H bridge basics

Fabien Le Mentec

texane@gmail.com

Abstract

This document describes the basics of H bridges. The primary goal is to learn more about electronics.

1 Introduction

1.1 Purpose

I wanted a small project to learn more about electronics basics. Since I regularly need to drive a DC motors, I chose to work on a home made H bridge board described in this document. Note that the circuit uses off the shelf parts and can be more complicated than needed and not efficient. A real application should use packaged H bridge circuits.

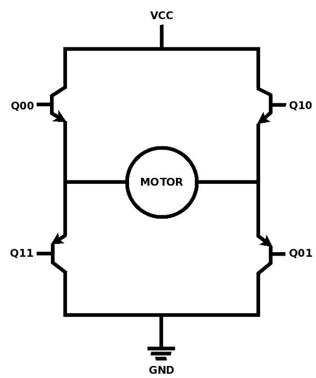
2 Features

The board features:

- 3 wires, safe H signaling interface, avoiding invalid transistor states
  - PWM,
  - FORWARD,
  - BRAKE.
- controlling software
  - coding done by XXX
- power stage driving up to XXX motors.

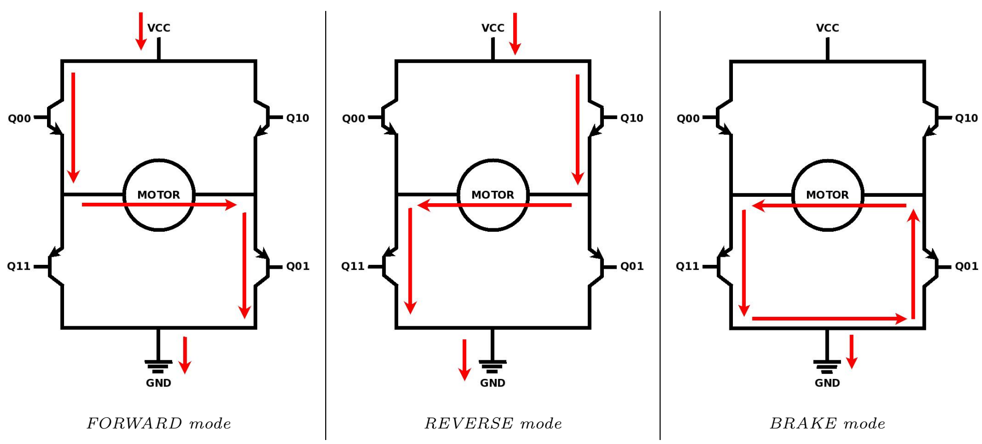
3 H bridge theory of operation

An H bridge is a circuit allowing to drive DC motors in both direction. The name comes from the typical circuit graphical representation which looks like the 'H' alphabet letter.



H bridge circuit

QXX are transistors allowing the current flowing through the DC motor to be electrically controlled. I am interested in 3 configurations:



4 H signaling interface

The H bridge expects a correct transistor configuration to work. Even worth, an incorrect setup can damage the circuit and motor. I designed a 3 wire signaling interface, called the H signaling interface, to address this issue:

Name	*Description*
FWD	controls the motor direction
PWM	controls the motor speed
BRAKE	slow the motor down until it stops

H control signals

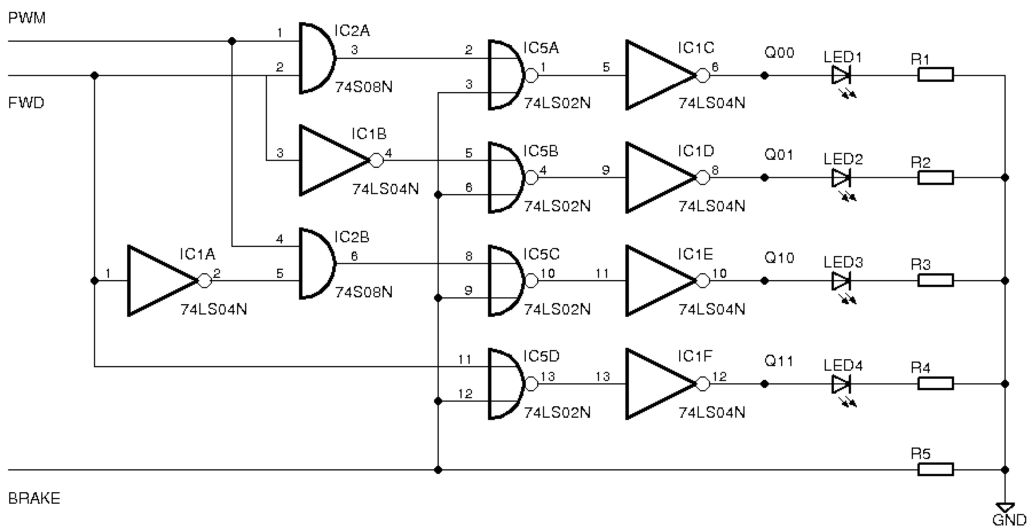
Since I have a software background it is easier for me to think in terms of C programming control structures. I thus express the H state machine in a C code which I then translate into logical statements. I use the resulting statement set to design the H state circuit using electronic gates.

```
if (BRAKE == 0)
{
  if (FWD == 1)
  {
    Q01 = 0;
    Q10 = 0;
    Q11 = 1;

    Q00 = PWM;
  }
  else /* REVERSE */
  {
    Q00 = 0;
    Q01 = 1;
    Q11 = 0;

    Q10 = PWM;
  }
}
else /* (BRAKE == 1) */
{
  Q00 = 1;
  Q01 = 1;
  Q10 = 1;
  Q11 = 1;
}
```

