Autonomous/Remote Control Mecanum Wheels Tesla Roadster

Tung Do

Electromechanical Engineering

Electromechanical Engineering Technology Department

College of Engineering

California State Polytechnic University, Pomona

Summary

- *Abstract:* This project aims to apply everything the student has learned in engineering about mechanics, machine elements, programming, and electronics to build a Mecanum Wheels Tesla Roadster to research the possibility of flexible movement in developing a smart vehicle network.

- *Keywords:* Mecanum, Robot, Wheel, Tesla, smart car, innovative vehicle, network

Table of Contents

1. List of figures and list of tables

Materials and dimensions

The material for this project will be the PLA used in the 3D printer since most of the parts of the project will be 3D printed. In addition, a small portion will use stainless steel, such as wheel axles, universal mounting hubs, and bolts. The roller can be improved with rubber or TPU to increase the friction for gripping.

	Length (mm)	Width (mm)	Height (mm)	Diameter (mm)	Weight (g)	Quantity
Mega Arduino	110	55	15	N/A		2
DC Motor	42	42	40	5	218	4
Li-Po Battery	140	45	40	N/A	412	1
Wheel case	N/A	45	N/A	75	82	4
Roller	50	N/A	N/A	10	8	40
Chassis (final)	300	200	80	N/A	560	1
Body	300	200	100	N/A	1470	1
Universal mounting hub	N/A	N/A	13	33		4
Wheel axle	55	N/A	N/A	3		3
M4 bolts and nuts	50	N/A	N/A	4		50
M3 bolts and nuts	10	N/A	N/A	3		50
Radio communication NRF24L01 module						2
Ultrasonic sensor HC-SR04 module						4
H-Bridge						2

Pre-built mecanum wheel						4
10000mAh power bank						2

2. Introduction

Throughout my life, I have seen many times that countries are fighting for oil and fossil fuels, which are finite and cause pollution and greenhouse gases. In my senior year, the energy crisis escalated between Russia and Ukraine, which made me think, what if I can find a solution to replace fossil fuels with environment-friendly electric vehicles to save the environment and people will no longer depend on fossil fuels? Therefore, in the Spring of 2022, I began my research with Dr. Scott Boskovich about researching and building a prototype of a Tesla Roadster with full functions, then implementing the mecanum wheels (omnidirectional wheels) on it to examine the potential of replacing the current wheel to push the potential of electric cars into smart cars. What if, in the future, humans use a system of intelligent vehicles that can communicate with each other and run-on autopilot? Then the mecanum wheel has a huge advantage in this system for using less space for rotating, changing lanes, and saving space. And when it comes to developing smart cars, my top choice is Tesla electric cars since they are optimized for this case. I designed my project based on the Tesla car's structure so everyone can have a realistic look at this project in practice.

"Mecanum wheel is an omnidirectional wheel design for a land-based vehicle to move in any direction" (Wikipedia, 2021). The mecanum wheel was invented by Bengt Ilon, a Swedish company Mecanum AB engineer, in 1973. The mecanum wheels are designed to achieve a 180-degree turn without taking much space, while the regular wheel with a driving wheel would take more time and space. This is useful in small areas such as industrial factories with loads of

merchandise and a minor pathway for the forklift to move around and pick up the merchandise. However, the current mecanum wheel design can only operate on flat hard surfaces and perform poorly on rough terrains.

The requirement for the wheels:

1. Based on its current purpose, the mecanum wheel should be designed to support daily support. Each pack of conventional wheels weighs an average of 60 lbs, which is a total of 240 lbs. On average, an everyday sedan can support 3000 lbs. Therefore, in our project scale, the vehicle should be able to handle a load of (3000-240)/240 = 11.5 times more significant than the wheels' weight.
2. For primary purposes such as working indoors, the speed of the wheels is optional to be high. However, if we want to implement it on a daily vehicle, the rate needs to be much higher for local and highway travel.
3. For advanced purposes, the project must implement autonomous functions such as object-avoiding and self-parking systems on the vehicle. This application is also helpful for daily cars.

Risk assessment:

1. Motor: may not be strong enough or may burn during operating
    - Backup plan: using a more robust motor or regulating the current and voltage through the motor with capacitors.
2. Material: may not be strong enough to support the required weight
    - Backup plan: redesign the vehicle or use a different material

3. Design Process

a. Analysis (software) simulation, theory

The Mecanum wheel is designed based on the principle of how an object moves when we take the resultant force from the diagonal wheels. The most important thing to remember when designing a mecanum wheel vehicle is that all the surface-touching roller axes must point toward the vehicle's center.

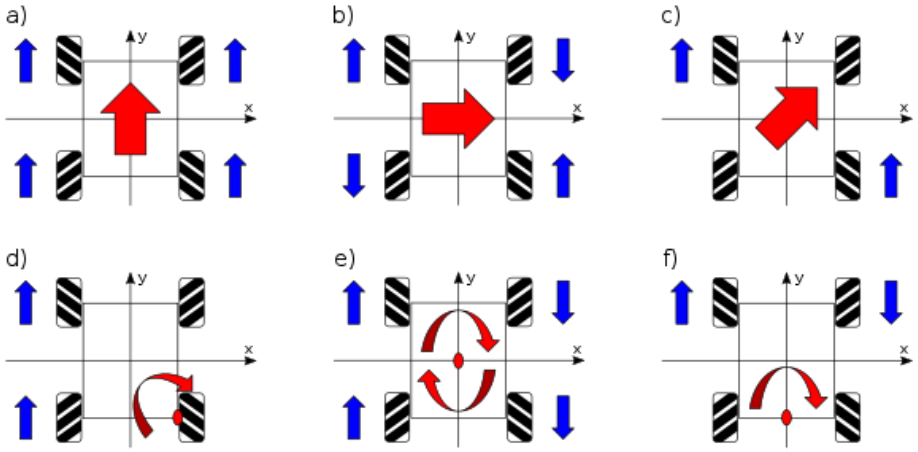


Figure 1. The Mecanum principle (Wikipedia)

To accomplish this, the Mecanum wheels have a unique design concept, with the actual wheel being a group of multiple smaller rollers at a specific angle, typically 45 degrees.

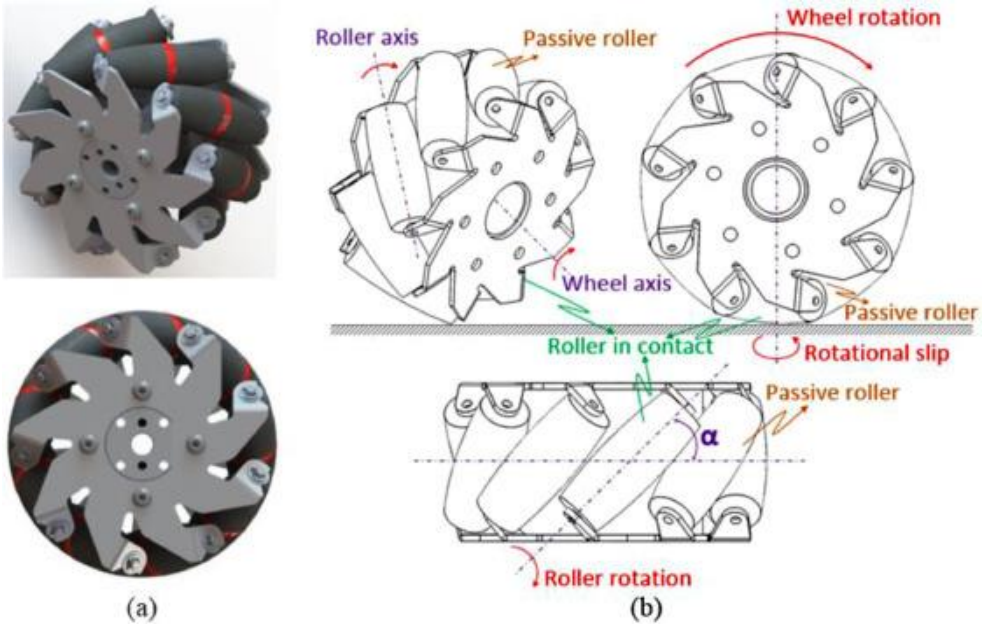
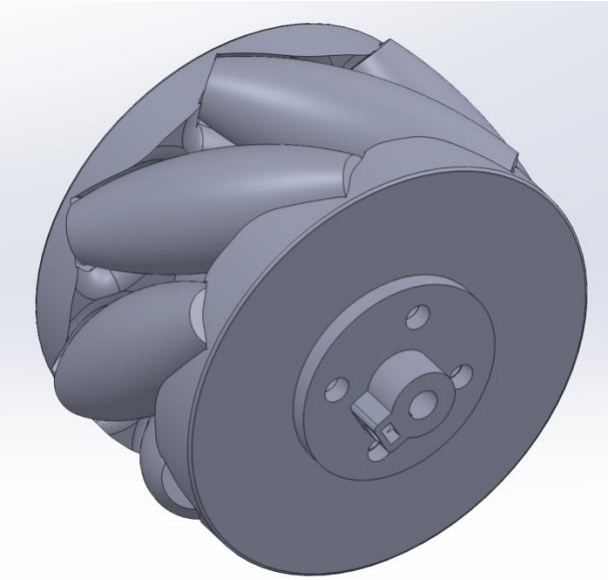


Figure 2. The Mecanum wheel design

b. Building the product:

The process is implemented efficiently to save building costs and time and to reduce mistakes to a minimum.

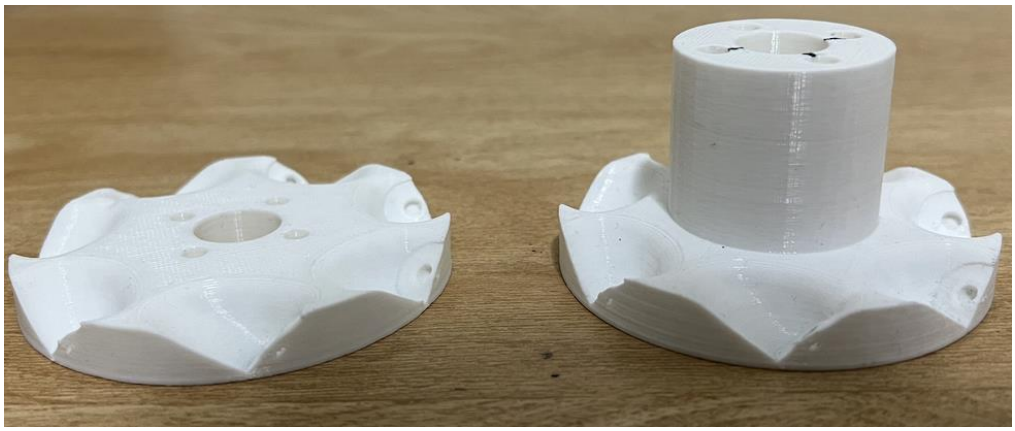
1. Using Solidworks, design the mecanum wheels, including the wheel cover and roller.



For the basic design, I followed the video named “Tutorial Mecanum Wheel SolidWorks PFA ULT 2020-2021” on YouTube, and then I modified some dimensions to fit the case of the project. I designed the wheel cover and the shaft coupler separately since I wanted to switch sides with the same wheel model to save time in the design process.

Figure 3. Mecanum wheel design using SolidWorks.

2. 3D print the wheel cover and rollers, cut the rod for the rollers’ axles, and assemble the wheels.



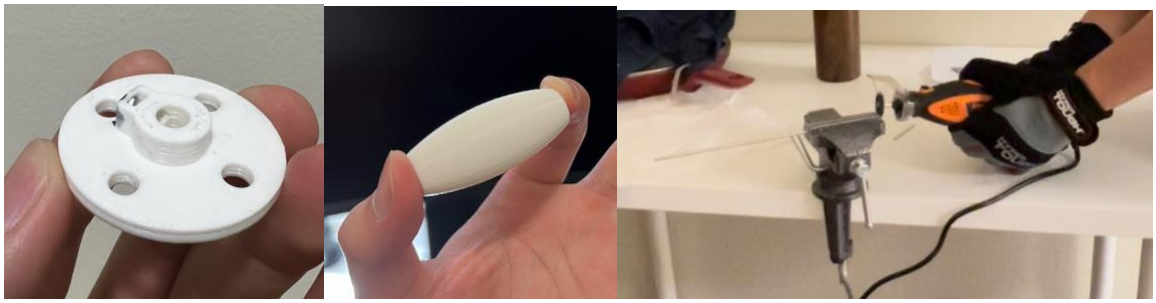
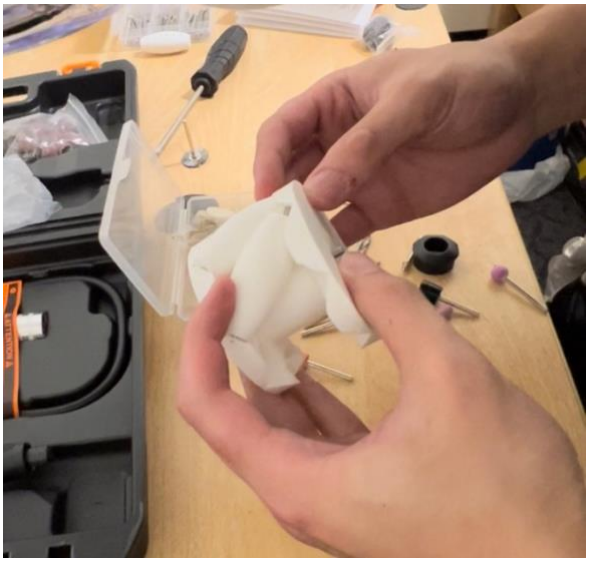


Figure 4. Manufacturing mecanum wheel process

This work needs to be done carefully due to its danger from tools and the precision of the axle length.



