REAL-TIME EMBEDDED SYSTEM
EXPERIMENT NO. 6

Self-Driving Car

Name: NAVARRO, ROD GERYK C.

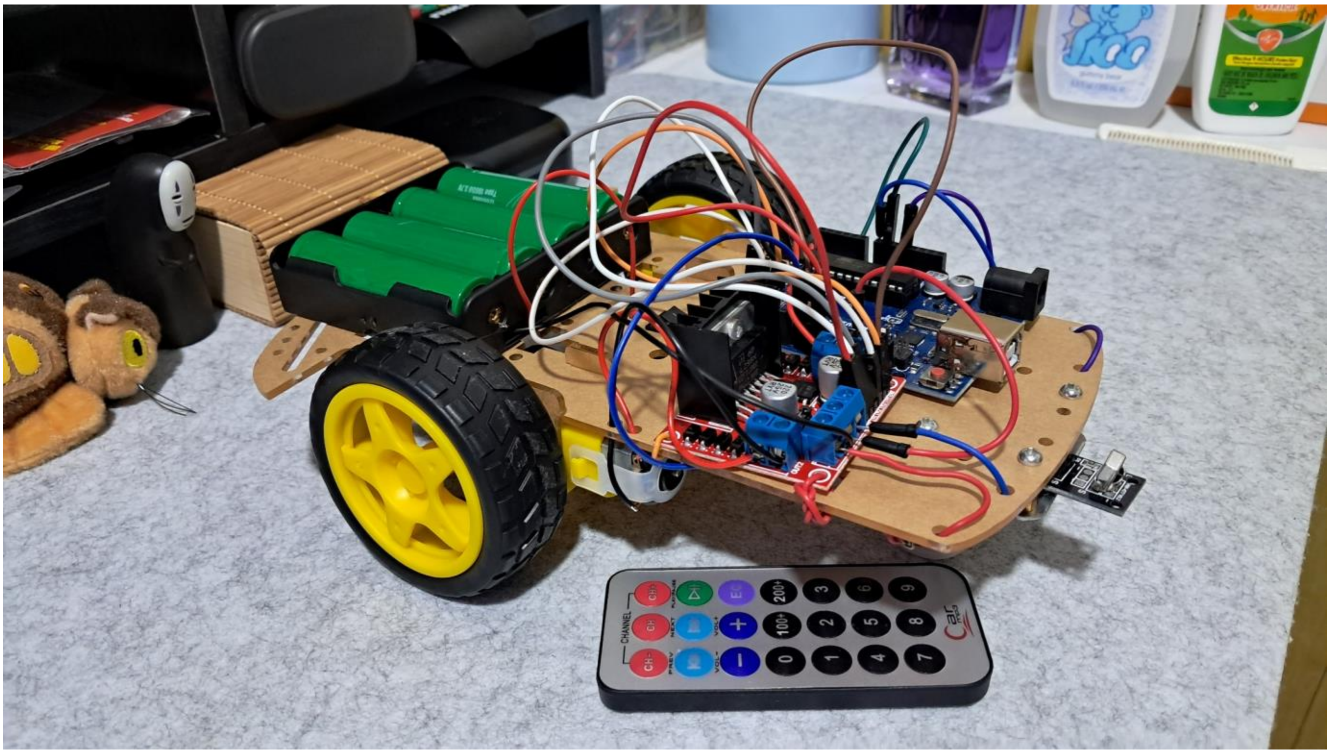
Course/Section: CPE161P-4/C1

Group No.: N/A

Date of Performance: 01/25/2025
Date of Submission: 02/2025

CYREL O. MANLISES, PH.D.
Instructor

DISCUSSION



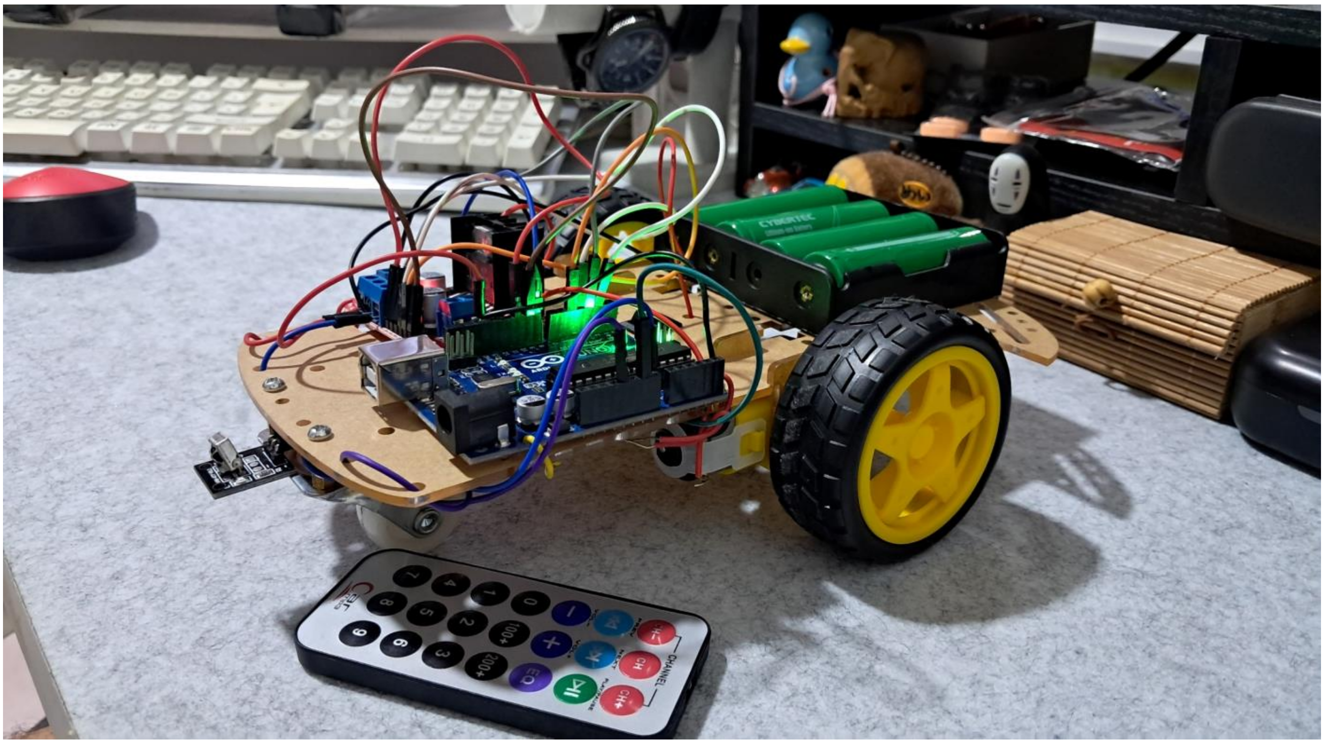


Figure 1: The Connection of the Motor Driver Board Module, DC Gear Motors, IR Sensor, 18650 Batteries, and Switch

This experiment is about building a real-time self-driving car. The first step in the experiment involved creating the physical structure of the car and connecting all its components properly. The components used in this experiment include a motor driver board module, which controls two high-powered DC gear motors. These motors cannot be connected directly to the microcontroller because they require more voltage than it can handle, which could cause damage. An infrared IR sensor is also used to receive signals from a wireless remote control, allowing the car to move in different directions. The DC gear motors power the car s wheels, while