Arduino:
Fast Power-MOSFET Driver Cookbook

Armin Schweizer

When driving standard Power-MOSFET's, an Arduino digital output meets two challenges: The output voltage swing of a digital output may/will be insufficient to drive the Power-MOSFET fully and the available output current may be too small to drive the Power-MOSFET fast.

This Arduino Fast Power-MOSFET Driver Cookbook demonstrates three inexpensive options -- based on easily available components – to drive Power-MOSFET's in a fast way in different voltage environments.

1 Introduction

In an Arduino controlled system, Power-MOSFET's enable a software to control (turn on and off) significant electrical power in a fast way. 'Fast' in this context means

- steep rising and falling edges (to keep efficiency high) and
- low delay (to allow e.g. exact PWM timing)

When an Arduino digital output (DOT) or a pulse width modulation output (PWM) needs to control a standard Power-MOSFET, it will need some support to overcome two obstacles:

- The Arduino DOT or PWM output voltage swing of the digital output between 0 and 5 Volts (or 0 and 3.3 Volts) may prove to be insufficient to fully drive the Power-MOSFET;
- The output current available from an Arduino DOT or PWM output is insufficient to drive the Power-MOSFET sufficiently fast.

The following proposes – in form of a cookbook – three solutions to solve these challenges. Inexpensive standard components are used, allowing you to complete the circuits quickly and possibly even from components you have available on shelf.

Note: The circuit designs of this 'Fast Power-MOSFET Driver Cookbook' are obviously not restricted to Arduino environments only! They can be used in any environment with 5 volt (or 3.3 volt) outputs.

2 General Information…

2.1 …on hardware

This information is valid for any kind of Arduino. The circuits have been designed and tested with the ATmega32U4 chip (on an Arduino Micro) with 5 Volts operating voltage for the Arduino. When using a 3.3 Volts version of an Arduino, it is recommended to reduce RB to approx. 2/3 of the value shown in the circuit diagrams below.

The bipolar junction transistors (BJT) shown in the circuit diagrams can be easily replaced by other transistor types. It is critical, that the h_{FE} (or Beta) is sufficiently high (meaning more than 100) with

collector currents I_C being in the range of some hundred milli-amperes. Experience shows, that some manufacturers data sheets are optimistic in that respect (!).

Make sure, that all wire lengths between the parts are kept short to reduce inductance and capacity (which could slow down the circuit). This includes not only those parts of the circuit shown in the diagrams, but also the wires to the motors, solenoids, lamps, inductors etc. Keep in mind, that the switching speed achieved with the shown circuits is quite high and controls significant power: As a result these circuits may emit radio-waves in the multi-megahertz range at significant power. Impact on radio, CB, television, telephony etc. might result.

No in-depth research has been invested in the supply current transients during output transitions (when the two output transistors Q2 and Q3 in sections 4.2 and 5.2 are conducting at the same time). During the test measurements we found, that the proposed push-pull driver circuits are switching so fast, that the gate capacity of the Power-MOSFET is the main generator of current spikes during output transitions.

We are looking into Power-MOSFETs (n-channel) used as 'low-side switches' here. However you will find, that the circuits shown in sections 4.2 and 5.2 can be adapted easily to support Power-MOSFETs (p-channel) used as 'high-side switches'.

The (outdated) RFK-25N18 Power-MOSFET has been used here for demonstration purposes only: The high capacity at the gate of this device proved it to be nasty enough for testing purposes and this ensures, that your preferred Power-MOSFET will be driven very fast.

2.2 …on software

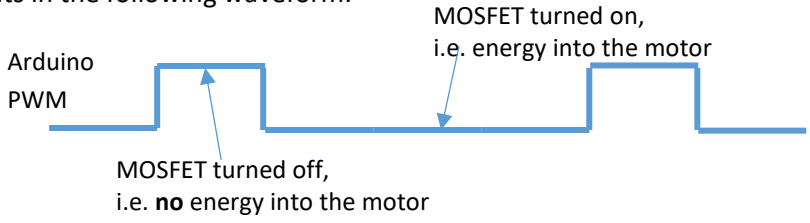
Please consider the logic-tables when writing the software. When using an Arduino DOT to control the Power-MOSFET, setting the DOT to HIGH will turn the Power-MOSFET off (=non-conducting) while a LOW will turn it on (= conducting).

