# Lego EV2 Large Motor Notes

## Cable Color Coding (if applicable):

* Red: VCC (Power)
* Black: GND (Ground)
* Yellow: PWM (Speed Control)
* Green: Direction Control

## ESP32 Pinout:

* GPIO 12: PWM signal to control motor speed.
* GPIO 13: Direction control (forward/reverse).
* GND: Common ground with the motor driver.

The **EV3 Large Servo Motor** uses a **6-wire cable** because it includes an **encoder** for precise motor control. However, the **L298N motor driver** only supports simple **DC motor control** (without encoder feedback).

### ****Understanding the EV3 Large Motor Wires****

The 6 wires inside the cable are typically **color-coded as follows**:

|  |  |  |
| --- | --- | --- |
| Wire Color | Function | L298N Connection |
| **Red** | Motor Power A | Motor Output A (OUT1) |
| **Black** | Motor Power B | Motor Output B (OUT2) |
| **Green** | Encoder GND | Not used |
| **White** | Encoder VCC | Not used |
| **Blue** | Encoder Channel A | Not used |
| **Yellow** | Encoder Channel B | Not used |

Since the **L298N cannot read the encoder**, we will **only connect the red and black wires** to drive the motor like a regular DC motor.

I discovered that White and Black if connected directly to 9V move the motor. So the above table the Red and Black are Black and White.

|  |  |  |  |
| --- | --- | --- | --- |
| LEGO Motor Pin | LEGO Cable Wire Color\* | BeagleBone Blue Connector/Pin | Description |
| 1 | white | M1/2 | Motor – |
| 2 | black | M1/1 | Motor + |
| 3 | red | E1/1 | GND |
| 4 | green | E1/2 | 3.3V |
| 5 | yellow | E1/3 | quadrature encoder A |
| 6 | blue | E1/4 | quadrature encoder B |

### ****Wiring EV3 Large Motor to L298N & ESP32****

#### ****L298N to ESP32****

* **L298N IN1** → ESP32 **GPIO 18**
* **L298N IN2** → ESP32 **GPIO 19**
* **L298N ENA** → ESP32 **GPIO 5** (for PWM speed control)
* **L298N GND** → ESP32 GND
* **L298N VCC** → **External Power Source (e.g., 9V or 12V battery pack)**

#### ****EV3 Large Motor to L298N****

* **Motor Red Wire** → **L298N OUT1**
* **Motor Black Wire** → **L298N OUT2**

### ****MicroPython Code for ESP32 (L298D)****

This code allows you to **control the speed and direction** of the EV3 Large Motor using **PWM signals**.

from machine import Pin, PWM

import time

# Define motor control pins

IN1 = Pin(18, Pin.OUT) # Direction 1

IN2 = Pin(19, Pin.OUT) # Direction 2

ENA = PWM(Pin(5)) # Enable (PWM for speed control)

ENA.freq(1000) # Set PWM frequency to 1kHz

def motor\_forward(speed):

ENA.duty(speed) # Set speed (0-1023)

IN1.value(1)

IN2.value(0)

def motor\_backward(speed):

ENA.duty(speed) # Set speed (0-1023)

IN1.value(0)

IN2.value(1)

def motor\_stop():

ENA.duty(0) # Stop motor

IN1.value(0)

IN2.value(0)

# Test motor

motor\_forward(512) # 50% speed forward

time.sleep(2)

motor\_backward(512) # 50% speed backward

time.sleep(2)

motor\_stop() # Stop motor

### ****Important Notes****

* The **L298N needs an external power supply** (9V or 12V recommended).
* **You will lose encoder feedback**, meaning you cannot get precise position control.
* If you want **precise movement control**, you need an **H-bridge with encoder support** like the **DRV8833** and an ESP32 library to read the encoder pulses.

Let me know if you want to **read the encoder** or need a **KiCad schematic** for this setup!

## How to use ENA and ENB

Yes, the **L298N motor driver** has two enable pins:

* **ENA (Enable A)** → Controls Motor A (OUT1 & OUT2)
* **ENB (Enable B)** → Controls Motor B (OUT3 & OUT4)

Since you are using **only one motor (EV3 Large Motor)**, you will connect **ENA** to your ESP32 for speed control.

