**Computer Aided Design and Prototyping**
(ME444 – Spring 2024)

# Guided Project- [RC Race/Battle Bot]

# Design Report

## Group No. 09


| | |
|---|---|
| Kashyap Subramanian | Junior - ME |
| Sidh Gurnani | Junior - ME |
| Cannon Elsbury | Senior - ME |
| Miguel Cremades | Junior - ME |

School of Mechanical Engineering
Purdue University


# Revision History

| S. No. | Date | Revision ID | Revision Details (Page No., Paragraph, Line No. etc.) | Author |
|---|---|---|---|---|
| 1. | 3/4/2024 | Revision A | Initial Release | Kashyap Subramanian, Miguel Cremades, Sidh Gurnani, Cannon Elsbury |

# Executive Summary

Problem Definition: This report describes Team 9's robot design and evaluates its performance. The problem statement was to design and build a robot that is capable of defeating other robots in a battlebot competition held within an arena full of obstacles. The robot must be a remote-controlled vehicle, limited to a maximum size of 10x10x10 inches and powered by a provided 9V rechargeable battery. It must be capable of moving forward/backward, controlled stopping, and left/right turning using the provided motors, with customization permitted. While it can use the provided wheels, modifications are allowed. For the battlebot competition, only cold weapons are permitted.

Design Description: The key aspects of the design includes a chassis inspired by a turtle's shell, designed to protect the robot's wheels by covering them up from all sides. The robot used 2 wheels, each powered by a separate motor for its motion. Additional support was provided using ball casters. The primary weapon designed was a "lift mechanism", built to get beneath the other robots' chassis and lifting it, thereby suspending the other robots' wheels, making them immobile. A third motor was to be used for the lift mechanism. The overall robot's dimensions were 10x10x4.35 inches . The idea was to make it large with a wide base and great stability, making it difficult to topple over and succumb to any weaponry similar to the lift mechanism designed on the bot. Additionally, a geared drivetrain was built to increase the torque of the driving motors, making the robot slow but sturdy and powerful.

Evaluation: The robot built won the intra-lab battlebot competition, but it had some significant flaws. Firstly, the team decided to not implement the lift mechanism right before the competition since the code for the mechanism was insufficient. Additionally, faulty tolerancing made the physical implementation difficult. Next, tolerancing for the ball-bearings were off by a few millimeters. This made them stick out and tilt the robot. Therefore, the lower back of the robot was scraping against the ground while it moved, making it excruciatingly slow and difficult to maneuver. The bulky design and high torque motion was a large positive, making our robot virtually unpinnable. It, however, struggled to catch up to the other robots and deal any substantial damage to them.

While the design idea seems to have been sound, its implementation could definitively have been better. First, there should have been more communication between the team while designing the chassis and the drivetrain. This would have made sure that everybody was on the same page. Next, 3D printing of the components should have been started a great deal of time before the competition in order to leave enough time for adjustments and reprinting. The robot coding should have also been started sooner and full-strength robot testing should have started at least a day prior to the competition. With this being said, the team learned a lot about the design and testing process, and the team looks forward to building from here.

# Table of Contents



# 1. Problem Definition

The overarching goal of the guided project was to design and prototype an RC car that is controlled through Blynk IOT to be able to either participate in a battlebot competition or a rally race. The winner of the former would be determined by the last one standing in the arena and the winner of the latter would be determined by the fastest lap achieved. The team ultimately chose to design a battlebot.

The arena itself is approximately a 9ft by 9ft square with walls that are around 6 inches high. There are also several traps that are present throughout: a flipper that will flip every 5 seconds, a trap that only has entrances and no exits, and a revolving set of wire hooks that will disconnect lose connections.

## 1.1 Design Requirements

Based off of the overall goal and structure of the arena, a set of design requirements can be constructed. The battlebot must be controlled by the Blynk IOT app, which makes it mandatory to have electronics on the car that control it. In addition, throughout yhe early ideation phase of the project, the team had created a list of important requirements that the battlebot should have to be able to win, listed below:

- Completely covered to not be affected by the wire hooks

- Be able to drive at an adequate speed, yet contain a lot of torque to be able to push other bots and get out of tricky situations
- Low center of mass to avoid flipping
- Low ground clearance to avoid being potentially picked up
- At least one offensive/defensive mechanism to be able to do damage and help the team through difficult situations during combatwe

## 1.2 Design Constraints

While there is an extensive list of requirements the team would like to have in the battlebot, there are some requirements that are mandated by the project guidelines. These requirements are listed below:

- Able to move forward, backward, right, left, and come to a stop
- Remote-controlled through the ESP32 microprocessor provided
- Able to drive with the provided motors
- Can drive with the wheels provided
- Must be powered by the provided 9V rechargeable battery, where a maximum of 2 connected in parallel is allowed
- Has a maximum size of 10x10x10 inches, with no weight limit
- Cold weapons only

These list of requirements don't necessarily limit what the team is able to do with the system, but it does, to an extent, dictate the design of the vehicle.

# 2. Design Description

## 2.1 Sub-system 1: Drive System

The drive system was designed to balance the tradeoff between maximize torque output to the wheels from the provided motors while still having adequate speed to maneuver around the arena. In addition, there was also the tradeoff between minimizing the footprint of the drive train in the chassis to allow adequate room for weaponry and the control system while also having enough rigidity and strength to withstand any offensive attacks from other teams without being demolished.