The above code reduces to the following set of logical statments:

```
Q00 = (FWD & PWM)      | BRAKE;
Q01 = (!FWD)           | BRAKE;
Q10 = ((!FWD) & PWM)   | BRAKE;
Q11 = FWD              | BRAKE;
```

These 4 statements are implemented by the following circuit:



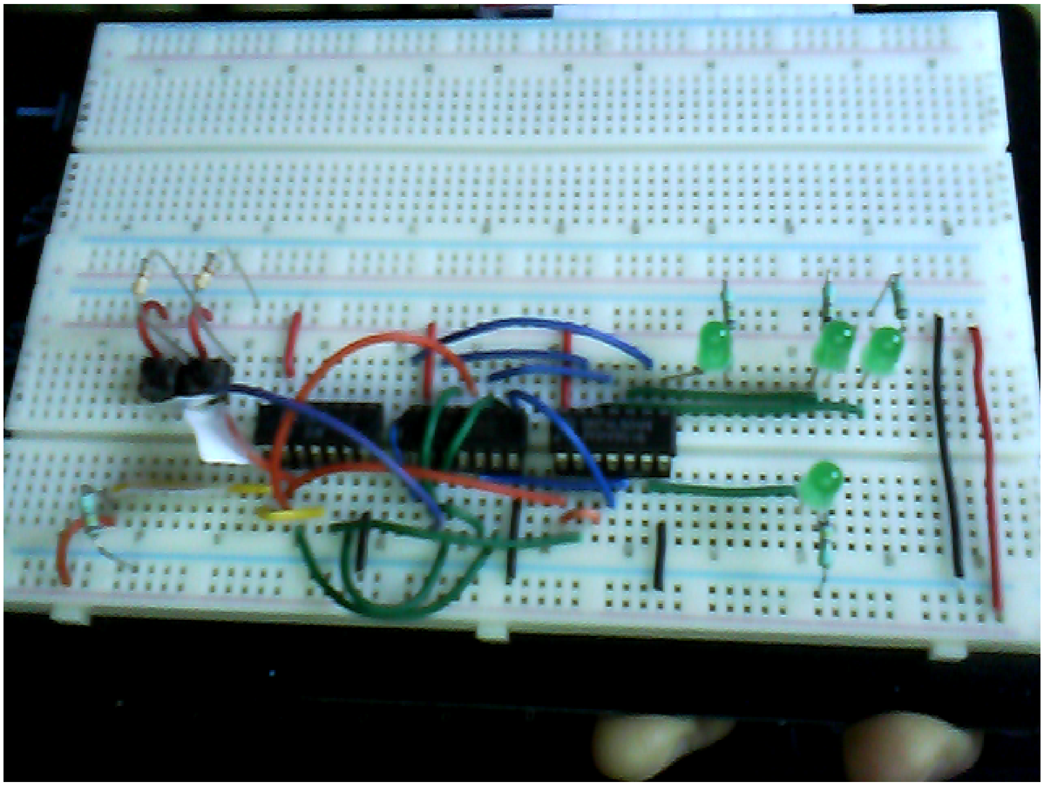
H control circuit

Note that I do not have OR gates, so I used NOR gates and inverters. It complexifies the circuit a bit. Apart from that, the logical statements are naively implemented using electronic gates.