### ****What Do the Switches on ENA and ENB Do?****

The small **jumpers (switches) on ENA and ENB** determine if **PWM control** is enabled:

* **Jumper ON** → The motor runs at full speed (PWM is disabled).
* **Jumper OFF** → You can control the speed using **PWM** from the ESP32.

Since we want to **control the speed**, **remove the jumper on ENA** and connect it to the ESP32.

### ****Updated Wiring for EV3 Large Motor and L298N****

#### ****ESP32 to L298N****

|  |  |  |
| --- | --- | --- |
| L298N Pin | ESP32 Pin | Function |
| **IN1** (Motor A) | **GPIO 18** | Direction 1 |
| **IN2** (Motor A) | **GPIO 19** | Direction 2 |
| **ENA** (Enable A) | **GPIO 5 (PWM)** | Speed Control |
| **GND** | **ESP32 GND** | Common Ground |
| **VCC** | **External Power (9V-12V)** | Motor Power |

#### ****EV3 Large Motor to L298N****

|  |  |
| --- | --- |
| EV3 Wire Color | L298N Pin |
| **Red** (Motor A) | **OUT1** |
| **Black** (Motor A) | **OUT2** |

**Important:**

* Remove the **ENA jumper** so the ESP32 can control the motor speed via PWM.
* The **ENB jumper can stay ON** since we are not using Motor B.

### ****MicroPython Code for ESP32 with L298N****

This code allows you to control both **direction** and **speed** of the EV3 Large Motor.

from machine import Pin, PWM

import time

# Define motor control pins

IN1 = Pin(18, Pin.OUT) # Direction 1

IN2 = Pin(19, Pin.OUT) # Direction 2

ENA = PWM(Pin(5)) # Enable (PWM for speed control)

ENA.freq(1000) # Set PWM frequency to 1kHz

def motor\_forward(speed):

ENA.duty(speed) # Set speed (0-1023)

IN1.value(1)

IN2.value(0)

def motor\_backward(speed):

ENA.duty(speed) # Set speed (0-1023)

IN1.value(0)

IN2.value(1)

def motor\_stop():

ENA.duty(0) # Stop motor

IN1.value(0)

IN2.value(0)

# Test motor

motor\_forward(512) # 50% speed forward

time.sleep(2)

motor\_backward(512) # 50% speed backward

time.sleep(2)

motor\_stop() # Stop motor

### ****Summary****

* **Remove the jumper on ENA** to enable speed control.
* **Connect ENA to ESP32 (GPIO 5) for PWM speed control.**
* **Motor direction is controlled via IN1 and IN2.**
* **Use a 9V or 12V power source for the L298N.**

Let me know if you need help with **encoder feedback** or a **KiCad schematic**!

## Input Voltage

The **L298N motor driver** has two power inputs:

* **12V (or 9V-12V input) → Main power for the motors**
* **5V (regulated output) → Power for the L298N logic circuit**

### ****How to Power the L298N Correctly****

#### ****Which Power Input to Use?****

* **Use the 12V input** (or 9V if you have a lower-voltage power supply).
* The L298N has a built-in **5V voltage regulator** that powers the logic circuit.
* The **5V pin is an output**, so you can use it to power the ESP32 if needed.

#### ****What Does the Jumper Above 12V/5V Do?****

* This is the **5V regulator jumper** that controls whether the L298N uses its internal **5V regulator**.
* If the jumper is **ON** (default):
  + The L298N’s **internal 5V regulator is active**.
  + The L298N takes **12V from the input** and converts it to **5V for its logic circuits**.
  + The **5V pin will act as an output**, and you can use it to power other components.
* If the jumper is **OFF**:
  + The internal 5V regulator is **disabled**.
  + You must **provide an external 5V supply** to the 5V pin for the logic circuit.

#### ****Recommended Jumper Setting****

* **Leave the jumper ON** if you are using **9V or 12V power** for the L298N.
* **Remove the jumper** if you want to power the ESP32 separately with **its own 5V source** and avoid interference.

