|  |  |
| --- | --- |
|  |  |
|  |  |

|  |
| --- |
| **Motor controller guide** |
|  |

Made for students, by students

Table of Contents

[1 Motor Control 3](#_Toc371512136)

[1.1 Motor Control System Overview 3](#_Toc371512137)

[1.2 Command Signals and Feedback 4](#_Toc371512138)

[1.3 Pulse Width Modulation (PWM) 5](#_Toc371512139)

[1.3.1 Sign Magnitude PWM Control 6](#_Toc371512140)

[1.3.2 Locked Anti-Phase PWM Control 7](#_Toc371512141)

[1.4 H-Bridge Motor Driver 8](#_Toc371512142)

[1.4.1 General Information 8](#_Toc371512143)

[1.4.2 Free Run Mode and Brake Mode 9](#_Toc371512144)

[1.4.3 Selecting H-Bridge Motor Driver Boards 11](#_Toc371512145)

[1.5 Tests Performed 14](#_Toc371512146)

[1.5.1 Free Running Tests 14](#_Toc371512147)

[1.5.2 Fast Switching Tests 15](#_Toc371512148)

[1.5.3 Pulse Width Modulation Duty Cycle 15](#_Toc371512149)

[1.5.4 Temperature Monitoring 15](#_Toc371512150)

[1.6 Testing Conclusions 15](#_Toc371512151)

[1.7 Software Abstraction 16](#_Toc371512152)

[1.7.1 Forward/Reverse Motion Control 16](#_Toc371512153)

[1.7.2 Steering Control for the DuneBot variation 17](#_Toc371512154)

[1.8 Things to discuss with teammates 18](#_Toc371512155)

# Motor Control

## Motor Control System Overview

In general, the purposes of a motor control system are to take a constant voltage input and to translate desired high-level motor commands. An example would be taking the command, “spin the motor at 50% of its maximum speed in the forward direction,” and translating it to the electrical commands the motor requires at the low-level to perform that operation. Many of the most commonly used motors require a change in the supplied voltage on one or more inputs in order to change their speed and spin direction, as in the case of a DC motor, or to change the rotational position of their axle in the case of a servo motor. For example, in order for many motors to spin at half of their speed, they require an input of 50% of their maximum operational voltage. This can be easily done at a lab bench, however in actual use, many devices' power supply systems are designed to provide only constant voltage outputs.[[1]](#footnote-1) The motor control system satisfies this need for variable motor operations and bridges the gap between the motor speed control code and the motor inputs.

Some more sophisticated motor control systems can also be implemented to apply low level feedback control. They measure the difference between what the motor was commanded to do and what the motor actually did, and then adjust the command signal in order to better match the desired output. For example, if the motor is commanded to go at 50% speed but only went at 49% speed, slightly more voltage is applied in the next command and the speed is re-measured to see if the motor spins at 50% or if more command adjustments are needed. To do this, motor control systems often include some kind of sensor, such as an encoder, that measures how much the motor has rotated. When this information is analyzed over a given period of time, the rotational speed of the motor can be calculated. The motor controller can then compare that measured speed with the desired command speed and then use the difference of these two speeds (often referred to as the error) to increase or decrease the voltage applied to better match the command speed.

Motor controller systems with feedback control are also frequently used in servo motors. The feedback portion is often pre-packed directly within the servo motor as well. This is particularly important in servo motors since they are designed to hold their axle at a specified rotational position, e.g. holding a robotic arm in a particular position, instead of defining the speed at which the motor is spinning.

Luckily, the basic concepts of a motor control system are the same for both a servo motor and a common “spinning” motor, such as a DC brushed motor. Hence, another benefit of a motor control system is that it makes your device more modular and motors can be replaced with minimal change to the high-level code. It is much easier to code a command to move a robotic arm to a specific location to then perform a task than it is to program each specific motor rotation, power level, and movement time. Overall, the motor control system acts as the interface between the main central processing unit (CPU) and the motors themselves, thus as will be explained, motor control systems consist of both software and hardware.

Depending on the type of motor, e.g. DC motors vs. servo motors, the actual motor command signals are different. The primary goals of DC motor control are to control the direction and speed, whereas for servo motors it is to control the angle, or position, of the motor. Both types of motors are discussed in this guide. Since motor control systems are fairly similar for many kinds of motors, this guide will mainly focus on voltage controlled DC brushed motors, which are one of the most common kinds of “spinning” motors. If you use a brushless motor, it will most likely require a special motor controller be used with it. Often, you need to buy the brushless motor controller that the motor manufacturer recommends. Brushed motors allow for more mix and matching of motors and motor controllers.

