Computer Aided Design and Prototyping

(ME444 – Spring 2024)

Guided Project- RC Race

Design Report

Group No. C6

Mariah Alsinan	Senior - ME
Everett Lamberson	Junior - ME
Chad Matulenko	Junior - ME
Heather Mello	Junior - MDE

School of Mechanical Engineering

Purdue University

Revision History

S. No.	Date	Revision ID	Revision Details (Page No., Paragraph, Line No. etc.)	Author
1	05Mar202	Revision A	Initial Release	Mariah Alsinan, Everett Lamberson, Chad Matuleno, Heather Mello

Executive Summary

Problem Definition

Design Requirements: The vehicle should be able to be remotely controlled to complete two laps around the rally race track. It must be able to move forward and backward, stop, and turn left and right. The vehicle had to be remote-controlled using the provided microprocessor, and be driven using the wheels and motors distributed to each team. Modifications on both wheels and motors were allowed.

Design Constraints: The vehicle was constrained by the parts distributed to each team. At a maximum, the vehicle was to be composed of four wheels, two DC motors, one microcontroller, one servo motor, one toggle switch, one 9V battery and battery clip, one mini breadboard, and one voltage regulator. In addition to these constraints, a size limit of 10x10x10 inches was also imposed.

Design Description

Design Subsystems: The subsystems were broken down into three main categories: steering, driving, and electronics. The steering system chosen was based on Ackerman geometry, the drive was direct drive of the rear wheels, and the electronics included a breadboard and motor controller with appropriate power supply to drive three main functions: forwards, reverse, and steering.

Final Design and Prototyping: One solid chassis was chosen to connect all of the subsystems using 3mm bolts and 5mm pins. The final manufactured parts on the prototype were primarily 3D printed using PLA, with the front steering component being laser cut.

Evaluation

Capabilities: The vehicle was able to move forwards/backwards, stop, and turn to a specific angle on command from the remote control app. It also satisfied the imposed component limitations, only using motors and controllers that were distributed. In addition ot this, the vehicle was also within the size limit.

Limitations: Overall, the RC Car was unsuccessful in its main objective of driving two laps around the track. Although many design requirements were met successfully, the functions of the vehicle were not well tested enough and were designed in an overcomplicated manner and ultimately failed during the demonstration.

Table of Contents

1. Problem Definition

For this guided project, all teams were tasked with creating an RC Toy Vehicle that would compete in either a Rally Race or a Battlebot competition. It was determined the team would design, prototype, and demonstrate a Rally Race Vehicle that is able to race for two laps around a track.

1.1 Design Requirements

Other than the overall goal of completing two laps around the racetrack, the vehicle also had some design requirements that had to be taken into consideration. The first of these that the vehicle had to satisfy was that it must be able to move forward and backward, stop, and turn left and right. The vehicle had to be remote-controlled using the provided microprocessor, the Esp32s. The vehicle also had to be driven with the provided motors, although modification of the motors was allowed. Similarly, the provided wheels must be used, and customization or modification was allowed. To power the vehicle, a 9V rechargeable battery is required. Last, the vehicle has a maximum size of 10x10x10 inches, with no weight limit or minimum size requirement.

1.2 Design Constraints

Because the vehicle was designed for the Rally Race and not the Battlebot Competition, the provided parts that the design was limited to included: Four wheels, two geared DC motors, one L298N Motor Controller, one MG290S servo motor, one Esp32s microcontroller, one toggle switch, one 9V battery, one mini breadboard, one battery clip, and one L780CV Voltage Regulator. In addition to the constraints imposed by the component restrictions, a size limit was also imposed. Although there was no weight limit, the size limit was 10x10x10 inches.

2. Design Description

2.1 Sub-system 1 - Steering Mechanism

For the steering system, an Ackerman geometry steering system was chosen. This was accomplished through a main laser cut linkage (drawing in Appendix B), two linkages that connect the wheels to the main linkage and chassis, a pin/connector to drive the main linkage from the servo, attachment points on the chassis, and 5mm pins at all connection points. See Figure 2.1.1 for a detailed assembly of the steering system.

