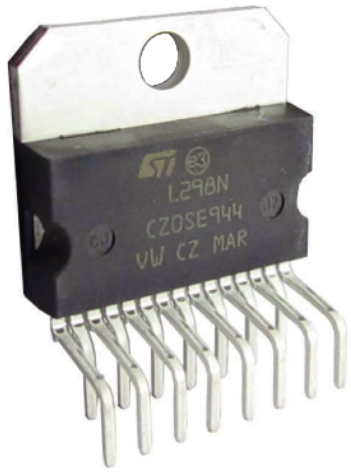
L298N Motor Diver

L298N

The L298N is a dual full-bridge motor driver designed to work with microcontrollers like Uno.



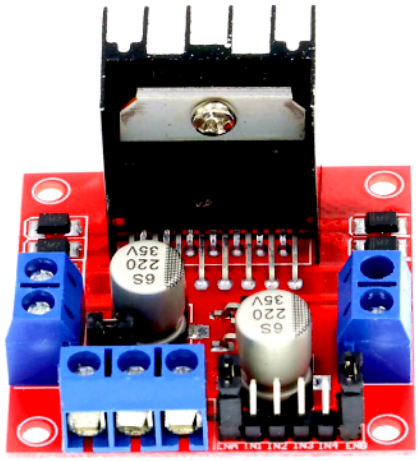


Image 1

Connections

Power Output Stage

The L298N integrates two power output stages (*A* & *B*). The power output stage is a bridge configuration and its outputs can drive an inductive load in common or differential mode, depending on the state of the inputs.

Input State

Each bridge is driven by means of four gates the input of which are *In1*, *In2*, *EnA* and *In3*, *In4*, *EnB*. The *In* is input. The *En* is enable. The *inputs* are used to set the bridge *states* with the *enable* used for pulse width modulation (PWM). PWM is a digital pulse length that varies a logic voltage varies from 0 to 5V. Thus PWM is used to adjust the motor speed. All inputs are Transistor-Transistor Logic (TTL) compatible exactly what the Atmega328P uses.

PWM

Pulse width modulation is a generated digital signal basically a square wave with ON and OFF time intervals. The ON and OFF times can be adjusted (or modulated) arbitrarily. By varying the amount of ON time called the duty cycle, the PWM behaves like an analog output by using digital

signals. The ON time for Arduino Uno uses 5 volts when turned HIGH and the OFF time uses 0 volts when turned LOW. On the Arduino Uno, the PWM pins are 3, 5, 6, 9, 10 and 11. The frequency of PWM signal on pins 5 and 6 is 980Hz and on other pins, is 490Hz. The PWM pins are labeled with ~ sign on the Arduino Uno board. By adjusting the duty cycle, the analog like output will supply a voltage range from 0 to 5 volts.[1]

$[T_{HIGH}]$ is the (ON time) when the signal is high.

$[T_{LOW}]$ is the (OFF time) when the signal is low.

The ***period*** - T is the time required for one complete cycle to pass a given point which is the sum of the ON and OFF times. The *period* is a constant where the ON and OFF times may vary.

$$T_{Total} = T_{HIGH} + T_{LOW}$$

The ***frequency*** - f is the number of cycles required to complete in 1 second.

$$f\,(Hz) = \frac{1\,cycle}{T\,(s)} = \frac{1}{T\,(s)}$$

The Arduino function *pulseIn*(pin, value) can use either the value HIGH or the value LOW on an Arduino pin. If the value used is HIGH, the function pulseIn() waits for the pin to go from LOW to HIGH and then starts timing, then waits for the pin to go LOW and then, it stops timing. The function returns the length of the pulse in ***microseconds*** (μs).[2] Therefore the equation becomes:

$$f = \frac{1}{T_{Total}\,(\mu s)} \cdot \frac{1\,000\,000\,(\mu s)}{1\,(s)} = \frac{1\,000\,000}{T_{Total}\,(s)}$$

The PWM ***duty cycle*** is:

$$Duty\ Cycle\ (percent) = \frac{T_{HIGH}}{T_{Total}} \cdot 100$$

Experiment with one of the PWM ~ pins with an input pin using the pulseIn() function.[3]