A few axles needed to be shortened, but overall, the first attempt at the model worked well.

However, this first design had many problems:

- The shaft coupler's first design was not good for the bolt to go through since the shaft holder and the bolt hole was in the same line, so they crossed each other, so I had to rotate the shaft 45 degrees compared to the holder.

- There is so much space between the two covers, and much of the rod material is revealed. Due to that, the wheel width is unnecessarily bigger, so I shortened the extrude inside by 5mm.
- The initial diameter of the rollers was too big, which made their clearance between each

other negative. Moreover, the roller's hole diameter was too big, giving them too much space to wiggle around; sometimes, they scratched each other and did not roll. It also created a clearance problem between the roller and the cover. Therefore, I reduced the roller's diameter and the hole's diameter.

- The shaft couple printed by PLA is not strong enough to hold the motor shaft, so I chose the universal mounting hub, which is made of steel and can hold the shaft using a better tighten screw.
- However, the hub can only hold the wheel on one axis. The wheel could wiggle around if the hole of the wheel cover had space, so I reduced the body's gap to fit the motor shaft by nearly 100%.

I adjusted the dimensions for bolts, motor shaft, and rollers' diameter and repeated until the result satisfied the requirement for good operation. Repeating the same procedure, I manufactured three more wheels, ready for testing.

Figure 5. Proper mecanum wheel

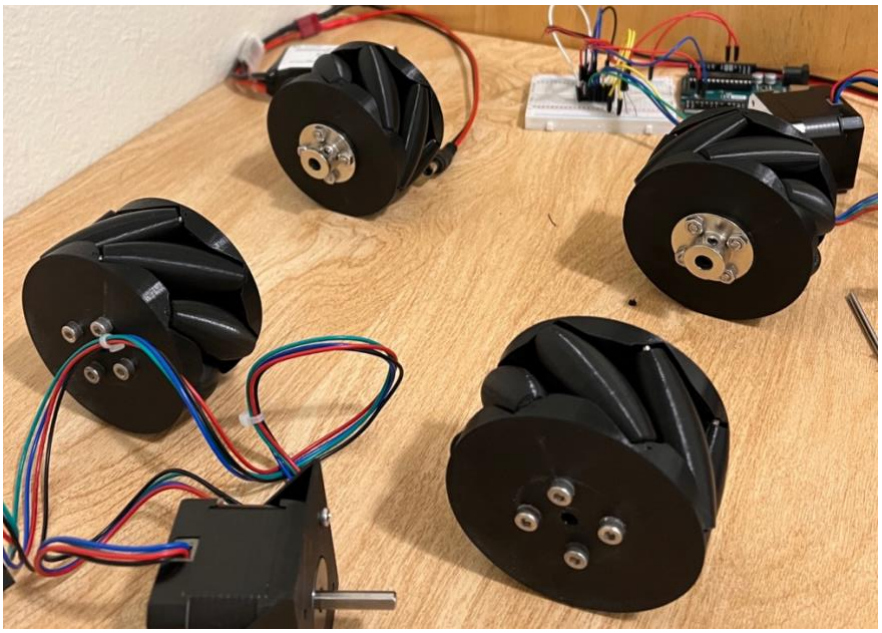


Figure 6. The whole mecanum wheel system

3. Using Solidworks, design the chassis.

The first idea when I designed the chassis was to create a platform with motor holders so that I could assemble the chassis and the wheels to test the fit and dimensions. It was a simple design with a small thickness to reduce the 3D print time. However, that thickness causes the chassis to be unstable and malfunction for testing.

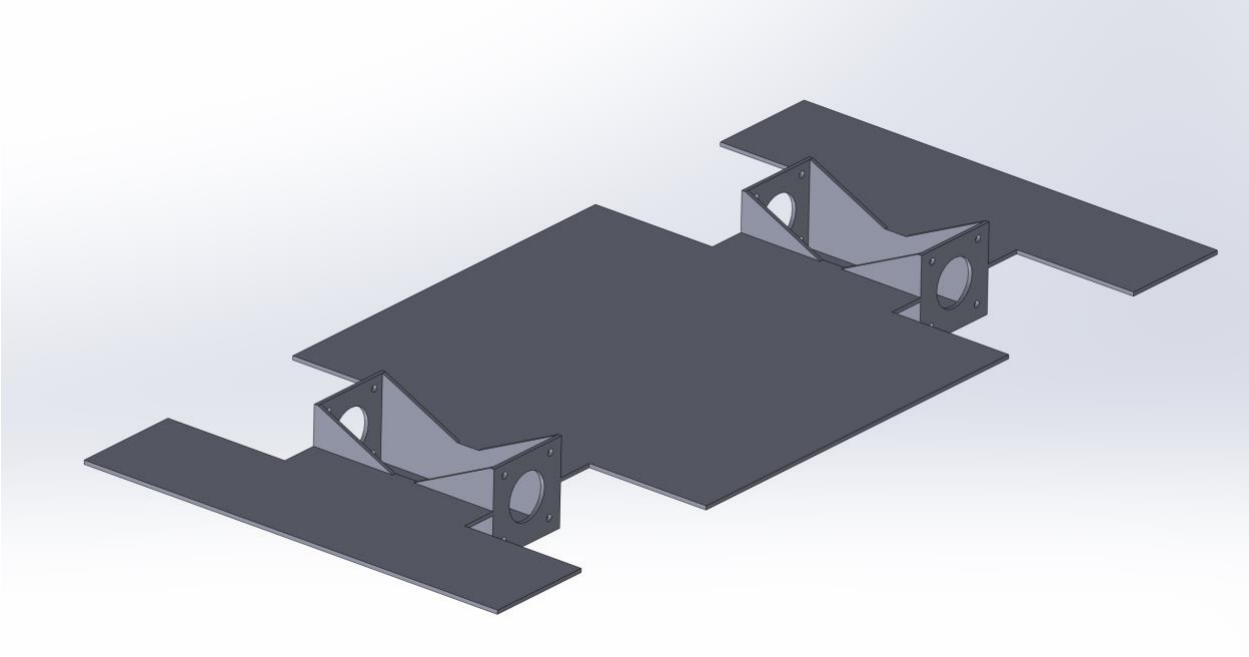


Figure 7. First chassis version

The second chassis was like the first one, with a minor update in the thickness. I also designed a chamber for Arduino Mega and the Li-Po battery beneath it because I planned to hide the power source beneath the furniture the same way a real Tesla car does.

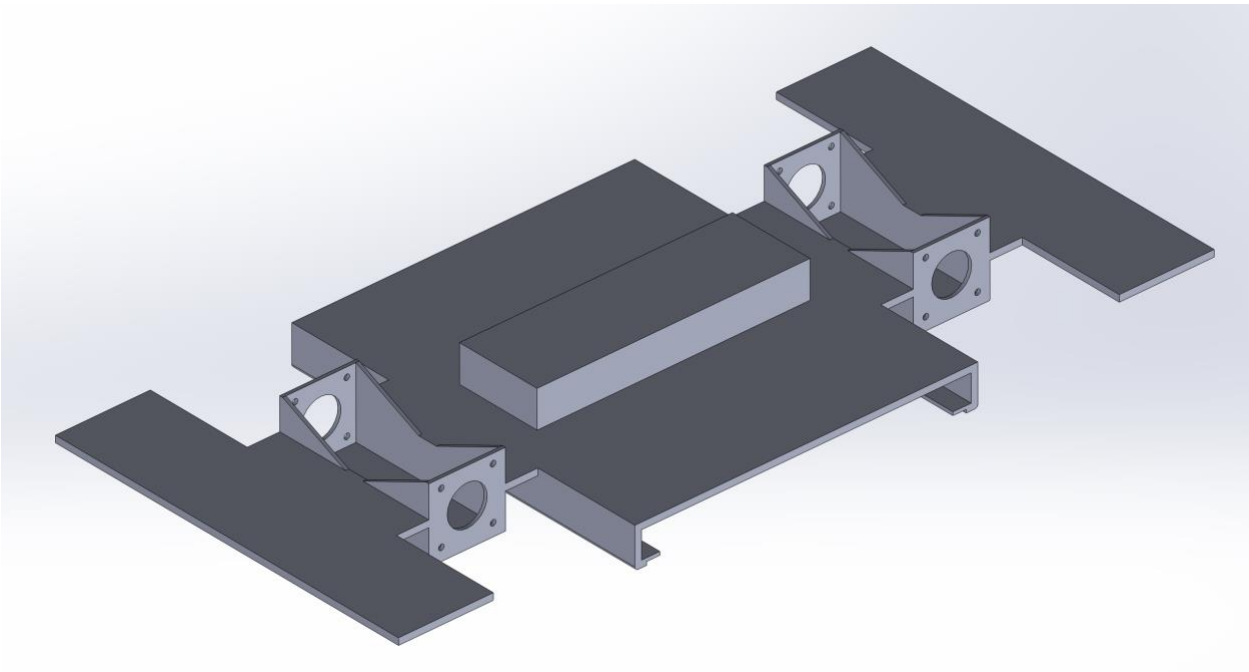


Figure 8. Second chassis version

However, I needed more time to finish other decorative features because I started focusing on the autonomous functions. Therefore, I modified the chassis to focus on the autonomous features, removing the chamber and increasing the thickness for stability. I designed the head and the tail more extensively to get more weight to balance with the inside body since the chassis body was acting as a rigid body and kept bending inward due to a load of electronic components. The spacing between the wheels is also increased to fit the simple design of the Tesla Roadster car. I also designed some attachment parts on the edge of the chassis for later assembly. The previous versions of the chassis were designed to hold stepper motors. However, stepper motors were unsuitable for this project and were changed to DC motors during the testing process. Therefore, this final version has different holders for stepper motors.