The following list contains some important features to consider when designing a motor control system. More details on each item are given throughout this guide.

1. Separate power supplies for motors and processors
2. Additional driving circuitry to convert control signals into power signals, e.g. H-bridge
3. Power required by the motors
4. Availability and ease-of-use of the control board output pins
5. Safety circuitry to prevent voltage spikes, motor damage, and avoid high thermal conditions
6. Protection diodes to protect against motor voltage spikes
7. Minimum noise level, both electrical and audible [[2]](#footnote-2)

The first part of this guide discusses local feedback control loop for each wheel/motor combination to handle the variability in the system while ensuring high performance. The second part describes control using PWM for both DC and servo motors. The third section describes the theoretical operation of the motors, which utilizes H-bridges as a robust means of specifying the spin direction speed of the DC motors.

## Command Signals and Feedback

As mentioned in the overview, it is not uncommon for motor control systems to incorporate some kind of feedback control in order to help ensure that the high end commands are being executed as desired. Regardless of whether the feedback is internal to the motor (as is the case with servo motors) or handled directly by the device designer, a motor will often have a performance sensor in order to include a feedback component to the motor controller system. The feedback control is rarely done by measuring the output voltage but rather by position and/or speed measurements because they are a direct measure of the desired effect. With spinning motors, a tachometer can be used for velocity measurements, and a motor shaft encoder can be used for both position measurements and velocity calculations. These measurements are sent back to either the CPU or the motor control system’s microcontroller in order to generate a feedback command to increase performance as shown in the following figure.

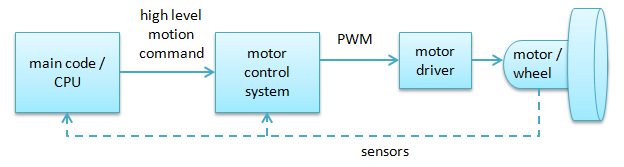


Figure 1: Basic diagram of motor control system command signal flow

The motor control system of the ModBot begins with the Intel Atom processor as the main CPU. The CPU calculates the desired speed values for each locomotion motor and sends these to a “helper board” that can generate the PWM signals. The ModBot system uses the Arduino Mega as the primary helper microcontroller board for motor control, and it receives the command signals from the Intel Atom via a serial connection.[[3]](#footnote-3) The Arduinos are well-suited for controlling individual motors as they can utilize pre-existing code libraries to allow for easy access to control individual output pins at the desired frequencies in order to generate the PWM signals. The Arduino PWM control signals are then sent to a motor driver circuit that boosts the power in order to turn the motors.

## Pulse Width Modulation (PWM)

One option to control the motor's operating speed is by using pulse width modulation. Pulse width modulation (PMW) is a common command signal for many kinds of devices, including most voltage controlled spinning motors and servo motors. PWM is a technique for motor control that uses a constant voltage input to the motor control system (such as a constant 5 V input from the power system’s 5 V rail), but PMW can vary the “perceived” voltage level supplied to the motor, hence allowing the motor to run at a variety of speeds. The term “perceived” is used because PWM works by very quickly turning “on” and “off” the voltage supply to the motor. This causes the motor to behave as it if were supplied a constant voltage equal to the average between how long the supplied voltage is “on” and how long it is “off”. By varying the percentage of time the voltage is actually “on,” the motor control system can control how fast the motor will spin.

For example, if the motor controller sends an input signal that is “on” half the time and is “off” the other half, the total signal coming out from the motor controller will look like the first square waveform in Figure 2. As a result, the motor will “see” an input voltage of 50% of the “on” constant voltage. Due to the high-low nature of the square wave, the part of the signal that is “on” is often referred to as sending a high signal, as that is the maximum or “highest” magnitude value that the overall signal can have. Likewise, the part of the signal that is “off” is referred to as a low signal. The ratio of the high and low signal can thereby change the motor’s perceived input voltage. If the high signal is sent 75% of each square wave period as shown in Figure 2, the motor will see an input voltage of 75% of the high signal, which is 75% of the motor control system’s constant voltage input. The percentage of each period where the signal is high is often referred to as the PWM’s duty cycle.

