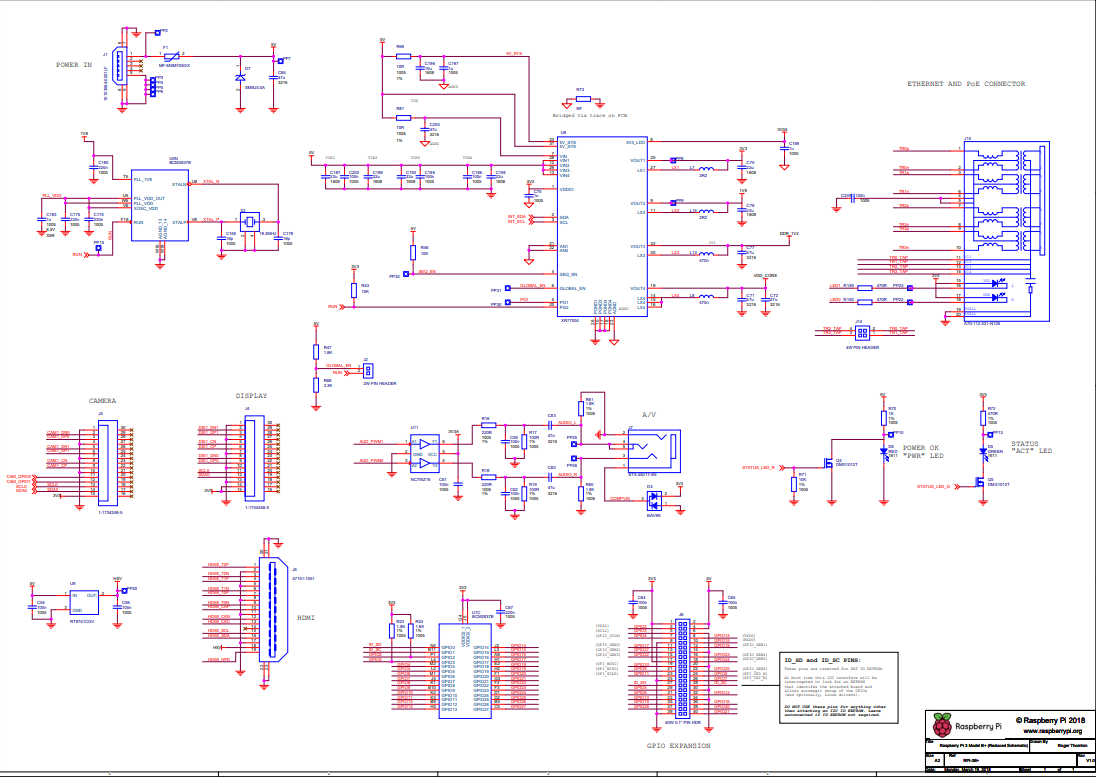


Figure 12 - Raspberry Pi 3B+ Schematic Diagram

## 

## L298N Motor Controller Schematic Diagram



Figure 13 - L298N Motor Controller Schematic Diagram

## 

# Appendix C – Parts List

Table 7 - Project Cost

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Quantity | Part | Manufacturer Part Number | Description | Unit Price |
| 1 | Raspberry Pi 2 Model B |  | A 900MHz quad-core ARM Cortex-A7 CPU  1GB RAM | **CA$**150 |
| 1 | Neuftech Stepper Motor Controller Dirve | L298n | Dual H-Bridge DC Stepper Motor Controller Dirve Module for Arduino Raspberry Pi | **CA$**10.99 |
| 1 | RCROKS RC Car | ‎B09WQZVKPK | 1/12 scale 2WD TOY-GRADE remote control car max speed can reach up to 28km/h | **CA$**69.99 |
| 1 | Gaming Wheel | N/A | Gaming wheel and pedals which are compatible with your PC or Laptop | N/A |
| 1 | Pi Camera | 8541582798 | Raspberry Pi Camera, KEYESTUDIO 5MP 1080p Camera Module with OV5647 Sensor Video Webcam for Raspberry Pi Model | **CA$**14.95 |
| 1 | PC or Laptop | N/A | PC or Laptop connected to Wi-Fi and gaming wheel | N/A |
| 1 | Power Bank | B08FSVYJ6D | Portable Battery for Mobile Phone (5.0v 2.0A 10000mAh Power Bank) | **CA$**79.73 |
|  |  |  |  | **Total:** **CA$**325.66 |

# Appendix D – References

<https://www.raspberrypi.com/products/raspberry-pi-3-model-b/>

<https://flask.palletsprojects.com/en/2.2.x/config/>

<https://learn.sparkfun.com/tutorials/raspberry-gpio/python-rpigpio-api>

<https://www.amazon.ca/dp/B00IVHQ0KI?psc=1&ref=ppx_yo2ov_dt_b_product_details>

<https://www.amazon.ca/dp/B073RCXGQS?ref=ppx_yo2ov_dt_b_product_details&th=1>

<https://www.amazon.ca/dp/B09WQZVKPK?psc=1&ref=ppx_yo2ov_dt_b_product_details>

<https://www.amazon.ca/dp/B01KBTNHS6?psc=1&ref=ppx_yo2ov_dt_b_product_details>

<https://www.amazon.ca/Complete-Starter-Raspberry-Module-Heatsink/dp/B08G8S63SL/ref=sr_1_3?crid=1D5AH0XNKG9IG&keywords=raspberry+pi+3+b%2B&qid=1680923108&sprefix=raspberry+pi+3+b%2Caps%2C83&sr=8-3>

<https://developer.mozilla.org/en-US/docs/Web/API/Gamepad_API/Using_the_Gamepad_API>

<https://maker.pro/raspberry-pi/tutorial/how-to-control-a-dc-motor-with-an-l298-controller-and-raspberry-pi>

<http://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics/>

# Appendix E - Contact Information

Name: Abdumalik Shamurov

Email Address: ashamurov@myseneca.ca

Cell Number: 647-687-9588