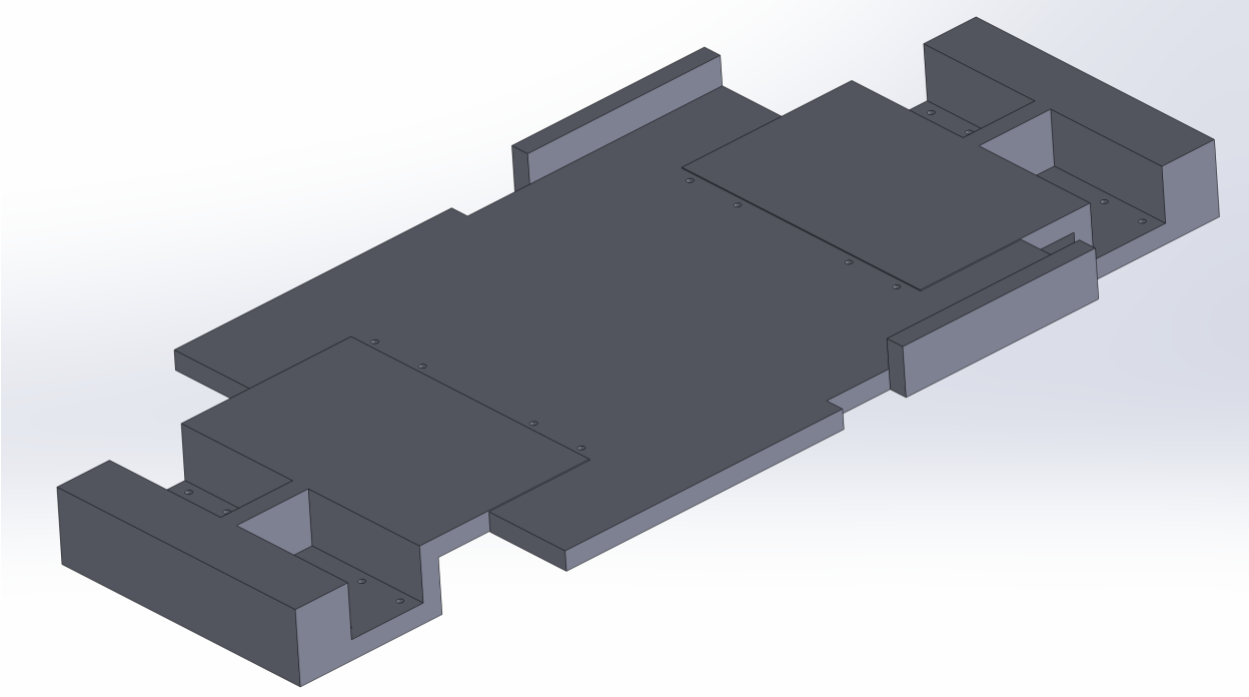
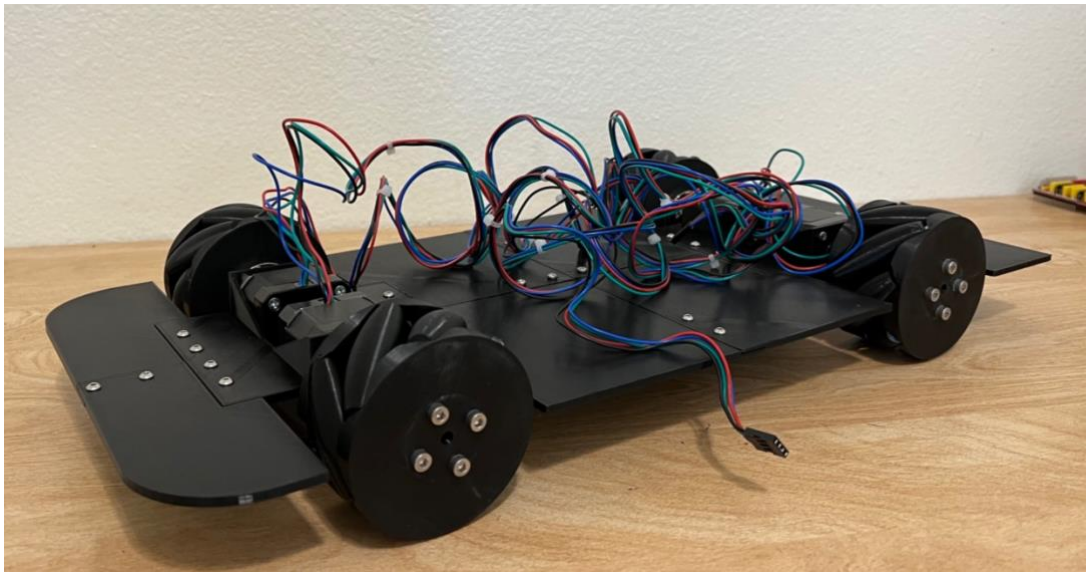


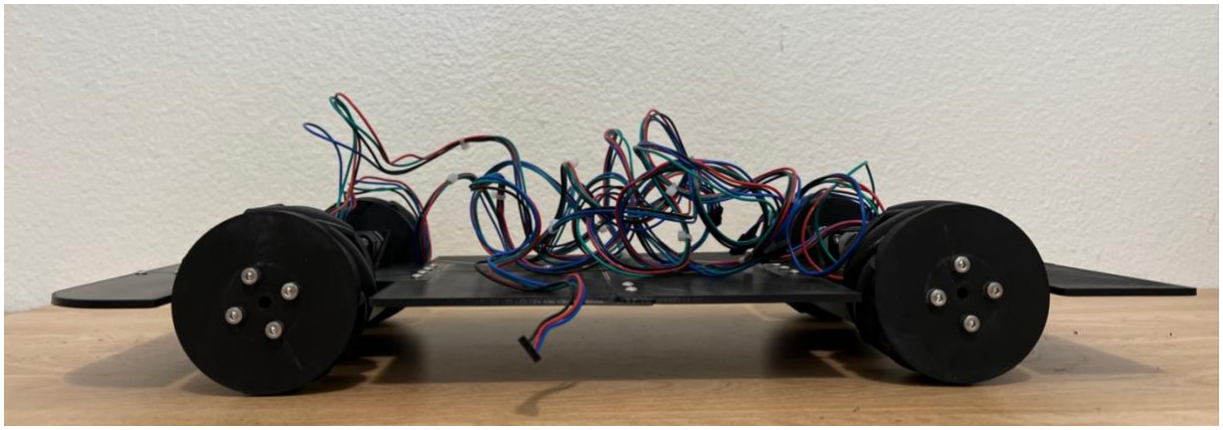
Figure 9. Final chassis version

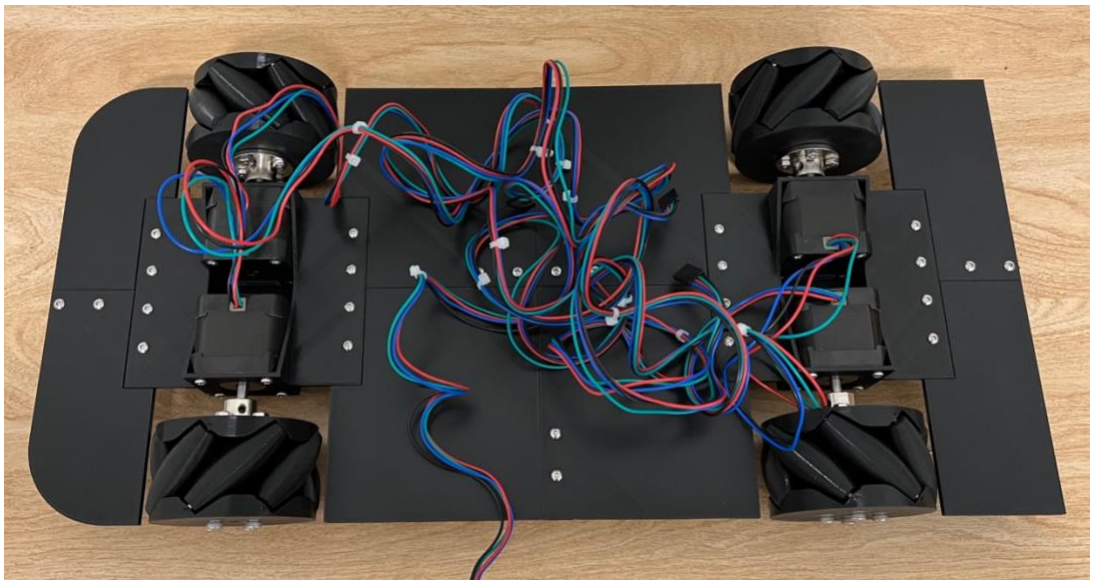
4. 3D print and assemble the chassis.

I had to split and print my chassis as multiple parts because my printer was too small.

Figure 10. 3D print the chassis as parts.







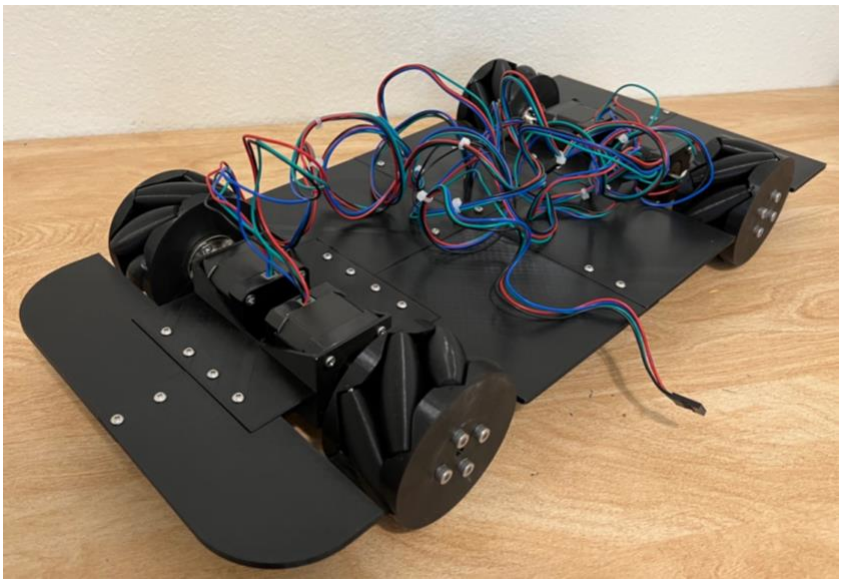
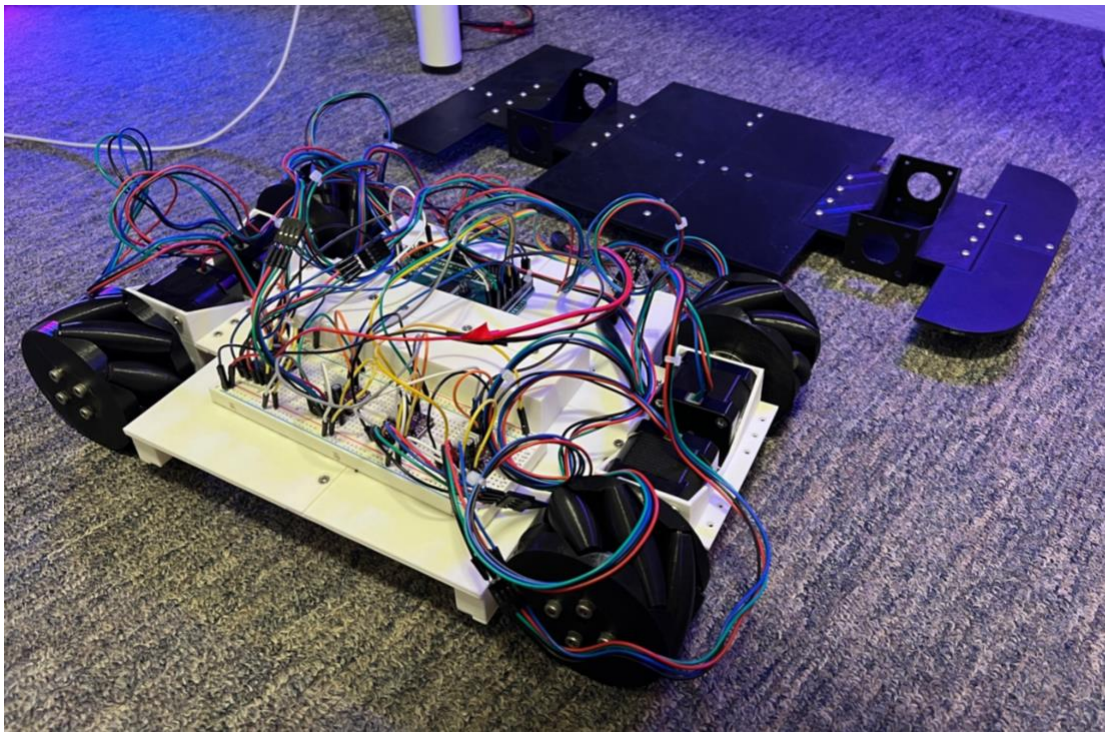