Ultimately, the drive train was achieved with the use of two motors driving a compound gear train to the wheels in the back of the chassis, and free-rotating metal ball casters in the front of the chassis for frictionless motion and enhanced stability.

The compound gear train consisted of the motor driving a set of bevel gears with a set of spur gears then driving the wheels. The bevel gears were used to pivot the axis of rotation 90 degrees from the motor to the wheels, creating more space inside the chassis for other components. The mechanical advantage of the system came from the gear ratio of the spur gears of 2.8 which took into account the balance between torque and speed. Shown below in Figure 1 is an isolated view of the cad showing the drive train demonstrating the path of movement from the motor to the wheels with Figure 2 showing the drive train in the full assembly.

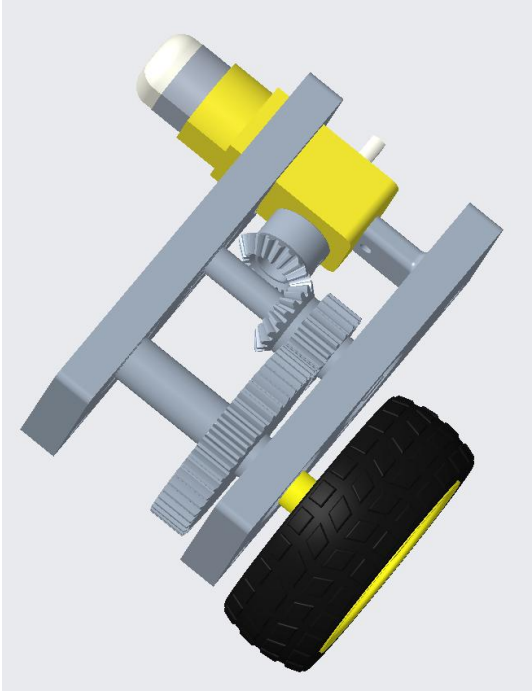


Figure 1. Isolated View of Drive Train

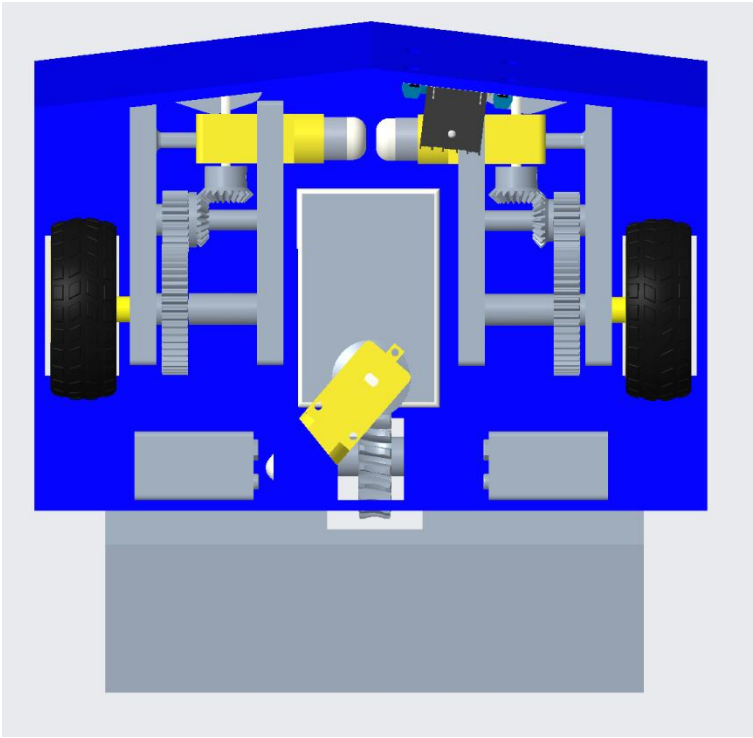


Figure 2. Top View of Robot showing Drive Train

## 2.2 Sub-system 2: Steering Mechanism

As discussed in the previous section, the bot was driven with two wheels in the rear and two metal ball casters in the front. This configuration allowed for very sharp rotations, turning the wheels in opposite directions, giving it the ability to pivot around the center of the bot. This steering relied on a lack of friction between the free-rotating ball casters and the tile surface of the arena. There was nothing else preventing rotation or motion besides these contact points. Shown below in Figure 3 is an image of the CAD showing the bottom of the chassis with the two wheels in the rear and the two ball casters in the front.