batteries serve as the main power source. A switch acts as a key to turn the car on and off. The wireless remote control is responsible for operating the car. It has designated buttons for moving the car forward, reversing, turning left or right, and stopping.

Navarro_EXP6_CPE161P.ino

```
#include<IRremote.hpp>

const int irRemotePin = 8; // the irRemote receiver pin
int motorLpwm = 10;
int motorRpwm = 9;
int motorSpeed = 60;
int turn = 10;

int motorRightPin1 = 2;
int motorRightPin2 = 3;
int motorLeftPin1 = 4;
int motorLeftPin2 = 5;

void setup() {
  Serial.begin(9600);
  pinMode(motorRightPin1, OUTPUT);
  pinMode(motorRightPin2, OUTPUT);
  pinMode(motorLeftPin1, OUTPUT);
  pinMode(motorLeftPin2, OUTPUT);
  IrReceiver.begin(irRemotePin, ENABLE_LED_FEEDBACK); // start IR receiver

}

void loop() {
  // check if IR signal is recieverd
 if (IrReceiver.decode()){
  Serial.println(IrReceiver.decodedIRData.decodedRawData, HEX); //print received IR data
  IrReceiver.printIRResultShort(&Serial); // print a IR result summary

  executeCommand(IrReceiver.decodedIRData.decodedRawData);
```