---

1. Arduino PWM Tutorial (https://create.arduino.cc/projecthub/muhammad-aqib/arduino-pwm-tutorial-ae9d71)

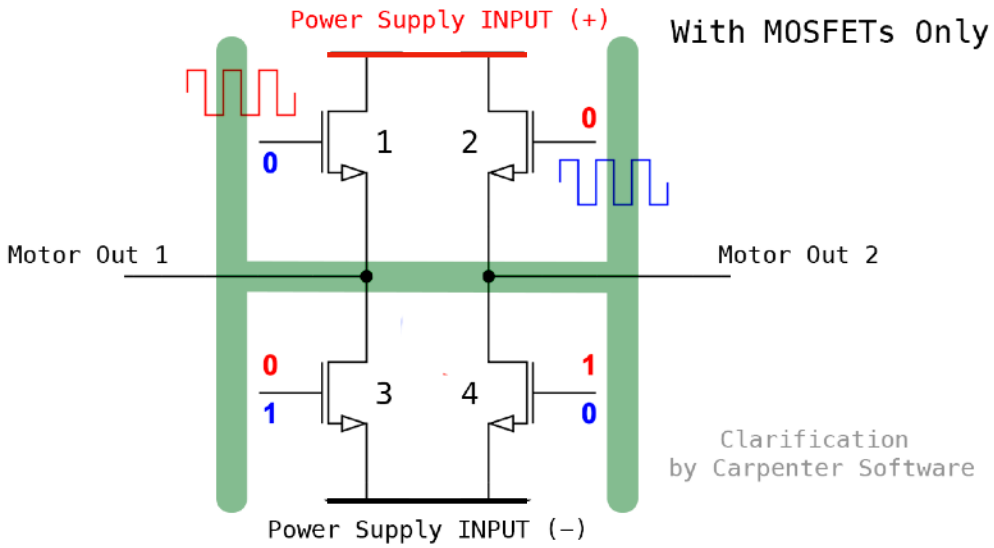
2. Arduino Website (https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/)

3. Duty Cycle Meter (https://create.arduino.cc/projecthub/boaz/duty-cycle-calculator-and-frequency-meter-f4d763)

H-Bridge Circuit

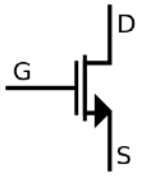
An H-bridge is an electronic circuit that switches the polarity of a voltage applied to a load. These circuits are often used in DC motors to run forwards or backwards. The L298 graphical representation of an H-bridge is shown as a circuit in Image 2.

Image 2



The four MOSFETs numbered 1 through 4 form a single H-bridge circuit. There is a power supply *input* and there is the *output* to the motor outs.

Image 3



N-Channel MOSFET

Image 3 shows a N-Channel MOSFET with the three connections, the *D*rain, the *S*ource, and the *G*ate. The discussion here is that the MOSFET *drains* the power supply and is *sourced* to the motor outs if the *gate* is actively *HIGH*.

In Image 4, the MOSFET gates are coupled with Logic *AND* gates where each *AND* gate controls the MOSFET as an electronic switch.

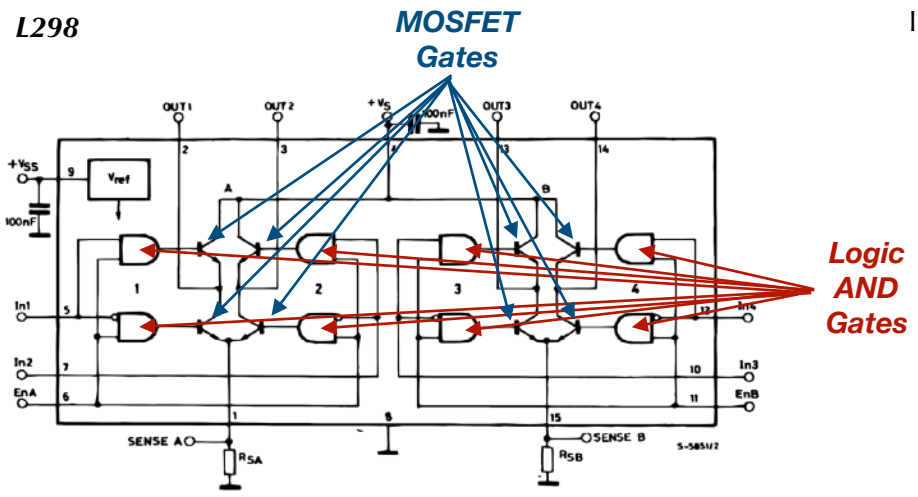


Image 4

The logic behind this circuit allows the four MOSFET at any one time to be in one of two states. There are other states but only two states are considered useful. Table 1 shows the two useful states. From Table 1, State 1 shows that MOSFET 1 is actively ON which in turn shown in Image 2 directs the positive power supply to motor out 1 and at the same time, it shows that MOSFET 4 is actively ON which in turn directs the negative power supply to motor out 2. State 2 shows the reverse of state 1. Any other state besides Table 1 would cause an electrical short.

