# Motor Control

R. Shek

## **Introduction**

We require our rover to be able to move freely and be able to find its way through the obstacle course, to do this we used two DC motors controlled by the Arduino, using an integrated H bridge circuit.

## **What is a DC motor and how is it implemented in this project?**

A picture containing text, indoor

Description automatically generatedA DC motor is made up of three main components [1], the stator, the rotor and the commutator and brushes. The stator is comprised of the two permanent magnets, the stator provides the magnetic field necessary to rotate the rotor, it. Next, the rotor, also known as the armature, the rotor is comprised of coils or discs of copper, ideally insulated as to prevent the formation of large eddy currents, these coils/discs behave as electromagnets when a current is passed through them, with the presence of the permanent magnets in the stator, the polarity induces a spinning motion. The final component is the commutator and brushes, the commutator is essentially a ring of copper which is split at equal intervals, when a current is sent through the opposing brush contacts the commutator sends a current through one of the coils, generating an electromagnet; this will, because of opposing polarities, cause both the armature and commutator to spin, this process repeats as the brush contacts cycle through the different sections of the commutator; to finish this off, a shaft is added and attached to the commutator and armature sections of the motor, the shaft provides a way for the mechanical energy produces by the motor to interact with the outside, in our case to provide mechanical energy to the wheels of our rover.

Fig. 1 Internal diagram of a DC motor.[2]

Diagram, schematic

Description automatically generated**How does the H-Bridge work?**

The H-bridge is the piece of the circuitry which allows the Arduino to communicate with the motors [3]. The H-Bridge is an integrated circuit consisting of 4 transistors arranged in a H pattern with the load motor placed in the cross bar. The priciple of how the H-bridge operates is through the use of transistors as switches, when a signal is sent through transistor A and transistor D, the current will flow through one end of the motor to the other, and due to the structure of the DC motor, the motor will start to spin in a certain direction. If instead, we turned on transistor B and C we would expect the current to travel through the opposite ends of the motor and as a result the motor will spin in the opposite direction. The structure of the H-bridge consists of two P-channel mosfets and two N-channel mosfets at the top and bottom, once we connect the H-bridge IC we are suppliying a 5V as Vm and through the use of the voltage outputs from the microcontroller we can send pulse width modulated (PWM) signals to the transistors which will decide how much current will drive the motors.

Fig. 2 Circuit Diagram of H-Bridge [5]

Diagram

Description automatically generated The PWM signal is the key factor which determines the speed of the motor [4]. If we wanted to change the speed of the motor then we would need to be able to tune the voltage driving the transistors, this should be impossible with a traditional digital output, as we know digital signals are either high or low; however by switching the signal on or off at very rapid signals and by changing the duty cycle of the square wave you can change the voltage from the minimum of 0V to the maximum voltage, for our metro M0 board this is 3.3V.

Fig. 3 Diagram of different PWM signals [4].

**Implementation in code**

Now that we have the logic down, we need to begin implementing this as code in the Arduino IDE. To begin we connected the Arduino board to the H-bridge, we did this by connecting 4 digital pins to the H-bridge module, this allows the module to be able to control the logic for the two wheels by sending two direction and enable signals to each wheel. In the IDE we began by setting up the pins into output mode so that the Arduino is able to send the four digital signals as output, then we chose to define some variables, namely a constant variable which sets the time interval in which we time each action performed by the rover, a variable to set the start time and one for the end time. To structure the code for the movement of the rover we sectioned the code and defined a function for each direction, forwards, backwards, left, and right. Each function looked something like figure 4.

Text

Description automatically generated

Fig. 4 Function for forward movement

The first thing we did was set the direction of both wheels, for our case its by sending a signal to the corresponding pins to the h-bridge module i.e., the DIR pins for the left and right motors. Then the crucial step was to be able to time the motors correctly, to do that we made use of the millis() function, this is an inbuilt timer inside the Arduino board which counts the time the program has been running, to be able to use the millis() function we have built a while function which runs for the time interval set by the start and end time, once the program enters this while loop it will set the speed of the wheels, setting the speed is done by using the analogWrite() function with the desired pin and an integer value ranging from 0-255 for the input parameters, this function is what sends the enable pwm signal; in our testing we discovered that a value of 127 as the input parameter was sufficient to reach a decent distance during a 3000 millisecond time interval for both the forward and reverse movements, when turning a value of 77 was used. Once the while loop has been surpassed by the program, we update the value of the start and end time, so the next function is able to run within the correct time interval.

**Issues and Refinements**

While developing a solution for the motor control we encountered a few complications. To begin with, when writing the code we kept running into an error when using the delay() function inside the Arduino, this presented an error as the delay function temporarily stops the Arduino from running until all the time has been elapsed, and since we are also using running a webserver from the Arduino this would present a huge complication since the delay function would also stop the webserver from running. Another issue we encountered was when we were building the while loop, the issue with Arduino sketches that differs them from regular C++ code is their setup loop structure, once the initial parameters are setup the sketch enters a part of the code which cycles through itself continuously. The issue with loops is that the program can enter an infinite loop where the program is essentially “stuck” in a segment of the code. To build a robust while loop which exits at the correct time was to update its parameters once the loop reaches its end so that the loop has updated parameters to work with, otherwise the loop will continue to work with its first instance. When tuning the motors we also found some problems, when setting the speed the motors should operate at we found that setting the motor to a low speed with a value less than 64 the motor would just not spin, this was due to the average voltage generated by the pwm signal being too low which would just not be enough to switch the transistors within the H-bridge module on and as such the current was not able to reach the motor.

## **References:**

1. IQS Articles. DC motors [Internet].IQS Directory; n.d. [cited 2022 June 07]. Available from: <https://www.iqsdirectory.com/articles/electric-motor/dc-motors.html>.
2. How does an electric motor work? (DC motors) [YouTube]. Jared Owens; n.d. [cited 2022 June 07]. Available from: <https://www.youtube.com/watch?v=CWulQ1ZSE3c>.
3. Build electronic circuits. What is a H-bridge [Internet]. Build electronic circuits; n.d. [cited 2022 June 08]. Available from: https://www.build-electronic-circuits.com/h-bridge/.
4. Arduino Docs. Basics of PWM [Internet]. Arduino Docs; n.d. [cited 2022 June 08]. Available from: <https://docs.arduino.cc/learn/microcontrollers/analog-output>.
5. EEERover technical guide. [Internet]. EEERover technical guide; n.d. [cited 2022 June 08]. Available from: https://github.com/edstott/EEERover/blob/main/doc/README.md.