Figure 11-14. The first chassis



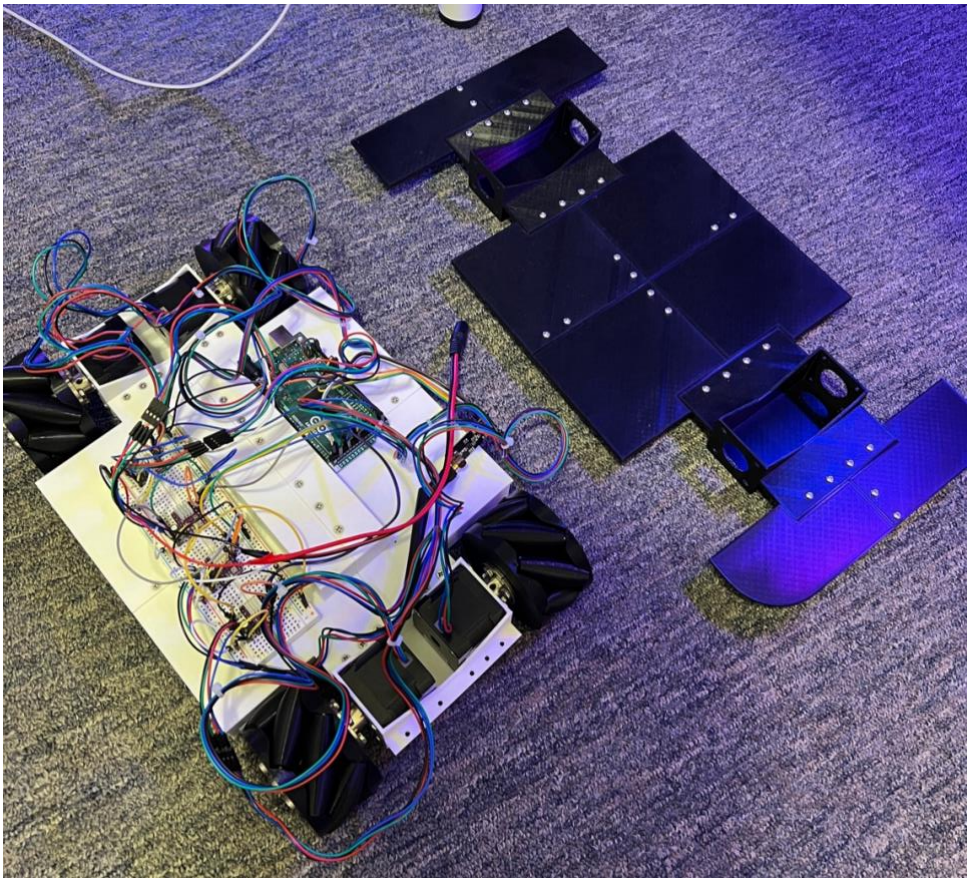


Figure 15-16. The second chassis

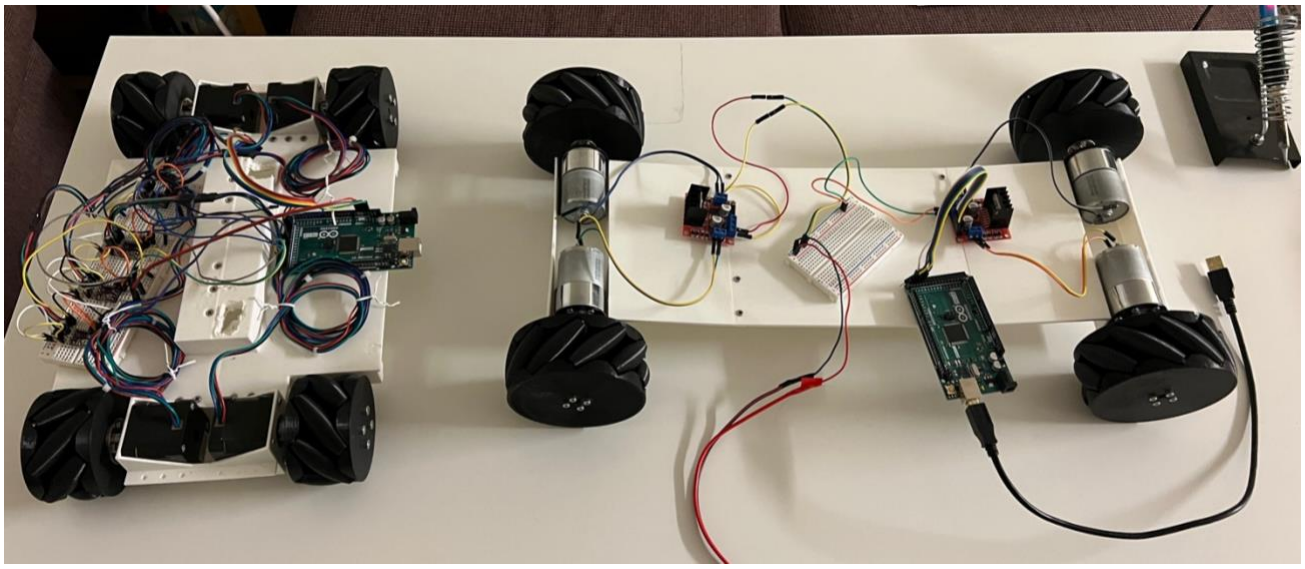


Figure 17. Prototype of final chassis

5. Using Solidworks, design the Tesla Roadster body based on the diagram from Google.

Since the Tesla Roadster has yet to be released, I must design it on SolidWorks from an old diagram found on Google.

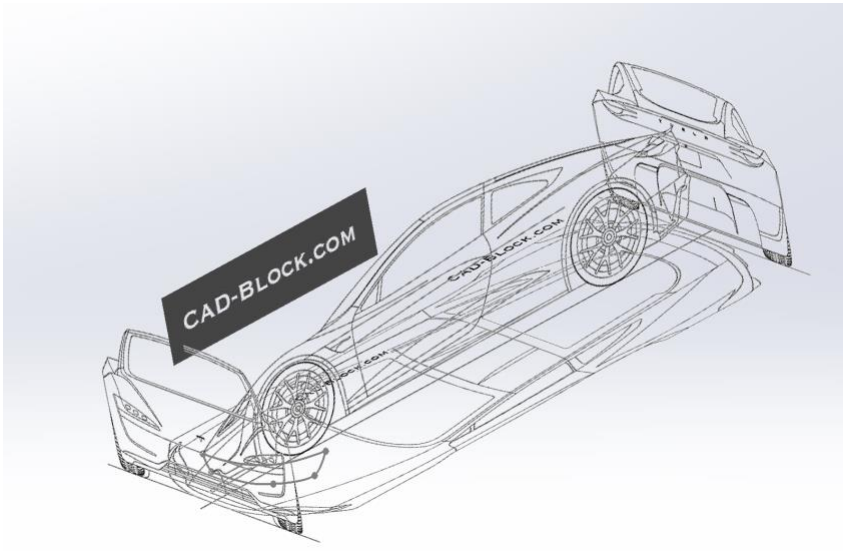


Figure 18. The Tesla Roadster's diagram from Google

I tried to model the Tesla Roadster using the Loft Surface feature in SolidWorks. The result looks magnificent, but I could not get it enclosed to save as a .stl file for 3D printing. There must have been some incompatible errors between the Loft Surface and the 3D sketch model.

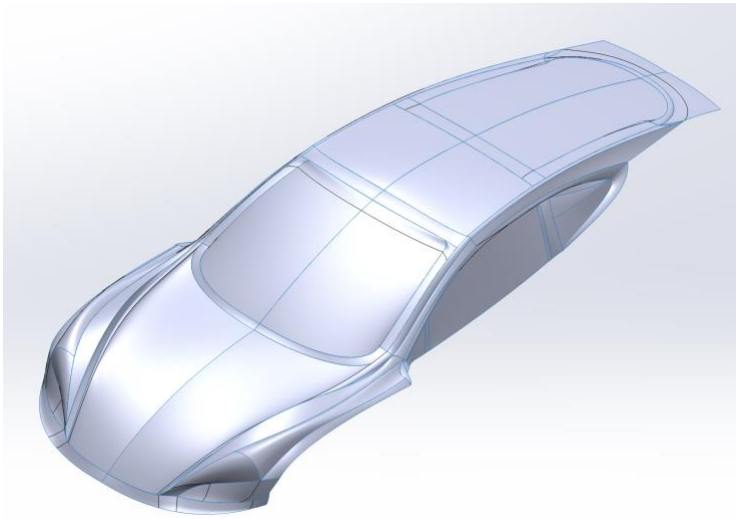


Figure 19. The first attempt to model the Tesla Roadster

Therefore, I must do the simple way of extruding and cutting since I would not have enough time to troubleshoot the problem with the first attempt. I made the result look the most like the diagram and enough for testing.

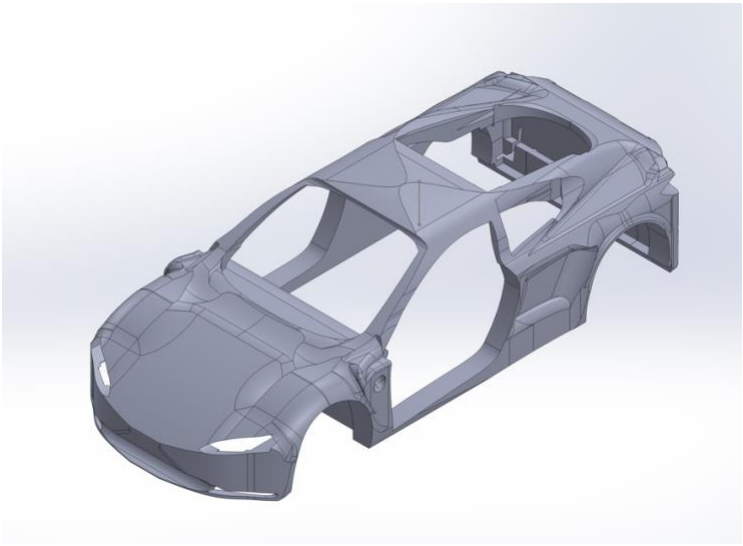


Figure 20. Result of modeling Tesla Roadster

6. 3D print and assemble the Tesla Roadster body.

Having the same problem with the small printer's bed size, I had to split the vehicle's body into multiple parts again and used glue to assemble it.

Figure 21-22. Assemble the Tesla Roadster from 3D-printed parts.

7. Assemble the final mechanical design.

The final mechanical design added four ultrasonic sensors to do object avoidance and self-parking in parallel. Due to the requirements for results, I asked Dr. Boskovich for permission to use the better quality mecanum wheels instead of the 3D printed ones with PLA material.