State	Image-2	MOSFET 1	MOSFET 2	MOSFET 3	MOSFET 4
1	**Red**	ON	OFF	OFF	ON
2	**Blue**	OFF	ON	ON	OFF

Table 1

The blues and reds of the zeros (0) and ones (1) in Image 2 represent the logic voltages applied to the MOSFET gates where (0) represents LOW and (1) represents HIGH. From Table 1, State 1 represents the red colors of the H-Bridge in Image 2 where the square wave at MOSFET 1 represents the PWM (analog like) input from 0 to 5 volts. State 2 represents the blue colors. The square wave next to the MOSFET gates is called the pulse width modulation as previously discussed where it too has the same two states, *HIGH* and *LOW* that varies (switches) with time.

There are different names applied to logic voltages which themselves are either in two states, either zero (0) volts or like in most devices, (5) volts. When using words like *ON*, *HIGH* and even the number *1,* they apply to the maximum logic voltage for example, the logic voltage for Arduino Uno at 5 volts. Of course, words like *OFF*, *LOW* and even the number *0,* they apply to the zero (0) volts.

An important question to consider, how can the MOSFET's gates be controlled in order to obtain the two possible states.

L298N Dual H-bridge Circuit

A single H-bridge is built with four solid-state switches (MOSFET). The L298N has a dual H-bridge circuit controlled by 8-AND logic gates (with 4 Input-Inverters). The dual-channel H-Bridge motor driver is capable of driving a pair of DC motors. The circuit of Image 4 was taken from ST-Microelectronics L298 dual full-bridge driver datasheet and it will be discussed further in more detail.

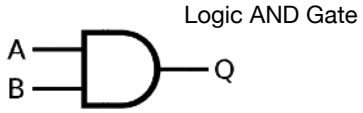


Image 5

The symbols in Image 4 are called logic *AND* gates as discussed where each logic gate is connected (coupled) to each gate of the MOSFET. The gate part of the MOSFET indicated by the letter *G* is shown in Image 3. The symbol for the logic gate shown in Image 5 represents an *AND* gate which has *two inputs* and *one output*.

The connections *A* and *B* are the inputs and *Q* is the output. Logic voltages are applied to each input that must be one of the two states which represent a switching circuit for the output. The inputs of the switching circuit can be set to either *LOW (0)* or *HIGH (1)*. They're not necessarily static in some cases but can change over time and can change states by using a digital signal like the PWM. If the PWM duty cycle to the MOSFET is set to 25 percent, then 25 percent for example from a 12 volt battery supply is 3 volts which is power supplied to a motor. Interesting question is how exactly does the PWM adjust the voltage to the L298N or is it the L298N converting the signal to a voltage.