When using an Arduino PWM to control the amount of power flowing to a device, please consider, the following:

- On an Arduino PWM (=analog output), setting a motor to a low speed (e.g. 25% energy) would be done by using the standard function analogWrite(pin, value);
- On a standard 8 bit PWM output the command and the wave-form at the Arduino-pin will look as follows:

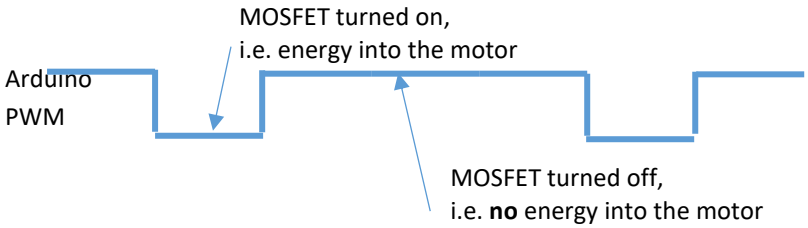
```
PWMpin = 11;      // Pin 11 as an example
AOTvalue = 64;    // on an 8 bit PWM (0..255) this will result in 25% high
analogWrite( PWMpin, AOTvalue );
```

This results in the following waveform:



- However, this is not what you want it to be! The motor will run with 75% energy instead of the intended 25% ! You can easily correct this by inverting the waveform by software (and still maintain the safe start/reset of the system):

```
PWMpin = 11;      // Pin 11 as an example
AOTvalue = 63;    //
analogWrite( PWMpin, (255 – AOTvalue) );  // results in 75% high
```



- Note: Depending on your application you would probably add some more logic to your software, e.g. to ensure, that a motor is not receiving a limited amount of energy which is insufficient to make it rotate (i.e. motor cooling is inactive) but burning energy in the motor and heating it up…

When using the Arduino development environment with default values, PWM's run with up to approx. 1 kHz only. This will require sometimes quite large and expensive inductors (e.g. when smoothing currents with a low-pass filter or when generating high voltages). Note that the PWM frequency can be massively increased, e.g. by changing the pre-scaler settings of the Arduino Timer/Counters used to drive a PWM (Example available here).

3 Quick & Dirty Power-MOSFET Driver Circuit

3.1 Approach

A minimal amount of components (and small board-space) is required with the common emitter circuit used in this quick & dirty approach. The bipolar junction transistor (BJT) Q1

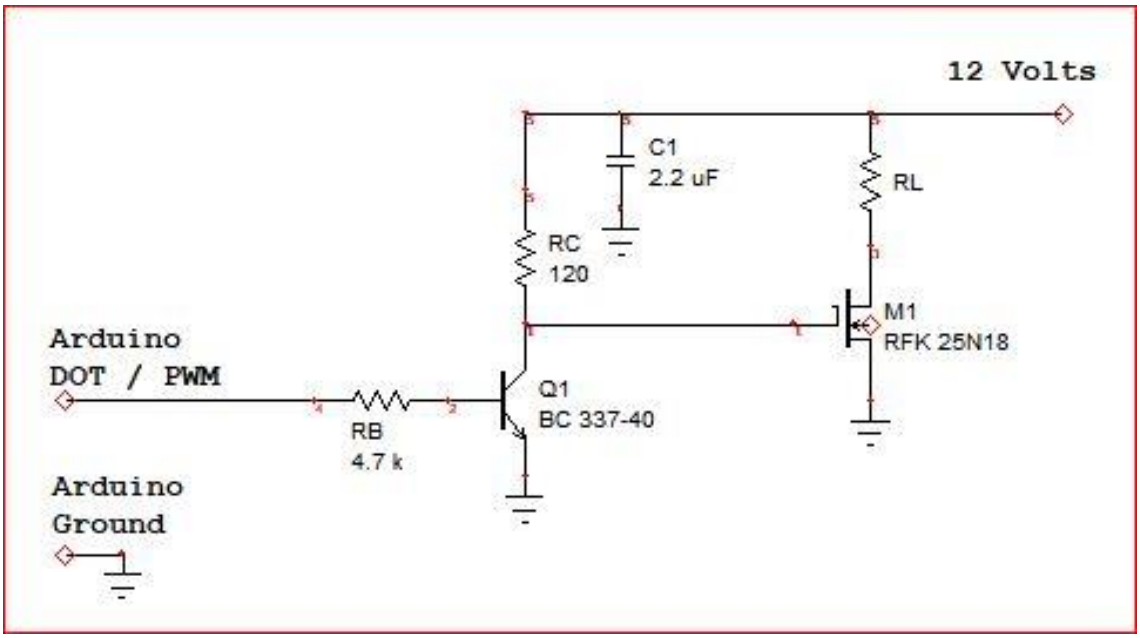
a) translates from the Arduino 5 volt environment to the power environment you need to control and
b) provides sufficient current into/from the gate of the Power-MOSFET to drive it faster than an Arduino DOT or PWM pin can do

The logic table looks as follows. It has to be noted, that after power-up or reset (tristate!) the Power-MOSFET is conducting until the ports have been configured by the Arduino's software: This may be a disadvantage in a real life application.