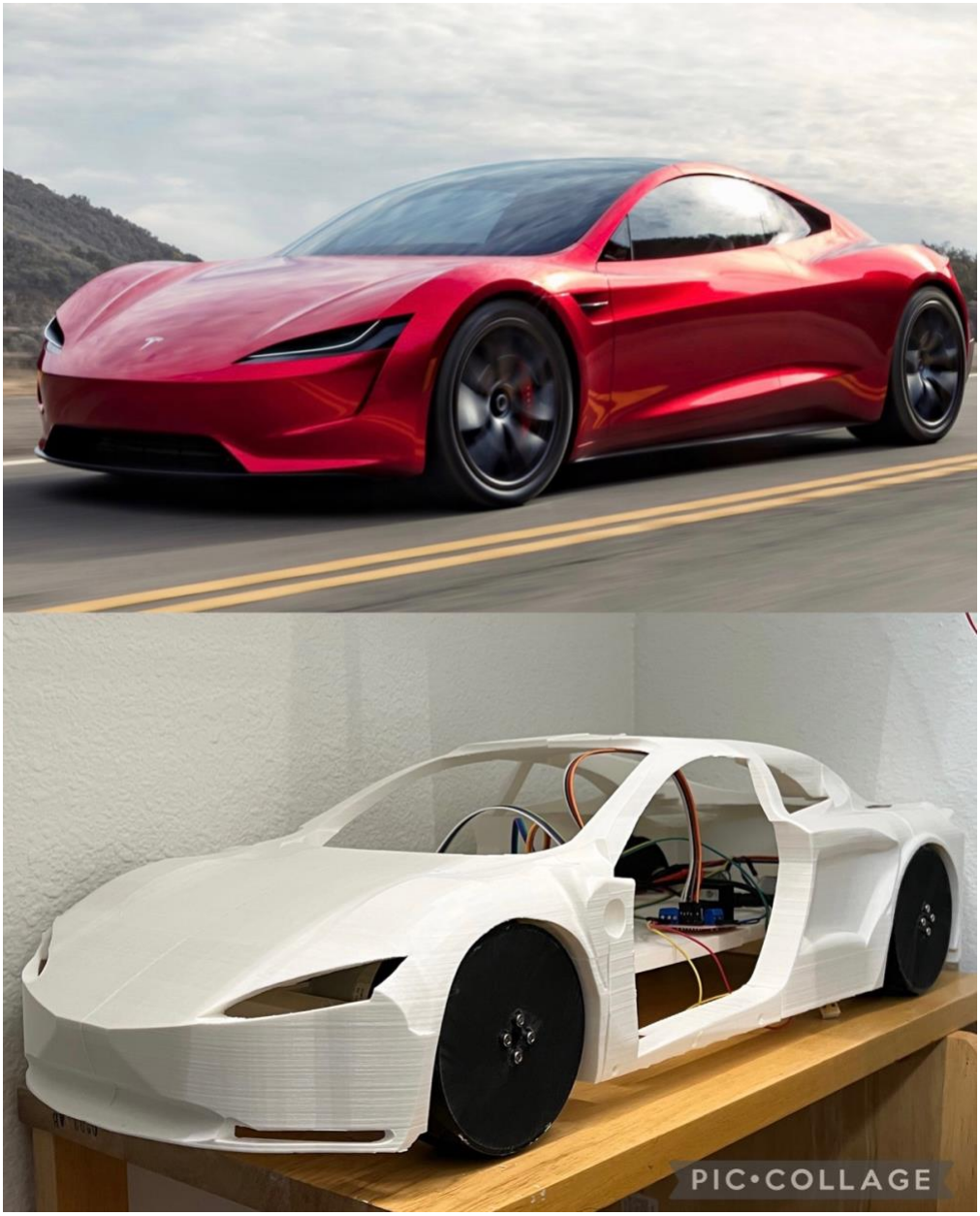


Figure 23. Comparison of final assemble with a photo of Tesla Roadster

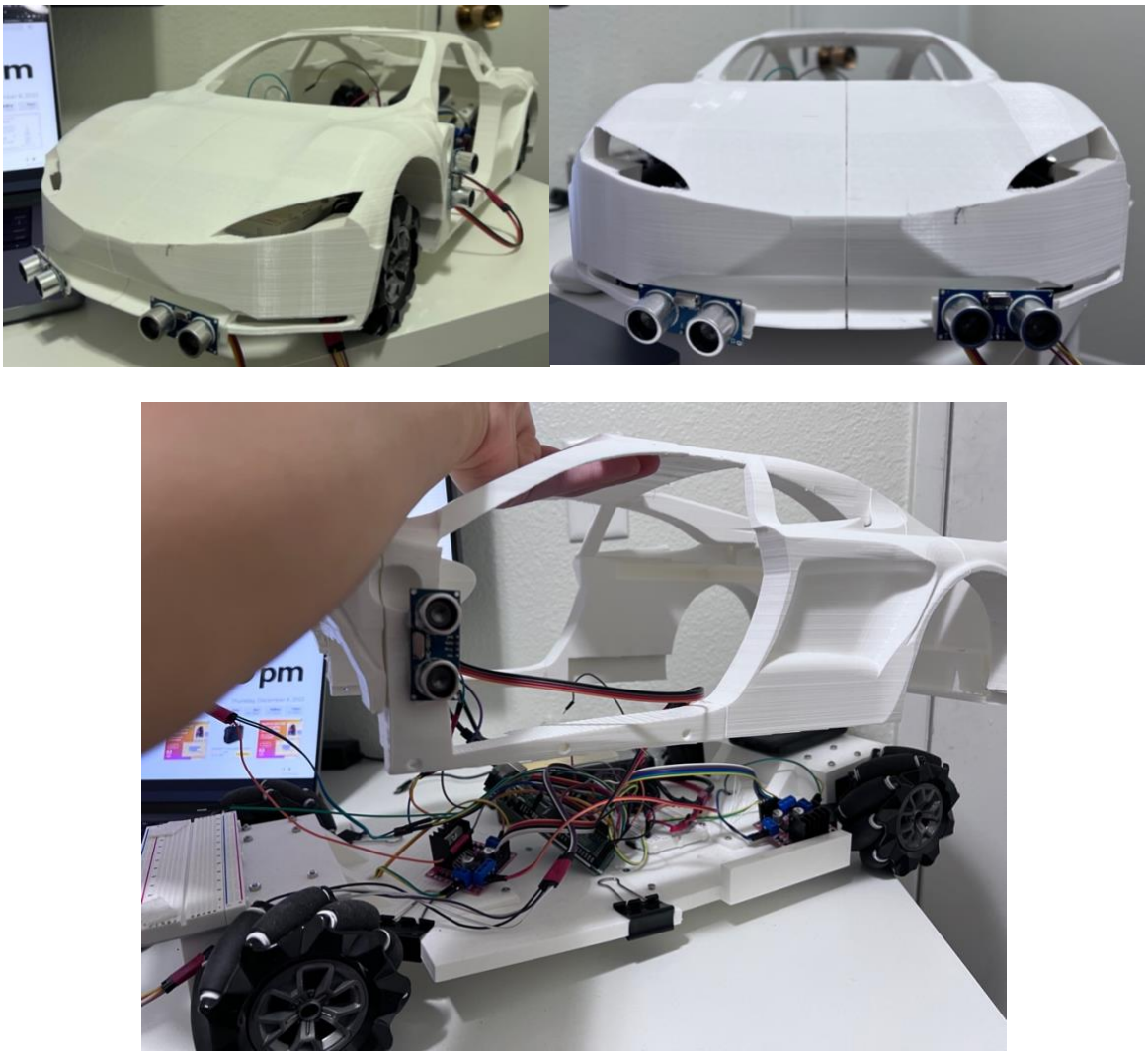


Figure 24-26. Final mechanical design with ultrasonic sensors

8. Connect all wheels, DC motors, H-bridges, and other electronic components to the Arduino board.

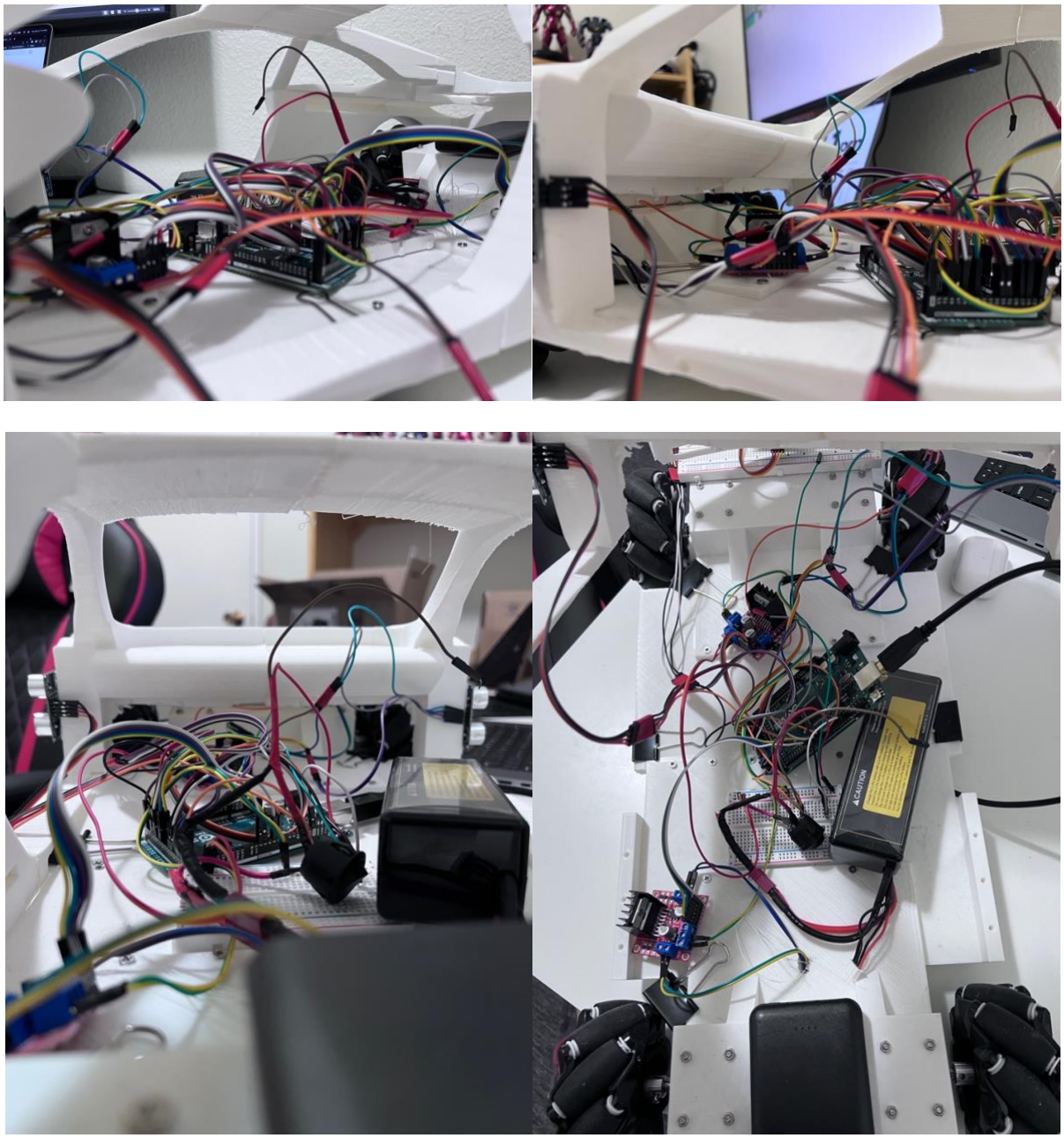


Figure 27-28. Inside view with electronic components

9. Test to ensure all wheels operate as expected together from the Arduino board.

I must ensure the DC motors are powered enough to run and support the body's load and chassis.

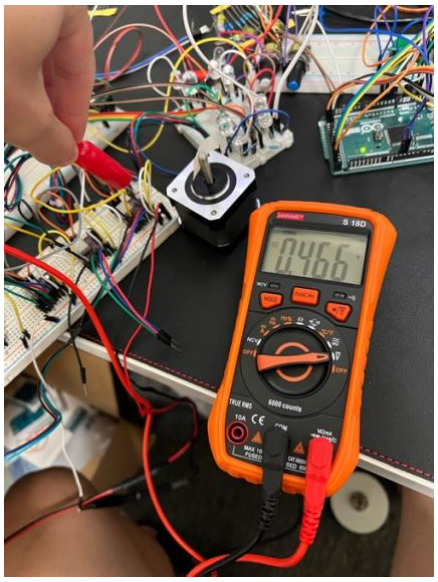


Figure 29. Testing

10. Build a handheld controller to control the wheel using a transceiver and receiver.

I made a controller with a potentiometer to switch between modes, including static (when the vehicle is not moving), manual control (the controller can control the vehicle), and autonomous functions (the car will automatically operate on its own). The manual control uses the NRF24L01 radio communication module to communicate with the same one on the vehicle. The range is about 160 ft and can be extended further using the antenna.

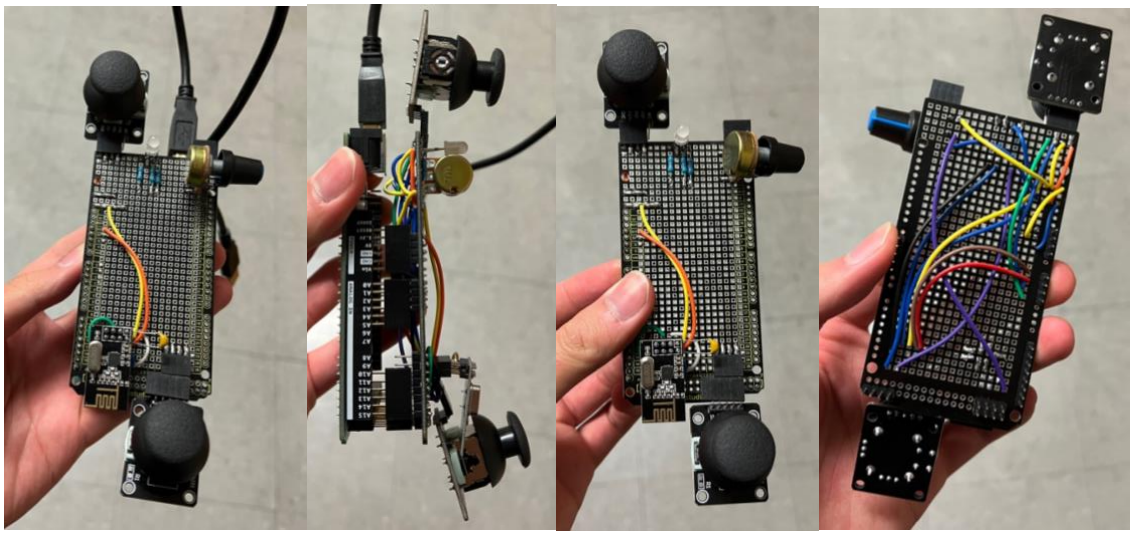


Figure 30. Controller for manual control

11. Program and test to ensure the following features:
    - The vehicle can be remote controlled using the controller

The code for the controller (transmitter):

transmitter.ino

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

//DECLARE VARIABLES
int vehicleState;
int chargingState;
int wiperState;

//Joysticks
#define j1PotX A2
#define j1PotY A3
#define j2PotX A0
#define j2PotY A1

//Servo
int wheelAngle;
int turnSteer;
int sideSteer;
int hoodAngle;
int leftDoorAngle;
int rightDoorAngle;
int leftWindowAngle;
int rightWindowAngle;

//Buttons
int leftSignalButtonPin = 3;
int newLeftSignalButton;
int oldLeftSignalButton;
int rightSignalButtonPin = 4;
int newRightSignalButton;
int oldRightSignalButton;
int emergencyButtonPin = 5;
int newEmergencyButton;
int oldEmergencyButton;
int chargingButtonPin = 2;
int newChargingButton;
int oldChargingButton;
int wiperButtonPin = 6;
int newWiperButton;
int oldWiperButton;

//Potentiometers
#define vehicleStatePin A15
int vehicleStatePot;
#define vehicleSpeedPin A14
#define headlightPin A13
int headlightPot;
#define hoodPin A12
#define leftDoor A11
#define rightDoor A10
#define leftWindow A9
#define rightWindow A8

//LEDs
int redPin = 27;
int greenPin = 25;
int bluePin = 23;
int leftSignalLEDState = 0;
int rightSignalLedState = 0;
int emergencyLEDState = 0;
```

```
//RF24
RF24 radio(49, 48);   // nRF24L01 (CE, CSN)
//const byte address[][6] = {"00001","00002","00003"}; // Address
const byte address[6] = "00001";