Truth Table for the *AND* Gate

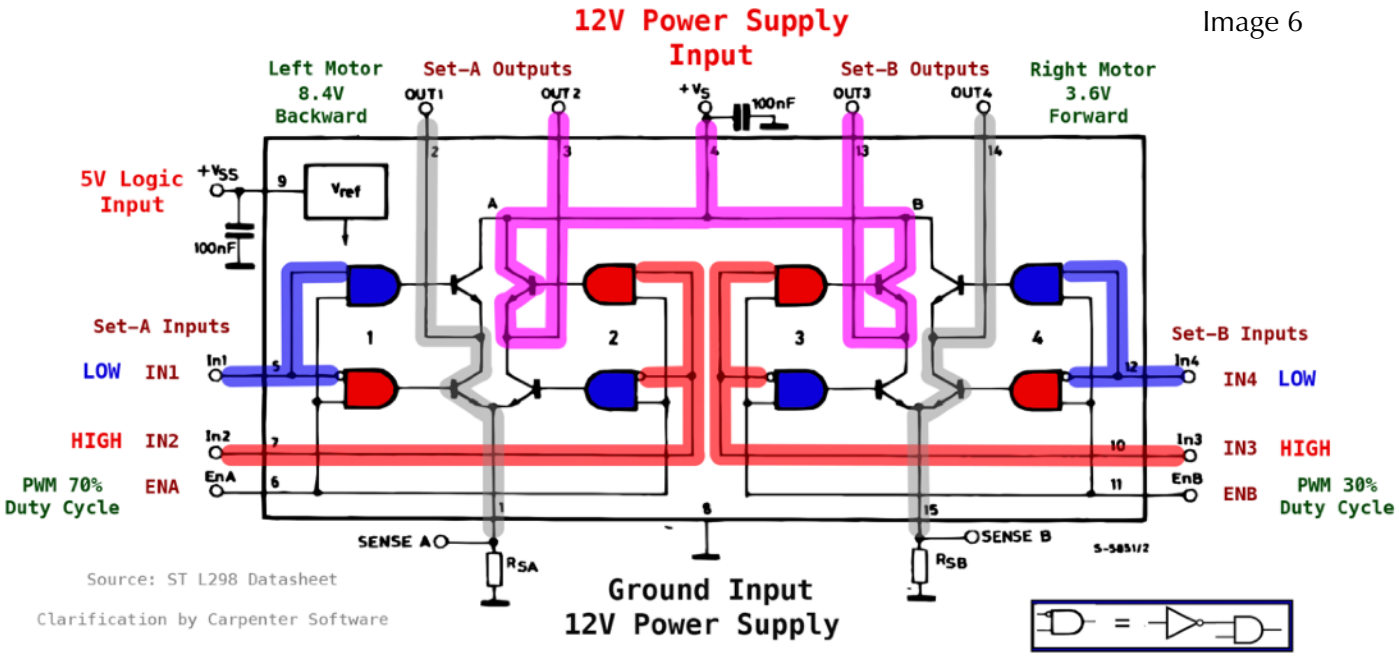
Table 2

AND Gate		
Inputs		**Output**
A	**B**	**Q**
0	0	0
0	1	0
1	0	0
1	1	1

The term *logic* can be represented by using *boolean* truth tables to describe the output results. In Table 2, when both inputs *A* & *B* are set to *HIGH (1)*, then the output Q becomes *HIGH (1)* otherwise the output is set to *LOW (0)*. The truth table shows all possible *A* and *B* input states as shown in Table 2. The answer is then the PWM activates the *AND* gate at 25 percent of the time.

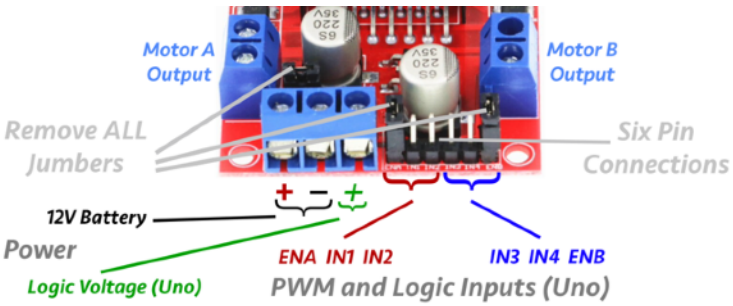
Image 6 may be daunting at first, yet the L298N circuit can be approached by divide and conquer. The inputs for the L298N shown in Image 6 as well as in Image 7 are *EnA, In1*, and *In2* which are used for the Motor ***A*** Output and the inputs *In3*, *In4*, and *EnB* which are used for Motor ***B*** Output. Notice these 6 inputs connected to the *AND* gates in Image 6.

Image 6



The 4 sets of *AND* gates cleverly designed are indicated by the numbered sets 1 through 4 not to be confused with Image 2. The left side, *EnA, In1*, and *In2* inputs will be discussed. When comparing Images 4 and 6, input *IN1* and input *ENA* are connected to two *AND* gates in set 1 but notice that one of the *AND* gates has a *circle*.

Image 7



Connections to the **L298N Motor Driver Module**

The circle represents an *inverted* input called the *complement* but here it is attached to only one of the inputs of each set of *AND* gates. In Image 9, the symbol represents the complement gate also called the inverted gate. Notice that the only inputs to set 1 are the *ENA* input and the *IN1* input where *IN2* is connected to set 2 which can be ignored for now. When *IN1* input is *LOW* indicated by the color blue its corresponding *AND* gate is also blue which indicates that its output is *LOW*.

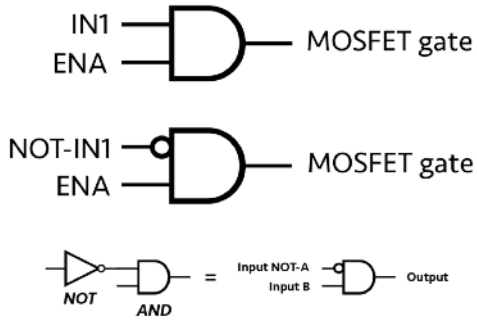


Image 8

The other logic gate is also connected by the *IN1* input but it is red because the circled input inverted the *AND* gate output to HIGH. The *ENA* input on the other hand carries the PWM signal which can be considered *HIGH* at some times. Whatever the *IN1* state might be, its counterpart will always have an opposite output.

