A PROJECT REPORT ON

TEJ4M COURSE SUMMATIVE

For
Todd Martinson
TEJ4M

By
Omar Alhalawani,
Yousef Alharbi,
In collaboration with Matthew Perrin

Report written by
Omar Alhalawani

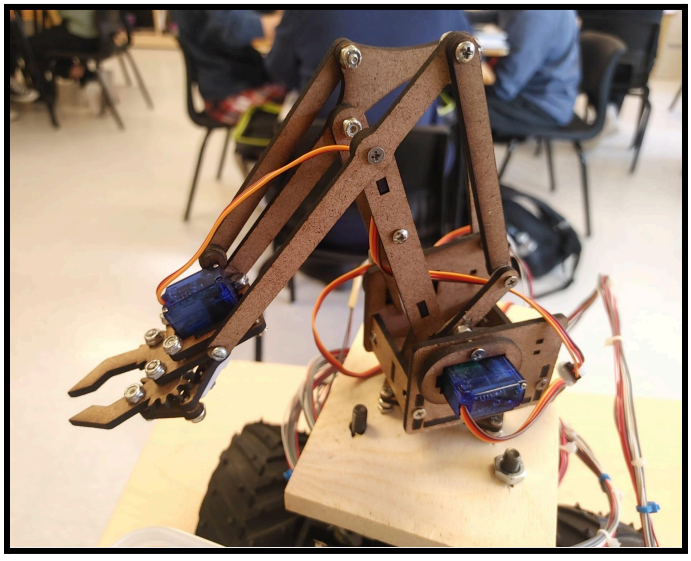
Table of Contents:

1. Introduction:

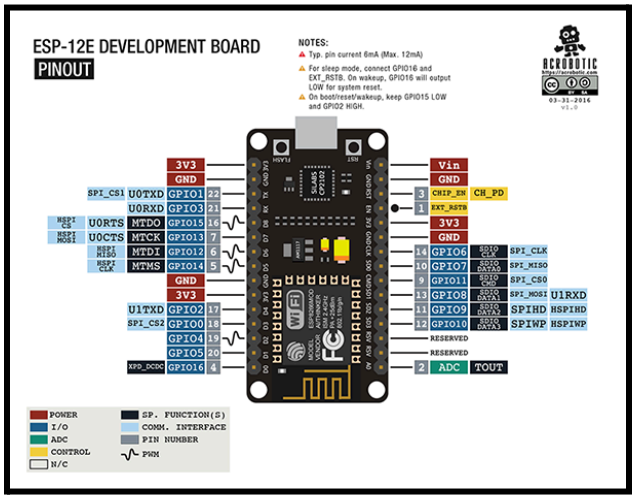
The TEJ4M Course Summative was a task to showcase our Computer Engineering skills. For my project, I decided to build a wireless car with a robot arm on top. I wanted to showcase my skills in the wireless Arduino field, so the car is controlled from my phone. I needed a car that could pick things up, place them in the storage container of the car, and transport them to a different location. This can be used by law enforcement to deal with dangerous unidentifiable objects that can pose a threat to human life. The robot could transport the object away and handle it with its robot arm. The robot would also have a camera that would live-stream footage of the object. In this report, I will thoroughly explain the design process, the implementation, the problems faced, and the future redesign.

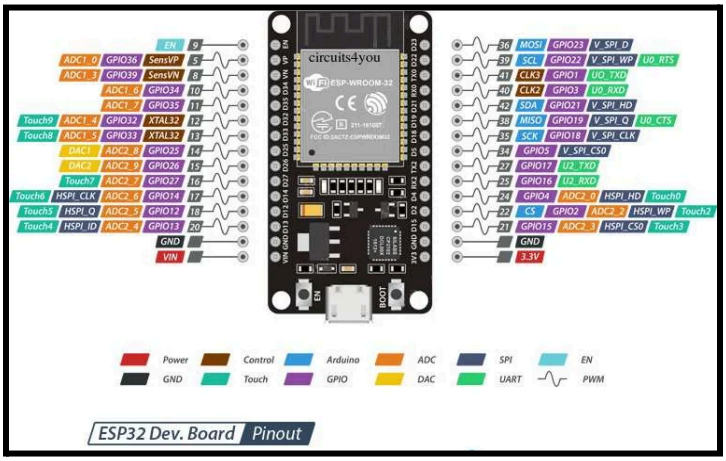
2. Robot Construction:

Here are pictures of the finished car and the robot arm.



