Remote Controlled 2 Wheel Drive

ESP32 Integration Guide

This project provides a complete breakdown of a 2-Wheel Drive (2WD) setup, including a full component list with estimated pricing, curated shop links with QR Codes (*links/prices are for the **Philippine market***), and source code. I've included detailed circuit diagrams to ensure a smooth assembly process.

I am a student developer taking Computer Science at Bicol University working on many side projects involving **IoT, Game Development, and Android Applications**. You can find more of my hardware experiments and software projects on my Github Profile.

If you notice any bugs in the code or improvements for the circuit design, I'd love to hear from you. Please reach out via email at **josefurei2019@gmail.com**.

Table Of Contents

Build Components, Prices & Shop Links:

Microcontroller:
For the Microcontroller, i've tested 2 products from different stores and they both worked as expected, however I used ENG LAB's ESP32 since it's cheaper and they come with a plastic case and a heatsink.

- Makerlab's ESP32 . . . . . ₱385
- ENG LAB.ph's ESP32 . . . . . ₱289 Type C, ₱279 Micro USB

  Includes:
  - Plastic Case
  - Heatsink

Makerlab's ESP32

ENG LAB's ESP32

Chassis:

Makerlab's 2 Wheel drive chassis . . . . . ₱289

Includes:
- Acrylic Chassis
- 2 DC motors
- 14500 4S li-on Battery **Holders**

Makerlab's 2 Wheel Drive Chassis

Batteries:
For the power source, **14500 Li-ion cells** are a convenient option as they fit into standard AA battery holders. However, I have opted for **18650 Li-ion cells** in this setup due to their significantly higher capacity and discharge rates.

Note: If you choose 18650 cells, you will need a dedicated battery holder and charger module, which I have listed in the 'Optional Components' section.

- Makerlab's 14500 Li-ion Battery . . . . . ₱99
- Makerlab's 18650 Li-ion Battery . . . . . ₱110 2200 mAh

14500 Li-ion Cells

18650 Li-ion Cells

Dual Motor Driver:
In this setup, I'm using the **TB6612FNG Motor Driver**. While the **L298N** is a more affordable alternative, it is much less efficient due to a significant internal voltage drop of **2V to 4V**. To ensure your motors receive maximum power and to keep the footprint small, I highly recommend sticking with the TB6612.

- fulabs.ph's TB6612 Dual Motor Driver . . . . . ₱142 Soldered Pins
- Makerlab's L298N Motor Driver . . . . . ₱85

TB6612 Dual Motor Driver

Makerlab's L298N Motor Driver

Optional but recommended components:

Voltage Regulator:

Makerlab's LM2596 . . . . . ₱155

18650 Battery Holder:

18650 3S Battery Holder . . . . . ₱69

Battery Charger Module:

TP4056 Battery Charger Module . . . . . ₱33 Type C w/protection

LM2596 Regulator

18650 Battery Holder

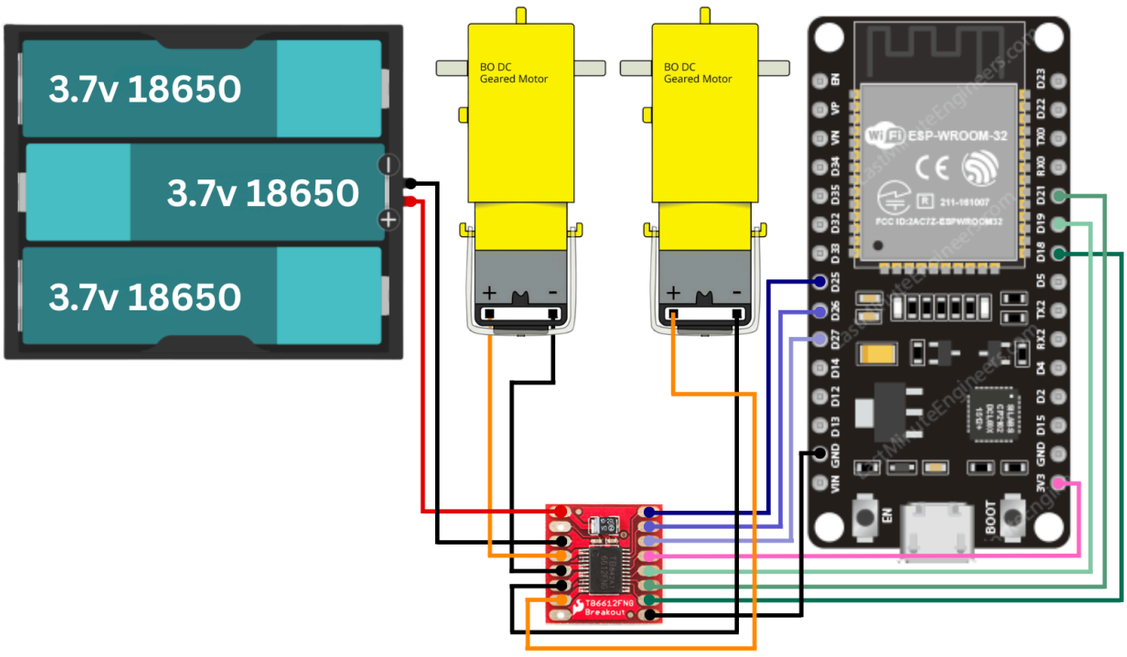
TP4056 Module

TOTAL PRICE: ₱1307

NOTE:

Total price is calculated WITH OPTIONAL COMPONENTS & WITHOUT SHIPPING. Prices may have changed at the time (February 2026) after this document was written.

Circuit Diagram



Pin Connections

TB6612FNG Motor Driver	Battery
VM	Battery (+)
GND	Battery (-)
TB6612FNG Motor Driver	**DC Motor**
A01	Motor A (+)
A02	Motor A (-)
B02	Motor B (-)
B01	Motor B (+)
TB6612FNG Motor Driver	**ESP32**
PWMA	GPIO 25
AI2	GPIO 26
AI1	GPIO 27
STBY	3v3
BI1	GPIO 18
BI2	GPIO 19
PWMB	GPIO 21
GND	ESP32 GND

Power Note:

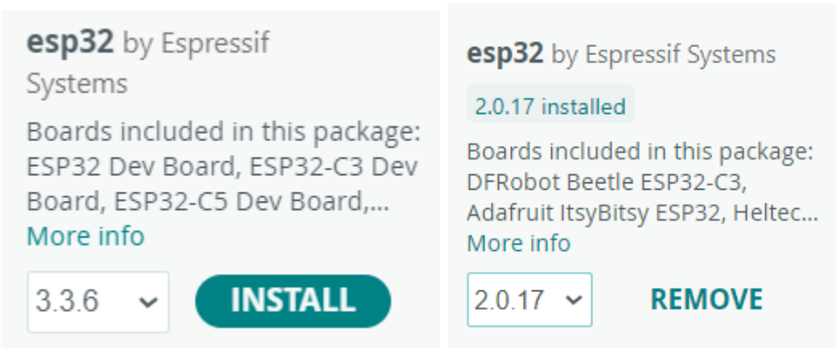
In this setup I used 3 18650 Li-ion Cells in series to get 11.5v for my DC motors, however, I am powering the ESP32 via a power bank through the USB port. Alternatively, you can use the **Vin pin**. If you choose this route, ensure your input voltage stays within the recommended **5V to 12V** range. Exceeding these limits can cause the onboard voltage regulator to overheat, leading to permanent hardware failure. **Crucial:** Always verify your specific ESP32 model's Vin specifications before connecting an external power source.

If you don’t have a power bank you can use, you can try using a step down voltage regulator.

Board Manager & Libraries (Arduino IDE)

1. Start Arduino IDE and go to File > Preferences at Additional boards manager URL paste this httpsraw.githubusercontent.comespressifarduino-esp32gh-pagespackage_esp32_index.json

2. Install esp32 by Espressif Systems in Arduino IDE boards manager. The latest version (as of writing this document) does not work on my system, however the version 2.0.17 works fine.



NOTE:

There might be version changes at the time (February 2026) of writing this document.

