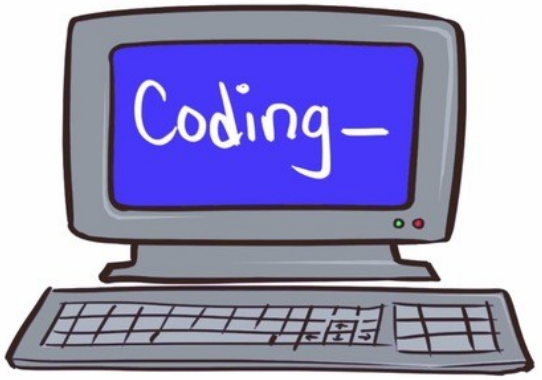
Class 39



hbridge

C++ vs C

C++ is C with some extra functionality added.

In C++ on the Arduino, these additions we can use a lot:

1) Classes & Objects (object oriented)
2) Functions with same name but diff parameter list (overloading)
3) Default values for function parameters

You could write plain C code for Arduino and it usually runs fine.

Arduino code skeleton

1) #include all needed libraries
2) in global code space, create global variables (incl objects)
3) in setup(), initialise everything as needed
4) in loop(), write repeating code
5) if needed, write any other functions needed

Declaring a variable

`int a;` a declaration. C now knows about it, and assigns a (empty) memory slot for it.

`a = 66;` a definition. C puts a good value there.

`int a=67;` a combination declaration/definition

(similarly)
Declaring a function

```
int mult2( int, int);
```

C will expect and handle a function defined later to that same parameter configuration

(ie taking 2 ints, and returning an int)

Defining a function

```
int mult2(int x, int y)
{
    return x * y ;
}
```

If the function was not declared earlier, it serves as a combo declaration/definition

Why separately declare a function?

So that we can place its definition further down our code,

and “calling” it early down our code won’t trigger a “function not declared” error.

More generally, it can often be easier to simply arrange our functions so that lower-down ones usually call further-up ones.

printf

printf("Speed = %d m/s Bearing %d degr\n", speed, bearing);

Speed = 5 m/s Bearing 125 degr

%s string

%d integer

%x integer in hex

%f float number

%c character

Recently, arduino supports Serial.printf()

Optional sizing of a field

%8.2f float printed as 8 characters, incl 2 decimals

%04d integer printed as 4 char, padded if nec (eg 0045)

Robot Construction ...

battery pack(s)

motors

servo(s)