3. Brainstorming and Concept Development:



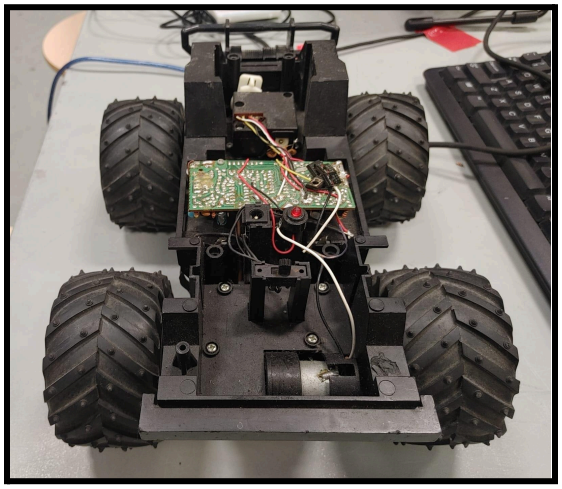


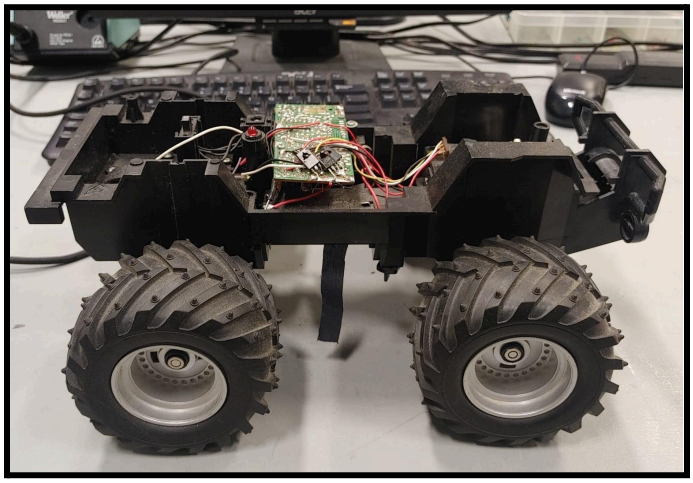
I researched multiple ESP boards to learn about the features each model had. In this project, I experimented with the ESP-8266 and the ESP-WROOM-32. The ESP-WROOM-32 was by far more capable than the ESP-8266, having 16 PWM pins while the ESP-8266 only had 4, which proved problematic when I planned to control 5 servo motors at the same time. After lots of research, experimentation, and corrupted ESP-WROOM-32 boards, I had to compromise and use an ESP-8266.

I also researched different RC cars and learned about their functionality. There were multiple options when it came to the acceleration of the car. The options included 2-wheel drive, 4-wheel drive, and all-wheel drive. I decided to go with a 2-wheel drive and an axle for steering at the front.

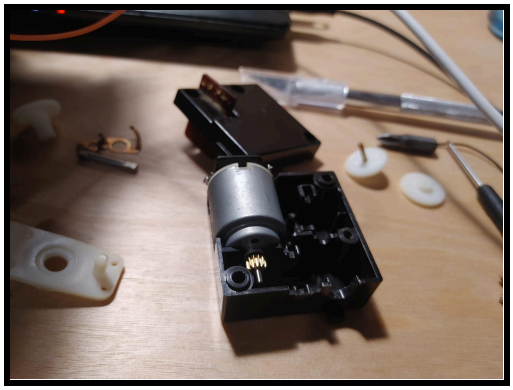
4. Design Selection and Modification:

To complete the project in the given period, Yousef Alharbi and I decided to reuse an old RC car we found in the classroom as shown in the figure below.