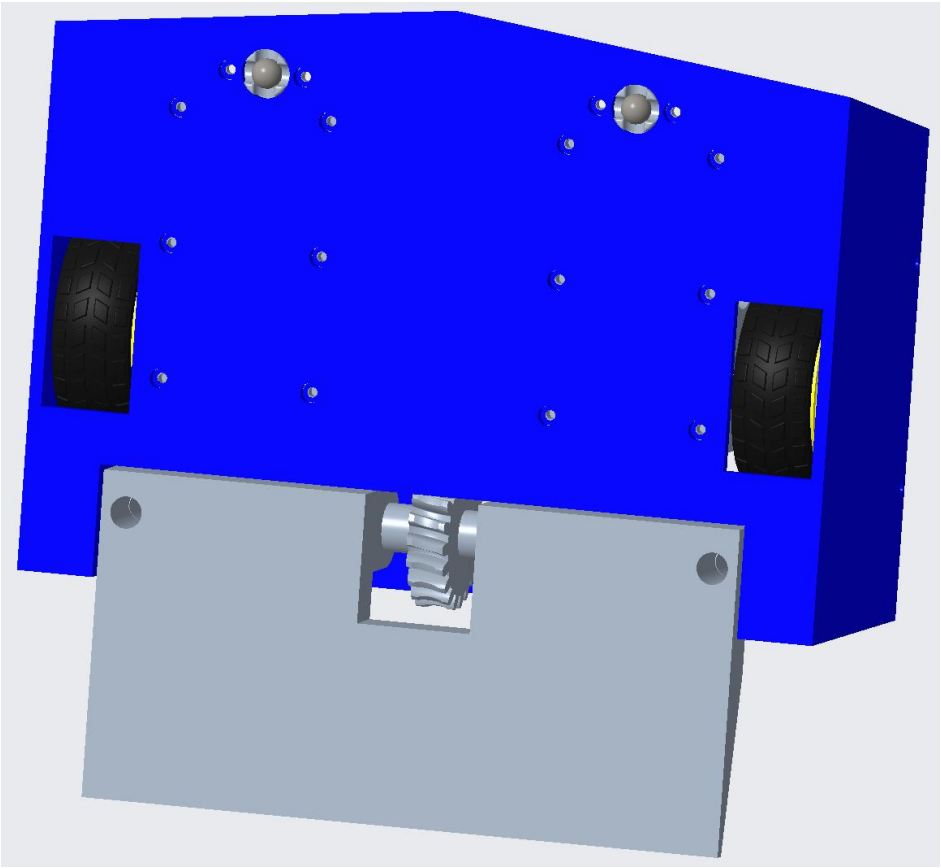


Figure 3. Bottom View of Robot

## 2.3 Sub-system 3: Defensive/Offensive systems

As mentioned in the design requirements, the team desired to have at least one offensive/defensive system that could be deployed. After extensive discussion, the team chose to use a lift assembly to be able to lift other bots as well as function as a guard.

The majority of the design decisions taken were primarily driven by the porotyping method that was being used: 3D printing. This means that while the team can design any system of their choosing, it had to be realizable by the manufacturing method.

A significant portion of the design was dictated by the team's requirements that were not explicitly stated in the requirements section of this report. The lift assmebly was intended to be driven by a motor, and in order to be able to lift up another battlebot, there had to be a relatively high amount of torque. This dictated that two gears be used, to reduce the amount of meshing errors between 3 or more gears and to also have the overall assembly be relatively tightly packaged. To avoid backdriving, a worm gear was used to drive a larger spur gear with a gear ratio of approximately 20:1.

A larger wedge was used instead of multiple smaller ones as a way to increase the surface area that was able to be used to lift up other battlebots. It would also make it much easier to deploy the weapon, as a smaller size wouldn't have allowed the team to be able to deploy the weapon as effectively whilst in combat. The wedge used was approximately 8 inched long, and had a relatively low gradient to maximize surface area and also be able to be used in a wide variety of situations.

Once those components were designed, the team was able to design the other components that interface with the chassis and be able to hold together the lift assembly. One notable item here is the decision to use two "halfshafts" as opposed to one longer shaft. This was done to potentially reduce 3D printing time and reduce the chances of the part delaminating in the 3D printers. It also made for much easier assembly, which is an implied requirement to be able to successfully complete this project.

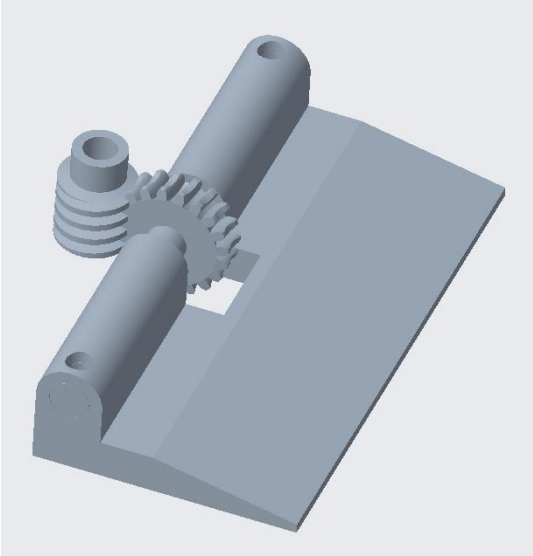


Figure 4. Isolated Lift System Assembly

## 2.4 Sub-system 4: Electronic System

The major electrical and electromechanic components used in the robot were two TT Dual DC Motors, one L298N Motor drive controller, one arduino ESP 32-S board, and one small breadboard.

The robot incorporated two TT Dual DC Motors for propulsion, controlled by an L298N Motor Drive Controller, which efficiently managed their operation with a voltage range of 3V to 12V and a maximum torque output of approximately 2.5 kg/cm. Driving its functionality was the Arduino ESP32-S board, equipped with dual-core processors running at speeds up to 240MHz, integrated Wi-Fi and Bluetooth capabilities, and ample GPIO pins for interfacing with sensors and actuators. The robot was controlled by pressing buttons on a mobile phone by connecting them via bluetooth. These electromechanical components worked in tandem to enable precise control and maneuverability in the robot's operations.

The team initially used three motors connected to 2 L298N Motor drive controllers attached to a single arduino ESP 32-S board. Unforunately, plugging the two L298N Motor drive controller boards to the single arduino burned the ESP 32-S chip instantly when the battery was

connected. Therefore, the team took the decision to reduce the number of motors used from three to two. This meant that the team could now use a single L298N Motor drive controller. The final circuit diagram used has been displayed in Figure 5.