Figure 2: *The Initialization of Variables and Components*

In this experiment, I built a real-time self-driving car using a motor driver module, DC gear motors, an infrared (IR) sensor, and a wireless remote control. The first step in coding was to initialize the variables needed to control the car s movement. I defined the IR sensor pin (irRemotePin = 8), which receives signals from the remote. I also assigned pins for the motor driver, including motorRightPin1, motorRightPin2, motorLeftPin1, and motorLeftPin2, which control the direction of the wheels. Additionally, I set motorLpwm and motorRpwm for speed control using the analogWrite() function. The motorSpeed variable was set to 60 to determine how fast the car moves, while the turn variable was set to 10 to adjust turning speed. These initial values helped organize the code and made it easier to control the car.

```
void setup() {
  Serial.begin(9600);
  pinMode(motorRightPin1, OUTPUT);
  pinMode(motorRightPin2, OUTPUT);
  pinMode(motorLeftPin1, OUTPUT);
  pinMode(motorLeftPin2, OUTPUT);
  IrReceiver.begin(irRemotePin, ENABLE_LED_FEEDBACK); // start IR receiver

}
```

Figure 3: The Setup of the Components

After initializing the variables, I worked on setting up the components in the setup() function. I used Serial.begin(9600); to enable serial communication, which allowed me to see the data received from the remote in the Serial Monitor. Then, I used pinMode() to define the motor driver pins as outputs, ensuring they could send signals to control the motors. The IR receiver was started using IrReceiver.begin(irRemotePin, ENABLE_LED_FEEDBACK);, which allowed it to detect signals from the remote.

```
int motorLpwm = 10;
int motorRpwm = 9;
int motorSpeed = 60;
int turn = 10;
```

```
void carStop(){
  digitalWrite(motorRightPin1, LOW);
  digitalWrite(motorRightPin2, LOW);
  analogWrite(motorLpwm, 0);

  digitalWrite(motorLeftPin1, LOW);
  digitalWrite(motorLeftPin2, LOW);
  analogWrite(motorRpwm, 0);
 // setMotorState(LOW, LOW, LOW, LOW);
}
```