### ****Final Wiring: ESP32 + L298N + EV3 Large Motor****

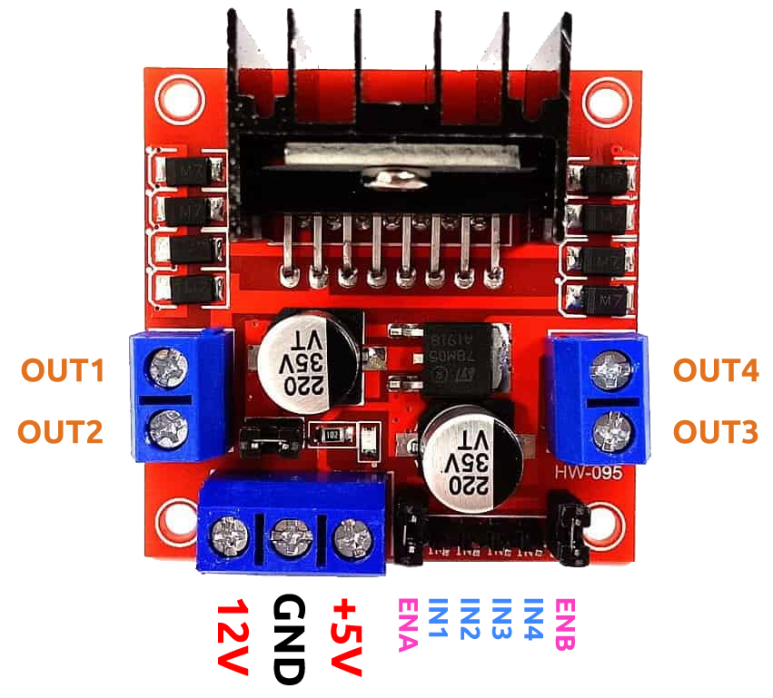
|  |  |  |
| --- | --- | --- |
| L298N Pin | ESP32 Pin | Function |
| **IN1** | **GPIO 18** | Direction 1 |
| **IN2** | **GPIO 19** | Direction 2 |
| **ENA** | **GPIO 5 (PWM)** | Speed Control |
| **GND** | **ESP32 GND** | Common Ground |
| **12V** | **External 9V-12V Power** | Motor Power |
| **5V** | (Leave unconnected, or use it to power ESP32 if needed) |  |