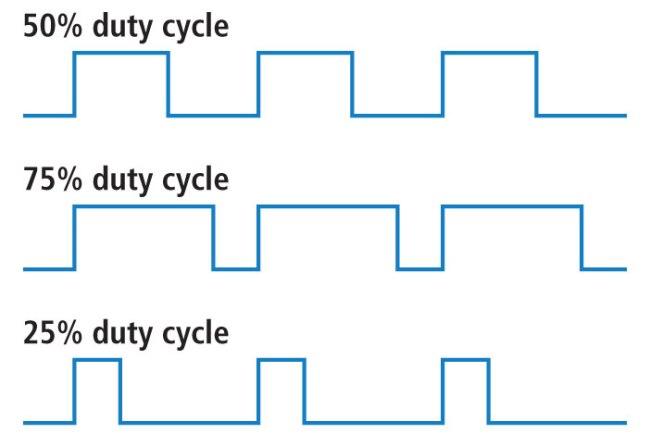


Figure 2: Various PWM Signals

The PWM technique works well for motors since they are mechanical in nature and their response times are much slower than most electronics. This means a series of rapidly changing high and low input voltages will make a motor behave just like an single constant voltage has been applied equal to the PWM’s duty cycle times the motor control system’s high voltage value.

(1)

In practice, these PWM signals are often generated on a microcontroller and then sent to a motor drivercircuit. The motor driver circuit is an important part of the motor control system that takes in both the constant voltage supply input and a second control signal used to set the output duty cycle. In many cases, the control signal is a PWM signal itself and the motor driver circuit simply amplifies this control signal. The control signal is usually at a lower voltage than the motor requires to be powered though, hence the voltage amplification. In the ModBot, the logic circuit works at a 5 V level, but the motors require a 24 V supply, so extra circuitry must be used to control a 24 V signal using the 5 V signal.

An H-bridge circuit is a commonly used motor driver circuit for brushed DC motors; it is described in more detail in the following sections but by itself it contains no feedback.[[4]](#footnote-4) Servo motors commonly include the motor driver circuit internally but still only require the same two inputs (a constant voltage supply and a command signal). More complicated motor control systems may still have at least the same two inputs but may perform additional functions such as feedback signals, surge protection, and other safety features.[[5]](#footnote-5) Two of the most common ways of controlling PWM signals are locked anti-phase PWM control and sign magnitude PWM control; they are detailed in the following two subsections.

### Sign Magnitude PWM Control

Instead of cycling between positive and negative voltages, sign magnitude PWM control cycles between either positive or negative voltage and the zero point, or no voltage. The voltage is either positive or negative based upon a single bit direction input, and the duty cycle is controlled by the PWM input. If the direction input is high, the motor will move forward at the speed dictated by the duty cycle on the PWM input. If the direction input is low, the motor will move backwards at the speed dictated by the duty cycle on the PWM input. If the duty cycle is 0%, the motor is stationary and the direction is irrelevant.

Controlling a brushed DC motor with a sign magnitude PWM control signal requires one PWM control input, one directional control input, and the voltage source. Sign magnitude PWM requires more pins than locked anti-phase PWM control but offers twice the precision control of the speed.

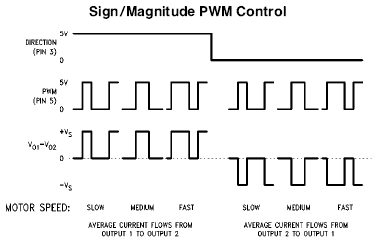


Figure 3: Sign/Magnitude PWM Control [[6]](#footnote-6)

### Locked Anti-Phase PWM Control

In locked anti-phase PWM control, the amount the voltage signal “swings” from a positive voltage to a negative voltage is used to control the motor. In this case, forward and reverse motion depend on the duty cycle of positive voltage versus negative voltage, with the 50% duty-cycle equaling a stationary motor. For example, if the voltage across the motor is positive for longer than it is negative, the motor will move forwards. The same is true for the opposite: if the voltage is negative for longer than it is positive, the motor will move backwards. Therefore, if the positive duty cycle is higher than 50%, the motor will move forwards; if it is lower than 50%, the motor will move backwards.

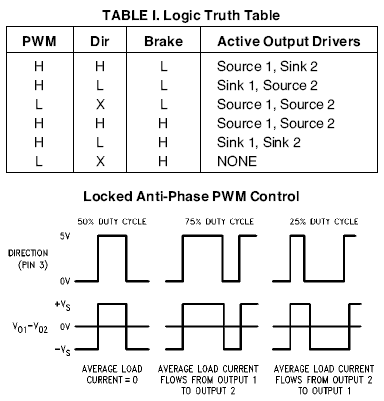
****

Figure 4: Locked Anti-Phase PWM Control [[7]](#footnote-7)

Controlling a brushed DC motor with a locked anti-phase PWM control signal only requires one control input and the voltage source. Requiring only one control input can be particularly valuable in embedded systems where the number of pins is limited and hence they are valuable. However, since the duty cycle is also being used to control both the speed and the direction, you have only half the preciseness in speed adjustments that you would using sign magnitude PWM control.