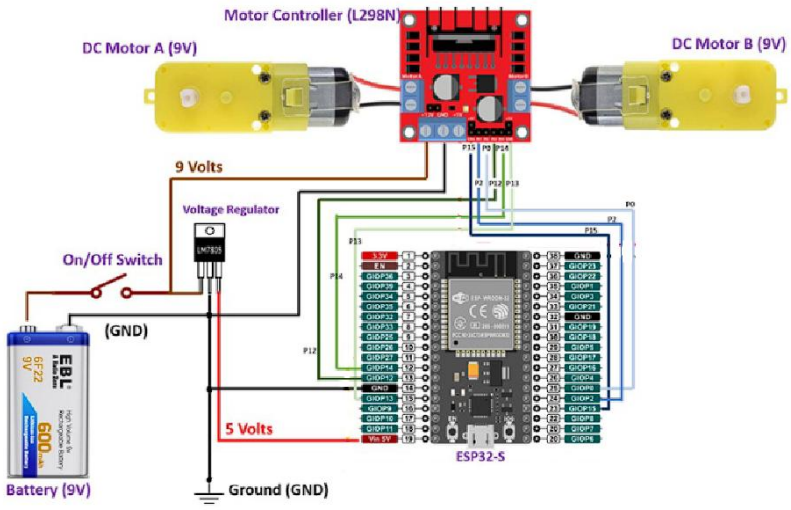



Figure 5: Circuit diagram used for the final robot

## 2.5 Final Design

For the final design an assembly was created to fit all of the subassemblies into the chassis and integrate all the systems created. This was a challenging task due to the space limitations of the battle Bot which were 10x10 inches. Multiple iterations of the chassis had to be made to accommodate for changes done in other components such as the lift assembly and the drivetrain. There also needed to be enough space to fit the electrical components of the bot like the motors and wiring to make sure it wouldn't interfere with any moving parts. This is why the controller had to be attached on the side wall due to a lack of space in the floor of the chassis. Planning ahead for the chassis would've made it easier to fit more components into the space available. The final assembly with and without the chassis as well as the chassis on its own can be seen below.