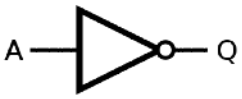


Image 9

Complement or Inverted Gate

COMPLEMENT Gate	
Input	**Output**
A	**Q**
0	1
1	0

Table 3

With that said referring back to Image 6 using Image 11, a larger view, follow the inputs of the supply voltage to the outputs for the motor outs. The gray shade shows how the current for the negative connection has bypassed the MOSFET when its gate is set to *LOW*. The same is true for the pink shade which shows how the current for the positive connection bypasses the MOSFET when its gate is set to LOW. What is important here, when comparing the two sets, 1 and 2, of the AND gates connected by the inputs *IN1* and *IN2* , the sets are also opposite which confirms state 2 of Table 1.

The truth table, Table 4, for the L298N *AND* gates sets 1 and 2 inputs: *ENA*, *IN1*, and *IN2*, shows that state 6 matches the inputs and outputs of Image 11 which also agrees with state 2 of Table 1. Table 4 also shows the output polarity of the supply voltage when the PWM is greater than zero (0). When the PWM is zero, the motor outputs are switched to off no matter what state configuration the digital inputs are set. When the PWM is active, only states 6 and 7 are useful which correlates back to Table 1.

Table 4

Truth Table: L298N Set-A Inputs: ENA IN1 IN2

	PWM	Input Gates				Output Polarity		
State	ENA	IN1	oIN1[2]	IN2	oIN2[2]	OUT 1	OUT 2	Motor
1	0	0	1	0	1	OFF	OFF	Stop[1]
2[a]	0	0	1	1	0	OFF	OFF	Stop[1]
3[b]	0	1	0	0	1	OFF	OFF	Stop[1]
4	0	1	0	1	0	OFF	OFF	Stop[1]
5	1	0	1	0	1	-	-	?[3]
6[a]	1	0	1	1	0	-	+	Backward
7[b]	1	1	0	0	1	+	-	Forward
8	1	1	0	1	0	+	+	?[3]

Key

Logic ***HIGH*** (1) - For the Arduino Uno, this is about 5V.

Logic ***LOW*** (0) - Zero (0) volts.

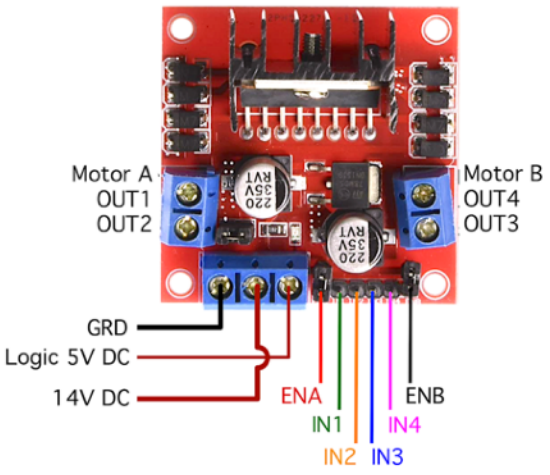
1. As the PWM signal for the ENA approached zero (0), the robot comes to a stop based on the *AND* truth table.

2. oIN1 and oIN2 are the inverted AND gates where the Inputs A are inverted as Input Not-A. See Image 8.

3. The questionable motor outputs might be designed so that the PWM does not cause an electrical short. Otherwise, this logic configuration is useless.