mount for imu

sonar

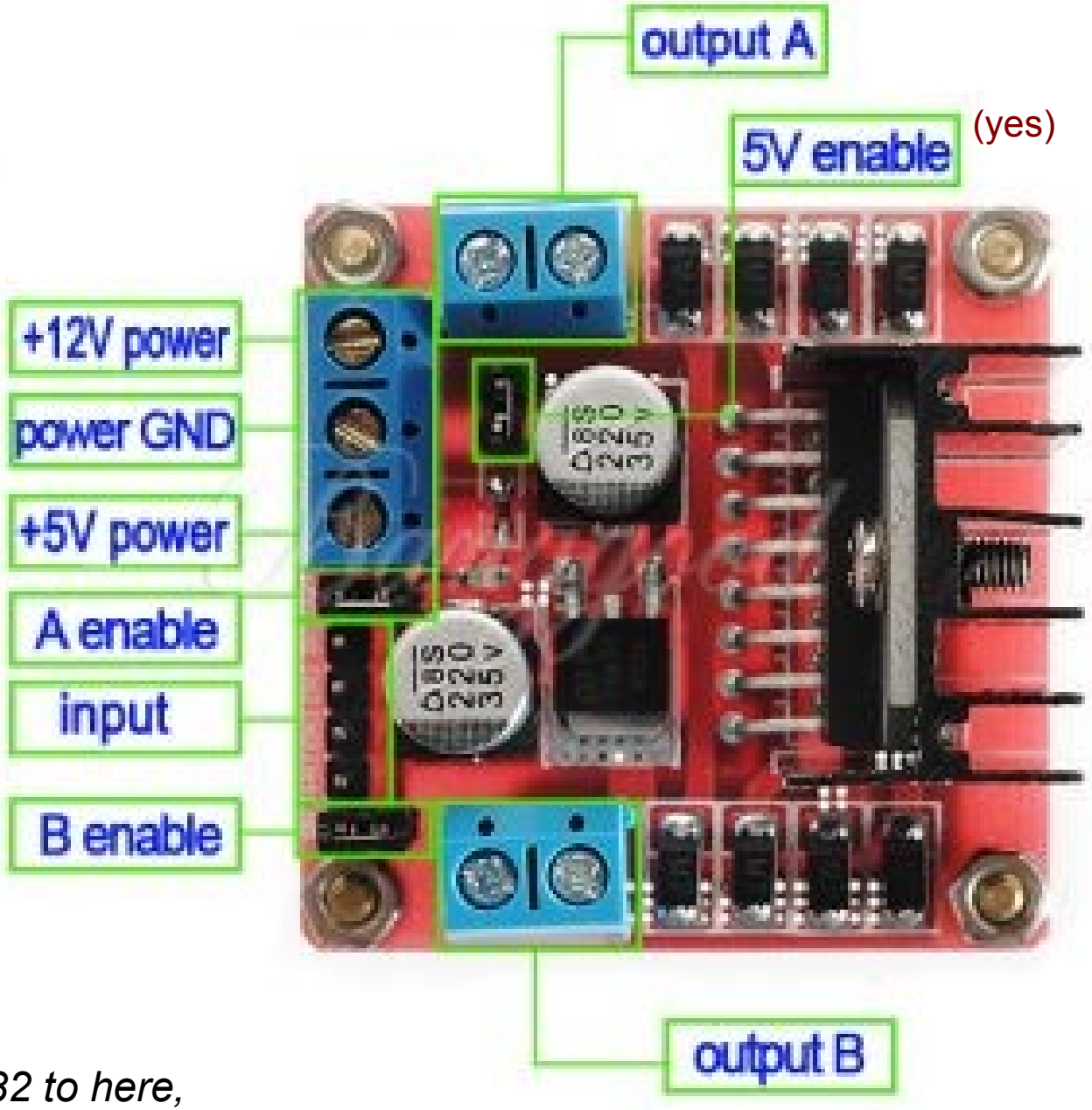
feelers? / bumpers?

6volt or 7.5volt battery pack
(4 or 5 AA cells)

"+12V" = Battery Pack positive
"power GND" = Battery Gnd
and GND from ESP32
"+5V" = not connected

A/B "enable" = PWM
for speed control.
(Remove these 2 jumpers)

In1/2 and In3/4 = run/direction
for motors A/B



*Note there **is** a GND from ESP32 to here,*
*but **no** 5V or 3.3V from ESP32 to here.*