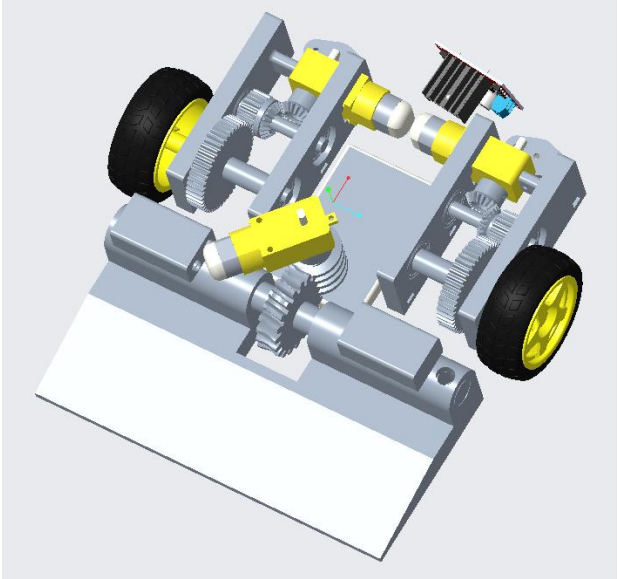


Figure 6: Final assembly without chassis

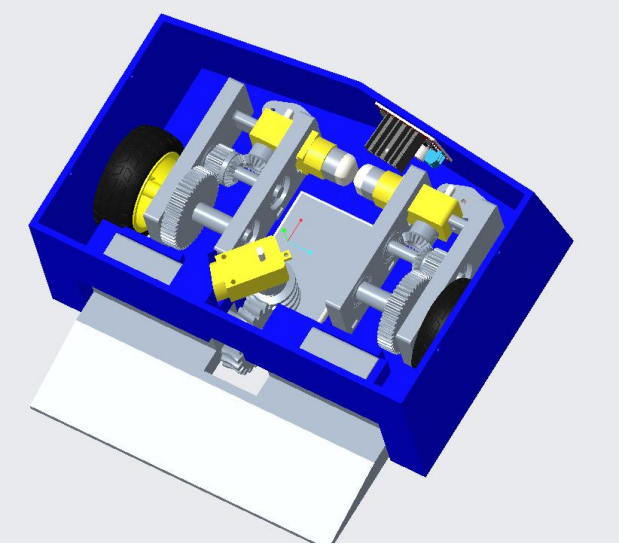


Figure 7: Final assembly with chassis

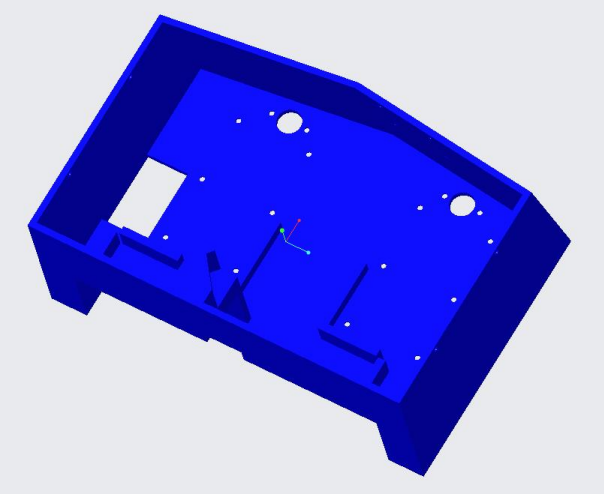


Figure 8: Chassis final iteration

## 2.6 Prototyping

Prototyping of the bot was achieved with the use of 3D printing and all of the materials provided: motors, metal balls, wheels, battery, control system and circuitry, etc. For 3D printing, both Polylactic Acid (PLA) and PLA Tough were used. PLA was used because it is very cheap and easy to print, while the PLA Tough has the same features as PLA while also having a higher impact resistance. Due to the ability to 3D print most of the components, the team did not have many limitations with manufacturability, although it was crucial to design the components knowing the benefits and drawbacks of 3D printing. A prime example of this was adding additional support from the bevel to spur gear to redinforce the bevel gear with additional material and limit the overhang from the teeth of the gears (shown below in Figure 9). Due to previous experience using 3D printing for manufacturing, the tolerancing was well known and could be used as an advantage while creating and assembling the bot, for example with the bearings and ball casters used (shown above in section 2.2).

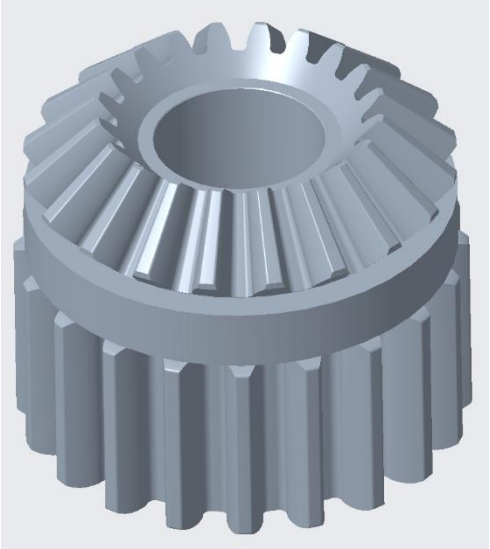


Figure 9. Bevel/Spur Gear with Additional Support

## 3. Results (Learning Outcomes)

The team had achieved most of what the members had sought to achieve, and in some regards, even exceeded the expectations set. The team never had the intention to win any competition, and the fact that the team's battlebot was able to achieve victory means that from that standpoint the team had exceeded expectations. That being said, the team can still learn a lot from the project, which can be applied in the subsequent final project.

Initially, the team had designed for more ground clearance, but due to a combination of mismeasurements and the team not properly accounting for the size of the wheel, there was signficantly less ground clearance. This had made it so that the torque generated by the drivetrain system was not just driving the wheels, but instead dragging the chassis and most of the vehicle weight along the ground. This greatly reduced the speed of the vehicle than the team had been anticipating. While the desired speed targets weren't achieved, the bot still had as significant amount of torque, which led to the victory in the intra-lab competition.

Additionally, the team could have done a better job from a systems integration point of view. The team sought to have the lift assembly controlled by a motor, but for a few reasons, this was not achievable. The first of these reasons is a time constraint, caused by the amount of time needed to print the chassis and other such components. Since this was necessary, the

team could have managed the balance between time and expectations better. The second of thses reasons was the numerous issues that the team faced while finally integrating their system together. There were signficant issues with the hardware that was being used, and the team spent a few hours trying to debug the problem: time which could have been better spent outside of debugging code. The last of these issues can be attributed to an oversight in the design phase. The team had planned to use 3 geared DC motors to run the drivetrain and the lift assembly. This meant that the current supplied to each of the components was too low, and thus could not allow for proper and ideal application of all systems. Since driving is one of the necesary requirements, the motor intended for the lift assembly was never implemented. The team also hadn't designed to use 2 batteries, which explains this limitation better.