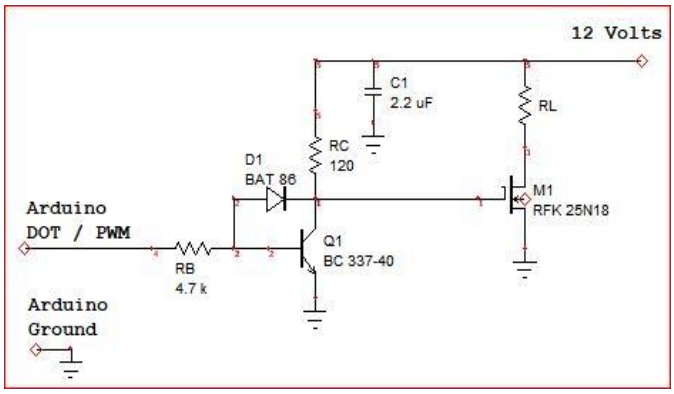
Electrical level at Arduino Pin (DOT or PWM)	**State of Power-MOSFET**
Tristate (after power-up or reset)	Turned on / Conducting current / 'low' on drain connector
LOW	Turned on / Conducting current / 'low' on drain connector
HIGH	Switched off / Blocking current / 'high' on drain connector

As another disadvantage you will find, that RC consumes a significant amount of power while Q1 is conducting, i.e. when the Power-MOSFET is switched off (see below in section 3.4).

3.2 Circuit

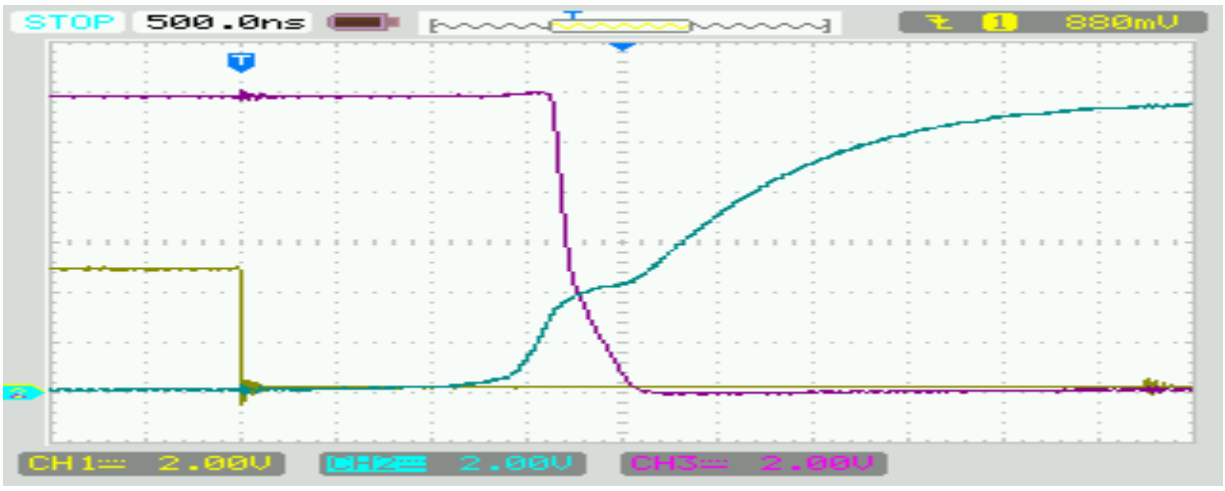


For the sake of completeness a slightly improved version is shown here too. It is adding a Schottky-diode with a small U_F of around 400 milli-volts to the transistor Q1 to prevent it from going into saturation – and consequently making Q1 faster when turning off. This comes with a cost however: The parts count is no longer minimized and it is approaching the dimension of the (better) circuits discussed in the following chapters.