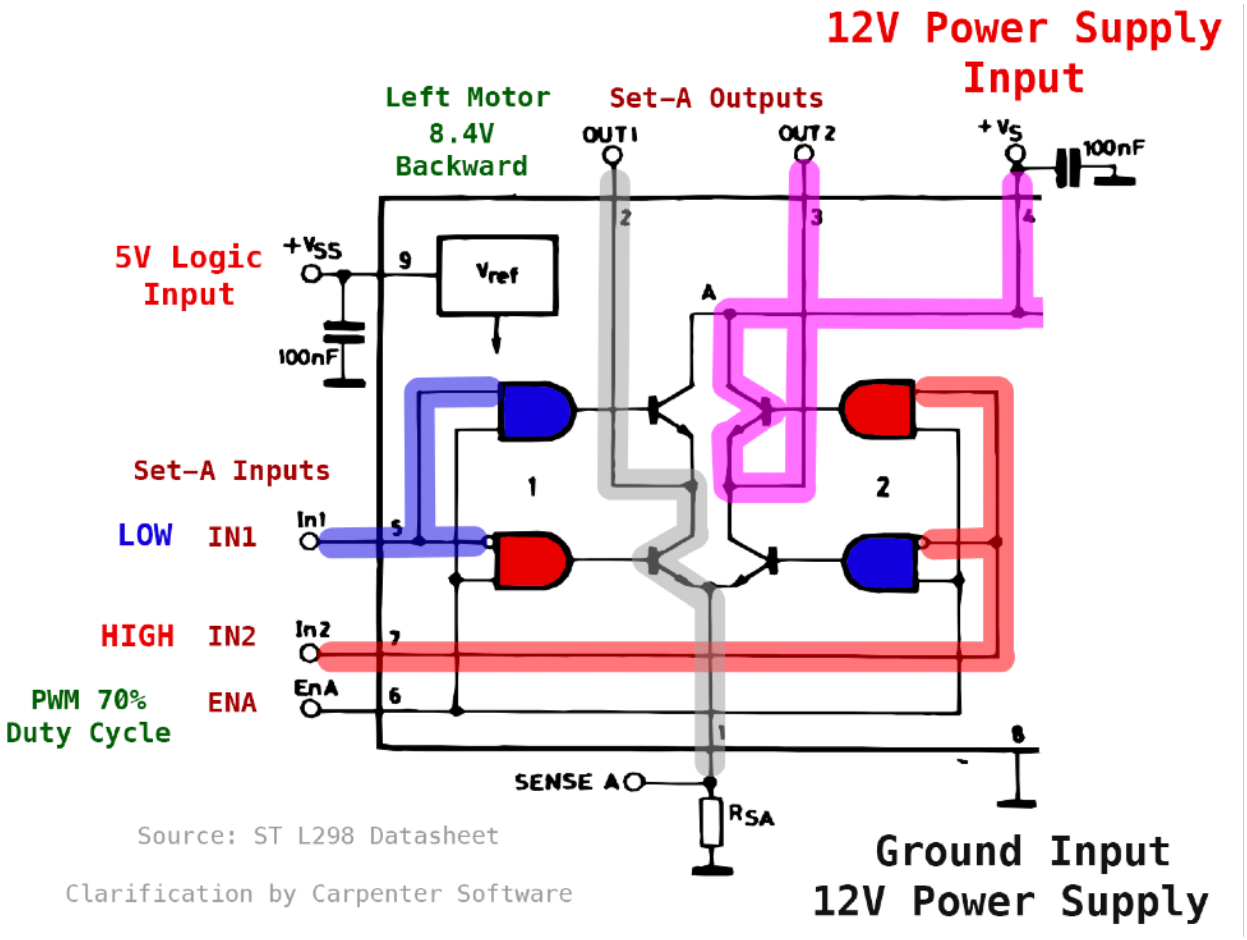
Note: How is the PWM affecting the inverted input to the AND gate when compared in the input state of IN1? The PWM, ENA in this case, is connected simultaneously to all four AND gates by IN1 and IN2. While the HIGH and LOW is switching back and forth in the circuit, all four MOSFETs are switching also at the same time.Therefore the PWM for example let's say state 6 is switching to state 2 and back again so PWM has no affect except the output voltage is adjusted by the duty cycle. Personally I cannot imagine that the circuit was designed where PWM is carried through to the motor outputs. Either way this could be analyzed.

Image 10




```
          Output                Input
-------------------------------------
Set A:    Motor A          ENA, IN1, IN2
Set B:    Motor B          ENB, IN3, IN4
```

Image 11



Upon closer scrutiny of Image 6, the motor moves backward because the output polarity is reversed from that of the forward polarity. The motor is using 8.4 volts from a 12 volt battery because the PWM is at 70 percent Duty Cycle. The recommended method to stop the motors is to set the PWM to *LOW* (0).