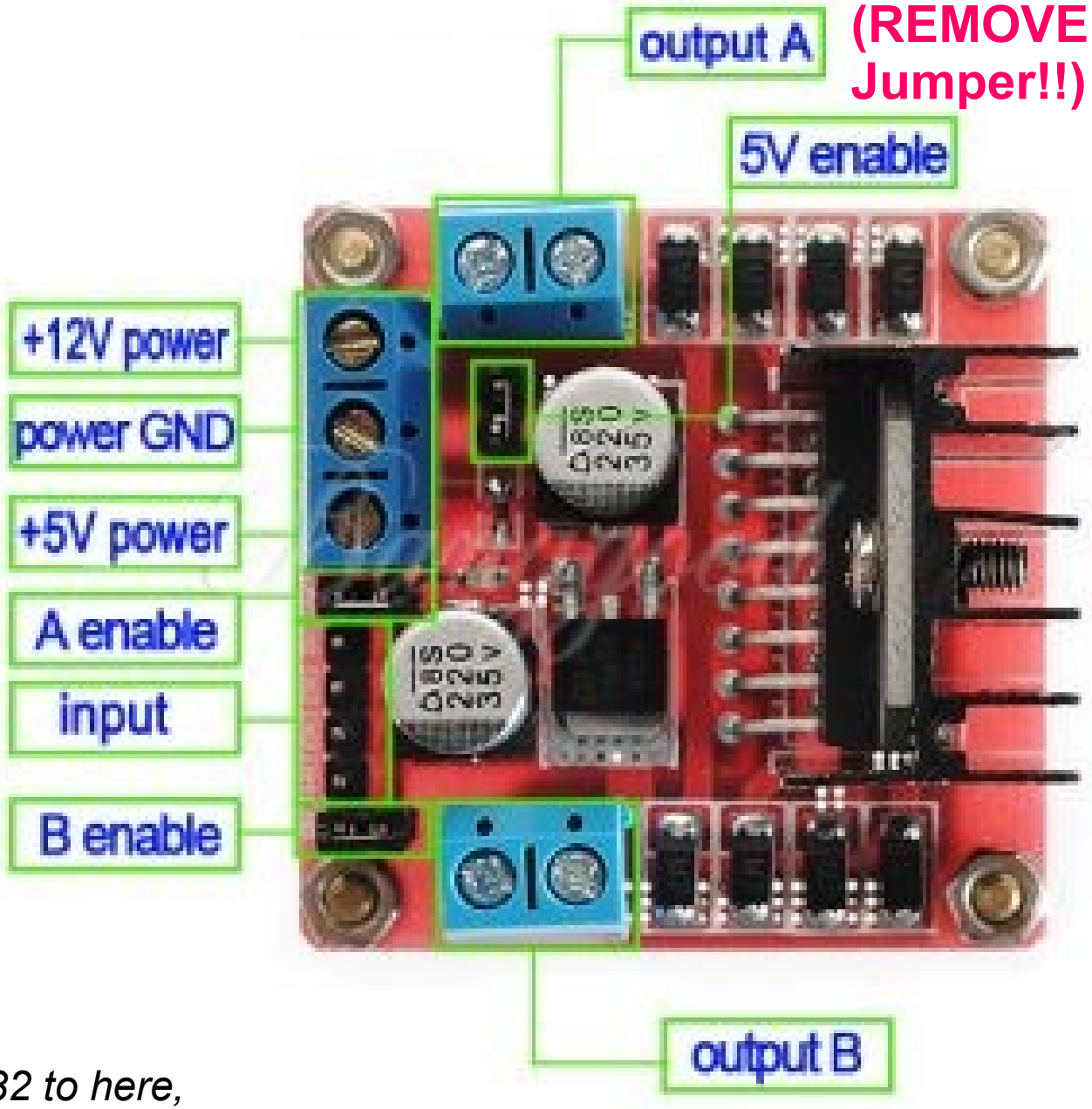
Your 2 motors A/B
(at 6 or 7.5V)

4.5volt battery pack
(3 AA cells)
Different connections:

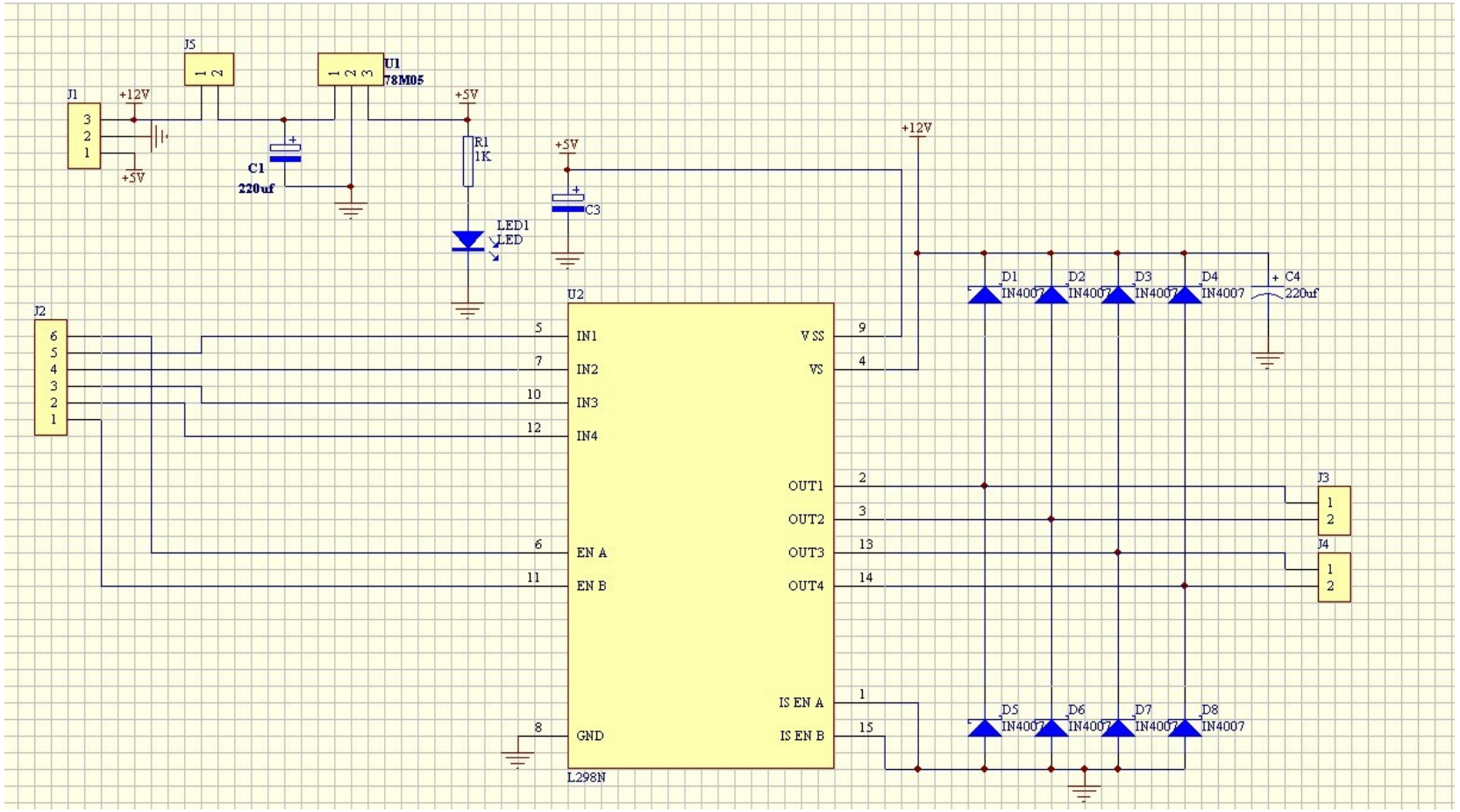
"+12V" = Battery Pack positive
"power GND" = Battery Gnd
and GND from ESP32
"+5V" = 5V from ESP32

A/B "enable" = PWM
for speed control.
(Remove these 2 jumpers)