## H-Bridge Motor Driver

### General Information

Controlling brushed DC motors using microcontrollers is often accomplished by using an H-bridge, which performs two important functions regarding motor control. The first function amplifies the motor’s command signal to be within the operational range of the motor. The motor's command signal is typically low power, while the motor typically requires a much higher power. As mentioned in the previous section, motor control systems often require a driver circuit to help in delivering high power to the motor since the microcontrollers that provide the PWM command signal usually cannot tolerate the high voltages or currents that the motors need. In order to translate the lower power PWM command signal, of a given duty cycle of *x*%, into a proportional input to the motor that is perceived as *x*% of the constant voltage supply, an H-bridge can be used. H-bridges can also be used for a second function: to control the direction that a motor spins by changing the direction that electrical current passes through the motor.

To better understand how this occurs it is helpful to examine a diagram of an H-Bridge as shown in Figure 5. A standard H-bridge has four switches (S1-S4), with two connected to each terminal of the motor (M). In most motor control circuits, these switches are actually transistors or MOSFETs, but for the purpose of these examples they can be considered actual switches. By controlling which switches are open or closed, the polarity of the applied voltage to the motor can be changed. Since a DC motor’s rotation direction depends on the polarity of the applied voltage (assuming that the motor is capable of reversing directions) you can then use the switches of an H-bridge to allow a motor to reverse direction without needing to invert the original power supply voltage. When S1 and S4 are closed and S2 and S3 are open, a voltage is applied across the motor from left to right, and the motor spins in one direction. When S2 and S3 are closed and S1 and S4 are opened, the polarity of the applied voltage is reversed and the motor spins in the other direction.

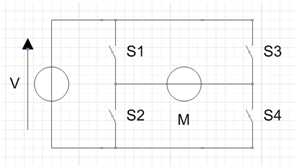


Figure 5: Basic H-Bridge Circuit

Putting these two H-bridge functions together, it becomes clear that the H-bridge can be used as driver circuit to control the speed and direction that the motor spins. If the current flowing through the closed switches is the constant supply voltage, and the PWM command signal is used to open and close one set of switches so that the switches are closed when the PWM signal is high (and vice versa), then the input experienced by the motor will be a constant supply voltage amplitude square wave with the same duty cycle as the PWM command cycle. Similarly, if the PWM signal is used to control the other set of switches, the speed of the motor can be controlled in the opposite direction.

### Free Run Mode and Brake Mode

In addition to controlling the direction and speed that the motor rotates, the H-bridge also has two operating modes in which it can break the spin of the motor: free-run mode and brake mode.

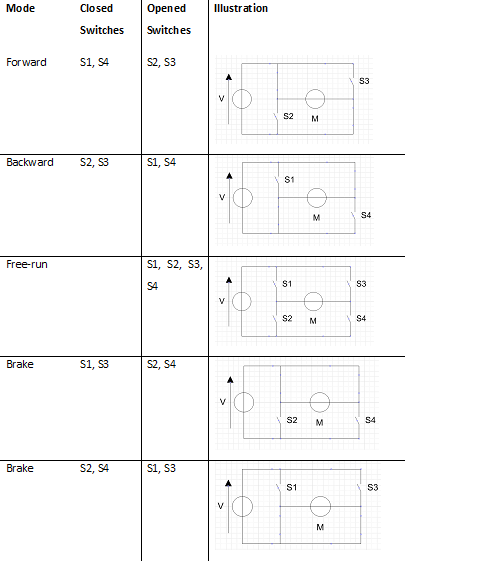
#### Free-Run Mode

Free-run mode allows your motors to come to a coasting stop, meaning the motors continue to rotate due to the robot’s momentum and stored energy within the motor until friction in the system causes them to stop. Opening all the H-bridge switches puts the motor into free-run mode. This isolates the motor from the electrical system and allows it to act as an inductor, maintaining a constant current if operating at a constant speed by storing energy in its windings. When the H-bridge switches are opened, the stored energy of the inductor allows the motor to continue running, but the power decreases due to mechanical and electrical losses. Because there are no electrical forces affecting the motor, and no voltage is applied to it, the motors can continue to rotate. The momentum of the robot also helps to propel the robot even after the motor power is removed. The motor then coasts to a stop after the remaining energy cannot overcome the friction of the system.