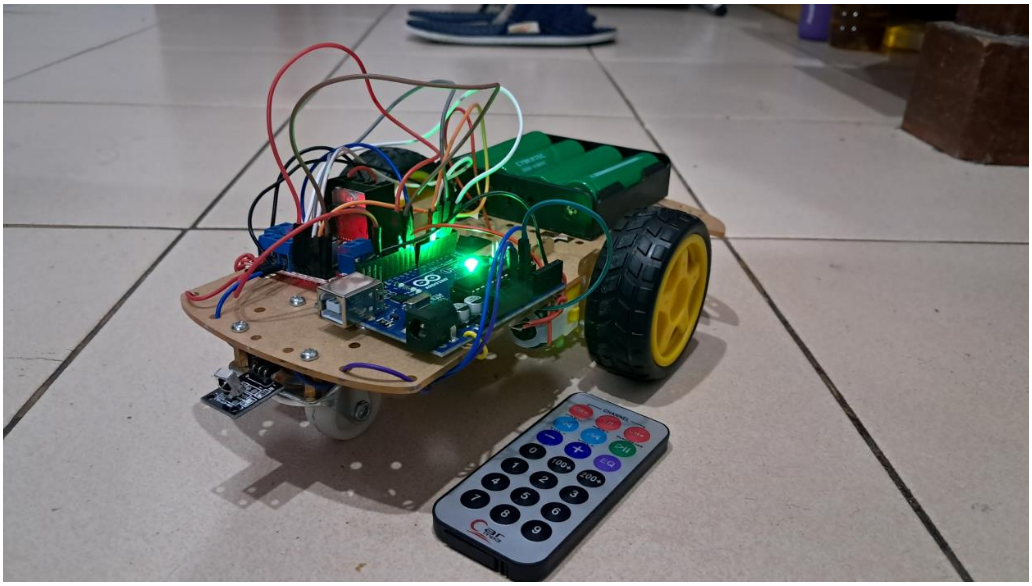


Figure 4: *Fixing the Errors*

The loop() function is responsible for continuously checking if the IR sensor receives a signal. When a button is pressed on the remote, IrReceiver.decode() detects the command, and the data is printed using Serial.println(). This helps monitor what signals are being received. The function executeCommand(IrReceiver.decodedIRData.decodedRawData); is then called to determine which command was received and take the appropriate action. Each button on the remote is assigned a unique hexadecimal code, which is matched in the executeCommand() function using a switch statement. For example, if the code 0xE718FF00 is received, the car moves forward, while 0xA55AFF00 makes it turn right. To move the car, different functions like moveForward(), moveBackward(), turnLeft(), turnRight(), and carStop() control the motor driver. Each function uses digitalWrite() to set the motor pins to HIGH or LOW, determining the direction of movement. Additionally, analogWrite() controls the speed of the motors.

Navarro_EXP6_CPE161P.ino

```
}

void loop() {
  // check if IR signal is recieverd
 if (IrReceiver.decode()){
  Serial.println(IrReceiver.decodedIRData.decodedRawData, HEX); //print received IR data
  IrReceiver.printIRResultShort(&Serial); // print a IR result summary

  executeCommand(IrReceiver.decodedIRData.decodedRawData);
  IrReceiver.resume(); // prepare IR for the next command
 }
}

void executeCommand(unsigned long command) {
  switch (command) {
        case 0xE718FF00: Serial.println("2"); moveForward(); break;  // Move forward
        case 0xAD52FF00: Serial.println("8"); moveBackward(); break; // Move backward
        case 0xF708FF00: Serial.println("4"); turnLeft(); break;     // Turn left
        case 0xA55AFF00: Serial.println("6"); turnRight(); break;    // Turn right
        case 0xE31CFF00: Serial.println("5"); carStop(); break;      // Stop the car
        default: break;
    }
}


void moveForward(){

  digitalWrite(motorRightPin1, HIGH);
  digitalWrite(motorRightPin2, LOW);
```