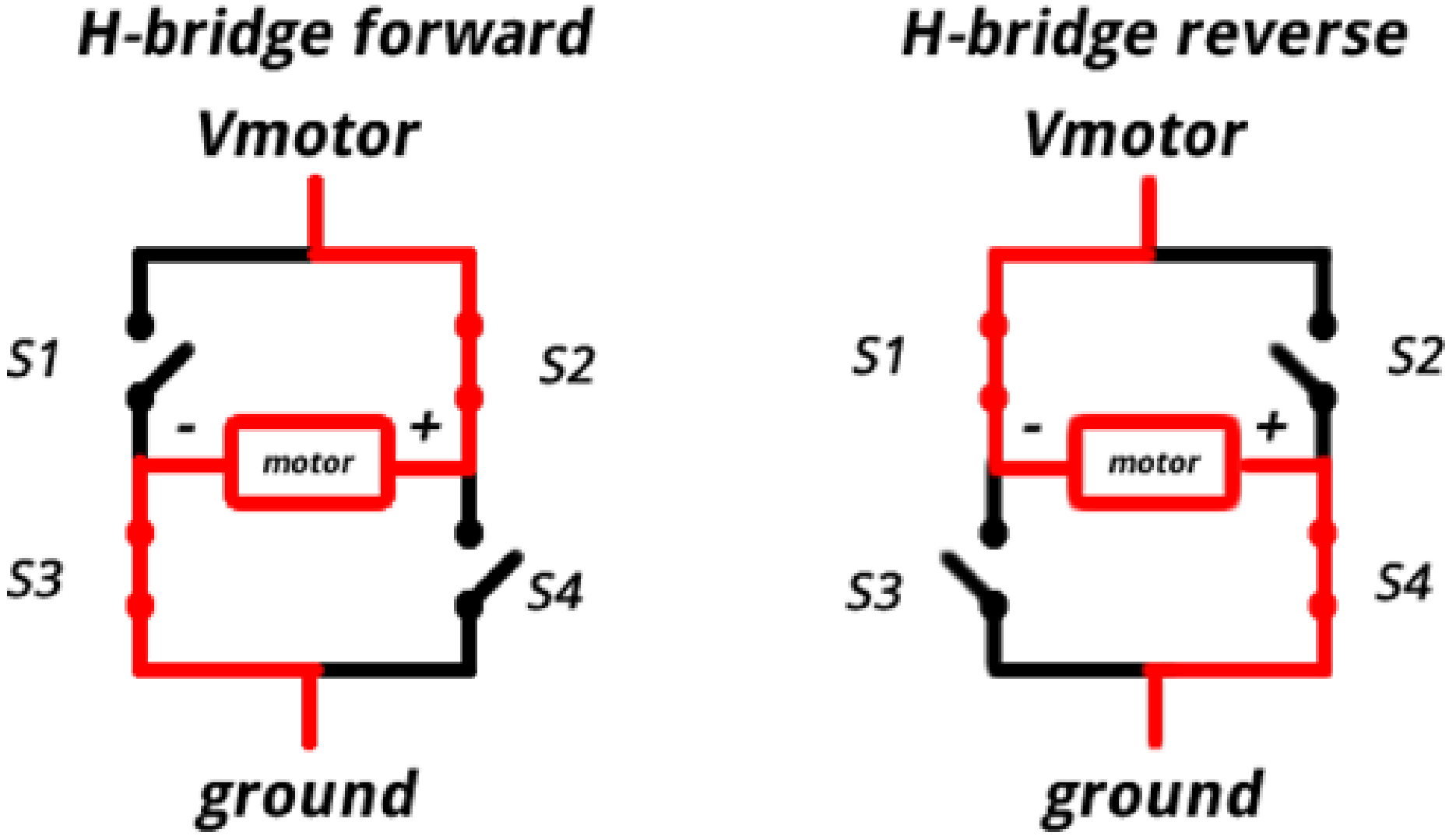
In1/2 and In3/4 = run/direction
for motors A/B



*Note there **is** a GND from ESP32 to here,*
*but **AND** 5V from ESP32 to here.*


J5
U1
78M05
J1
+12V
+5V
C1
220uf
R1
1K
LED1
LED
C3
D1
IN4007
D2
IN4007
D3
IN4007
D4
IN4007
C4
220uf
J2
U2
IN1
IN2
IN3
IN4
EN A
EN B
GND
VSS
VS
OUT1
OUT2
OUT3
OUT4
IS EN A
IS EN B
L298N
J3
J4
D5
IN4007
D6
IN4007
D7
IN4007
D8
IN4007