#### Brake Mode

When placed in brake mode, the electrical properties of the motors are used to provide active braking to allow for faster deceleration. No power is applied to the motor, but it is still connected to the electrical system of the robot. Closing S1 and S3 and opening S2 and S4 means that the motor will have equal voltage on both terminals and will fully discharge quickly. Similarly, S2 and S4 may be closed while S1 and S3 are opened, resulting in the same brake mode. This works by causing an electrical force to build up within the H-bridge, which in turn works to actively stop the rotation of the motor. Table 1 below summarizes the H-bridge modes discusses as well as their corresponding switch orientations.

Table 1. H-Bridge Operation Modes



#### Free-Run Mode versus Brake Mode

Both modes have different advantages and the level to which they are used often depends on the quality of the motors and the weight and maximum momentum of the load (i.e. the weight and maximum momentum of the ModBot).

Although brake mode can be advantageous for certain operations because it allows for a faster stop, it can cause some motors to wear considerably faster. The built-up energy in the motor discharges suddenly, creating spikes in voltage that may damage the control circuit. These spikes may even damage the motor and cause it to overheat, particularly if there is a lot of starting and stopping. Even if this is handled using methods like those described in the Power Board and Battery Selection Guide, the motor and gearing can suffer abuse and they can wear out that much faster.[[8]](#footnote-8) Free-run mode is gentler on the motors, but a coasting stop may not offer the best performance.

#### Supplementary Braking

An additional way to stop the motors is to enable free-run mode for a coasting stop, but utilize an exterior braking method, such as brake pads. In this manner, the brake pads do all of the work, protecting the motors from the negative effects described in the previous section, while increasing system friction to stop faster than normal free-run.

#### ModBot Configuration

The ModBot robot can be run in any of these modes but utilizes a free-run mode coasting stop without braking pads as its default mode of operation. In order to obtain a better performance, higher level control algorithms are utilized with robot position feedback as will be discussed in future documentation. This is a far more complex approach but allows for additional “fancier” maneuvers. Low level feedback control, however, should also be applied for standard operation speed control.[[9]](#footnote-9)

#### H-Bridge Short Circuits

One final note regarding the operation of an H-bridge: the user must be careful to not cause a short circuit, which in turn can cause the H-bridge circuit to overheat and damage components. It is important to see that closing both S1 and S2 at the same time will create a short circuit. Similarly, S3 and S4 cannot both be closed at the same time. The control signals (microcontroller code) must be checked to ensure that both of these cases are avoided.

### Selecting H-Bridge Motor Driver Boards

There are several ways in which an H-bridge circuit can be made. A standard H-bridge itself can be assembled from only a few components. The transistors may also be available as a pre-packaged chip, thus saving the assembly time needed to solder.

A more convenient option is to purchase a commercially-available H-bridge board which may run only $20-40. Commercial boards are available as a kit or as a fully-assembled board and often contain various safety features that would otherwise have to be designed, such as built-in logic to prevent short circuits and thermal protection.

Regardless of which option you use, it is important to test your H-bridge and motor subsystem before full implementation. Despite the promised operation characteristics of some purchased H-bridges, early design tests of the ModBot resulted in a significant amount of heat being generated by the motors. This was caused by the motors drawing excessive current from the motor controller. Excess heat can be dangerous to the robot’s components and could possibly lead to complete failure or even fire. Because of this, the amount of heat generated by the motor controller needed to be minimal to safely operate the robot. Unfortunately, low cost motor drivers have a general reputation of not being very robust and are prone to complete failures, so tests were performed on a variety of motor controller options. The tests were of course focused on motor controllers that would meet the following ModBot requirements:

* Provide a minimum of 8 amps of current, with an operating voltage of 24 volts
* Higher efficiency, with low heat generation (i.e. it won't burn out during an extended operation period)
* Provide for heat dissipation via heat sink
* Have a small enough form factor to safely be contained within the robot

Extensive testing was performed on a variety of motor controller options over a period of years and the following two motor drivers were determined to best satisfy the requirements:

1. Cytron 10A Single DC Motor Driver (MD10C)
2. Sabertooth dual 12 A Motor Driver

These two’s specifications are introduced in the next two subsections. They are then compared in the following testing section to show the decision process used in making a final selection.