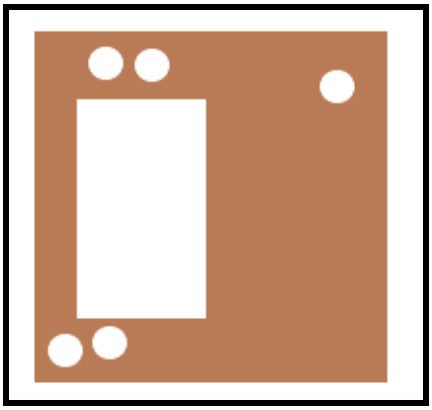


I studied the wiring of the RC car, and once I understood how the car operates, I took the PCB board out and left both the DC motor at the back and the steering axle motor at the front. However, after testing both motors, I decided to take the turning axle motor out (and make my own as it was very weak and would not steer properly. I decided to replace the old motor with a powerful HS-322HD servo motor.



Old turning axle motor

HS-322HD



Wooden Mount for Servo

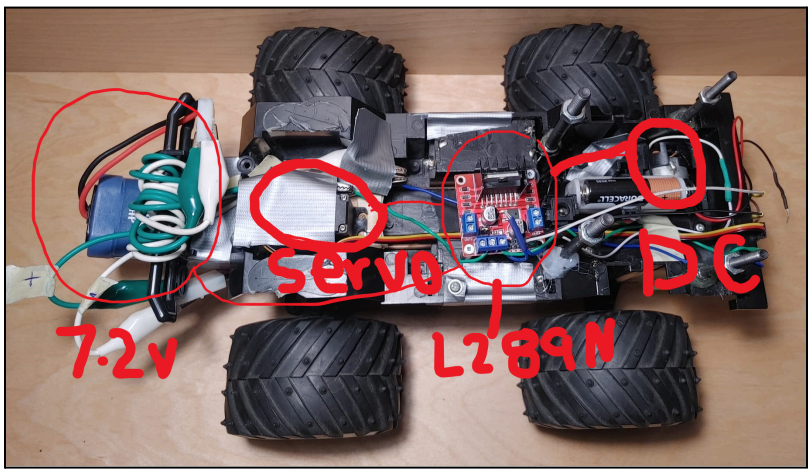
The motor required a mount so in collaboration with Mr. Perrin's woodworking skills, my partner and I designed and shaped this mount using a piece of wood. This mount was used to hold the H2-322HD servo motor in place to control the steering axle of the car.

5. Implementation:

5.1 Mounting the steering axle servo motor:

As explained in the Design Selection and Modification section, the old turning axle servo motor was not functioning properly so we had to design a mount with specific measurements to house the new steering axle servo motor.

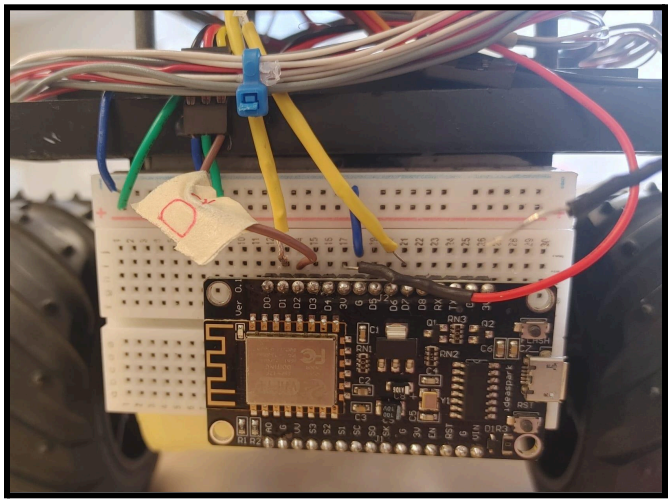
5.2 Installing the L298N Motor Driver Module:



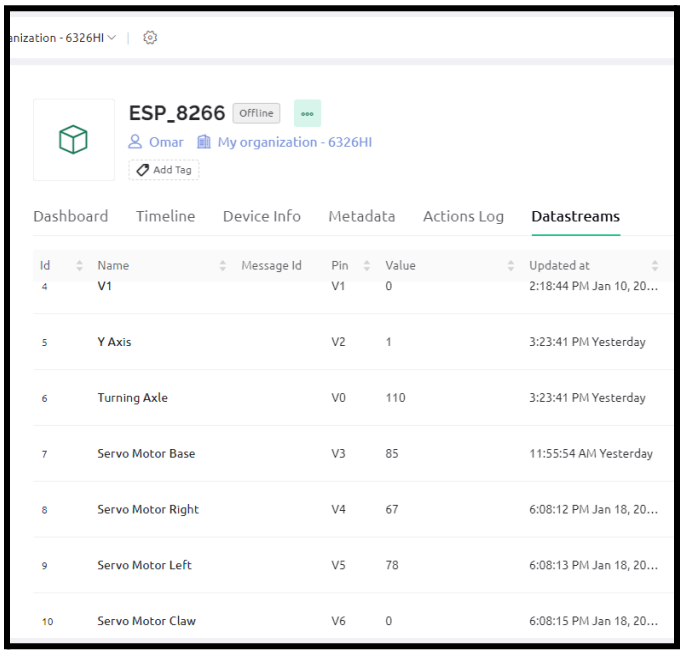
To power the motors of the car, I decided to use an L298N Motor Driver Module. I placed it where the old board used to be and hooked up a 7.2 V battery to its input port. 5V went to the turning axle servo