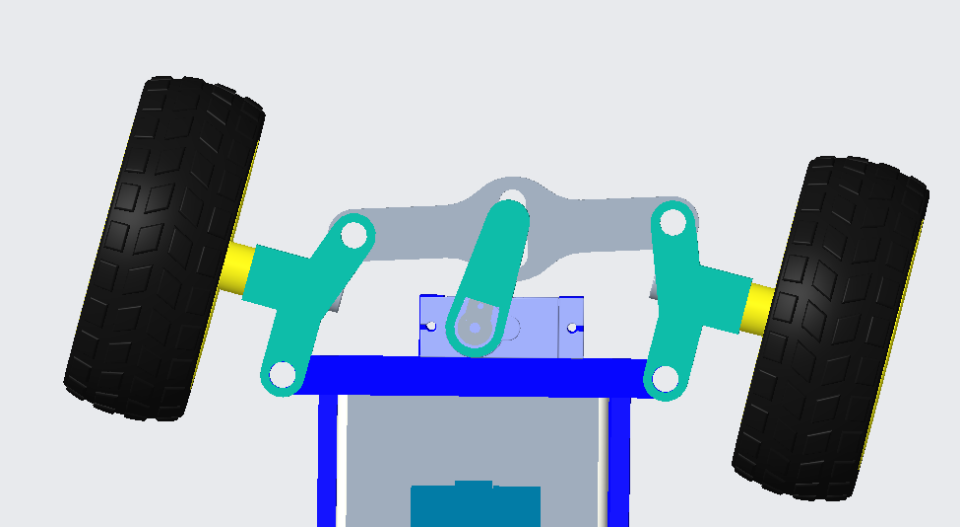


Fig. 2.1.1: Steering system in assembly

2.2 Sub-system 2 – Drive System

For the drive system, a two-motor direct drive system was chosen. To accomplish this, a motor carriage holds the motors via 3mm bolts to supply power to each wheel individually. See Figure 2.2.1 for a detailed view of the drive system.