#### Cytron 10A Single DC Motor Controller (MD10C)

The Cytron 10A Single DC Motor Controller (MD10C), available for $14.33, is designed to continuously drive a high current brushed DC motor up to 13 A. It offers several enhancements such as support for both locked-antiphase PWM signal and sign-magnitude PWM signals. It also uses solid state components, resulting in faster response time and eliminating the wear and tear of a mechanical relay. These properties help to control the robot more robustly and can include efficient braking operation. According to its data sheet,[[10]](#footnote-10) the MD10C has been designed with the capabilities and features of:

* Bi-directional control for 1 brushed DC motor.
* Support motor voltage ranges from 3 V to 25 V.
* Maximum current up to 10 A continuous and 15 A peak (10 second).
* 3.3 V and 5 V logic level input.
* Solid state components provide faster response time and eliminate the wear and tear of a mechanical relay.
* Fully NMOS H-Bridge for better efficiency and no heat sink is required.
* Speed control PWM frequency up to 10 kHz.
* Support both locked-antiphase and sign-magnitude PWM operation.
* Dimension: 75 mm x 43 mm

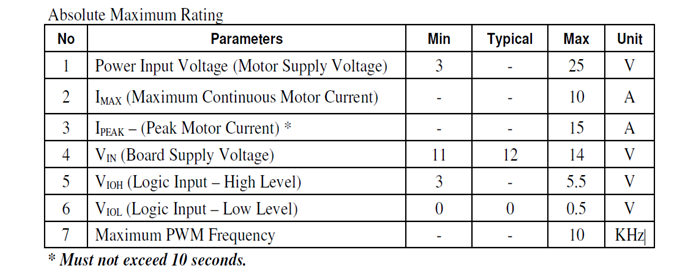


Figure 6: Features of Cytron motor driver

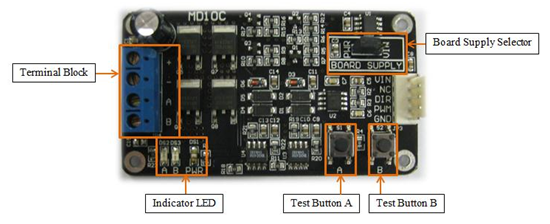


Figure 7: Cytron motor driver

#### Sabertooth Dual 12A Motor Driver

The Sabertooth is a bit more expensive at $80, but it can continuously supply two DC brushed motors with 12 A each. Peak currents of 25 A are achievable for a few seconds if necessary. The Sabertooth provides four different operating modes and can handle a variety of input signals, both analog and digital, providing a good number of control options. The Sabertooth has a built in 1 A Switching 5 V battery eliminator circuit (BEC) that can power additional electronics. The Sabertooth also offers a lithium cutoff mode that allows it to operate safely with lithium ion and lithium polymer battery packs. The Sabertooth's PWM transistors are also switched at ultrasonic speeds (32 kHz) for silent operation. The Sabertooth has the following specifications: [[11]](#footnote-11)

* Bi-directional control for 2 brushed DC motors.
* Support nominal motor voltage ranges from 6 V to 24 V, with an absolute max of 30 V.
* Maximum current up to 12 A continuous and 25 A peak (a few seconds).
* 5 V logic level input.
* Solid state components provide faster response time and eliminate the wear and tear of a mechanical relay.
* Speed control PWM frequency up to 32 kHz.
* Support both locked-antiphase and sign-magnitude PWM operation as well as a variety of other inputs.
* Dimension: 64 x 75 x 16 mm

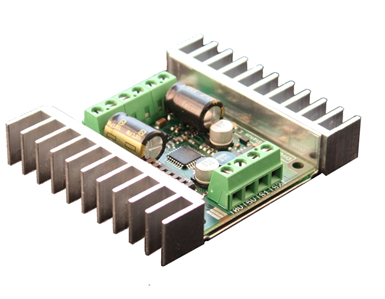


Figure 8: Sabertooth motor driver

## Tests Performed

When selecting any electronics device, it is always important to test its performance. Values listed on a data sheet are usually from an optimal configuration with a specific system; operational values likely differ in your configuration. There are a few standard parameters that should be monitored to identify the best motor driver:

* Heat dissipation
* Temperature
* Handling current spikes during fast directional switching
* Any kind of breakdown

For the ModBot system, the following tests were performed. It is highly recommended that you perform similar tests for any motor controller system that is being considered, as again many off-the-shelf motor controllers do not live up to their data sheets and the two were only found after many dead ends.