motor while the rest went to the DC motor that controlled the acceleration of the car. 2.2V is little to run a dc motor so the car was a little slow. However, this was planned as the car would be housing a robot arm which was a little fragile, so speed was better kept to a minimum.

5.3 Coding the ESP-8266:



For this project, I used an ESP-8266 wireless arduino board. I decided to use the Blynk application to control the car's functions. I first started by making an account on Blynk, creating a new project, and then I started coding in Arduino. The Arduino code was fairly simple as all the actual work was done in the Blynk app. The ESP-8266 connected to my phone's hotspot, allowing me to send commands to the board through the Blynk application.



In the app, I set up different data streams that would be used to send data to the Arduino board through virtual pins. The Arduino board then takes the data from the virtual pin and performs tasks based on the given value of the data. Below is the code with comments explaining what each line of code does.

```
/*Omar Alhalawani and Yousef AlHarbi
  TEJ4M Summative
  January 17, 2024
  Code to control Wifi Controlled Car
*/
```

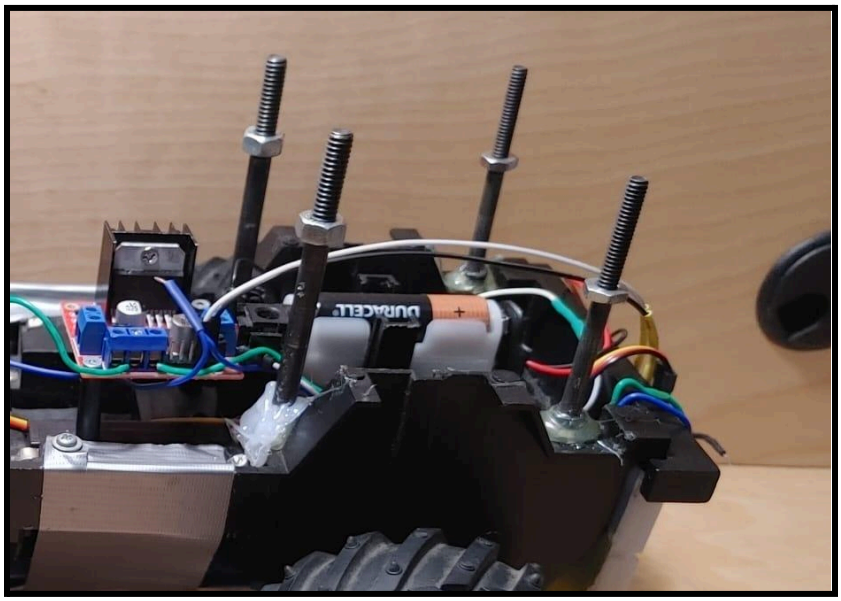
```
#include <Servo.h>
// Using your ESP8266 with the Blynk App
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPL2eHbschXn"
#define BLYNK_TEMPLATE_NAME "LED TEST"
#define BLYNK_AUTH_TOKEN "ywruN00h9CkLvgjhYEsprtdllJa9MG_n"
#include <ESP8266WiFi.h> //library for wifi connection between ESP-8266 and
phone
#include <BlynkSimpleEsp8266.h>
Servo myservo_turning; // create Servo object to control steering axle
int servoPin_turning=D2;// variable to store the servo position
char ssid[] = "OmarLG"; // wifi SSID
char pass[] = "omarali123"; //wifi password

void setup(){
  delay(100);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass); // this will send Blynk the
required data for connection
  pinMode(D7, OUTPUT); // setup pin for dc motor controlling the backward
acceleration
  pinMode(D1, OUTPUT); // setup pin for dc motor controlling the forward
acceleration
  myservo_turning.attach(servoPin_turning);  //attach pin D2 to turning axle
servo
}
BLYNK_WRITE(V0){ // this function will send data from the Blynk app switch to
the ESP8266
  int pin1 = param.asInt(); // this takes the value of a slider from Blynk and
puts it into 'pin1'
  myservo_turning.write(pin1); //the pin value goes from 80 to 140
}
BLYNK_WRITE(V2){// this function will send data from the Blynk app switch to
the ESP8266
  int pin2 = param.asInt(); // this takes the value of a slider from Blynk and
puts it into 'pin2'
  if (pin2 == 2){   // if slider is pulled up, value is 2 and car accelerates
forward
    digitalWrite(D1, HIGH);
    digitalWrite(D7, LOW);   }
  else if (pin2 == 0){ // if slider is pulled down, value is 0 and car
accelerates backward
    digitalWrite(D1, LOW);
```