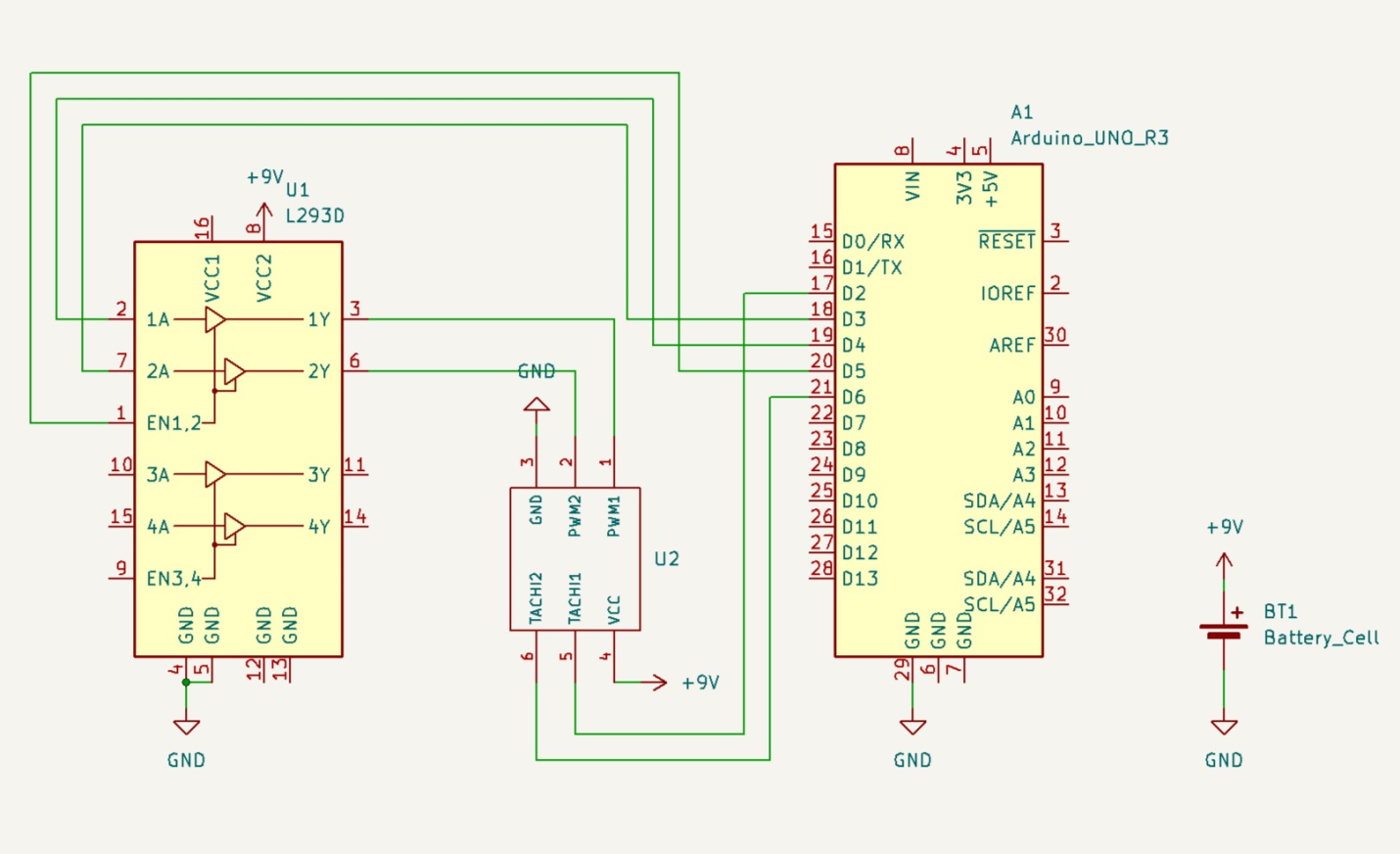
|  |  |
| --- | --- |
| EV3 Wire Color | L298N Pin |
| **Red** (Motor A) | **OUT1** |
| **Black** (Motor A) | **OUT2** |