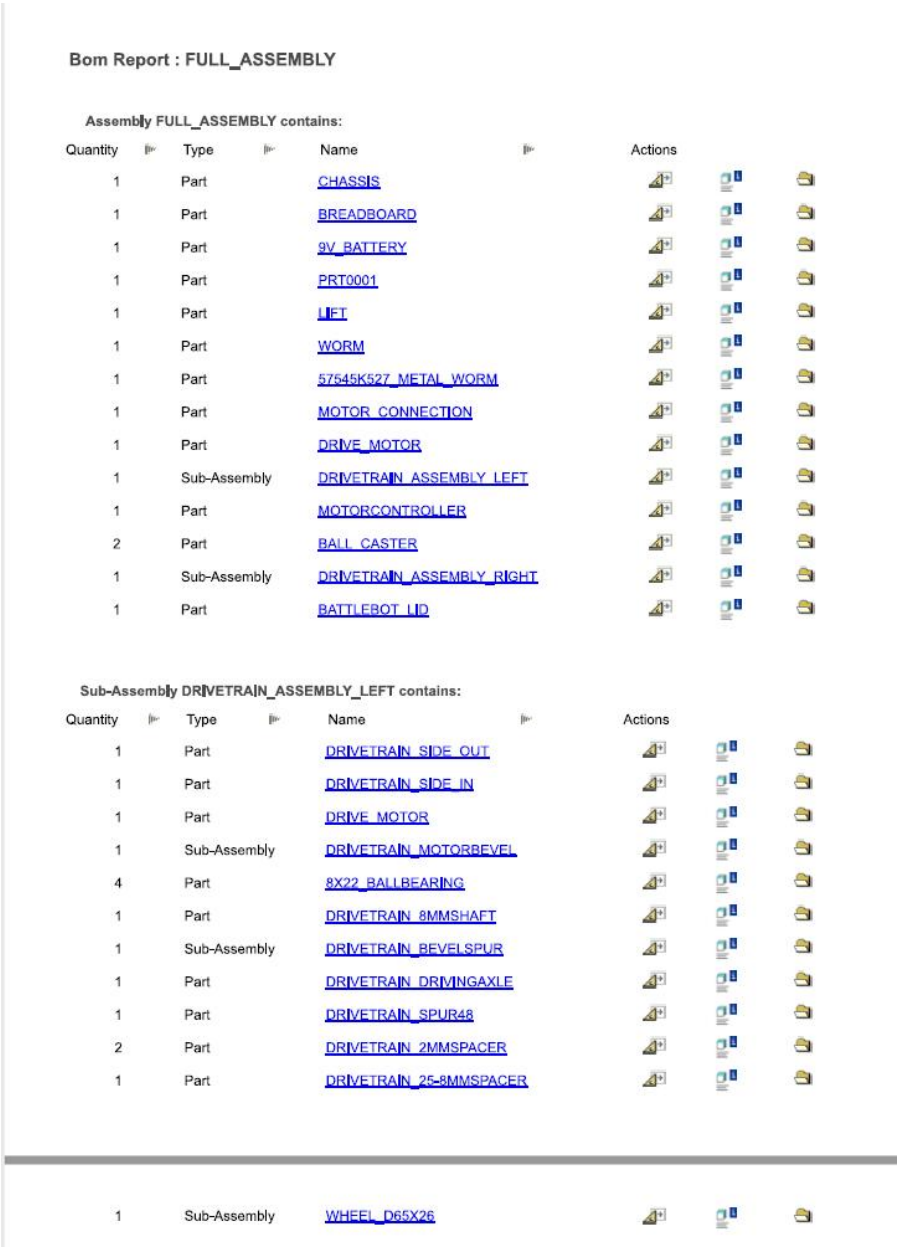
# 4. Conclusion

As mentioned earlier, the battlebot was able to win the intra-lab competition that it had participated in. The victory was made possible by the bukly size of the bot and the relatively low ground clearance. While the lift assembly wasn't able to be powered by the motor, it was still assembled onto the car and was low enough to the ground to be able to lift a certain specification of vehicles. The battlebot ultimately came second to last in the final battle, and a lot of it can be attributed to the slow speed of the bot itself. Ironically, the bot was flipped over in the final battle due to two other vehicles with wedges driving significantly faster and attacked the bot from either side, leaving the team's bot vulnerable and thus flipped over.

The main cause of the loss can be attributed to the excessively low ground clearance that was talked about earlier. In order to improve on this shortcoming, the team can priortize systems integration and start to think about it from the initial design phase as well. In addition, the team can also account better for errors, especially for critical components such as that one, and design for more error in mind.

# 5. Appendices

- *Part Drawings – No laser cut part*
- *Assembly Drawings - No laser cut part*
- *BOM*



**Bom Report : FULL_ASSEMBLY**

**Assembly FULL_ASSEMBLY contains:**

| Quantity | Type | Name | Actions |
|---|---|---|---|
| 1 | Part | CHASSIS | |
| 1 | Part | BREADBOARD | |
| 1 | Part | 9V_BATTERY | |
| 1 | Part | PRT0001 | |
| 1 | Part | LIFT | |
| 1 | Part | WORM | |
| 1 | Part | 57545K527_METAL_WORM | |
| 1 | Part | MOTOR_CONNECTION | |
| 1 | Part | DRIVE_MOTOR | |
| 1 | Sub-Assembly | DRIVETRAIN_ASSEMBLY_LEFT | |
| 1 | Part | MOTORCONTROLLER | |
| 2 | Part | BALL_CASTER | |
| 1 | Sub-Assembly | DRIVETRAIN_ASSEMBLY_RIGHT | |
| 1 | Part | BATTLEBOT_LID | |

**Sub-Assembly DRIVETRAIN_ASSEMBLY_LEFT contains:**

| Quantity | Type | Name | Actions |
|---|---|---|---|
| 1 | Part | DRIVETRAIN_SIDE_OUT | |
| 1 | Part | DRIVETRAIN_SIDE_IN | |
| 1 | Part | DRIVE_MOTOR | |
| 1 | Sub-Assembly | DRIVETRAIN_MOTORBEVEL | |
| 4 | Part | 8X22_BALLBEARING | |
| 1 | Part | DRIVETRAIN_8MMSHAFT | |
| 1 | Sub-Assembly | DRIVETRAIN_BEVELSPUR | |
| 1 | Part | DRIVETRAIN_DRIVINGAXLE | |
| 1 | Part | DRIVETRAIN_SPUR48 | |
| 2 | Part | DRIVETRAIN_2MMSPACER | |
| 1 | Part | DRIVETRAIN_25-8MMSPACER | |

| | | | |
|---|---|---|---|
| 1 | Sub-Assembly | WHEEL_D65X26 | |

**Sub-Assembly DRIVETRAIN_MOTORBEVEL contains:**

| Quantity | Type | Name | Actions |
|---|---|---|---|
| 1 | Part | DRIVETRAIN_BEVEL | |
| 1 | Part | DRIVETRAIN_BEVELTOMOTOR | |
| 1 | Part | DRIVETRAIN_MOTORBEVEL_PLUG | |