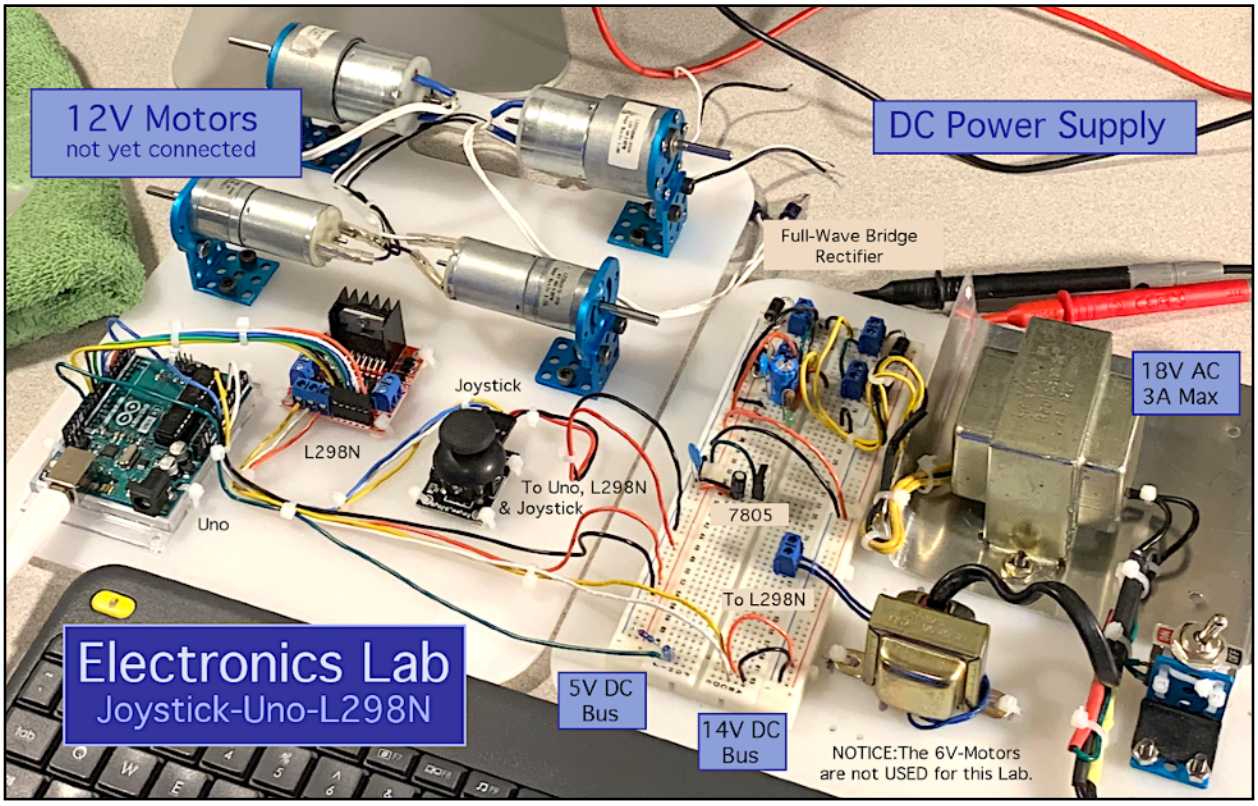
The connections to the L298N shows the Power and the Inputs and Outputs in Image 10. The wiring diagram in Image 12 shows the complete testing circuit with image 13 showing the complete Electronics Lab for the Joystick, Uno and the L298N.

Electronics Lab

In an electronics laboratory, whether at home, at school or at professional lab, safety comes first. Image 12 shows a desktop electronics lab boards. One is the power supply and the other is the study board for the joystick, Uno and the L298N. The board also has 6 volt and 12 volt motors to test while programming the MCU. Study each of the electronic components before assembling the lab boards. Use a Lab Notebook to maintain information needed for each component. With the Lab Notebook, the experiments may finally illustrate a circuit affirmation.

The electronic schematic is shown in Image 13. It was designed knowing how each electrical component operated and how to connect the components. Careful lab notes were taken along the way and afterwards, reviewing the lab notes bore new ideas for the next design. Who knows what the new design will bring.