Code

The source code for this project and many more tutorials are hosted on my Github Profile for version control and accessibility.

```
#include <WiFi.h>
#include <WebServer.h>

// WiFi Configuration
const char* ssid = "My2WheelDrive";
const char* password = "password123";

// Motor Pins
const int motorA_PWM = 25; // Speed control
const int motorA_IN1 = 26; // Direction 1
const int motorA_IN2 = 27; // Direction 2

const int motorB_PWM = 18; // Speed control
const int motorB_IN1 = 19; // Direction 1
const int motorB_IN2 = 21; // Direction 2

WebServer server(80);
int currentSpeed = 200;

// HTML Interface
// The R"=====( ... )=====" allows to write raw HTML
const char HTML_CONTENT[] PROGMEM = R"=====(
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <style>
    body { font-family: sans-serif; text-align: center;
background: #222; color: white; }
    .btn { width: 80px; height: 80px; margin: 10px; background:
#444; color: #00e676; border: 2px solid #00e676; border-radius:
10px; font-size: 24px; }
    .btn:active { background: #00e676; color: black; }
    .slider { width: 80%; margin-top: 40px; }
  </style>
</head>
<body>
  <h2>Robot Control</h2>
  <div><button class="btn" ontouchstart="send('forward')"
ontouchend="send('stop')">▲</button></div>
  <div>
    <button class="btn" ontouchstart="send('left')"
ontouchend="send('stop')">◀</button>
```

```
    <button class="btn" ontouchstart="send('right')"
ontouchend="send('stop')">▶</button>
  </div>
  <div><button class="btn" ontouchstart="send('backward')"
ontouchend="send('stop')">▼</button></div>

  <p>Speed: <input type="range" min="150" max="255"
class="slider" onchange="fetch('/speed?val=' + this.value)"></p>

  <script>
    function send(cmd) { fetch('/' + cmd); }
  </script>
</body>
</html>
)=====";

// Motor Logic
void drive(int speedA, int speedB, int a1, int a2, int b1, int
b2) {
  digitalWrite(motorA_IN1, a1); digitalWrite(motorA_IN2, a2);
  digitalWrite(motorB_IN1, b1); digitalWrite(motorB_IN2, b2);
  analogWrite(motorA_PWM, speedA);
  analogWrite(motorB_PWM, speedB);
}

void setup() {
  Serial.begin(115200);

  // Set all motor pins as outputs
  pinMode(motorA_PWM, OUTPUT); pinMode(motorA_IN1, OUTPUT);
pinMode(motorA_IN2, OUTPUT);
  pinMode(motorB_PWM, OUTPUT); pinMode(motorB_IN1, OUTPUT);
pinMode(motorB_IN2, OUTPUT);

  // Start WiFi Access Point, basically a hotspot
  WiFi.softAP(ssid, password);

  // Display ssid and ip address to serial monitor
  Serial.print("Connect to WiFi: "); Serial.println(ssid);
  Serial.print("Visit: "); Serial.println(WiFi.softAPIP());

  // When the user connects to the wifi and go to the ip
address,
  // the HTML_CONTENT will be served / shown to the user,
  // it contains the controls as written in the html
  server.on("/", []() { server.send(200, "text/html",
HTML_CONTENT); });
```

```
  // These will run based on what button the user taps
  server.on("/forward",  []() { drive(currentSpeed,
currentSpeed, 1, 0, 1, 0); server.send(200); });
  server.on("/backward", []() { drive(currentSpeed,
currentSpeed, 0, 1, 0, 1); server.send(200); });
  server.on("/left",     []() { drive(currentSpeed,
currentSpeed, 0, 1, 1, 0); server.send(200); });
  server.on("/right",    []() { drive(currentSpeed,
currentSpeed, 1, 0, 0, 1); server.send(200); });
  server.on("/stop",     []() { drive(0, 0, 0, 0, 0, 0);
server.send(200); });

  // Basically grabs the speed from the slider created on the
HTML_CONTENT
  server.on("/speed", []() {
    if (server.hasArg("val")) currentSpeed =
server.arg("val").toInt();
    server.send(200);
  });

  server.begin();
}

void loop() {
  // serve client
  server.handleClient();
}
```