**Sub-Assembly DRIVETRAIN_BEVELSPUR contains:**

| Quantity | Type | Name | Actions |
|---|---|---|---|
| 1 | Part | DRIVETRAIN_BEVEL_SPUR | |
| 1 | Part | DRIVETRAIN_SPUR | |

**Sub-Assembly WHEEL_D65X26 contains:**

| Quantity | Type | Name | Actions |
|---|---|---|---|
| 1 | Part | RIM_65MM | |
| 1 | Part | TIRE_65MM | |

**Sub-Assembly DRIVETRAIN_ASSEMBLY_RIGHT contains:**

| Quantity | Type | Name | Actions |
|---|---|---|---|
| 1 | Part | DRIVETRAIN_SIDE_OUT_RIGHT | |
| 1 | Part | DRIVETRAIN_SIDE_IN | |
| 1 | Part | DRIVE_MOTOR | |
| 4 | Part | 8X22_BALLBEARING | |
| 1 | Part | DRIVETRAIN_8MMSHAFT | |
| 1 | Part | DRIVETRAIN_DRIVINGAXLE | |
| 2 | Part | DRIVETRAIN_2MMSPACER | |
| 1 | Part | DRIVETRAIN_BEVELSPUR_ASM | |
| 1 | Part | DRIVETRAIN_SPUR48 | |
| 1 | Part | DRIVETRAIN_25-8MMSPACER | |
| 1 | Sub-Assembly | DRIVETRAIN_MOTORBEVEL | |

---

| | | | |
|---|---|---|---|
| 1 | Sub-Assembly | WHEEL_D65X26 | |

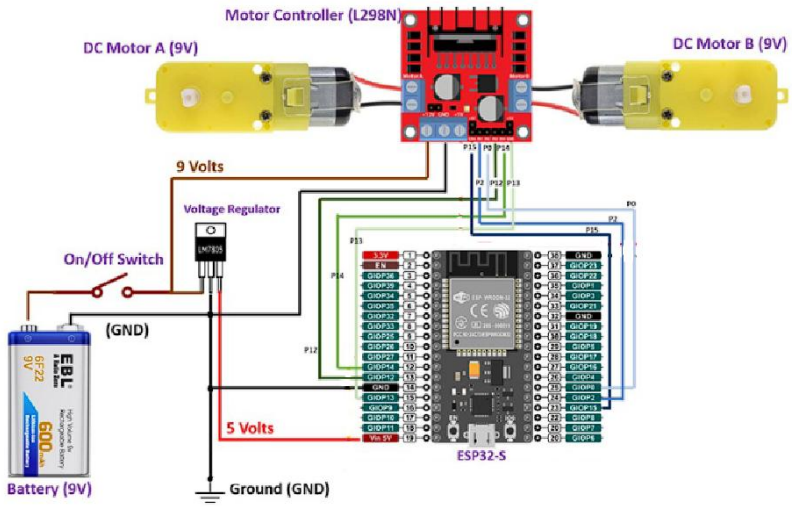
Summary of items for FULL_ASSEMBLY:

| Quantity | Type | Name | Actions |
|---|---|---|---|
| 1 | Part | CHASSIS | |
| 1 | Part | BREADBOARD | |
| 1 | Part | 9V_BATTERY | |
| 1 | Part | PRT0001 | |
| 1 | Part | LIFT | |
| 1 | Part | WORM | |
| 1 | Part | 57545K527_METAL_WORM | |
| 1 | Part | MOTOR_CONNECTION | |
| 3 | Part | DRIVE_MOTOR | |
| 1 | Part | DRIVETRAIN_SIDE_OUT | |
| 2 | Part | DRIVETRAIN_SIDE_IN | |
| 2 | Part | DRIVETRAIN_BEVEL | |
| 2 | Part | DRIVETRAIN_BEVELTOMOTOR | |
| 2 | Part | DRIVETRAIN_MOTORBEVEL_PLUG | |
| 8 | Part | 8X22_BALLBEARING | |
| 2 | Part | DRIVETRAIN_8MMSHAFT | |
| 1 | Part | DRIVETRAIN_BEVEL_SPUR | |
| 1 | Part | DRIVETRAIN_SPUR | |
| 2 | Part | DRIVETRAIN_DRIVINGAXLE | |
| 2 | Part | DRIVETRAIN_SPUR48 | |
| 4 | Part | DRIVETRAIN_2MMSPACER | |
| 2 | Part | DRIVETRAIN_25-8MMSPACER | |
| 2 | Part | RIM_65MM | |
| 2 | Part | TIRE_65MM | |
| 1 | Part | MOTORCONTROLLER | |
| 2 | Part | BALL_CASTER | |
| 1 | Part | DRIVETRAIN_SIDE_OUT_RIGHT | |
| 1 | Part | DRIVETRAIN_BEVELSPUR_ASM | |