Navarro_EXP6_CPE161P.ino

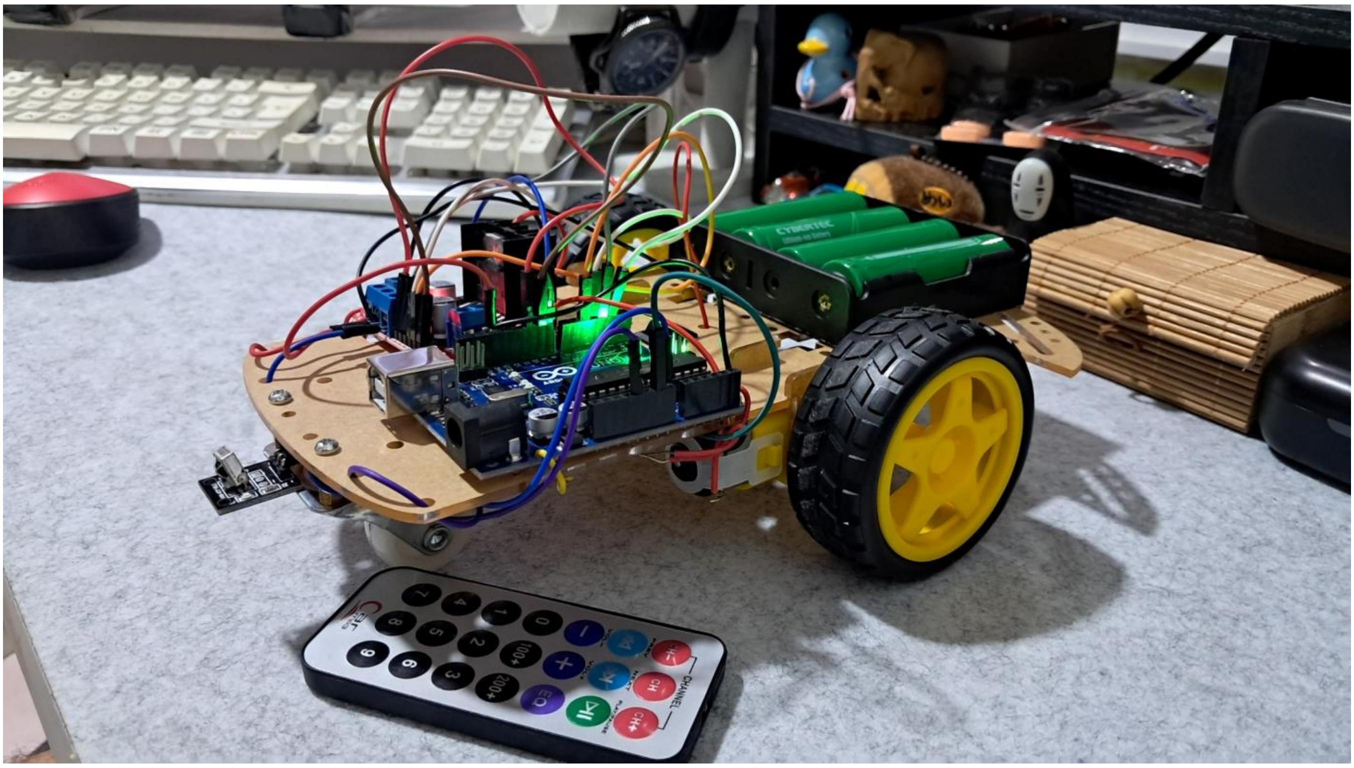
```
void moveForward(){

  digitalWrite(motorRightPin1, HIGH);
  digitalWrite(motorRightPin2, LOW);
  analogWrite(motorLpwm, motorSpeed);

  digitalWrite(motorLeftPin1, HIGH);
  digitalWrite(motorLeftPin2, LOW);
  analogWrite(motorRpwm, motorSpeed);

 // setMotorState(HIGH, LOW, LOW, HIGH);
}

void moveBackward(){
 digitalWrite(motorRightPin1, LOW);
 digitalWrite(motorRightPin2, HIGH);
 analogWrite(motorLpwm, motorSpeed);

 digitalWrite(motorLeftPin1, LOW);
 digitalWrite(motorLeftPin2, HIGH);
 analogWrite(motorRpwm, motorSpeed);
  //setMotorState(LOW, HIGH, HIGH, LOW);
}

void turnLeft(){
 digitalWrite(motorRightPin1, HIGH);
 digitalWrite(motorRightPin2, LOW);
 analogWrite(motorLpwm, motorSpeed-turn);

 digitalWrite(motorLeftPin1, LOW);
```