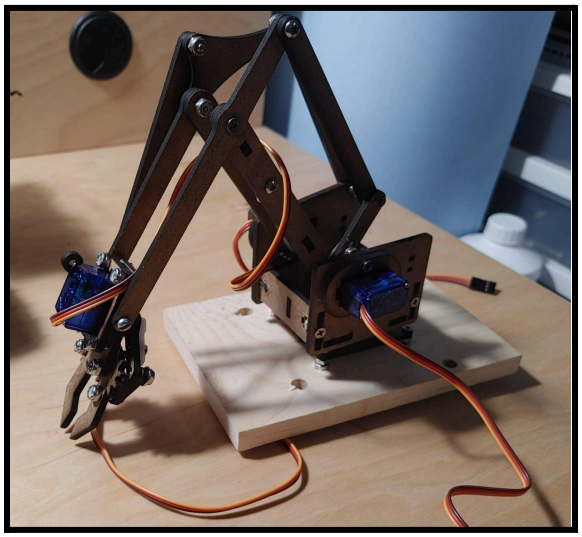
```
    digitalWrite(D7, HIGH);    }
  else{                  // if slider is not pulled, value is 1 and car does not
accelerate
    digitalWrite(D1, LOW);
    digitalWrite(D7, LOW);   }}
void loop(){
  Blynk.run();}
```

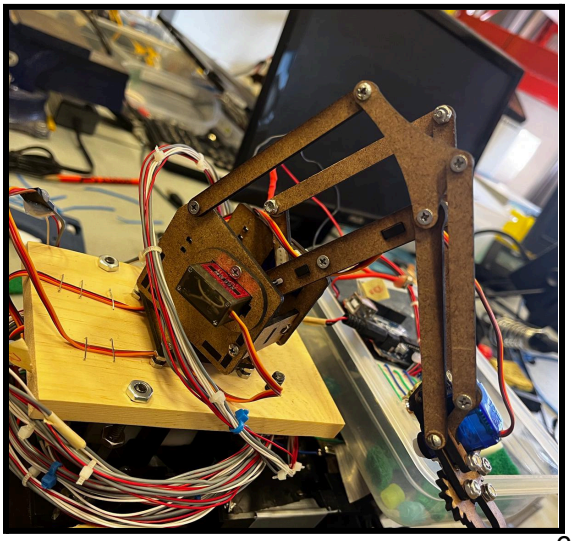
5.4 Mounting the Robotic Arm:

Before wiring the Robotic arm to the car, I had to figure out a way to mount it. We researched different options which included laser cutting, 3d printing, or using wood. We decided to go with making the mount out of wood. With the much-needed help of Mr. Perrin, we worked together to design a wooden mount for the arm.



We first started by modifying the car by cutting off some of its plastic to make room for the bolts. We glued the bolts on the car and set nuts in place to elevate the robot arm. We then took the dimensions of the arm and made the mount. The arm sat in the mount and the mount was placed in the bolts and screwed in place.