Image 12



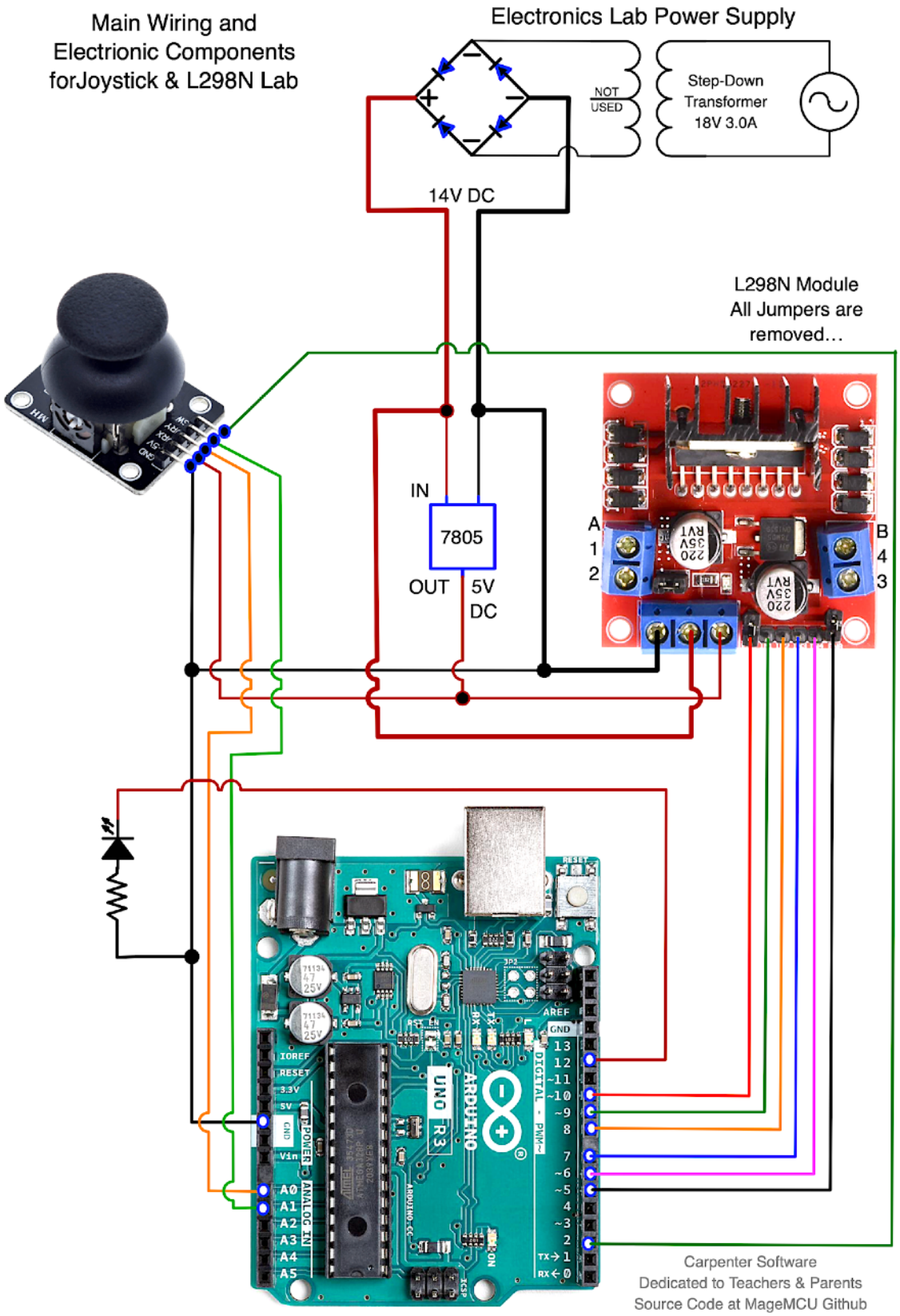
When one compares the lab work with the electronic schematic there should not be any *surprises*. Always double check the wiring connections for correctness.

The real fun begins when programming these devices together. The observations and the behavior as one programs the microcontroller are all recorded in the lab notebook.

Once the main circuit for the overall schematic has been confirmed, it's time to build the robot.

These three main components, the joystick, the microcontroller, Uno, and the L298N motor driver, are enough to get stated in robotics at home. This project could take an entire year to study and to build a robot. Enjoy and have fun.

Image 13


Main Wiring and
Electrionic Components
forJoystick & L298N Lab
Electronics Lab Power Supply
NOT
USED
Step-Down
Transformer
18V 3.0A
14V DC
L298N Module
All Jumpers are
removed…
IN
7805
OUT
5V
DC
A
1
2
B
4
3
Carpenter Software
Dedicated to Teachers & Parents
Source Code at MageMCU Github

DISCLAIMER (Carpenter Software & Github)

Any Information, including the Electronic Schematics and the Software (AI_ES_S) is provided as is, without any representation or warranty of any kind, either express or implied, including without limitation any representations or endorsements regarding the use of, the results of, or performance of the AI_ES_S, its appropriateness, accuracy, reliability, or correctness.

The entire risk as to the use of AI_ES_S is assumed by Licensee (MIT).

Carpenter Software LC (AUTHOR) does not assume liability for the use of this AI_ES_S. In no event will the AUTHOR be liable for additional direct or indirect damages including any lost profits, lost savings, or other incidental or consequential damages arising from any mishap, or the use or inability to use these AI_ES_S, even if AUTHOR has been advised of the possibility of such damages.

The MageMCU GitHub account is a living workspace used by the AUTHOR for the AUTHOR and if by consequence the AI_ES_S is used by a Licensee then the Licensee is solely responsible.

Parents and Teachers

Parents and Teachers: This Github Account is restricted to minors. When working with electricity of any kind, they should be directed and instructed by a responsible mentor. A USER (Licensee) must be at least 13 years of age when using GitHub.

Minors (Children) and Teenagers

All children and teenagers, please consult with your parent (your guardian, your mentor or your teacher) before experimenting with any of these electronic circuits. Never work with electricity alone. Electricity Kills.

The code may be download by MageMCU at GitHub. The manifest chosen contain some of the most inexpensive items. The circuit has been tested over several years. Anyone who wants to volunteer to teach, please educate our children for the world to come…

–Jesse Carpenter

Carpenter Software
revision 20211202 edited…
revision 20211108 edited…
revision 20210906 Jesse Carpenter