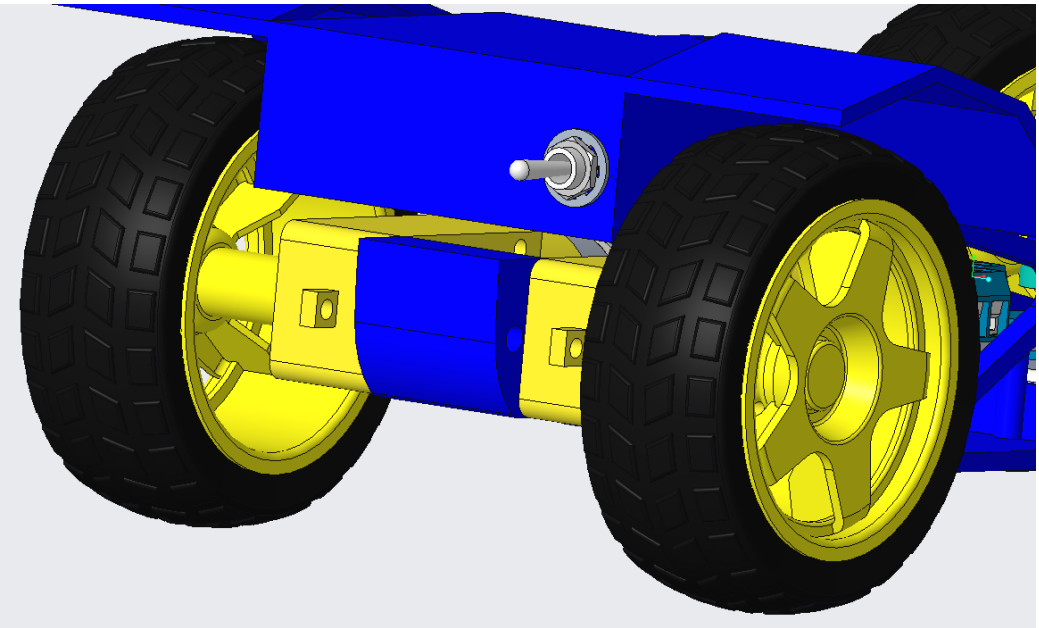


Fig. 2.2.1: Motor drive system

2.3 Sub-system 3 – Electronic System

An Esp32s Microcontroller was used to control the Motor Driver, DC Motors, and Servo. The Esps32 Microcontroller is a single board computer with a wireless microcontroller chip that can be integrated with Arduino and has Wi-Fi and Bluetooth capabilities. The L298N Dual H Bridge Motor Driver was used to drive the two DC Motors, motors A and B. The two enable pins located on the L298N are used to control the speed of the motors using pulse-width modulation, which generates variable voltage for the motors. The four logic pins control the direction of each motor, allowing them to go forward or reverse. However, the L298N was not used to control the MG920S servo, it instead simply gave it power, since all the logic electronics are located within the servo. A circuit diagram for these components can be seen in Appendix E.

The MG920S Servo was selected to control the steering system because of its angular precision. In the Arduino code, which can be seen in Appendix A, the turning function is extremely simple, as the only variable being adjusted is the angle of the servo. The DC motors

were chosen to drive the system and were coded to run in sync with each other. In an initial design, the forward and reverse functions were integrated with motor power. However, after testing, it was decided that the motor power adjustments were unnecessary for the RC race, since it was more effective to have the motors always run at their maximum power. Instead of slowing down for turns, it was decided that short back and forth movements would be more effective than attempting to fully integrate these functions, since it would likely be too difficult to drive during the demonstration. Because of this, all motor power adjustment controls were removed from the code, so only three simple functions were used for the actual demonstration:

1. Forward: Motors A and B run forward using a button on the Blynk App
2. Reverse: Motors A and B run backward using a button on the Blynk App
3. Turning: Servo angle is adjusted on a slider on the Blynk App

2.4 Final Design

For the final design, one solid chassis was chosen. While this meant a less easily alterable design, it provided more structural support than multiple pieces. The car body itself is fairly long, at about 10” in length. This was done to accommodate a layout where the breadboard and motor controller were positioned in front of each other, in front of the motors. The motor carriage described in Section 2.2 was implemented via a suspension system to the chassis. In addition, there was a pocket included above the motor to house the battery and toggle switch. From an aesthetics point of view, fenders were added to the rear wheels, which in essence made the car look like a drag race car. There were also structural supports added to either side of the breadboard/controller housing area to ensure that weight was distributed towards the wheels and away from the interior corner on the main chassis and also to ensure that the breadboard and controller were sufficiently being held in place. An assembly view can be seen in Appendix C.

2.5 Prototyping

A vast majority of the components were 3D printed using the Purdue ME 3D Print lab on Pulse XE printers for small components like the steering components and Taz 6 printers for the chassis and motor carriage. The front main steering linkage was laser cut out of 1/8” thick aluminum plate. 3mm bolts were used to connect all stationary components such as the rear motors and as the center post for the suspension system, and 5mm pins were used for the steering system

connections. The 3D printed holes were designed at a 5.2mm diameter for the 5mm pins, which ended up having a transition fit or slight interference fit when the pins were inserted. Holes for the bolts holding the motors were designed to have the same diameter as the holes included on the motor, which allowed for a clearance fit of the bolts. Images of the final prototype can be seen in Appendix F, and a BOM is included in Appendix D.

3. Results

On the day of the competition, the RC car experienced a major failure with the steering system that caused the car to only complete about ¼ of a lap out of the 2 required for timing. At first, the team believed that it may have been an issue with the servo motor, so a new servo was replaced. However, this servo failed on the ¼ lap completed. With 30 minutes left in the lab period, there was not enough time to implement another solution to fix the steering, however it is believed that the failure was caused by one of two things that further testing could discern the root cause: either the servo was overloaded by the amount of torque required to steer the car, or the software used was over-rotating the servo. Whichever root cause may have been happening, it was causing the servo to make a buzzing noise and overheat.

This critical failure was indicative of a much larger issue with the design process, and that was the design was made too complicated for what was needed to be accomplished in this project across pretty much all subsystems. After observing other teams' cars, successful cars had smaller, single piece chassis with steering controlled by the motors, and without any type of suspension systems. Our team's decision to include a suspension system was possibly one of the biggest over-complications of the design of the car that was made. This one decision caused the chassis to be made into its long shape by pushing the controller and breadboard forward, and creating an inefficient use of space, especially for this type of competition. Not including the suspension system would have almost no negative impacts, as the surface of the track was mostly flat, and the ramp was smooth and might not have imparted a critical amount of stress to the car.

