Port Said University , Faculty of Engineering , Computer and Control

Systems Department

A car controlled
with mobile app
and gyroscope

by Mohamed Hesham, Ayman Moataz and Yassmin Osama

Prepared by second year students from second year of computer and control systems department

Abstract

This report presents the design and implementation of a WiFi-controlled robot car using the ESP32 microcontroller platform. The project involves the integration of hardware and software components to enable remote control and monitoring of the robot car through a dedicated mobile application. The ESP32 microcontroller is employed for its robust WiFi capabilities, making it an ideal choice for wireless communication. The mobile application provides an intuitive interface for users to send commands and receive real-time feedback from the robot. The report details the hardware setup, software architecture, and communication protocols employed in the project. Additionally, the results of practical testing and potential applications of the WiFi-controlled ESP32 robot car are discussed. The successful implementation of this project showcases the feasibility and versatility of utilizing WiFi connectivity for remote control applications, opening avenues for further advancements in the field of robotics and IoT.

Disclaimer

We declare the following to be our own work, unless otherwise referenced, as defined by the University’s policy on plagiarism.

Acknowledgments

Thanks are extended to the Faculty of Engineering Port Said University and Computer and Control Department and the Environment (formerly the Faculty of Engineering) for giving us a way to self-learn and improve ourselves . Thanks, are also extended to the Flexible Learning (Teaching and Learning) from our Advisors and Professors

Dr. Islam Shalaan
Dr. Osama Refaat

Other help;
Eng Youssef Khaled
Eng Ahmed Elgendy

Contents

Figures and tables

Figures

1 Introduction

In the age of smart technologies, the convergence of mobile applications and sensor-driven control systems has ushered in a new era of interactive and dynamic functionalities. This report unveils the development of an innovative smart car control system, where the fusion of a mobile application and gyroscope technology allows users to guide the vehicle with intuitive gestures. At this project lies the ESP32 microcontroller, orchestrating seamless communication with the L298N Motor Driver and drawing power from a reliable 12V battery. As we delve into the intricate details of each component, we aim to shed light on the transformative potential of this control system, paving the way for enhanced user experiences and opening avenues for applications in diverse fields.

2 Components

2.1 Esp32

The ESP32 is a low-cost, low-power system on a chip microcontroller with integrated Wi-Fi and Bluetooth capabilities . It is designed for mobile devices, wearable electronics, and IoT applications . The ESP32 employs either a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, Xtensa LX7 dual-core microprocessor or a single-core RISC-V microprocessor . It includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules . The ESP32 is capable of functioning reliably in industrial environments, with an operating temperature ranging from –40°C to +125°C [1]. It also has state-of-the-art features, such as fine-grained clock gating, various power modes, and dynamic power scaling.

Fig1.Esp32

2.2 L298N Motor Driver

The L298N motor driver can control up to 4 DC motors . To control 4 DC motors, you can connect two L298N motor driver modules to your esp32 microcontroller .

1. your microcontroller and the other two DC motors.
2. Connect the power supply to both L298N motor driver modules.
3. Program your esp32 microcontroller to control the speed and direction of the DC motors.

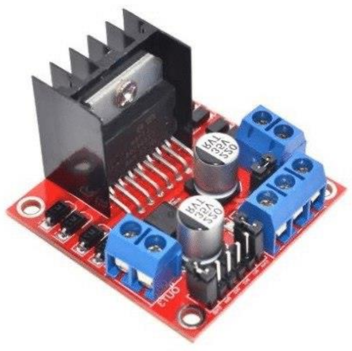


Fig2. L298N Motor Driver

2.3 Car kit

We need a car kit with 4 dc motors, but you can replace kit with wood panels to form the outer frame.

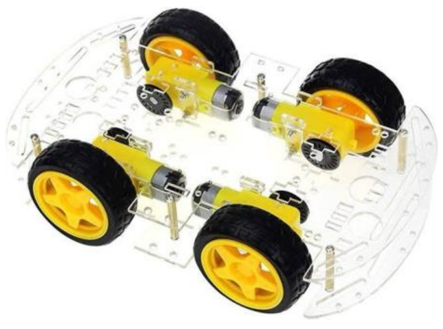


Fig3. Car kit

2.4 Jumper Wires

Connecting Wires are easily compatible with Esp32 and easy to connect all our parts. The wires as in Fig6 below.

Fig4.Jumper Wires

2.5 12V Battery

As all our parts between 3.3V and 12V so we used 3 batteries of 3V and resistance to keep our components safe.