3.3 Switching waveforms

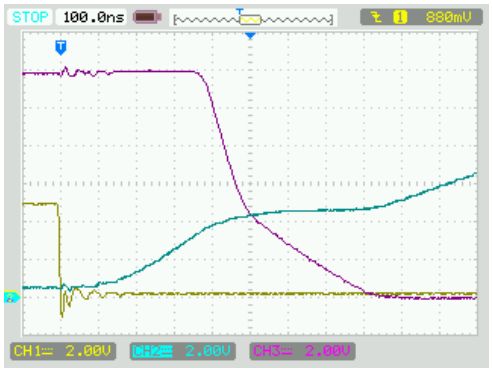
3.3.1 Turn Power-MOSFET on



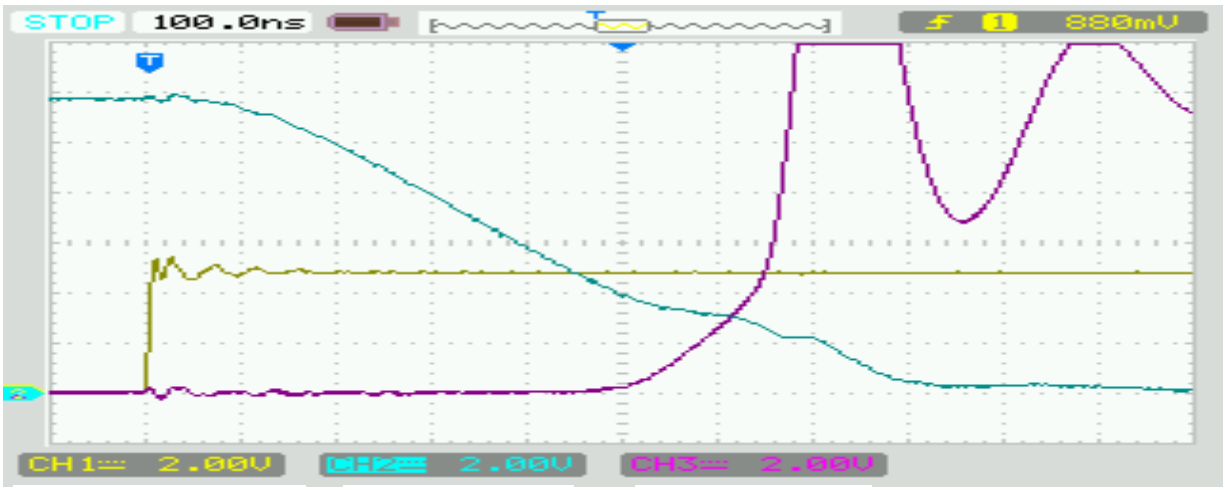
Arduino DOT / PWM Gate PowerFET Drain PowerFET

To turn the Power-MOSFET on, the Arduino needs to move the DOT / PWM output from high to low. The voltage at the collector of Q1 and consequently the voltage level of the drain of the Power-MOSFET is increasing and so does the load current when the Power-MOSFET turns on. Note, that this happens quite slowly, this takes about 2 micro-seconds (the scope is set to 500 nano-seconds per division on this 1st picture).

When looking at the improved version, there is a significant gain in speed visible (the scope is set to 100 nano-seconds per division in this 2nd picture). However it is still significantly slower than the circuits shown in following chapters.



3.3.2 Turn Power-MOSFET off



Arduino DOT / PWM Gate PowerFET Drain PowerFET

To turn the Power-MOSFET off, the Arduino needs to move the DOT / PWM output from low to high. The voltage at the collector of Q1 and consequently the voltage level at the gate of the Power-MOSFET is decreasing. This happens not too fast and it takes more than 500 nano-seconds until the Power-MOSFET turns to the off-state. For now ignore the huge damped sine-wave of the drain of the Power-MOSFET which results from the (cable-)inductance and capacitance of my experimental setup.

The improved circuit option (with diode D1) will not provide any advantage in this situation and the resulting switching speeds and scope-picture are essentially the same (and so are not shown here).

3.4 Adapting it to different voltages

In order to adapt this circuit to higher operating voltages on the Power-MOSFET's side, you need only to proportionally adapt RC:

Voltage	RC
10	100 Ohms / 1 Watt
12	120 Ohms / 1.2 Watts
15	150 Ohms / 1.5 Watts
24	220 Ohms / 3 Watts
30	270 Ohms / 4 Watts
36	330 Ohms / 5 Watts

3.5 Conclusion

This circuit topology is extremely simple and needs only very few parts. It is fast enough to switch a Power-MOSFET in a standard Arduino PWM environment (with PWM-frequencies up to 1 kHz) or even more if you modify the prescaler settings of the ATmega microcontroller chip.

A clear disadvantage of this mini-circuit is the missing control of the Power-MOSFET during tristate: The Power-MOSFET will be switched on (e.g. during power-up and reset of the microcontroller). This may be unacceptable in many applications.

The main disadvantage with respect to the circuit is the high amount of energy burnt in RC (and the PCB-board-space and cost consumed by such a large device). In fact this is the main motivator for using push-pull driver configurations: See more about this approach in the rest of this document.