Another serious over-complication was with the steering system. The team believed that using the Ackerman geometry would allow for the car to sustain high power to both motors while going around the corners. As it turned out, the corners were very tight and even if the steering system did not have a critical failure, the car would have traversed turns in 3-point turns, or possibly an even higher amount of points, which would have seriously reduced the time to complete laps by more than the amount of time lost by reducing power to the motors to cause turning. If the steering design chosen was steering by reducing motor power to either motor, the car would have been less complex in that there would be no need for a mechanical steering system, and there would have been no need for a servo in our system.

Despite all of the over-complications of this car, there was one design choice that was made to be simple, and that was direct drive of the wheels by the motors. In the team's observation of

other cars tested, most cars used direct drive. The use of a transmission was not used, but that would have significantly increased the complexity of the car for the possibility at higher speeds if the acceleration was high enough. In creating a simple mathematical model, it was determined that using a gear ratio of 1:1 (ie. direct drive) would produce a desirable speed at an acceleration that was not too low. The model is available at this link:

https://www.desmos.com/calculator/abv96mp6lo

This model considers the mass of the car, the stall torque/no load speed, the gear ratio, and friction of the system to create a model of velocity of the cart as a function of time, as well as plotting torque output and acceleration. As can be noticed by changing the gear ratio n, higher gear ratios reduce the torque and therefore the acceleration but increases the final speed of the car. In the end the speed predicted by model was sufficient compared to the time it would take to reach a steady state speed (~0.5 seconds).

4. Conclusion

In conclusion, the design did not fully achieve the goals of the RC car, which is to complete two full laps on the track with the control using the App and Arduino. Considering the ability of the RC car to complete only a quarter of a lap, and that the servo motor was not the main issue with the design, it was assumed that the issue was mainly due to the complexity of the design and the use of a steering system. For future steps, a direct drive steering system could replace the Ackerman geometry implemented in our design to reduce the complexity of the design.

5. Appendices

Appendix A: Arduino Code

```
#define BLYNK_PRINT Serial

#define BLYNK_TEMPLATE_ID "TMPL2etJNGrQ9"
#define BLYNK_TEMPLATE_NAME "TeamC6 Feb13"
#define BLYNK_AUTH_TOKEN "OCPa1gdYJQGu7GrxvRdXttwV8haGrn0U"

#include <BlynkSimpleEsp32.h>

char auth[] = "OCPa1gdYJQGu7GrxvRdXttwV8haGrn0U";

#include "ESP32Servo.h"

int servoPin = 27;     //specify pin numbers according to the circuit diagram
int servoAngle = 90;
int enablePinA = 15;
int in1PinA = 2;
int in2PinA = 0;
int enablePinB = 13;
int in1PinB = 12;
int in2PinB = 14;

Servo servo1;  //claim the servo motor object

void setup()
{
  Serial.begin(115200);
  delay(100);
```

```
  Blynk.begin(auth, "PAL3.0", "14Boone45");

  servo1.attach(servoPin);    //set up the servo pin as pin 27
  servo1.write(servoAngle);    //initialize the servo motor at 90 degree.
  pinMode(in1PinA, OUTPUT);    //set in1PinA(2) as an output pin
  pinMode(in2PinA, OUTPUT);    //set in2PinA(0) as an output pin
  pinMode(enablePinA, OUTPUT); //set enablePinA(15) as an output pin
  pinMode(in1PinB, OUTPUT);    //set in1PinB(12) as an output pin
  pinMode(in2PinB, OUTPUT);    //set in2PinB(14) as an output pin
  pinMode(enablePinB, OUTPUT); //set enablePinB(13) as an output pin
}

BLYNK_WRITE(V0)
{
  digitalWrite(enablePinA, param.asInt());  //run the gearhead motor A forward
  digitalWrite(in1PinA, HIGH);
  digitalWrite(in2PinA, LOW);
  digitalWrite(enablePinB, param.asInt());  //run the gearhead motor B forward
  digitalWrite(in1PinB, HIGH);
  digitalWrite(in2PinB, LOW);
}
BLYNK_WRITE(V1)
{
  digitalWrite(enablePinA, param.asInt());  //run the gearhead motor A backward
  digitalWrite(in1PinA, LOW);
  digitalWrite(in2PinA, HIGH);
  digitalWrite(enablePinB, param.asInt());  //run the gearhead motor B backward
  digitalWrite(in1PinB, LOW);
  digitalWrite(in2PinB, HIGH);
}
```

```
BLYNK_WRITE(V2)
{
  servo1.write(param.asInt());//control the servo angle
  delay(10);
}

void loop() {
  Blynk.run();
}
```