//CONSTRUCT DATA PACKAGE
// Max size of this struct is 32 bytes - NRF24L01 buffer limit
struct Data_Package {
  byte j1PotX;
  byte j1PotY;
  byte j2PotX;
  byte j2PotY;
  byte vehicleState;
  byte vehicleSpeed;
  byte wheelAngle;
  byte headlightPot;
  byte leftSignalLEDState = 0;
  byte rightSignalLEDState = 0;
  byte emergencyLEDState = 0;
  byte chargingState = 0;
  byte hoodAngle;
  byte wiperState = 0;
  byte leftDoorAngle;
  byte rightDoorAngle;
  byte leftWindowAngle;
  byte rightWindowAngle;
  byte leftSignalLEDState_4;
  byte rightSignalLEDState_4;
};
Data_Package data; //Create a variable with the above structure

//SETUP
void setup() {
  //Serial
  Serial.begin(9600);

  //Radio communication
  radio.begin();
//  radio.openWritingPipe(address[1]); // address used is "00002"
  radio.openWritingPipe(address);
  radio.setAutoAck(false);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.setChannel(9);

  // Set initial default values
  data.j1PotX = 127; // Values from 0 to 255. When Joystick is in resting position, the value is in the middle, or 127.
  data.j1PotY = 127;
  data.j2PotX = 127;
  data.j2PotY = 127;
  data.vehicleState = 1;
  data.vehicleSpeed = 100;
  data.wheelAngle = 88;
  data.headlightPot = 0;
  data.leftSignalLEDState = 0;
  data.rightSignalLEDState = 0;
  data.emergencyLEDState = 0;
  data.chargingState = 0;
  data.hoodAngle = 0;
  data.wiperState = 0;
  data.leftDoorAngle = 0;
  data.rightDoorAngle = 0;
```

```
  data.rightDoorAngle = 0;
  data.leftWindowAngle = 0;
  data.rightWindowAngle = 0;

  //pinModes
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  pinMode(headlightPin, INPUT);
  pinMode(leftSignalButtonPin, INPUT);
  pinMode(rightSignalButtonPin, INPUT);
  pinMode(emergencyButtonPin, INPUT);
}

//EXECUTION
void loop() {
  radio.stopListening();  //Set the module as trasnmitter
  read_vehicleState();
  read_vehicleSpeed();
  read_headlightBrightness();
  read_joystick();
  read_dataSteeringWheel();
  read_leftSignalState();
  read_rightSignalState();
  read_emergencyState();
  read_chargingState();
  read_hoodAngle();
  read_wiperState();
  read_leftDoorAngle();
  read_rightDoorAngle();
  read_leftWindowAngle();
  read_rightWindowAngle();
  radio.write(&data, sizeof(Data_Package));   // Send the whole data from the structure to the receiver
}



//SWITCH CASE FUNCTION
void read_vehicleState() {
  vehicleStatePot = map(analogRead(vehicleStatePin), 0, 1023, 0, 2);
  if (vehicleStatePot >= 0 && vehicleStatePot < 1) {
    digitalWrite(redPin, HIGH);
    digitalWrite(greenPin, LOW);
    digitalWrite(bluePin, LOW);
    vehicleState = 1;
  }
  else if (vehicleStatePot >= 1 && vehicleStatePot <= 2) {
    digitalWrite(redPin, LOW);
    digitalWrite(greenPin, HIGH);
    digitalWrite(bluePin, LOW);
    vehicleState = 2;
  }
}

//READ SIGNAL STATE FUNCTION
void read_leftSignalState() {
  newLeftSignalButton = digitalRead(leftSignalButtonPin);
  if (oldLeftSignalButton == 0 && newLeftSignalButton == 1) {
    if (data.leftSignalLEDState == 0) {
      data.leftSignalLEDState = 1;
    }
```

```
    else {
      data.leftSignalLEDState = 0;
    }
  }
  oldLeftSignalButton = newLeftSignalButton;
}
void read_rightSignalState() {
  newRightSignalButton = digitalRead(rightSignalButtonPin);
  if (oldRightSignalButton == 0 && newRightSignalButton == 1) {
    if (data.rightSignalLEDState == 0) {
      data.rightSignalLEDState = 1;
    }
    else {
      data.rightSignalLEDState = 0;
    }
  }
  oldRightSignalButton = newRightSignalButton;
}
void read_emergencyState() {
  newEmergencyButton = digitalRead(emergencyButtonPin);
  if (oldEmergencyButton == 0 && newEmergencyButton == 1) {
    if (data.emergencyLEDState == 0) {
      data.emergencyLEDState = 1;
    }
    else {
      data.emergencyLEDState = 0;
    }
  }
  oldEmergencyButton = newEmergencyButton;
}

//READ CHARGING STATE FUNCTION
void read_chargingState() {
  newChargingButton = digitalRead(chargingButtonPin);
  if (oldChargingButton == 0 && newChargingButton == 1) {
    if (data.chargingState == 0) {
      data.chargingState = 1;
    }
    else {
      data.chargingState = 0;
    }
  }
  oldChargingButton = newChargingButton;
}

//READ DOOR ANGLE
void read_leftDoorAngle() {
  data.leftDoorAngle = map(analogRead(leftDoor), 0, 1023, 0, 179);
}
void read_rightDoorAngle() {
  data.rightDoorAngle = map(analogRead(rightDoor), 0, 1023, 0, 179);
}

//READ WINDOW ANGLE
void read_leftWindowAngle() {
  data.leftWindowAngle = map(analogRead(leftWindow), 0, 1023, 0, 179);
}
void read_rightWindowAngle() {
  data.rightWindowAngle = map(analogRead(rightWindow), 0, 1023, 0, 179);
}
```

```
246  //READ WIPER STATE FUNCTION
247  void read_wiperState() {
248    newWiperButton = digitalRead(wiperButtonPin);
249    if (oldWiperButton == 0 && newWiperButton == 1) {
250      if (data.wiperState == 0) {
251        data.wiperState = 1;
252      }
253      else {
254        data.wiperState = 0;
255      }
256    }
257    oldWiperButton = newWiperButton;
258  }
259
260  //READ HEADLIGHT FUNCTION
261  void read_headlightBrightness() {
262    data.headlightPot = map(analogRead(headlightPin), 0, 1023, 0, 255);
263  }
264
265  //READ VEHICLE SPEED FUNCTION
266  void read_vehicleSpeed() {
267    data.vehicleSpeed = map(analogRead(vehicleSpeedPin), 0, 1023, 0, 255);
268  }
269
270  //READ HOOD ANGLE FUNCTION
271  void read_hoodAngle() {
272    data.hoodAngle = map(analogRead(hoodPin), 0, 1023, 0, 179);
273  }
274
275  //READ JOYSTICK FUNCTION
276  void read_joystick() {
277    // Read all analog inputs and map them to one Byte value
278    data.j1PotX = map(analogRead(j1PotX), 0, 1023, 0, 255); // Convert the ana
279    Serial.println(data.j1PotX);
280    data.j1PotY = map(analogRead(j1PotY), 0, 1023, 0, 255);
281    Serial.println(data.j1PotY);
282    data.j2PotX = map(analogRead(j2PotX), 0, 1023, 0, 255); // Convert the ana
283    data.j2PotY = map(analogRead(j2PotY), 0, 1023, 0, 255);
284  }
285
286  //READ DATA FOR STEERING WHEEL FUNCTION
287  void read_dataSteeringWheel() {
288    turnSteer = map(analogRead(j2PotX), 0, 1023, 0, 179);
289    sideSteer = map(analogRead(j1PotX), 0, 1023, 0, 179);
290    if (sideSteer < 80 || sideSteer > 100) {
291      data.wheelAngle = sideSteer;
292    }
293    if (turnSteer < 80 || turnSteer > 100) {
294      data.wheelAngle = turnSteer;
295    }
296  }
297
```

The code for vehicle in manual control (receiver):

receiver1.ino

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <L298NX2.h>

//DECLARE VARIABLES
float dt1 = 1.5;

//RF24
RF24 radio(49, 48);   // nRF24L01 (CE, CSN)
const byte address[6] = "00001";
unsigned long lastReceiveTime = 0;
unsigned long currentTime = 0;



// Pin definition
const unsigned int EN_FL = 23;
const unsigned int IN1_FL = 25;
const unsigned int IN2_FL = 24;

const unsigned int IN1_FR = 27;
const unsigned int IN2_FR = 26;
const unsigned int EN_FR = 22;

const unsigned int EN_RL = 29;
const unsigned int IN1_RL = 31;
const unsigned int IN2_RL = 33;

const unsigned int IN1_RR = 30;
const unsigned int IN2_RR = 32;
const unsigned int EN_RR = 28;

// Initialize both motors
L298NX2 Fmotors(EN_FL, IN1_FL, IN2_FL, EN_FR, IN1_FR, IN2_FR);
L298NX2 Rmotors(EN_RL, IN1_RL, IN2_RL, EN_RR, IN1_RR, IN2_RR);

// Initial speed
unsigned short FLspeed = 128;
unsigned short FRspeed = 128;
unsigned short RLspeed = 128;
unsigned short RRspeed = 128;

int wheelSpeed;

//CONSTRUCT DATA PACKAGE
// Max size of this struct is 32 bytes - NRF24L01 buffer limit
struct Data_Package {
  byte j1PotX;
  byte j1PotY;
  byte j2PotX;
  byte j2PotY;
  byte vehicleState;
  byte vehicleSpeed;
  byte wheelAngle;
  byte headlightPot;
  byte leftSignalLEDState;
  byte rightSignalLEDState;
  byte emergencyLEDState;
  byte chargingState;
  byte hoodAngle;
```

```
  byte wiperState;
  byte leftDoorAngle;
  byte rightDoorAngle;
  byte leftWindowAngle;
  byte rightWindowAngle;
  byte leftSignalLEDState_4;
  byte rightSignalLEDState_4;
  byte vehicleMovement;
};
Data_Package data; //Create a variable with the above structure

//SETUP
void setup() {
  //Radio communication
  radio.begin();
  radio.openReadingPipe(1, address);
  radio.setAutoAck(false);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.setChannel(9);

  //Serial
  Serial.begin(9600);
}

//EXECUTION
void loop() {
  checkConnection();  // Check whether there is data to be received
  radio.startListening(); //  Set the module as receiver
  if (radio.available()) {
    radio.read(&data, sizeof(Data_Package)); // Read the whole data and store it into the 'data' structure
    lastReceiveTime = millis(); // At this moment we have received the data
  }
  switch (data.vehicleState) {
    case 2:
      Serial.println("case 2");
      delay(dt1);
      Serial.print("j1PotX = ");
      Serial.println(data.j1PotX);
      Serial.print("j1PotY = ");
      Serial.println(data.j1PotY);
      runMotor_2();
      delay(dt1);
      break;
    case 3:
      runMotor_3();
      break;
    case 4:
      break;
  }
}


////STEPPER MOTOR FUNCTION
void runMotor_3() {
  // if (data.vehicleMovement == 1) {
  //   moveSidewaysRight();
  //   delay(10);
  // }
  // else {
  //   stopMoving();
```

```
  // }
  //Execute the steps
  Fmotors.setSpeedA(255);
  Fmotors.setSpeedB(255);
  Rmotors.setSpeedA(255);
  Rmotors.setSpeedB(255);
}
void runMotor_2() {
  // if (data.j1PotY > 160) {
  //   rotateLeft();
  //   // delay(dt1);
  // }
  // else if (data.j1PotY < 100) {
  //   rotateRight();
  //   // delay(dt1);
  // }
  if (data.j1PotX > 160) {
    turnLeft();
    // delay(dt1);
  }
  else if (data.j1PotX < 100) {
    turnRight();
    // delay(dt1);
  }
  else if (data.j2PotY < 30) {
    moveBackward();
    // delay(dt1);
  }
  else if (data.j2PotY > 220) {
    moveForward();
    // delay(dt1);
  }
  else if (data.j2PotX < 100) {
    rotateRight();
    // delay(dt1);
  }
  else if (data.j2PotX > 160) {
    rotateLeft();
    // delay(dt1);
  }
  else {
    stopMoving();
    // delay(dt1);
  }
  // Execute the steps
  Fmotors.setSpeedA(FLspeed);
  Fmotors.setSpeedB(FRspeed);
  Rmotors.setSpeedA(RRspeed);
  Rmotors.setSpeedB(RLspeed);
}
void moveForward() {
  Fmotors.backwardA();
  Fmotors.backwardB();
  Rmotors.forwardA();
  Rmotors.backwardB();
}
void moveBackward() {
  Fmotors.forwardA();
  Fmotors.forwardB();
  Rmotors.backwardA();
  Rmotors.forwardB();
```

```
184   }
185   void turnRight() {
186     Fmotors.backwardB();
187     Rmotors.forwardA();
188   }
189   void turnLeft() {
190     Fmotors.backwardA();
191     Rmotors.backwardB();
192   }
193   void rotateLeft() {
194     Fmotors.forwardA();
195     Fmotors.backwardB();
196     Rmotors.forwardA();
197     Rmotors.forwardB();
198   }
199   void rotateRight() {
200     Fmotors.forwardB();
201     Fmotors.backwardA();
202     Rmotors.backwardA();
203     Rmotors.backwardB();
204   }
205   void moveRightForward() {
206     Fmotors.backwardA();
207     Rmotors.forwardA();
208   }
209   void moveRightBackward() {
210
211   }
212   void moveLeftForward() {
213     Fmotors.backwardB();
214     Rmotors.backwardB();
215   }
216   void moveLeftBackward() {
217
218   }
219   void stopMoving() {
220     Fmotors.stop();
221     Rmotors.stop();
222   }
223
224   //CHECK CONNECTION FUNCTION
225   void checkConnection() {
226     // Check whether we keep receving data, or we have a connection between the two modules
227     currentTime = millis();
228     if ( currentTime - lastReceiveTime > 1000 ) { // If current time is more then 1 second since we
229       resetData(); // If connection is lost, reset the data. It prevents unwanted behavior, for exa
230     }
231   }
232
233   //RESET DATA FUNCTION
234   void resetData() {
235     // Reset the values when there is no radio connection - Set initial default values
236     data.j1PotX = 127;
237     data.j1PotY = 127;
238     data.j2PotX = 127;
239     data.j2PotY = 127;
240     data.vehicleState = 1;
241     data.vehicleSpeed = 100;
242     data.wheelAngle = 88;
243     data.headlightPot = 0;
244     data.leftSignalLEDState = 0;
```

```
245     data.rightSignalLEDState = 0;
246     data.emergencyLEDState = 0;
247     data.chargingState = 0;
248     data.hoodAngle = 0;
249     data.wiperState = 0;
250     data.leftDoorAngle = 0;
251     data.rightDoorAngle = 0;
252     data.leftWindowAngle = 0;
253     data.rightWindowAngle = 0;
254     data.leftSignalLEDState_4 = 0;
255     data.rightSignalLEDState_4 = 0;
256
257 }
```