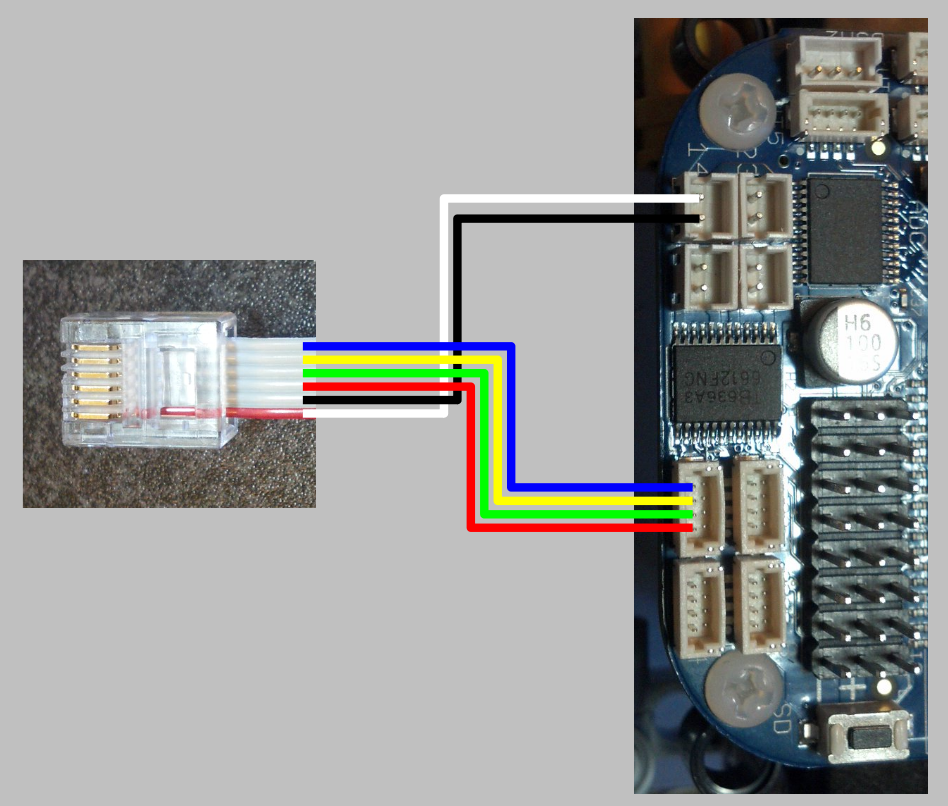
### ****Summary****

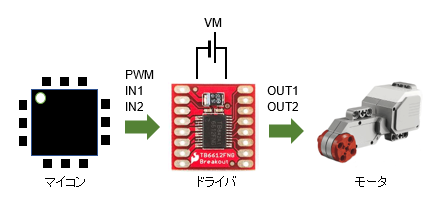
✅ **Use the 12V input** for motor power.  
✅ **Keep the 5V jumper ON** if using 9V-12V input.  
✅ **Remove the jumper** if you power ESP32 separately with a 5V supply.

Let me know if you need a **KiCad schematic** for this setup!









## Use L9110S motor drives

from machine import Pin, PWM

import time

# Define motor control pins

IN1 = Pin(18, Pin.OUT) # Direction 1

IN2 = Pin(19, Pin.OUT) # Direction 2

# Use PWM for speed control

speed\_pwm = PWM(IN1, freq=1000)

def motor\_forward(speed):

speed\_pwm.duty(speed) # Set speed (0-1023)

IN1.value(1)

IN2.value(0)

def motor\_backward(speed):

speed\_pwm.duty(speed) # Set speed (0-1023)

IN1.value(0)

IN2.value(1)

def motor\_stop():

speed\_pwm.duty(0) # we need this to stop the motor

IN1.value(0)

IN2.value(0)

# Test motor

motor\_forward(512) # 50% speed forward

time.sleep(2)

motor\_backward(512) # 50% speed backward

time.sleep(2)

motor\_stop() # Stop motor

## Read the Encoders (Changed from what chatGPT suggested)

For reading the encoders we use the following wires:

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | red | E1/1 | GND |
| 4 | green | E1/2 | 3.3V |
| 5 | yellow | E1/3 | quadrature encoder A |
| 6 | blue | E1/4 | quadrature encoder B |

Connection to ESP32 (C3 Zero Used)

|  |  |  |
| --- | --- | --- |
| EV3 Wire | Function | ESP32 Pin (C3Zero) |
| **Blue** | Encoder A (Signal) | **GPIO 4** (or any input) |
| **Yellow** | Encoder B (Signal) | **GPIO 5** (or any input) |
| **Red** | Encoder Ground | **GND** |
| **Green** | Encoder Power | **3.3V or 5V** |

|  |
| --- |
| from machine import Pin  import time  # Define Encoder Pins  ENC\_A = Pin(4, Pin.IN, Pin.PULL\_UP) # Encoder A  ENC\_B = Pin(5, Pin.IN, Pin.PULL\_UP) # Encoder B  encoder\_count = 0 # Tracks motor position  def encoder\_callback(pin):  global encoder\_count  if ENC\_B.value() == 1: # Check B signal to determine direction  encoder\_count += 1 # Forward  else:  encoder\_count -= 1 # Reverse  # Attach interrupt to read encoder pulses  ENC\_A.irq(trigger=Pin.IRQ\_RISING, handler=encoder\_callback)  # Main loop  while True:  print("Encoder Position:", encoder\_count)  time.sleep(0.1) |

## Reference Links

<https://lechnology.com/2017/03/using-lego-mindstorms-motors-with-beaglebone-blue/>

<https://en.depfields.com/dcmotor-pwmcontrol/>