Appendix B: Part Drawings

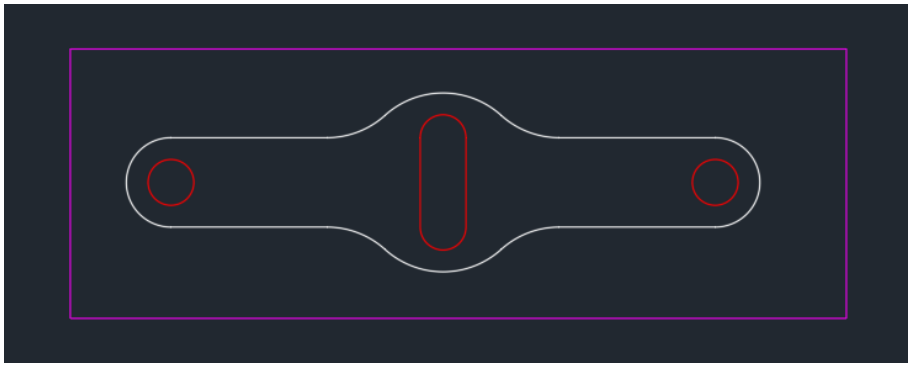


Figure B1: Part Drawing used for laser cutting the front steering linkage.

Appendix C: Assembly Drawings

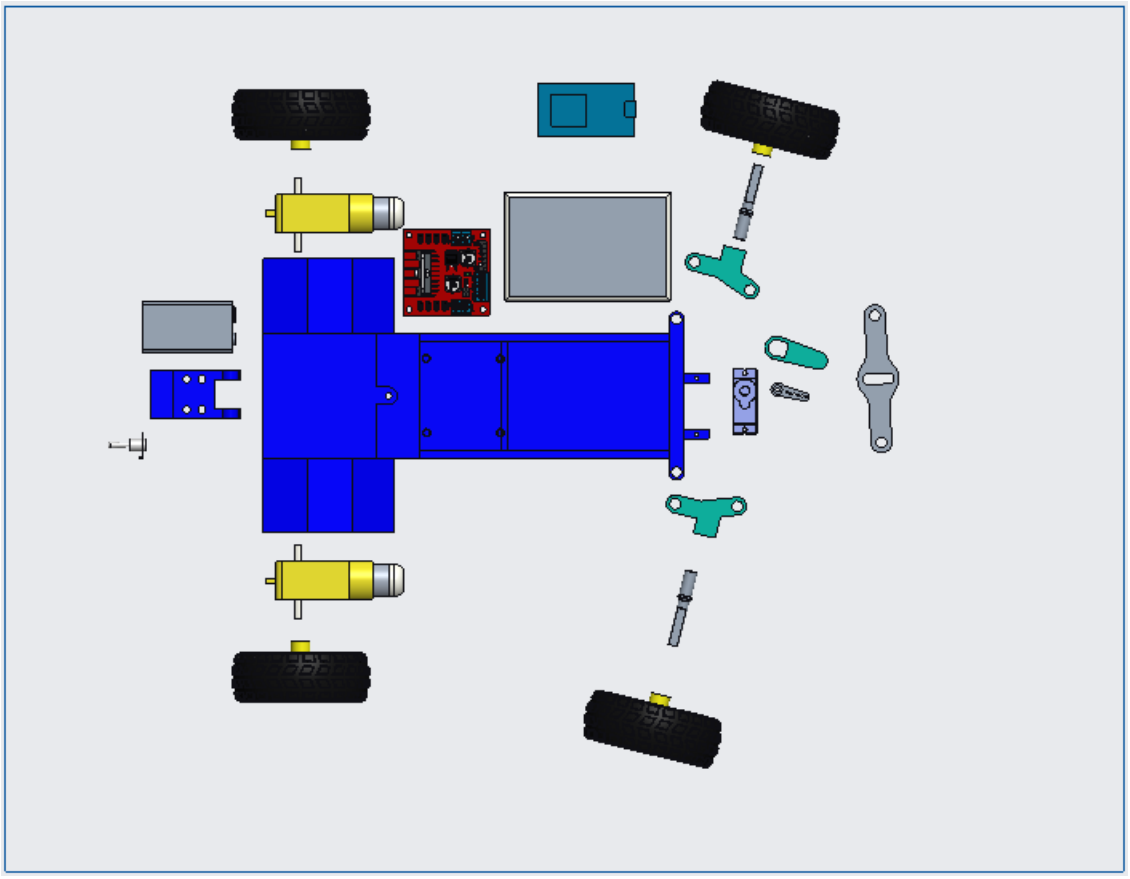


Figure C1: Exploded view of the final assembly

Appendix D: Bill of Materials

Top Level

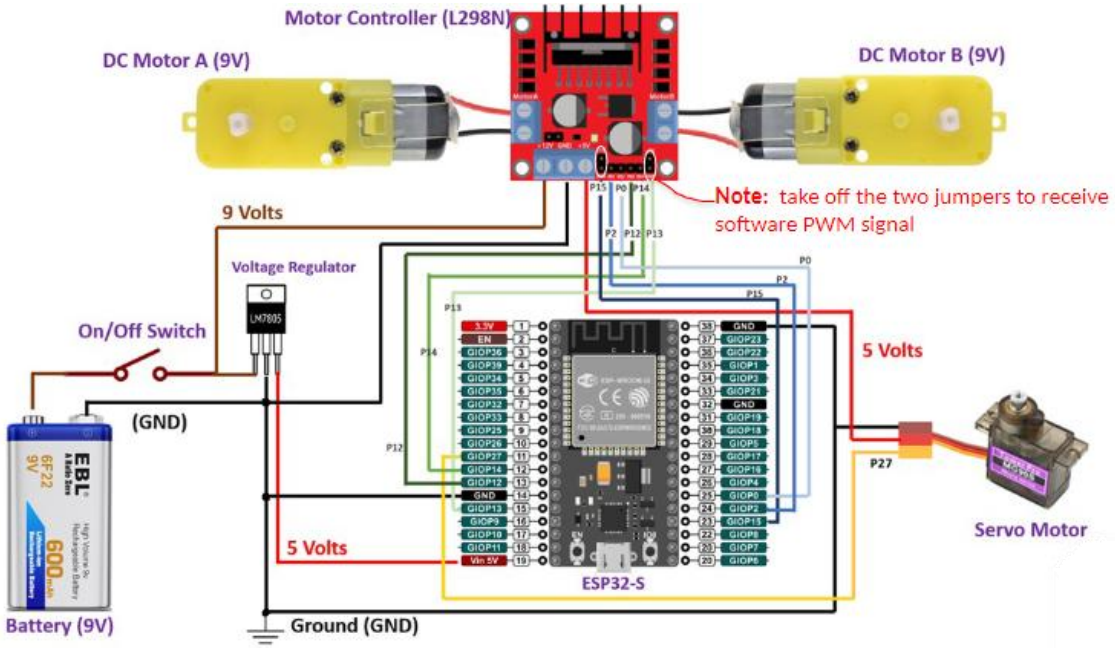
Item	QTY	Description	Material
1	1	Chassis	PLA
2	1	Motor carriage	PLA
3	1	Steering System	Assembly
4	1	Breadboard	COTS
5	1	Motor controller	COTS
6	4	Wheels	COTS
7	2	DC Brushless Motors	COTS

Steering System

Item	QTY	Description	Material
1	2	Steering to Wheel linkage	PLA
2	1	Main steering linkage	1/8” Al

3	1	Servo and arm	COTS
4	1	Servo to steering pin	PLA
5	4	5mm pins	COTS
6	2	Wheel shafts	PLA

Appendix E: Circuit Diagram



Appendix F: Final Prototype Images