---

| | | | |
|---|---|---|---|
| 1 | Part | BATTLEBOT_LID | |

- *Circuit Diagram*




- *Arduino Code*

```
#define BLYNK_PRINT Serial

#define BLYNK_TEMPLATE_ID "TMPL2W0-oy2W5"
#define BLYNK_TEMPLATE_NAME "BattleBotv2"
#define BLYNK_AUTH_TOKEN "vsh9LkPyBzWq5Qz2hNmg6KTKcsJlV80c"

#include <BlynkSimpleEsp32.h>
char auth[] ="vsh9LkPyBzWq5Qz2hNmg6KTKcsJlV80c";

//defining pins
int enablePinA = 15;
int in1PinA = 2;
int in2PinA = 0;

int enablePinB = 13;
int in1PinB = 12;
int in2PinB = 14;

/*
int enablePinC = 6;
int in1PinC = 7;
int in2PinC = 8;
*/

int direction = 0;

void setup() {
```

```
  Serial.begin(115200);
  delay(100);
  Blynk.begin(auth, "NETGEAR39", "phobicwater900");

  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
  /*
  pinMode(enablePinC, OUTPUT);
  pinMode(in1PinC, OUTPUT);
  pinMode(in2PinC, OUTPUT);
  */
}

BLYNK_WRITE(V4)
{
  direction = param.asInt();
  if (param.asInt() == 1)
  {
    Serial.println(param.asInt());
    digitalWrite(enablePinA, param.asInt());  //run the gearhead motor A forward
    digitalWrite(enablePinB, param.asInt());  //run the gearhead motor B forward
    digitalWrite(in1PinA, HIGH);
    digitalWrite(in2PinA, LOW);
    digitalWrite(in1PinB, HIGH);
    digitalWrite(in2PinB, LOW);
    //direction = 1;
  }

  if (param.asInt() == -1)
  {
    Serial.println(param.asInt());
    digitalWrite(enablePinA, param.asInt());  //run the gearhead motor A backward
    digitalWrite(enablePinB, param.asInt());  //run the gearhead motor B backward
    digitalWrite(in1PinA, LOW);
    digitalWrite(in2PinA, HIGH);
    digitalWrite(in1PinB, LOW);
    digitalWrite(in2PinB, HIGH);
    //direction = -1;

  }
  if (param.asInt() == 0)
```

```
  {
    Serial.println(param.asInt());
    digitalWrite(enablePinA, param.asInt());  //stop the gearhead motor A
    digitalWrite(enablePinB, param.asInt());  //stop the gearhead motor B
    digitalWrite(in1PinA, LOW);
    digitalWrite(in2PinA, LOW);
    digitalWrite(in1PinB, LOW);
    digitalWrite(in2PinB, LOW);
    //direction = 0;

  }
}

BLYNK_WRITE(V1)
{
  if (param.asInt() == -1) //left
  {
    Serial.println(param.asInt());
    digitalWrite(enablePinA, param.asInt());  //turn the gearhead motor A
    digitalWrite(enablePinB, param.asInt());  //turn the gearhead motor B
    digitalWrite(in1PinA, HIGH);
    digitalWrite(in2PinA, LOW);
    digitalWrite(in1PinB, LOW);
    digitalWrite(in2PinB, HIGH);
  }

  if (param.asInt() == 1) //right
  {
    Serial.println(param.asInt());
    digitalWrite(enablePinA, param.asInt());  //turn the gearhead motor A
    digitalWrite(enablePinB, param.asInt());  //turn the gearhead motor B
    digitalWrite(in1PinA, LOW);
    digitalWrite(in2PinA, HIGH);
    digitalWrite(in1PinB, HIGH);
    digitalWrite(in2PinB, LOW);
  }
  if (param.asInt() == 0)
  {
    Serial.print(direction);
    if (direction == 1)
    {
      digitalWrite(enablePinA, param.asInt());  //turn the gearhead motor A
      digitalWrite(enablePinB, param.asInt());  //turn the gearhead motor B
      digitalWrite(in1PinA, HIGH);
      digitalWrite(in2PinA, LOW);
```

```
      digitalWrite(in1PinB, HIGH);
      digitalWrite(in2PinB, LOW);
    }
    if (direction == -1)
    {
      digitalWrite(enablePinA, param.asInt());  //turn the gearhead motor A
      digitalWrite(enablePinB, param.asInt());  //turn the gearhead motor B
      digitalWrite(in1PinA, LOW);
      digitalWrite(in2PinA, HIGH);
      digitalWrite(in1PinB, LOW);
      digitalWrite(in2PinB, HIGH);
    }
    if (direction == 0)
    {
      digitalWrite(enablePinA, param.asInt());  //turn the gearhead motor A
      digitalWrite(enablePinB, param.asInt());  //turn the gearhead motor B
      digitalWrite(in1PinA, LOW);
      digitalWrite(in2PinA, LOW);
      digitalWrite(in1PinB, LOW);
      digitalWrite(in2PinB, LOW);
    }
  }
}

/*
BLYNK_WRITE(V2) //Lift
{
  if (param.asInt() == 1) {
      digitalWrite(enablePinC, param.asInt());
      Serial.println(param.asInt());
      digitalWrite(in1PinC, HIGH);
      digitalWrite(in2PinC, LOW);
      delay(3000);
  }
  if (param.asInt() == 0) {
      digitalWrite(enablePinC, param.asInt());
      Serial.println(param.asInt());
      digitalWrite(in1PinC, LOW);
      digitalWrite(in2PinC, LOW);
      delay(3000);
  }
}
*/

void loop() {
```

```
  Blynk.run();
}
```