Reference	*Quantity*	*Description*
SN74LS02N	1	quad 2 input NOR gates
SN74LS04N	1	hex inverter
SN74LS08N	1	quad 2 input AND gates
Resistors	5	2.4k ohms
LEDS	4	

part list

Picture of the prototyped circuit, with switchs added for testing:



H control circuit prototype

5 Power stage

TODO

6 PWM generation

IN PROGRESS

6.1 PWM basics

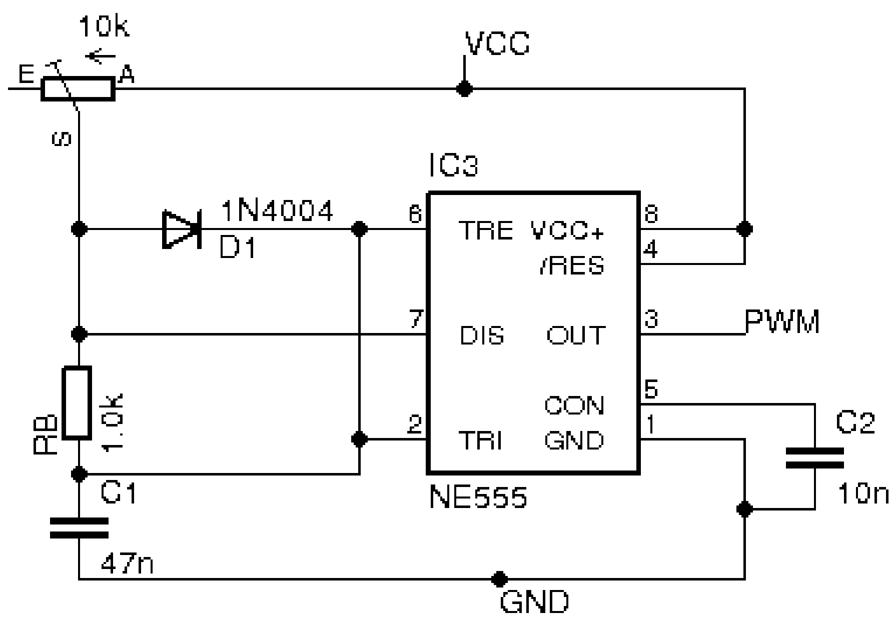
The motor speed is controlled using a PWM signal.
TODO: explain duty cycle, averaged total current as a function of time

6.2 PWM using transistors

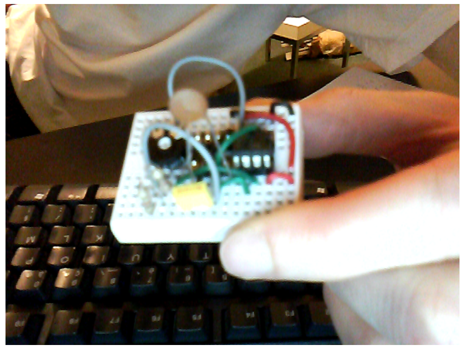
TODO

6.3 PWM using a NE555

I use a NE555 chip in an astable setup to generate a PWM signal. The Philips 555 application note has been used as a reference for this section. The circuit is shown below:



NE555 astable setup



NE555 astable circuit

2 things to note:

- a pot is use to control the duty cycle by varying the voltage divider ratio,
- a diode between pins 6 and 7 has been added to allow a duty of less than 50%.

The documentation gives the following formula:

$$\boxed{Frequency = \frac{1.49}{((Ra+2*Rb)*C1)}}$$

In the above circuit, the following values are used:

- Ra = 0 to 10k,
- Rb = 1k,
- C1 = 47n.

By substitution, we have:

$$\boxed{\frac{1.49}{(2*10^3+10*10^4)*47*10^{-9}} <= frequency <= \frac{1.49}{2*10^3*47*10^{-9}}}$$

Thus:

$$\boxed{2642 <= frequency <= 15851}$$

- TODO: old result from a different configuration
- TODO: picture of the oscilloscope screen

The frequencies can be checked using an oscilloscope:

$$\boxed{2272 <= frequency <= 10000}$$

6.4 PWM using an AVR microcontroller

TODO

7 H controlling software

TODO

8 Status

- PWM / FORWARD / BRAKE signaling interface: PROTOTYPED
- control software: TODO
- power stage: TODO
- documentation: STARTED

9 Conclusion

TODO

10 Further readings

10.1 H bridge projects

- http://embedded-lab.com/blog/?p=1159
- http://www.mcmanis.com/chuck/robotics/tutorial/h-bridge
- http://www.modularcircuits.com/h-bridge_secrets1.htm
- http://www.solarbotics.net/library/circuits/driver_4varHbridge.html
- http://www.solarbotics.net/library/circuits/driver_buf_h.html
- http://www.robotroom.com/HBridge.html
- http://www.robotroom.com/BipolarHBridge.html
- http://www.societyofrobots.com/schematics_h-bridgedes.shtml

10.2 Controlling software

- http://www.seattlerobotics.org/encoder/200001/simplemotor.htm

11 TODO

- PWM using ne555
  - oscilloscope pictures
  - wider duty cycle range
  - how to compute the duty range
  - more about the 555
- Atmel AVR or ne555 or RC network + variable resistor to control motor speed
- controlling software