4 Push-Pull Driver Using Complementary Bipolar Junction Transistors

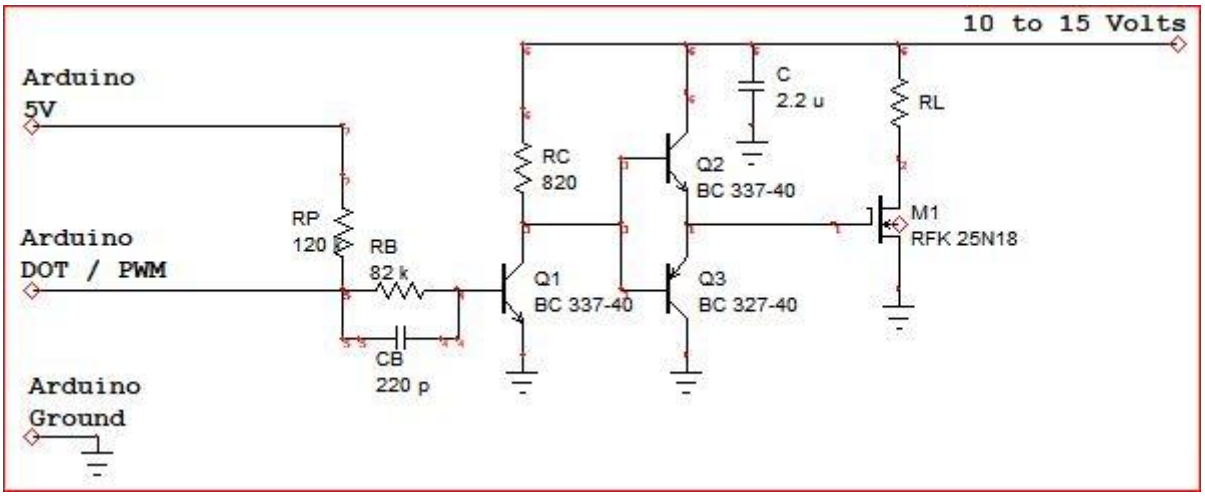
4.1 Approach

This circuit design is using complementary bipolar junction transistors (BJT) of NPN and PNP type. This approach is helpful in those cases where you want to achieve highest speed with lowest number of parts. The circuit uses transistor Q1 to change the voltage levels. The push/pull-approach shown here is also called a 'class B' amplifier. The use of complementary transistors (Q2 and Q3) allows to use the transistors as emitter followers, making this circuit very fast. Note, that the two complementary transistors need *not* be 'paired' (specially selected) as it was the case in older music amplifiers using 'class B' circuits.

This circuit has been designed to work safely also during power-up and reset. The logic table looks as follows:

Electrical level at Arduino Pin (DOT or PWM)	State of Power-MOSFET
Tristate (after power-up or reset)	Switched off / Blocking current / 'high' on drain connector
LOW	Turned on / Conducting current / 'low' on drain connector
HIGH	Switched off / Blocking current / 'high' on drain connector

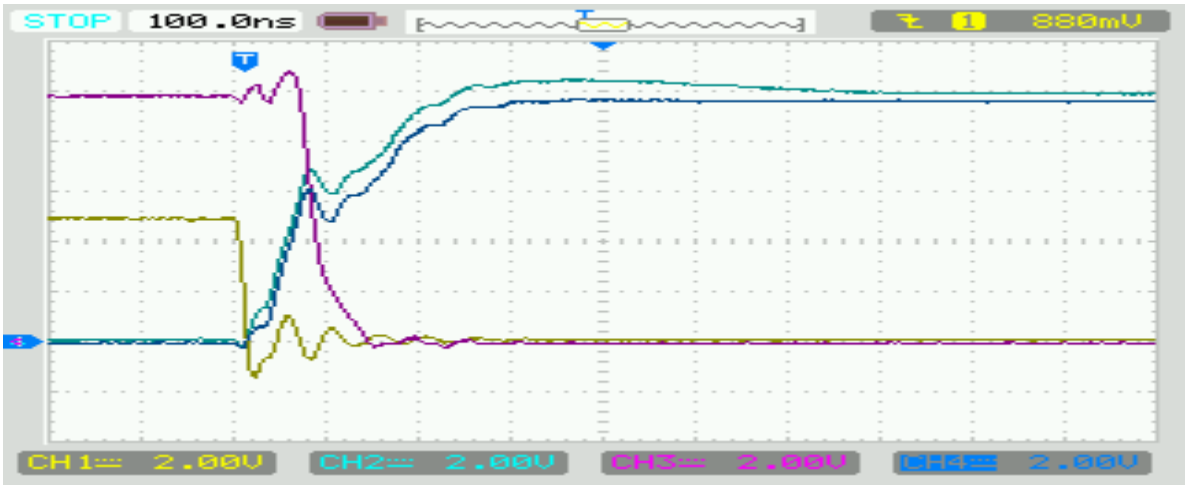
4.2 Circuit



The current required from the Arduino chip while switching (in either direction) is higher than during the stable periods. Capacitor CB (a ceramic type) ensures sufficient current can drive the circuit during switching while keeping the current low (and the Arduino chip cooler) during the stable periods.