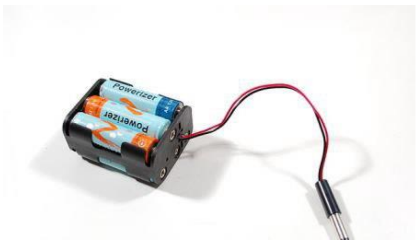


Fig5.Battery

3 Project Work Idea

The idea of a car robot using an ESP32 with an XY Remote application involves creating a controllable car robot that can be wirelessly operated via a smartphone using the XY Remote app. Here's how it generally works:

1- Hardware Setup:

ESP32: The ESP32 serves as the central control unit for the car robot.Motor Control: Connect motors and their control circuits to the ESP32. Motor drivers or an H-bridge can be used for controlling the car's movement.

2- Programming the ESP32:

Use Arduino IDE or PlatformIO to program the ESP32 microcontroller.Program the ESP32 to establish a Wi-Fi connection and communicate with the XY Remote application.

3- XY Remote Application:

Install the XY Remote app on a smartphone. Utilize the XY Remote interface to send control signals to the ESP32 via the local Wi-Fi network.

4- Controlling the Car Robot:

Through the XY Remote app, the user can interact with a virtual joystick or directional buttons on their smartphone screen. The XY Remote app transmits these control signals over Wi-Fi to the ESP32.

The ESP32 receives these signals and translates them into specific motor commands to control the car's movement, such as forward, backward, left, right, etc. This setup enables remote operation of the car robot through a wireless connection established between the smartphone running the XY Remote app and the ESP32 controlling the car's motors.

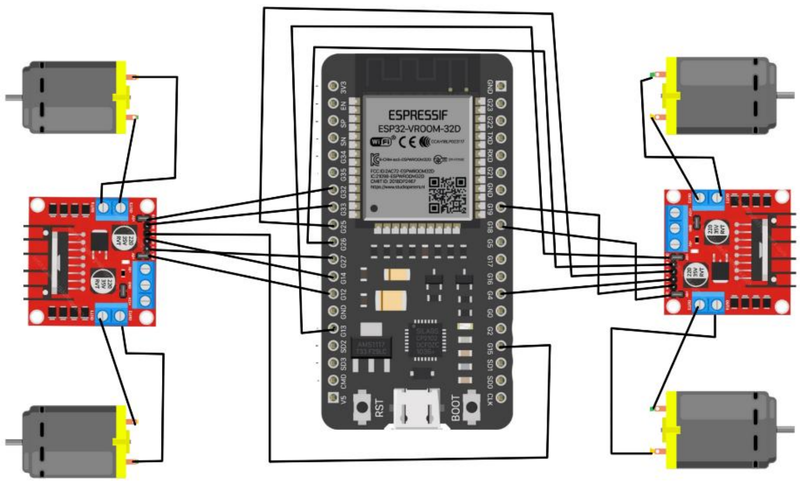
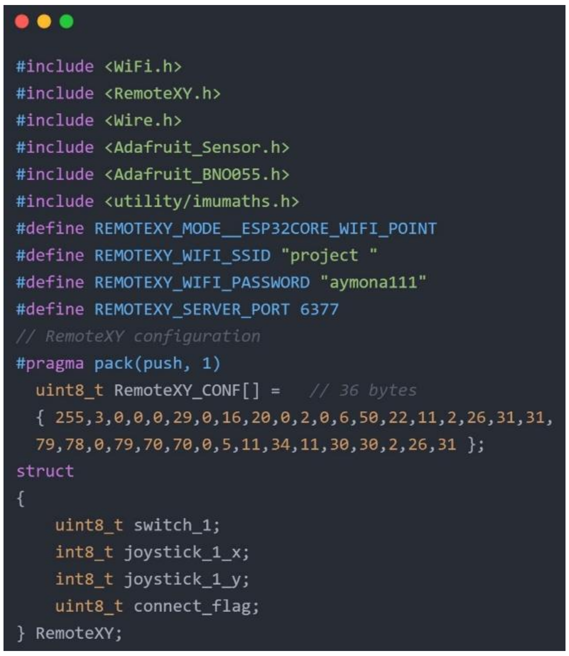