- The car can run autonomously and implement activities like object avoidance and self-parking in parallel.

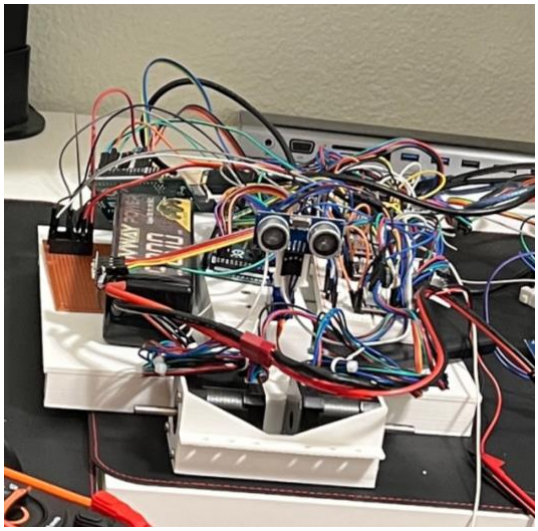


Figure 31. Prototype for autonomous functions

Object avoidance:

object avoidance.ino

```
#include <L298NX2.h>
#include <NewPing.h>

#define MAX_DISTANCE 500

#define SONAR_NUM 4      // Number of sensors.

unsigned int cm[SONAR_NUM];         // Where the ping distances are stored.
unsigned long currentTimer = 0;
unsigned long previousTimer = 0;

NewPing sonar[SONAR_NUM] = {   // Sensor object array.
  NewPing(48, 49, MAX_DISTANCE), // Each sensor's trigger pin, echo pin, and max distance to ping.
  NewPing(46, 47, MAX_DISTANCE),
  NewPing(44, 45, MAX_DISTANCE),
  NewPing(40, 41, MAX_DISTANCE)
};

int n1 = random(150, 350);
int n2 = random(150, 175);


float temp = 23.7; //Temp in C degree
float factor = sqrt(1 + temp / 273.15) / 60.368;

const unsigned int EN_FL = 2;
const unsigned int IN1_FL = 25;
const unsigned int IN2_FL = 24;

const unsigned int IN1_FR = 27;
const unsigned int IN2_FR = 26;
const unsigned int EN_FR = 3;

const unsigned int EN_RL = 6;
const unsigned int IN1_RL = 31;
const unsigned int IN2_RL = 33;

const unsigned int IN1_RR = 30;
const unsigned int IN2_RR = 32;
const unsigned int EN_RR = 5;

L298NX2 Fmotors(EN_FL, IN1_FL, IN2_FL, EN_FR, IN1_FR, IN2_FR);
L298NX2 Rmotors(EN_RL, IN1_RL, IN2_RL, EN_RR, IN1_RR, IN2_RR);


int speed = 255;
int directionA = L298N::FORWARD;
int value = 0;
int distanceL;
int distanceR;
int distanceFL;
int distanceFR;
int distance;

unsigned short FLspeed = speed;
unsigned short FRspeed = speed;
unsigned short RLspeed = speed;
unsigned short RRspeed = speed;
```

```
60  void setup() {
61    Serial.begin(9600);
62    setSpeed();
63
64    previousTimer = millis();
65  }
66
67  void loop() {
68    currentTimer = millis();
69    distanceFL = (float)sonar[2].ping_median(5) * factor;
70    distanceFR = (float)sonar[3].ping_median(5) * factor;
71
72    if (distanceFR <= 40 || distanceFL <= 40) {
73      moveBackward();
74      delay(100);
75      stopMoving();
76      delay(500);
77      moveBackward();
78      delay(n2);
79      stopMoving();
80      delay(300);
81      if (currentTimer - previousTimer >= 1UL) {
82        distanceFL = (float)sonar[2].ping_median(5) * factor;
83        distanceFR = (float)sonar[3].ping_median(5) * factor;
84        Serial.print("previousTimer = ");
85        Serial.println(previousTimer);
86        Serial.print("currentTimer = ");
87        Serial.println(currentTimer);
88        Serial.print("distanceFL = ");
89        Serial.println(distanceFL);
90        Serial.print("distanceFR = ");
91        Serial.println(distanceFR);
92        previousTimer = currentTimer;
93      }
94      if (value == 0) {
95        distanceR = (float)sonar[0].ping_median(5) * factor;
96        distanceL = (float)sonar[1].ping_median(5) * factor;
97        Serial.print("distanceR = ");
98        Serial.println(distanceR);
99        Serial.print("distanceL = ");
100       Serial.println(distanceL);
101       value = 1;
102     }
103     if (distanceR >= distanceL) {
104       if (distanceR < 20) {
105         stopMoving();
106       }
107       else {
108         Serial.println("Now rotate right ");
109         rotateRight();
110         delay(n1);
111       }
112       value = 0;
113     }
114     if (distanceL > distanceR) {
115       if (distanceL < 20) {
116         stopMoving();
117       }
118       else {
119         Serial.println("Now rotate left");
120         rotateLeft();
```

```
      delay(n1);
    }
    value = 0;
  }
 }
 else {
  Serial.print("distanceFL = ");
  Serial.println(distanceFL);
  Serial.print("distanceFR = ");
  Serial.println(distanceFR);
  moveForward();
 }
 // rotateRight();
}

void setSpeed() {
 FLspeed = speed;
 FRspeed = speed;
 RLspeed = speed;
 RRspeed = speed;
 Fmotors.setSpeedA(FLspeed);
 Fmotors.setSpeedB(FRspeed);
 Rmotors.setSpeedA(RRspeed);
 Rmotors.setSpeedB(RLspeed);
}

void moveForward() {
 Fmotors.setSpeedA(165);
 Fmotors.setSpeedB(165);
 Rmotors.setSpeedA(165);
 Rmotors.setSpeedB(165);
 Fmotors.backwardA();
 Fmotors.backwardB();
 Rmotors.forwardA();
 Rmotors.backwardB();
}
void moveBackward() {
 Fmotors.setSpeedA(255);
 Fmotors.setSpeedB(255);
 Rmotors.setSpeedA(255);
 Rmotors.setSpeedB(255);
 Fmotors.forwardA();
 Fmotors.forwardB();
 Rmotors.backwardA();
 Rmotors.forwardB();
}
void turnRight() {
 Fmotors.backwardB();
 Rmotors.forwardA();
}
void turnLeft() {
 Fmotors.backwardA();
 Rmotors.backwardB();
}
void rotateLeft() {
 Fmotors.setSpeedA(255);
 Fmotors.setSpeedB(255);
 Rmotors.setSpeedA(255);
 Rmotors.setSpeedB(255);
 Fmotors.backwardA();
 Fmotors.forwardB();
```

```
182     Rmotors.forwardA();
183     Rmotors.forwardB();
184   }
185   void rotateRight() {
186     Fmotors.setSpeedA(255);
187     Fmotors.setSpeedB(255);
188     Rmotors.setSpeedA(255);
189     Rmotors.setSpeedB(255);
190     Fmotors.backwardB();
191     Fmotors.forwardA();
192     Rmotors.backwardA();
193     Rmotors.backwardB();
194   }
195   void moveRightForward() {
196     Fmotors.backwardA();
197     Rmotors.forwardA();
198   }
199   void moveRightBackward() {
200
201   }
202   void moveLeftForward() {
203     Fmotors.backwardB();
204     Rmotors.backwardB();
205   }
206   void moveLeftBackward() {
207
208   }
209   void stopMoving() {
210     Fmotors.stop();
211     Rmotors.stop();
212   }
```

Self-parking:

auto parking.ino

```
#include <L298NX2.h>
#include <NewPing.h>

#define MAX_DISTANCE 500

#define SONAR_NUM 4      // Number of sensors.

unsigned int cm[SONAR_NUM];         // Where the ping distances are stored.
unsigned long currentTimer = 0;
unsigned long previousTimer = 0;

NewPing sonar[SONAR_NUM] = {   // Sensor object array.
  NewPing(48, 49, MAX_DISTANCE), // Each sensor's trigger pin, echo pin, and max distance to ping.
  NewPing(46, 47, MAX_DISTANCE),
  NewPing(44, 45, MAX_DISTANCE),
  NewPing(40, 41, MAX_DISTANCE)
};

int n1 = random(150, 350);
int n2 = random(150, 175);

int dist[12];
int command = 0;

float temp = 23.7; //Temp in C degree
float factor = sqrt(1 + temp / 273.15) / 60.368;

const unsigned int EN_FL = 2;
const unsigned int IN1_FL = 25;
const unsigned int IN2_FL = 24;

const unsigned int IN1_FR = 27;
const unsigned int IN2_FR = 26;
const unsigned int EN_FR = 3;

const unsigned int EN_RL = 6;
const unsigned int IN1_RL = 31;
const unsigned int IN2_RL = 33;

const unsigned int IN1_RR = 30;
const unsigned int IN2_RR = 32;
const unsigned int EN_RR = 5;

L298NX2 Fmotors(EN_FL, IN1_FL, IN2_FL, EN_FR, IN1_FR, IN2_FR);
L298NX2 Rmotors(EN_RL, IN1_RL, IN2_RL, EN_RR, IN1_RR, IN2_RR);


int speed = 255;
// int directionA = L298N::FORWARD;
int value = 0;
int distanceL;
int distanceR;
int distanceFL;
int distanceFR;
// int distance;

unsigned short FLspeed = speed;
unsigned short FRspeed = speed;
unsigned short RLspeed = speed;
unsigned short RRspeed = speed;