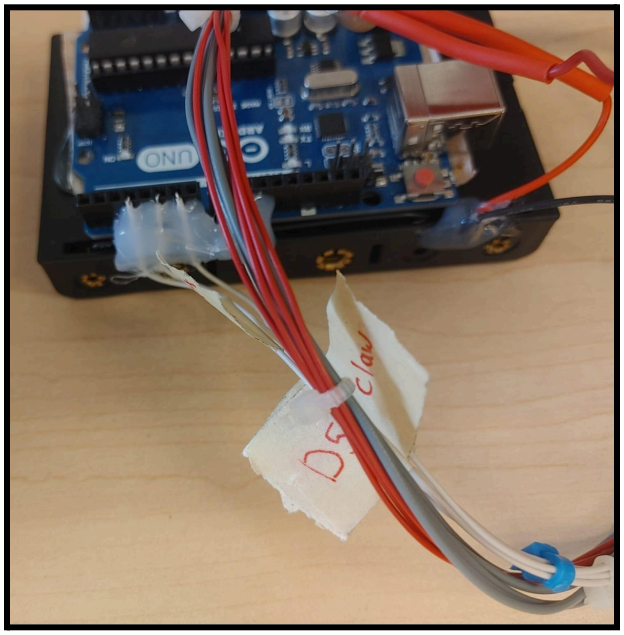


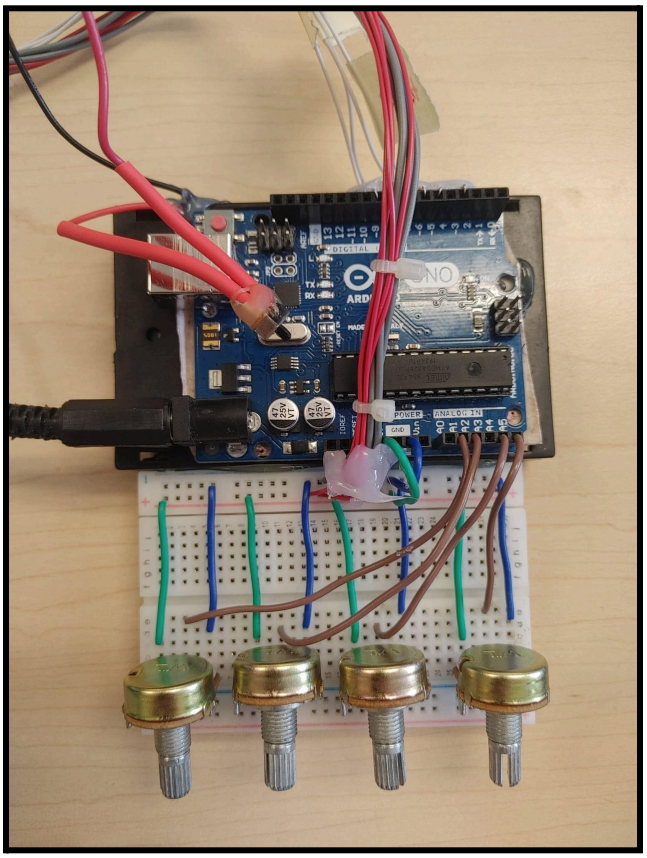


6. Testing and Troubleshooting:

6.1 Controlling the Robotic Arm:

When we first started the project, we planned to control the robotic arm wirelessly using the Blynk app. However, this proved difficult as the Blynk application was not controlling the servo motors on the arm properly. Running out of time, in a last-ditch effort to save the project, I made a makeshift wired controller using an Arduino Uno, a 9V battery pack, 4 potentiometers, and lots of wires.





Since we used an Arduino Uno board instead of the ESP-8266, I had to write a separate program for it.