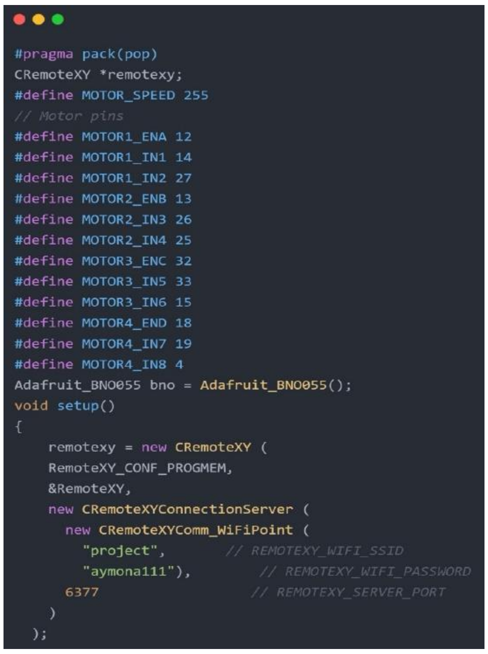
Fig6.Project Circuit

3.1 Program Code

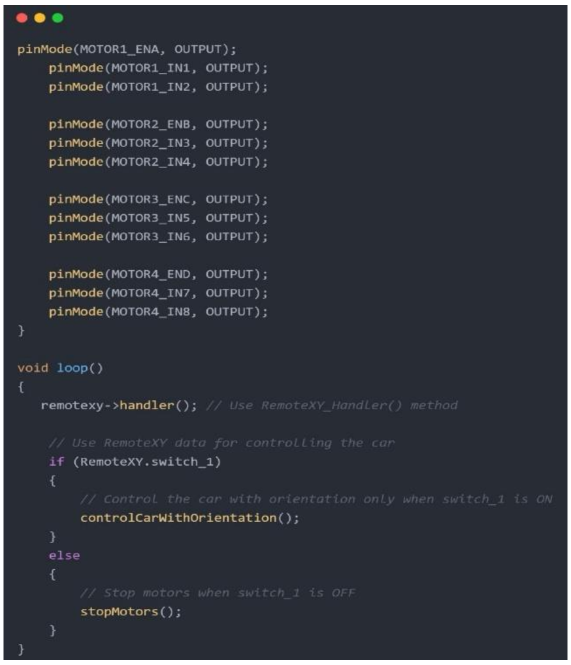
After Connecting all components as we sayed we need to know how the code will run.