```

```
void setup() {
  Serial.begin(9600);
  setSpeed();

  previousTimer = millis();
}

void loop() {
  delay(2000);
  if (value == 0) {
    for (int i = 0;i < 12; i++) {
      dist[i] = (float)sonar[0].ping_median(5) * factor;
      Serial.println(dist[i]);
      moveForward();
      delay(300);
      moveBackward();
      delay(80);
      // Serial.println("move forward");
      stopMoving();
      delay(200);
    }
    // value = 1;
    stopMoving();
    delay(300);
    for (int i = 0; i < 12; i++) {
      if (dist[i] < 35) {
        command = 0;
      }
      else {
        command = 1;
      }
    }
  }
  value = 1;
  // stopMoving();
  // delay(300);
  // for (int i = 0; i < 10; i++) {
  //   if (dist[i] < 35) {
  //     command = 0;
  //   }
  //   else {
  //     command = 1;
  //   }
  // }
  if (command == 1) {
    moveBackward();
    delay(600);
    moveForward();
    delay(80);
    stopMoving();
    delay(200);
    moveRightSideways();
    delay(1200);
    moveLeftSideways();
    delay(80);
    stopMoving();
    // command = 0;
  }
  command = 0;

  // moveLeftSideways();
}
```

```
125  void setSpeed() {
126    FLspeed = speed;
127    FRspeed = speed;
128    RLspeed = speed;
129    RRspeed = speed;
130    Fmotors.setSpeedA(FLspeed);
131    Fmotors.setSpeedB(FRspeed);
132    Rmotors.setSpeedA(RRspeed);
133    Rmotors.setSpeedB(RLspeed);
134  }
135
136  void moveForward() {
137    Fmotors.setSpeedA(255);
138    Fmotors.setSpeedB(255);
139    Rmotors.setSpeedA(255);
140    Rmotors.setSpeedB(255);
141    Fmotors.backwardA();
142    Fmotors.backwardB();
143    Rmotors.forwardA();
144    Rmotors.backwardB();
145  }
146  void moveBackward() {
147    Fmotors.setSpeedA(255);
148    Fmotors.setSpeedB(255);
149    Rmotors.setSpeedA(255);
150    Rmotors.setSpeedB(255);
151    Fmotors.forwardA();
152    Fmotors.forwardB();
153    Rmotors.backwardA();
154    Rmotors.forwardB();
155  }
156  void moveLeftSideways() {
157    Fmotors.setSpeedA(255);
158    Fmotors.setSpeedB(255);
159    Rmotors.setSpeedA(255);
160    Rmotors.setSpeedB(255);
161    Fmotors.forwardA();
162    Fmotors.backwardB();
163    Rmotors.forwardB();
164    Rmotors.forwardA();
165  }
166  void moveRightSideways() {
167    Fmotors.setSpeedA(255);
168    Fmotors.setSpeedB(255);
169    Rmotors.setSpeedA(255);
170    Rmotors.setSpeedB(255);
171    Fmotors.backwardA();
172    Fmotors.forwardB();
173    Rmotors.backwardB();
174    Rmotors.backwardA();
175  }
176  void rotateLeft() {
177    Fmotors.setSpeedA(255);
178    Fmotors.setSpeedB(255);
179    Rmotors.setSpeedA(255);
180    Rmotors.setSpeedB(255);
181    Fmotors.backwardA();
182    Fmotors.forwardB();
183    Rmotors.forwardA();
184    Rmotors.forwardB();
185  }
```

```
185   }
186   void rotateRight() {
187     Fmotors.setSpeedA(255);
188     Fmotors.setSpeedB(255);
189     Rmotors.setSpeedA(255);
190     Rmotors.setSpeedB(255);
191     Fmotors.backwardB();
192     Fmotors.forwardA();
193     Rmotors.backwardA();
194     Rmotors.backwardB();
195   }
196   void moveRightForward() {
197     Fmotors.backwardA();
198     Rmotors.forwardA();
199   }
200   void moveRightBackward() {
201
202   }
203   void moveLeftForward() {
204     Fmotors.backwardB();
205     Rmotors.backwardB();
206   }
207   void moveLeftBackward() {
208
209   }
210   void stopMoving() {
211     Fmotors.stop();
212     Rmotors.stop();
213   }
```

12. Validating the performance.

c. Experimental testing and results and data

Calculation:

The chassis weight = 560g

The motors weight = 218*4 = 872g

Tesla Roadster body weight = 1470g

Total amount of load on the wheels = 560 + 872 + 1470 = 2902g

Total wheels weight = (wheel case + 10*roller weight)*4 = (82+10*8)*4 = 648g

⇨ The expected weight that the wheels can carry: 648*11.5 = 7452g

Table of mecanum wheel performance

Materials of mecanum wheel				Terrain	
TPU case	PLA case	TPU roller	PLA roller	Carpet floor	Flat surface

yes		yes		Not working	Not working
yes			yes	Not working	Not working
	yes	yes		Not working	Not working
	yes		yes	Working	Working

4. Discussion

a. Detail of the build-up of the artifact

Architecture:

1. Processor: Arduino Mega boards, 2 H-bridges
2. Sensor: NRF24L01 Transceiver Module, ultrasonic sensors HC-SR4
3. Motor: 4 DC motors
4. AWD or FWD: AWD
5. Driving transmission: directly from motor shafts
6. Terrain: flat surface, trying to do it on outdoor surfaces
7. Battery: 11.1V, 50C, 5200 mAh, LiPo battery
8. Capacitor: 47~100uF
9. LEDs

b. Equipment used

1. Rotary tools, clamp, vise, screwdrivers, plyers, driller, solder, wire stripper, electronic tape
2. Ipad for sketching ideas.
3. Computer with 3D CAD and 3D print software (SolidWorks, Ultimaker Cura) for designing and modeling

4. 3D printer for printing
5. Voltmeter for checking voltage and current values.
6. Computer with Arduino IDE for programming

c. Setup procedures and final results

I had many troubles with keeping everything together during the setup and assembly. When I designed the mecanum wheels, it took time to understand the mechanism of how these wheels work and how they are designed so the roller can touch the ground and roll ideally. When I created the Tesla Roadster, I had to base it on the online diagram since the vehicle has yet to be released, and there was no pre-model I could find on a website such as Thingiverse. And the chassis was also a problem since it is the most important besides the wheels. With a good chassis, the wheel can run properly. And my chassis did not have any suspension, which, when it bends inward, will cause the wheels to be lifted from the ground. However, I fixed it by flipping the DC motor holder upside down and using the Tesla Roadster body weight distribution load to keep the wheel on the ground. The PLA mecanum wheels were good, but not for an actual experiment. I tried to make the roller from TPU material, which is soft like rubber, to get more friction. Still, it did not work since the roller was initially distorted, causing it to fail to run from the beginning and making the whole wheel unable even to rotate. Therefore, I had to switch to actual mecanum wheels. This showed how precisely the manufacturing process needs to be for this specific type of wheel.

At the beginning of this project, I planned to use stepper motors. They worked but were too slow and noisy. My stepper motor drivers keep heating up and draw too much power and heat my Arduino board. Therefore, I had to switch to DC motors with two H-bridge. The wheel could handle about 3000g of load for the electrical side. However, it was with the 11.1V battery, while

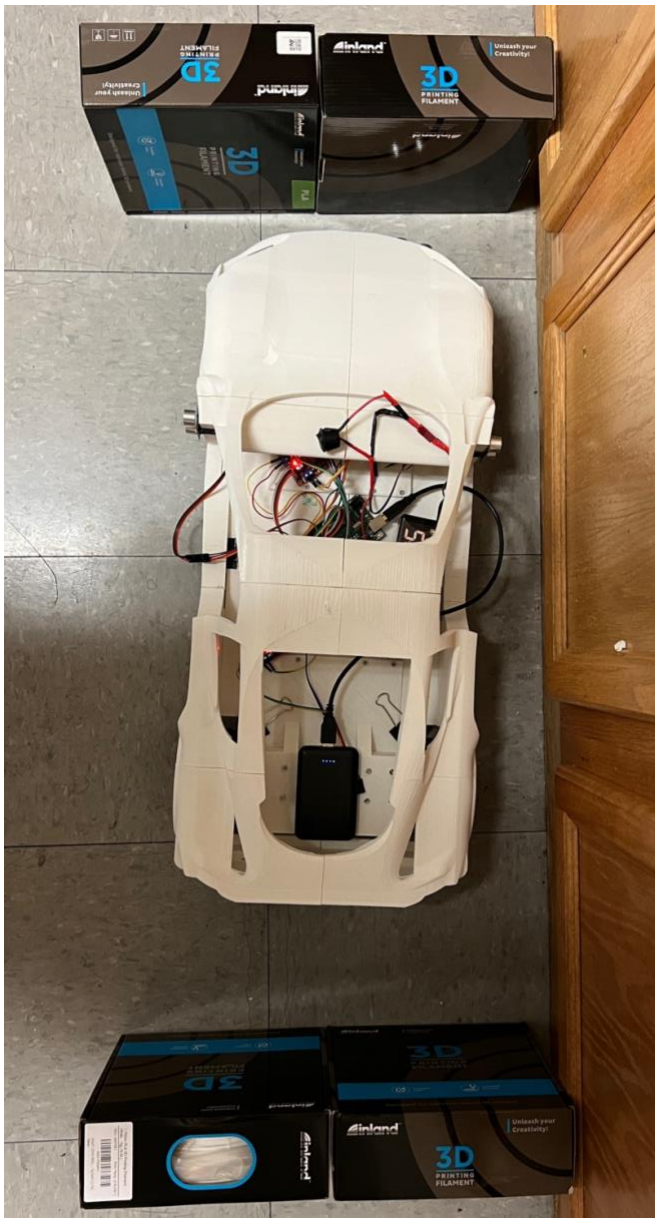
the H-bridge can handle up to 36V, and I believe 24V is enough to handle 7500g as expected. The reason that I could not experiment with the 24V battery was that the limit voltage of my DC motors was 12V. The wheel could go at a high enough speed with the remote control. The rate of the wheels depends on how much PWM we give to it and the voltage applied. However, if we want to implement autonomous functions, the wheel should go reasonably fast to sense the environment and keep everything safe.

The wheel can do object avoidance and self-parking features. If I have more time to learn and use more powerful resources such as Nucleo board and Raspberry Pi, I can make it follow the lead, stay in lane, and do lane changing.

However, there are many drawbacks to this type of wheel. The sideways movement is not practical for cars and trucks, which mostly move forward. I used the sideways movement sparingly for my project, even for object avoidance. The only time this sideways movement is helpful is doing parallel parking; anything else would cause an accident in real life.

The mecanum wheel durability is much lower than the conventional wheels, especially at high speed, which means higher maintenance costs and will only be suitable for highway performance if we have better material for roller manufacture. The mecanum wheel also has much inertia in movement; without a standardized manufacturing process, it may not meet the standard quality for daily use and cause many more problems with actuation.

Moreover, the mecanum wheel performance is not good on other terrains besides the smooth concrete ground. The lack of friction makes ensuring that every roller touches the land more challenging. And working with the mecanum wheel also means losing half of the power from each wheel when moving since they cancel half of each in other directions.



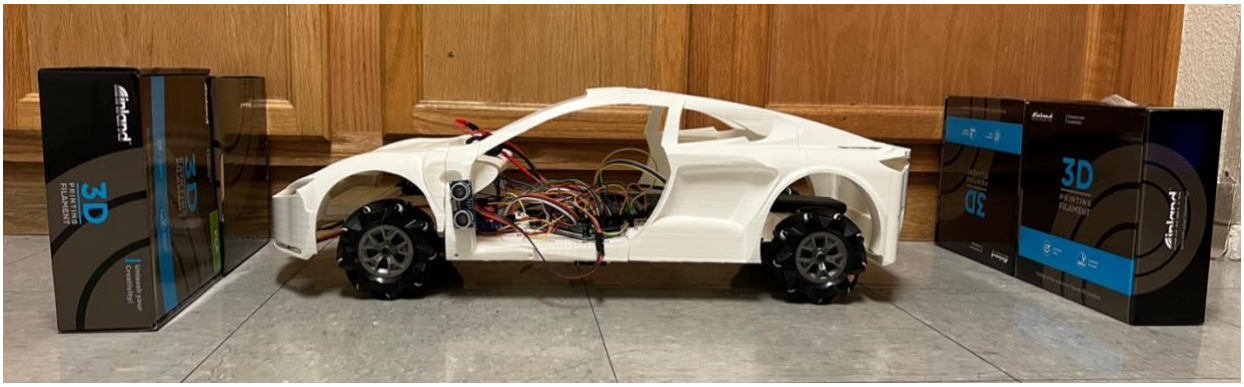


Figure 32-33. Self-parking in parallel

5. Conclusions

In conclusion, my project did not yield the best result because of the lack of suitable materials and a proper manufacturing process. However, it proved that developing smart cars from electric cars is possible, and we can even create a network to connect smart cars when the algorithms are completed. Still, the algorithm should be developed based on the conventional wheels and their work mechanism since it is standardized and optimized for daily use. The mecanum wheels are suitable for omnidirectional functions, but their disadvantages outweigh the advantages unless we can optimize the mecanum wheels better.

6. References

"Arduino Mecanum Wheel Robot." *Youtube*, uploaded by How To Mechatronics, May 28th, 2019, https://www.youtube.com/watch?v=83tVkgT89dM&list=WL&index=1.

"Mecanum Wheel." *Wikipedia*, Wikimedia Foundation, 14 Nov. 2021, https://en.wikipedia.org/wiki/Mecanum_wheel.

https://www.sciencedirect.com/science/article/pii/S0957415820301318

"Tutorial Mecanum Wheel SolidWorks PFA ULT 2020-2021" *Youtube*, uploaded by Maher Bahri, Jan 14th, 2021, https://www.youtube.com/watch?v=O36jcFvxf88