H-Bridge


H-bridge forward
Vmotor
S1
S2
-
motor
+
S3
S4
ground
H-bridge reverse
Vmotor
S1
S2
-
motor
+
S3
S4
ground

The Truth Table

Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

main Library calls:

(See L298N.h)

Constructor: L298N motor1(Aen, Ain1, Ain2);

motor1.setSpeed(pwmval);

motor1.getSpeed();

motor1.run(direction);

motor1.forward(); or .backward()

motor1.forwardFor(time); backwardFor(...)

motor1.stop();

motor1.isMoving();

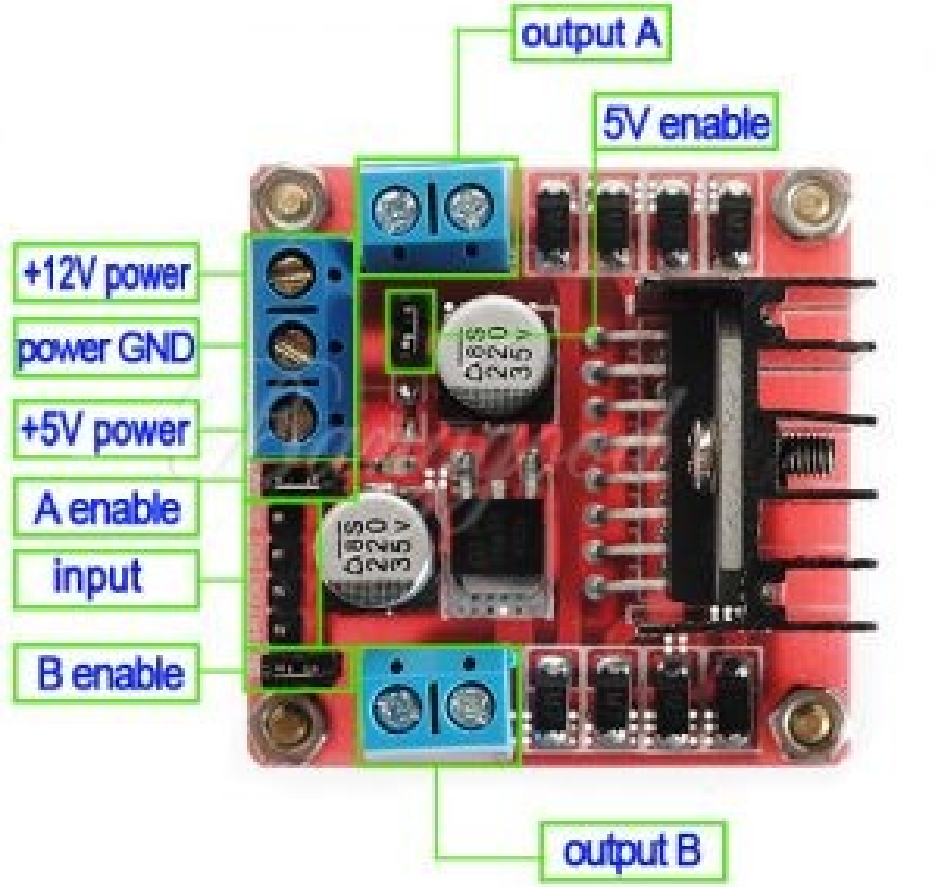
Using servos also?

The “+5V power” and “power GND”

should be a legitimate 5V supply for a couple of RC servos,

if you use battery pack 6 or 7.5V.

Their control pin would still come from a ESP32 gpio pin as normal.



Hbridge_blynk.ino

We need to add motor speed/run commands

Input – from blynk calls

Output – to the L298 library calls