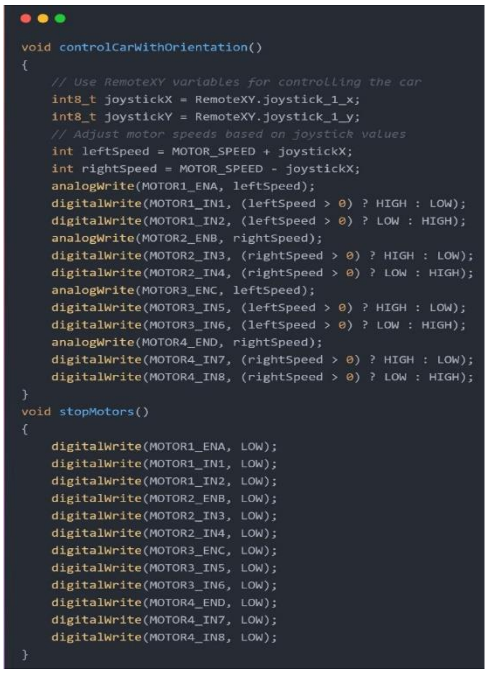
```
#include <WiFi.h>
#include <RemoteXY.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>
#define REMOTEXY_MODE__ESP32CORE_WIFI_POINT
#define REMOTEXY_WIFI_SSID "project "
#define REMOTEXY_WIFI_PASSWORD "aymona111"
#define REMOTEXY_SERVER_PORT 6377
// RemoteXY configuration
#pragma pack(push, 1)
  uint8_t RemoteXY_CONF[] =   // 36 bytes
  { 255,3,0,0,0,29,0,16,20,0,2,0,6,50,22,11,2,26,31,31,
  79,78,0,79,70,70,0,5,11,34,11,30,30,2,26,31 };
struct
{
    uint8_t switch_1;
    int8_t joystick_1_x;
    int8_t joystick_1_y;
    uint8_t connect_flag;
} RemoteXY;
```



```
#pragma pack(pop)
CRemoteXY *remotexy;
#define MOTOR_SPEED 255
// Motor pins
#define MOTOR1_ENA 12
#define MOTOR1_IN1 14
#define MOTOR1_IN2 27
#define MOTOR2_ENB 13
#define MOTOR2_IN3 26
#define MOTOR2_IN4 25
#define MOTOR3_ENC 32
#define MOTOR3_IN5 33
#define MOTOR3_IN6 15
#define MOTOR4_END 18
#define MOTOR4_IN7 19
#define MOTOR4_IN8 4
Adafruit_BNO055 bno = Adafruit_BNO055();
void setup()
{
    remotexy = new CRemoteXY (
    RemoteXY_CONF_PROGMEM,
    &RemoteXY,
    new CRemoteXYConnectionServer (
      new CRemoteXYComm_WiFiPoint (
        "project",        // REMOTEXY_WIFI_SSID
        "aymona111"),         // REMOTEXY_WIFI_PASSWORD
      6377                  // REMOTEXY_SERVER_PORT
    )
  );
```



```
pinMode(MOTOR1_ENA, OUTPUT);
    pinMode(MOTOR1_IN1, OUTPUT);
    pinMode(MOTOR1_IN2, OUTPUT);

    pinMode(MOTOR2_ENB, OUTPUT);
    pinMode(MOTOR2_IN3, OUTPUT);
    pinMode(MOTOR2_IN4, OUTPUT);

    pinMode(MOTOR3_ENC, OUTPUT);
    pinMode(MOTOR3_IN5, OUTPUT);
    pinMode(MOTOR3_IN6, OUTPUT);

    pinMode(MOTOR4_END, OUTPUT);
    pinMode(MOTOR4_IN7, OUTPUT);
    pinMode(MOTOR4_IN8, OUTPUT);
}

void loop()
{
   remotexy->handler(); // Use RemoteXY_Handler() method

    // Use RemoteXY data for controlling the car
    if (RemoteXY.switch_1)
    {
        // Control the car with orientation only when switch_1 is ON
        controlCarWithOrientation();
    }
    else
    {
        // Stop motors when switch_1 is OFF
        stopMotors();
    }
}
```



```
void controlCarWithOrientation()
{
    // Use RemoteXY variables for controlling the car
    int8_t joystickX = RemoteXY.joystick_1_x;
    int8_t joystickY = RemoteXY.joystick_1_y;
    // Adjust motor speeds based on joystick values
    int leftSpeed = MOTOR_SPEED + joystickX;
    int rightSpeed = MOTOR_SPEED - joystickX;
    analogWrite(MOTOR1_ENA, leftSpeed);
    digitalWrite(MOTOR1_IN1, (leftSpeed > 0) ? HIGH : LOW);
    digitalWrite(MOTOR1_IN2, (leftSpeed > 0) ? LOW : HIGH);
    analogWrite(MOTOR2_ENB, rightSpeed);
    digitalWrite(MOTOR2_IN3, (rightSpeed > 0) ? HIGH : LOW);
    digitalWrite(MOTOR2_IN4, (rightSpeed > 0) ? LOW : HIGH);
    analogWrite(MOTOR3_ENC, leftSpeed);
    digitalWrite(MOTOR3_IN5, (leftSpeed > 0) ? HIGH : LOW);
    digitalWrite(MOTOR3_IN6, (leftSpeed > 0) ? LOW : HIGH);
    analogWrite(MOTOR4_END, rightSpeed);
    digitalWrite(MOTOR4_IN7, (rightSpeed > 0) ? HIGH : LOW);
    digitalWrite(MOTOR4_IN8, (rightSpeed > 0) ? LOW : HIGH);
}
void stopMotors()
{
    digitalWrite(MOTOR1_ENA, LOW);
    digitalWrite(MOTOR1_IN1, LOW);
    digitalWrite(MOTOR1_IN2, LOW);
    digitalWrite(MOTOR2_ENB, LOW);
    digitalWrite(MOTOR2_IN3, LOW);
    digitalWrite(MOTOR2_IN4, LOW);
    digitalWrite(MOTOR3_ENC, LOW);
    digitalWrite(MOTOR3_IN5, LOW);
    digitalWrite(MOTOR3_IN6, LOW);
    digitalWrite(MOTOR4_END, LOW);
    digitalWrite(MOTOR4_IN7, LOW);
    digitalWrite(MOTOR4_IN8, LOW);
}
```

Fig7.Program Code

4 Conclusion

Our test project succeeded, and the car moved in the correct directions
We concluded that the car meets all our requirement:
the car robot project using an ESP32 and the XY Remote application offers an engaging way to create a controllable vehicle that can be maneuvered wirelessly through a smartphone. By integrating the ESP32 microcontroller with motor control and programming it to communicate via Wi-Fi with the XY Remote app, users can remotely operate the car robot using the virtual controls on their smartphone's screen. This project combines hardware setup, microcontroller programming, and mobile app interaction to demonstrate remote-controlled robotics, making it an educational and entertaining endeavor for learning about wireless communication and control systems in a practical setting.

5 References

[1] https://srituhobby.com/how-to-make-a-wifi-control-car-with-the-new-blynk-app-step-by-step , 2/1/2024

[2] https://www.hackster.io/lightthedreams/wifi-rc-car-with-remotexy-b9526d 2/1/2024

[3] ESP32 - Wikipedia , 1/1/2024

[4] ESP32 Wi-Fi & Bluetooth SoC | Espressif Systems , 1/1/2024