### Free Running Tests

For the free running tests, motor drivers were supplied with power and their attached motors were allowed to spin for over an hour. The temperature was monitored on both the controller and motor.

### Fast Switching Tests

In order to check for current spikes, the motor controllers were powered and the associated microcontroller was programmed to switch the motors between forwards and backwards motion at predetermined intervals. Both current and temperature were monitored, and the test was performed multiple times with switching at intervals of 10 seconds, 1 second, and 500 milliseconds. Each test was run for 20 minutes.

### Pulse Width Modulation Duty Cycle

The microcontroller was programmed to send PWM signals to the motor controller to control the speed of the motors. Varying duty cycles of 25%, 50% and 75% were used to verify changes in speed and direction.

### **Temperature Monitoring**

Throughout all of the previous tests, the heat dissipation properties of the motor controllers were monitored. Additionally, it is important to perform tests that simulate the usage of the robot under the extremes of the expected operating ranges.

To determine how much heat the motor controller would generate during operational use, a 3 Ω/300 W resistive load was applied in place of a motor to draw 6.5 A from the robot’s batteries. This was done to simulate the current draw when the motor suddenly changes directions, which can lead to excessive heating in inefficient motor controllers. The motor controller was placed in the robot’s enclosure and left for 30 minutes, and the temperature inside of the box was monitored throughout the test.

It is important that this test be run under the maximum stress that the motor controllers are expected to experience. It is ideally carried out for the full length of the expected runtime (30 minutes for the ModBot), but actual test time may be limited by the charge capacity of the batteries used. It is also far better to test with the resistive load than with the motors, since resistive load tests are potentially dangerous to costly motors, and this test only tests the motor controllers and not the motors themselves (which can be tested separately as well).

## Testing Conclusions

The selected motor controllers obtained excellent results in all the tests described in the previous section. The peak operating temperature of the Cytron was 61°C, and the peak temperature of the Sabertooth was 48°C. The cost of the Sabertooth was $80 for two motor channels, while the cost of the Cytron was $14.33 for single motor. The temperature results of both motor controllers are plotted below in Figure 9.

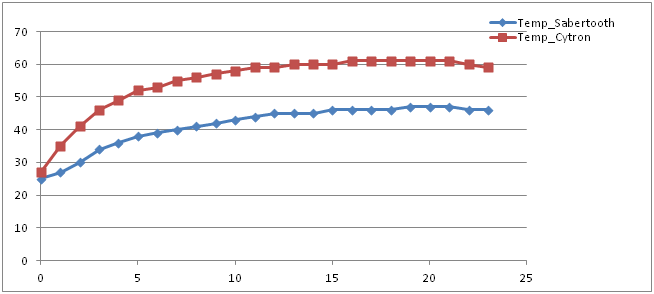


Figure 9: Temperature (Celsius) vs. Time (Minutes) Comparison

In the end, the Cytron 10A DC motor controller was chosen over the Sabertooth for the following reasons:

* The Cyton was more cost effective at $14.33 as compared to the Sabertooth at $80, or $40/motor as it can control two.
* The Cyton was light in weight and compact in dimensions, and could be easy fit in the compact design.
* It is NMOS fabricated, thus it was much less likely to cause current shocks to the user.
* There was little difference in the temperature of the two motor controllers after the 25 min temperature monitoring test.
* The added features of the Sabertooth were potentially valuable for other applications, but weren’t needed for the ModBot’s operations.

## Software Abstraction

This section outlines a few of the key translations that were put in place to translate the robot’s programmed code into the actual motor actions. More information on the commands available can be found in future documentation.

### Forward/Reverse Motion Control

#### Enable

The selected motor controller has an enable pin. When a HIGH voltage was applied to this input, the motor was active. Whenever any motion is required from the motor, the enable pin would need to be pulled HIGH.

#### Speed Control

The motor controller regulates the speed of the motor depending on the PWM input on its speed control terminal. The higher the duty cycle of the PWM, the higher the motor speed and similarly, the lower the duty cycle of PWM, the lower the motor speed.

The digital speed inputs of 0-255 were mapped directly with the PWM values of 0-255 on an Arduino pin to enable a linear control of the motor speed. If desired, the scale can be modified to suit your needs. A 0% duty cycle corresponds to an input of 0, and a 100% cycle corresponds to an input of 255, with a linear relationship in between.

For safety reasons and to preserve the life of the motor, speed values less than 10 were considered 0, and the enable pins were pulled low. This was implemented as very low speeds of the motor often are not powerful enough to put the robot into motion (i.e. overcome static friction), which could overheat or otherwise damage the motor. Microcontroller speed input values in excess of 255 were considered HIGH and reduced to the allowable value of 255 before being sent to the motor controller as well.