4.3 Switching waveforms

4.3.1 Turn Power-MOSFET on

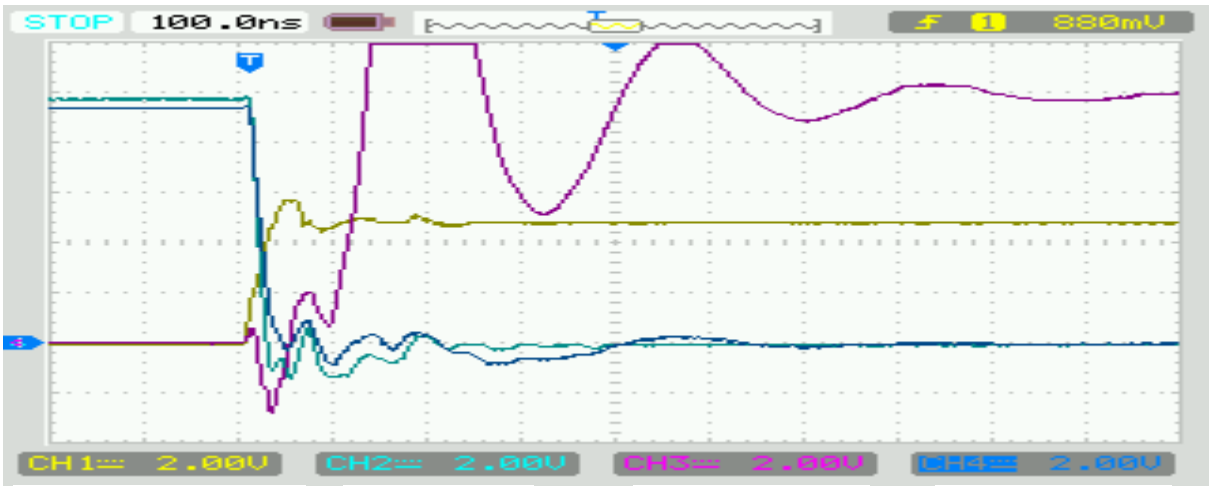


Arduino DOT / PWM Collector Q1 Drain PowerFET Gate PowerFET

To turn the Power-MOSFET on, the Arduino needs to move the DOT / PWM output from high to low. The collector of Q1 and consequently the emitters of Q2 and Q3 – these transistors are both used as

emitter followers – react within less than 100 nanoseconds. And so does the drain of the Power-MOSFET as well as the load current.

4.3.2 Turn Power-MOSFET off



Arduino DOT / PWM Collector Q1 Drain PowerFET Gate PowerFET

To turn the Power-MOSFET off, the Arduino needs to move the DOT / PWM output from low to high. The collector of Q1 and consequently the emitters of Q2 and Q3 -- working as emitter followers – react within much less than 100 nanoseconds. And so does the drain of the Power-MOSFET. And within less than 100 nano-seconds the load current has been turned off (see first 'bump' in the red curve). For now ignore the huge damped sine-wave of the drain of the Power-MOSFET which results from the (cable-)inductance and capacitance of my experimental setup.

4.4 Adapting it to different voltages

In order to adapt this circuit to higher operating voltages on the Power-MOSFET's side, you need only to proportionally adapt RC:

Voltage	RC
10	820 Ohms / ¼ Watt
12	1 kOhm / ¼ Watt
15	1.2 kOhm / ¼ Watt
24	1.8 kOhm / ½ Watt
30	2.5 kOhm / ½ Watt
36	3.3 kOhm / ½ Watt

4.5 Conclusion

This circuit provides high speed switching (rising and falling edges as well as short delays) making it a good choice for all sorts of power control (motor speed, power conversion etc.). In addition it uses a low number of parts (low cost, small PCB-board space). Some designers (e.g. chip designers) may consider it a disadvantage, that NPN and PNP transistors are used.