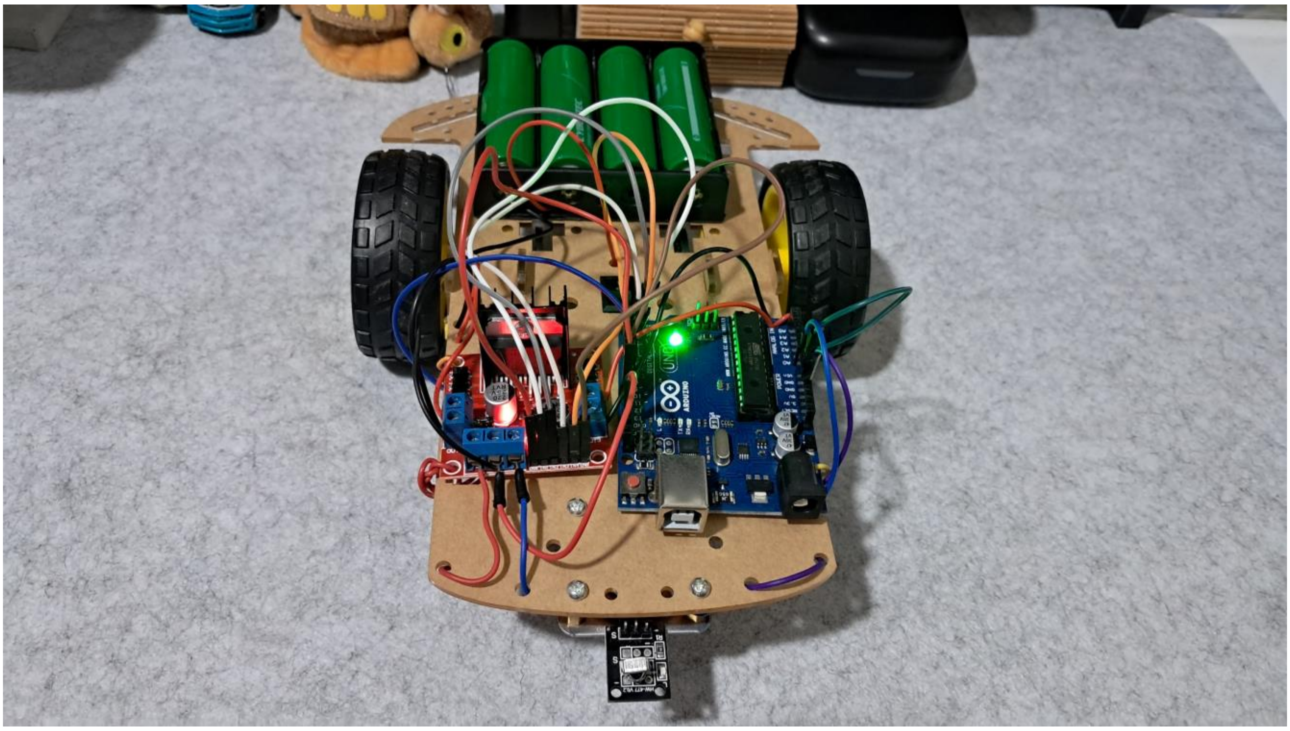
Navarro_EXP6_CPE161P.ino

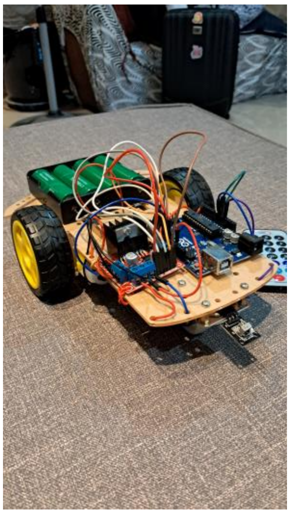
```
 analogWrite(motorRpwm, motorSpeed+turn);
  //setMotorState(LOW, HIGH, LOW, LOW);
}

void turnRight(){
  digitalWrite(motorRightPin1, LOW);
  digitalWrite(motorRightPin2, LOW);
  analogWrite(motorLpwm, motorSpeed+turn);

  digitalWrite(motorLeftPin1, HIGH);
  digitalWrite(motorLeftPin2, LOW);
  analogWrite(motorRpwm, motorSpeed-turn);
  //setMotorState(LOW, LOW, LOW, HIGH);
}


void carStop(){
  digitalWrite(motorRightPin1, LOW);
  digitalWrite(motorRightPin2, LOW);
  analogWrite(motorLpwm, 0);

  digitalWrite(motorLeftPin1, LOW);
  digitalWrite(motorLeftPin2, LOW);
  analogWrite(motorRpwm, 0);
 // setMotorState(LOW, LOW, LOW, LOW);
}
```

Figure 5: Loop Function and other Necessary Functions

The loop() function is responsible for continuously checking if the IR sensor receives a signal. When a button is pressed on the remote, IrReceiver.decode() detects the command, and the data is printed using Serial.println(). This helps monitor what signals are being received. The function executeCommand(IrReceiver.decodedIRData.decodedRawData); is then called to determine which command was received and take the appropriate action. Each button on the remote is assigned a unique hexadecimal code, which is matched in the executeCommand() function using a switch statement. For example, if the code 0xE718FF00 is received, the car moves forward, while 0xA55AFF00 makes it turn right. To move the car, different functions like moveForward(), moveBackward(), turnLeft(), turnRight(), and carStop() control the motor driver. Each function uses digitalWrite() to set the motor pins to HIGH or LOW, determining the direction of movement. Additionally, analogWrite() controls the speed of the motors.