Code:

```
/*Omar Alhalawani and Yousef Alharbi
  TEJ4M Summative
  Code to control robot arm.
  The code consists of controls for 4 servo motors.
*/
#include <Servo.h>
Servo base;  // create servo object to control the base servo
Servo left;  // create servo object to control the left servo
Servo right; // create servo object to control the right servo
Servo claw;  // create servo object to control the claw servo
```

```
int basepot = A1;  // analog pin used to connect the base potentiometer
int leftpot =  A2; // analog pin used to connect the left potentiometer
int rightpot = A4; // analog pin used to connect the right potentiometer
int clawpot = A5;  // analog pin used to connect the claw potentiometer
int baseval;    // variable to read the value from the basepot pin
int leftval;    // variable to read the value from the leftpot pin
int rightval;   // variable to read the value from the rightpot pin
int clawval;    // variable to read the value from the clawpot pin

void setup() {
  base.attach(6);  // attaches the servo on pin 6 to the base servo
  left.attach(3);  // attaches the servo on pin 3 to the left servo
  right.attach(5); // attaches the servo on pin 5 to the right servo
  claw.attach(9);  // attaches the servo on pin 9 to the claw servo
}
void loop() {
  baseval = analogRead(basepot);            // reads the value of the
potentiometer (value between 0 and 1023)
  baseval = map(baseval, 0, 1023, 0, 180);     // scale it for use with the
servo (value between 0 and 180)
  base.write(baseval);                  // sets the servo position according to
the scaled value
  delay(15);                           // waits for the servo to get there

  leftval = analogRead(leftpot);            // reads the value of the
potentiometer (value between 0 and 1023)
  leftval = map(leftval, 0, 1023, 0, 180);     // scale it for use with the
servo (value between 0 and 180)
  left.write(leftval);                  // sets the servo position according to
the scaled value
  delay(15);                           // waits for the servo to get there

  rightval = analogRead(rightpot);            // reads the value of the
potentiometer (value between 0 and 1023)
  rightval = map(rightval, 0, 1023, 0, 180);     // scale it for use with the
servo (value between 0 and 180)
  right.write(rightval);                  // sets the servo position according
to the scaled value
  delay(15);                           // waits for the servo to get there

  clawval = analogRead(clawpot);            // reads the value of the
potentiometer (value between 0 and 1023)
```

```
  clawval = map(clawval, 0, 1023, 0, 180);     // scale it for use with the
servo (value between 0 and 180)
  claw.write(clawval);                  // sets the servo position according to
the scaled value
  delay(15);                           // waits for the servo to get there
}
```

6.2 Power Issues with the Steering Axle:

Halfway through the project, the steering axle servo motor stopped turning all the way. I initially thought it was a voltage issue, so I increased the voltage input through the L289N. However, this was a huge mistake as I have completely fried the L289N circuit board. After replacing the L289N, I did some research and found out that it could be a current issue. So to overcome this, I attached two 7.2 HEX batteries in parallel to double the current and this proved effective. The steering axle servo motor now turns fully.

7. Redesign:

7.1 Amazon Camera Scam:

Part of this design was implementing a camera that would live-stream footage of the object to the user's phone through a web server. After researching for 4 days on what camera to purchase, what the required code is, and how to design the web server, I bought the ESP 32 Camera Module from Amazon. Although the reviews were very positive, the module was defective and I had no spare time to acquire another model.

In the future, more research will be needed before purchasing electronics from Amazon. I plan to implement the camera in the future after the project has been submitted.

7.2 Controlling the Robot Arm with Blynk:

If I were to do this project again, I would use an arm that uses the same servo motor I used for the steering axle as that motor worked perfectly with Blynk. This way, I would be able to control the arm wireless, thus fulfilling the purpose of this project which is to deal with dangerous objects from a distance so no human is at risk of being harmed.

7.3 Aesthetics:

The finished product looked like a prototype. This is because we used different materials for different aspects of the project, and most of those materials did not match in aesthetics. It is good to keep aesthetics in mind as it makes a product look more capable and appealing than its bad-looking counterparts.

7.4 Reduce the Weight and Use Powerful Motors:

The initial plan was to have a slow-moving car. However, the car was slower than initially planned. This is because it housed two 7.2 Hex batteries and a robotic arm, all of which slowed down the car. In the future, reducing the weight and adding a more powerful motor would be more effective in making the car faster.

8. Conclusion:

Although not all of the initial design was successful, this project has gone better than expected. From concept to completion, there were many obstacles to overcome, but these also provided chances for growth and innovation. The finished project, a hybrid-controlled remote-controlled automobile with a custom-designed robotic arm, is proof of the strength of determination, creativity, and technical ability.

Special Thanks To

Matthew Perrin: Crucial to the completion of this project as his woodworking skills helped us immensely with the creation of the wood mounts.

Todd Martinson: Suggested the idea of installing two batteries in parallel to double the current, which greatly improved the steering of the car.

Ross Morrison: Supplied the ESP-WROOM-32 and helped fix complications in the robotic arm's movement.