5 Push-Pull Driver Using NPN Bipolar Junction Transistors

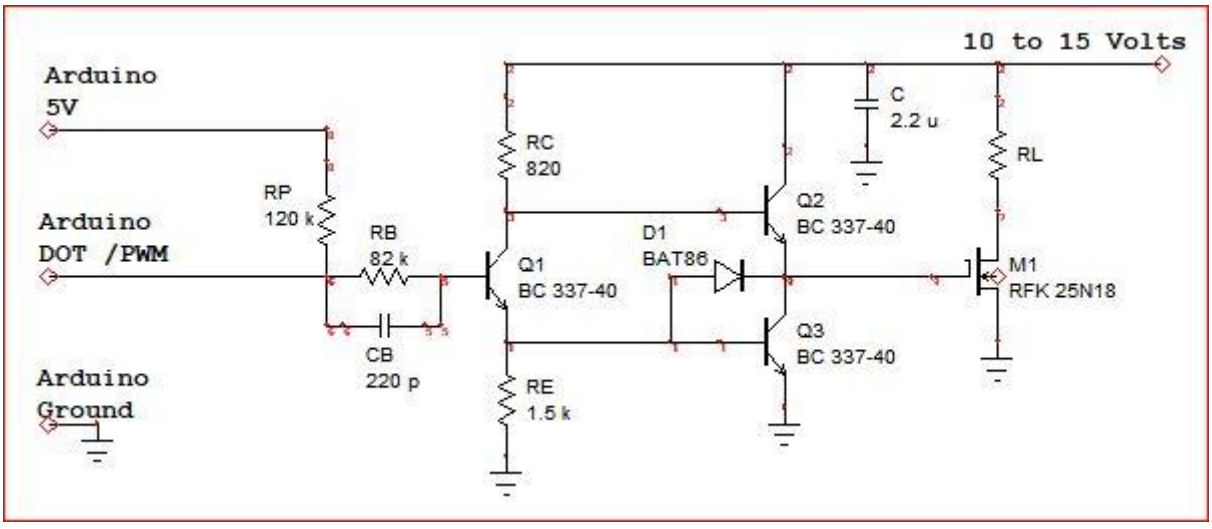
5.1 Approach

This circuit design is using NPN transistors of the same type. It is helpful in those cases where you want to use NPN transistors only (restrictions of cost, availability in stock etc.). The circuit uses transistor Q1 to change the voltage levels and a transistor used as emitter follower (Q2) which can both switch quite fast. However transistor Q3 is used in a common-emitter configuration and this setup has a tendency to switch comparably slow. By using the Diode D1 – a small Schottky diode (BAT 86 used here) with an U_F of approx. 400 milli-volts – the transistor can be prevented from reaching saturation. As a result it will be able to switch full speed (see below). Without the diode D1 the Power-MOSFET turn-on will be 5 to 10 times slower.

This circuit has been designed to work safely also during power-up and reset. The logic table looks as follows:

Electrical level at Arduino Pin (DOT or PWM)	**State of Power-MOSFET**
Tristate (after power-up or reset)	Switched off / Blocking current / 'high' on drain connector
LOW	Turned on / Conducting current / 'low' on drain connector
HIGH	Switched off / Blocking current / 'high' on drain connector

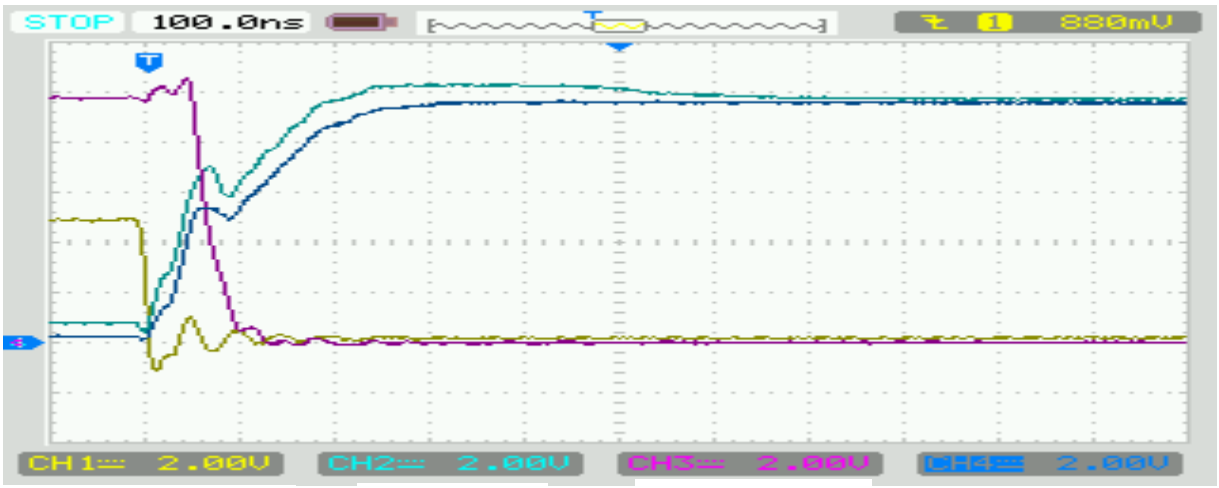
5.2 Circuit



The current required from the Arduino chip while switching (in either direction) is higher than during the stable periods. Capacitor CB (a ceramic type) ensures sufficient current can drive the circuit during switching while keeping the current low (and the Arduino chip cooler) during the stable periods.