CYBERTEC
CHANNEL
CH-
CH
CH+
100+
200+
EQ





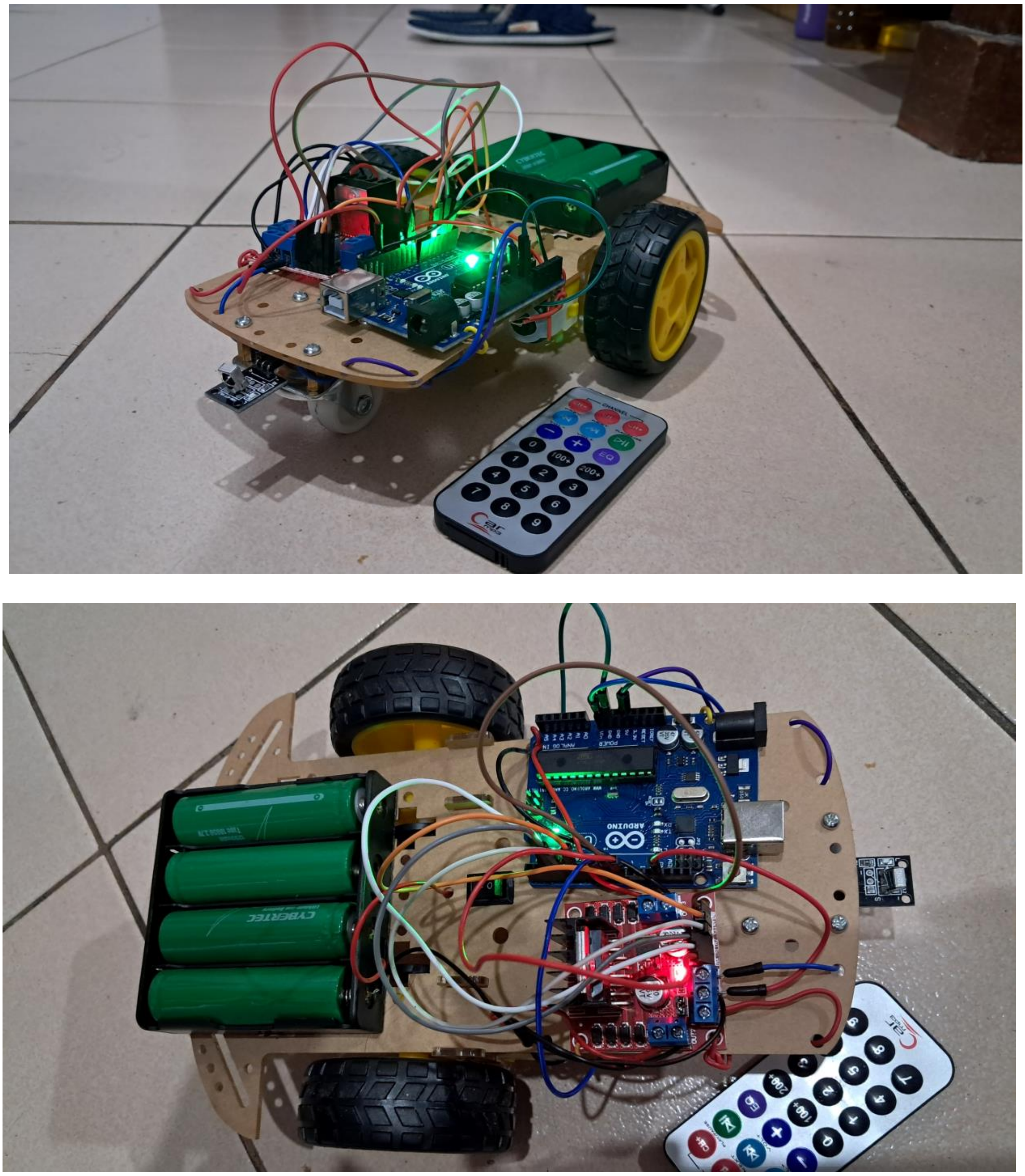


Figure 6: *Working Prototype.*

This is the final working prototype of the self-driving car experiment, which successfully meets all the requirements.

CONCLUSION

In this experiment, I successfully built and programmed a self-driving car that responds to remote control signals. By connecting a motor driver module, DC gear motors, an IR sensor, and batteries, I was able to create a working system that allowed the car to move in different directions. The coding process involved initializing variables, setting up the components, and writing functions to control movement. Through this, I learned how to integrate hardware and software to create an interactive system.

The experiment was a great learning experience. I encountered some errors while coding, but using Serial Monitor helped me debug and fix issues. The use of functions like moveForward(), turnLeft(), and executeCommand() made the code more organized and easier to understand. This project improved my understanding of motor control, IR communication, and embedded systems. In the future, I can enhance this self-driving car by adding sensors for obstacle detection or improving its autonomy.