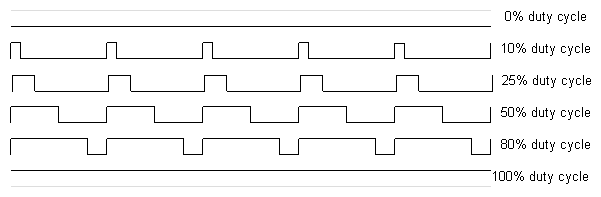


Figure 10: PWM description

#### Direction

When the drive motors were placed on opposite sides of the ModBot, they had to be mounted as mirror images of each other. Therefore, in order to get the robot to move forwards or backwards, the direction pin assignments for the two motors must be opposite. For example, the two motors are essentially facing opposite directions, so in order to move the robot forward, the left motor would have to spin clockwise (facing out from the robot) and the right motor would have to spin counter-clockwise (facing out from the robot). It is important to keep in mind the orientation of your motors and reflect this in your associated microcontroller code.

#### Calibration

Since the chosen motor controller was selected to match the specifications of the motors used in the ModBot, no specific calibration was required to operate the motors. If calibration is required, it is important to follow the manufacturer instructions provided with your motor and motor controller.

### Steering Control for the DuneBot variation

The DuneBot variation is more car-like than the ModBot, and it is meant to be used outdoors. It also utilizes different steering than the ModBot. Steering control programming for the DuneBot, which drives like a car but with each wheel having its own independent steering servo, consists of the following two components.

#### Angle

The values used in the robot’s code were mapped directly onto the servo motors controlling the steering. The actual angle thresholds, or how far the wheels were allowed to be turned, were left to the programmer to process and determine.

#### Calibration

Calibration can be the most time consuming part of the steering control system. Unfortunately, many servos do not perfectly align to a 50% duty cycle being the servo’s neutral position and likewise the servo may reach their extreme positions before a 0% or a 100% duty cycle command is given. The DuneBot had four servos controlling each of the wheels’ direction and each one needed to be calibrated independently.

The method of calibration used in the DuneBot involved sending angle commands to each servo using the serial port in order to determine the threshold values. The motors were turned to the extreme right and then to the extreme left, and these values were noted and added to the code as the thresholds. This ensured that the wheels would not be able to turn past these threshold angles in order to avoid damaging the robot. Furthermore, this also helped establish how the range of commands translated to the range of motion of the servo.

In the case of the DuneBot, the steering servo motors were able to move a complete 180 degrees before encountering the physical limitations of the link that connected them to the wheel.

## Things to discuss with teammates

There are a few important things to discuss with your teammates when choosing a motor controller. You should double check that it matches what the power system is designed for (see the Power Board and Battery Selection Guide). You also should confirm what type of motor is used in your system; is it brushed or brushless? Make sure your controller will work with the motor type. Also, make sure there is room in your system for the motor controller. The controller will take up space, and may need to be by a fan, so make sure the housing/chassis designers leave room for it.

1. More information on the power supply systems can be found in the Power Board and Battery Selection Guide. [↑](#footnote-ref-1)
2. Items 5-7 are discussed in the Power Board and Battery Selection Guide. [↑](#footnote-ref-2)
3. More information on the Arduino serial connection can be found in the Microcontroller guide. [↑](#footnote-ref-3)
4. More information on H-bridges can be found in Section 1.4. [↑](#footnote-ref-4)
5. More information on feedback can be found in Section 1.2. [↑](#footnote-ref-5)
6. Plot from: <http://rbsfm.org/am/index.php?option=com_content&task=view&id=425>. [↑](#footnote-ref-6)
7. Plot from: <http://rbsfm.org/am/index.php?option=com_content&task=view&id=425>. [↑](#footnote-ref-7)
8. See Power Board and Battery Selection Guide Section for more on the safety circuitry. [↑](#footnote-ref-8)
9. See Microcontroller guide for more on low-level control. [↑](#footnote-ref-9)
10. The capabilities are directly from the datasheet, which can be found at: <http://www.cytron.com.my/usr_attachment/MD10C_Rev2.0_User's_Manual.pdf> [↑](#footnote-ref-10)
11. The specifications are directly from the datasheet, which can be found at: http://www.dimensionengineering.com/datasheets/Sabertooth2x12.pdf [↑](#footnote-ref-11)