5.3 Switching waveforms

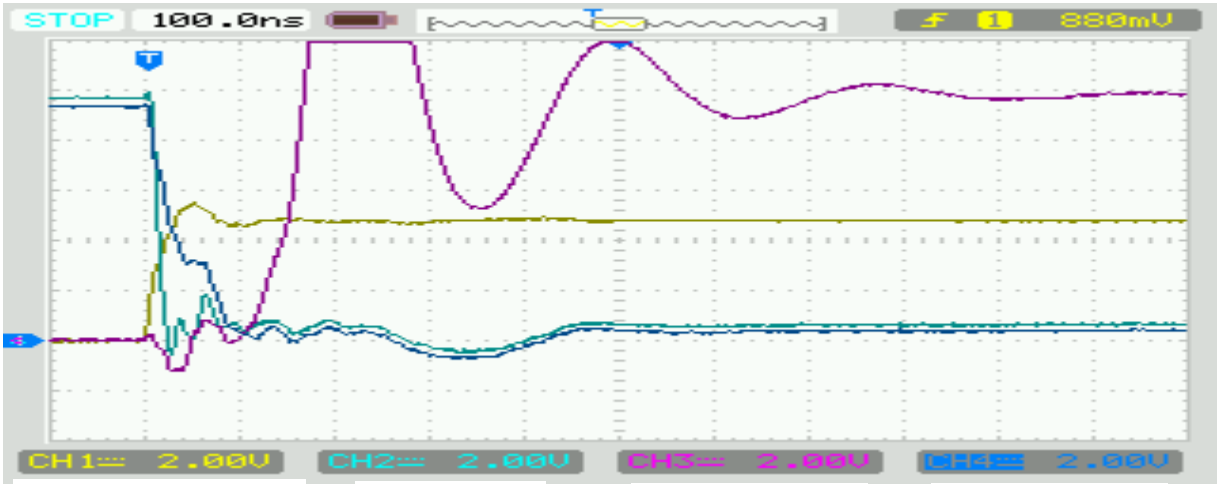
5.3.1 Turn Power-MOSFET on



Arduino DOT / PWM | Collector Q1 | Drain PowerFET | Gate PowerFET

To turn the Power-MOSFET on, the Arduino needs to move the DOT / PWM output from high to low. The collector of Q1 – and consequently the Base of Q2, which differs from Q3 collector only by UBE – reacts within less than 100 nanoseconds. At the same time the current into the base of Q3 is turned off, resulting in the turn-off of Q3. And so the drain of the Power-MOSFET starts to conduct the load current.

5.3.2 Turn Power-MOSFET off



Arduino DOT / PWM | Collector Q1 | Drain PowerFET | Gate PowerFET

To turn the Power-MOSFET off, the Arduino needs to move the DOT / PWM output from low to high. The collector of Q1 – and consequently the emitter of Q2, which is working as emitter-follower here – reacts within less than 100 nanoseconds. At the same time the current into the base of Q3 is turned on, resulting in the turn-on of Q3. And so the drain of the Power-MOSFET (see first small 'bump' in the red curve) is starting to block the load current. For now ignore the huge damped sine-wave of the drain of the Power-MOSFET which results from the (cable-)inductance and capacitance of my experimental setup.

5.4 Adapting it to different voltages

In order to adapt this circuit to higher operating voltages on the Power-MOSFET's side, you need only to proportionally adapt RC:

Voltage	RC
10	820 Ohms / ¼ Watt
12	1 kOhm / ¼ Watt
15	1.2 kOhm / ¼ Watt
24	1.8 kOhm / ½ Watt
30	2.5 kOhm / ½ Watt
36	3.3 kOhm / ½ Watt

5.5 Conclusion

This circuit shows a speed equivalent to the one in section 4 and allows large Power-MOSFET's to be driven with many hundred kilohertz (e.g. for pulse width modulation PWM) or to be turned on and off within less than 100 nano-seconds (e.g. to generate high voltages in push-converters with good efficiency). Depending on your design restrictions, it can be an advantage, that only NPN transistors are used and that all devices can easily be purchased on the market. The disadvantage of this approach is the slightly higher number of components, resulting in somewhat higher cost and slightly more PCB board-space.

6 Summary

Typically a Power-MOSFET driver circuit will be implemented with discrete parts only, when integrated Power-MOSFET driver chips or modern Power-MOSFET designs (logic level MOSFET's, intelligent MOSFET's etc.) are too expensive or unavailable for the specific requirement. With the three Power-MOSFET driver solutions presented above, large Power-MOSFET's can be driven in a fast way (fast meaning steep rising/falling edges as well as low delays) and at a very moderate cost (the parts cost a few cents and there is not too much costly PCB-board space required). Each solution can satisfy different needs, however in general the solution in section 4 is probably my recommended solution for many/most cases.