

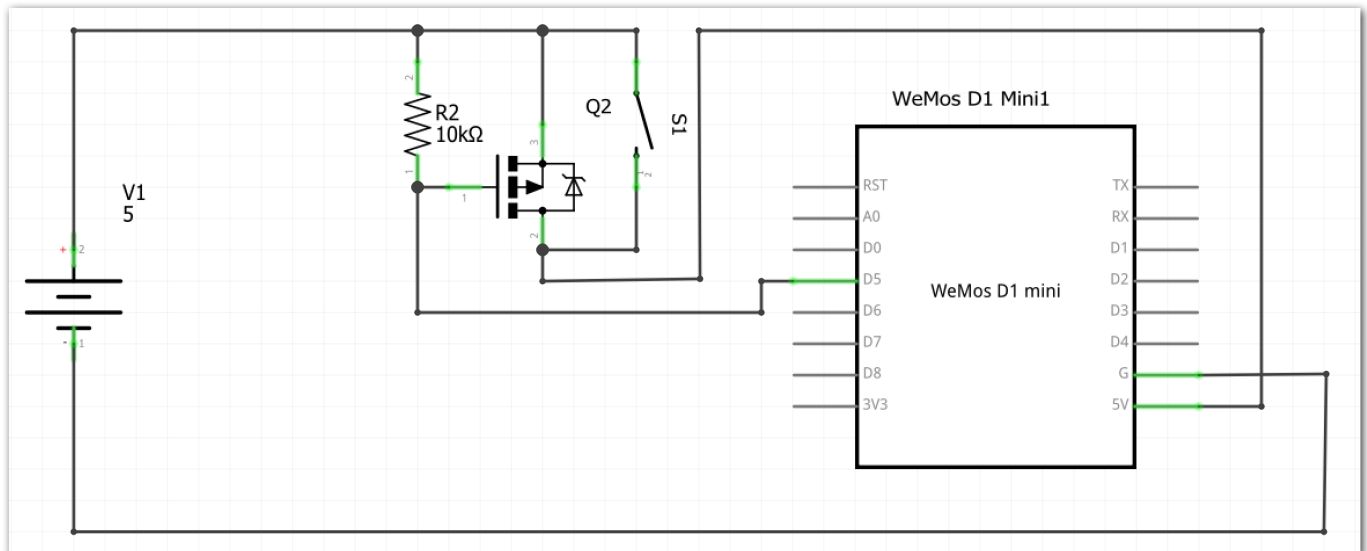
## Controlling Wemos D1 power with P-Channel MOSFET

[ Note: this is the continuation of [another question](#), the first one being my attempt to implement the same idea using an N-channel MOSFET. Based on the answers I received there, I went out, bought a P-channel MOSFET, and found myself with even more questions than before :) ]

I am trying to implement something similar to this: <https://youtu.be/nbMfb0dlvYc?t=4m27s>

The idea is that the microcontroller (a Wemos D1 Mini in my case) is powered up by an external switch. As soon as it starts up, it sets up a MOSFET so that it can keep its power connection until it finishes whatever job it is set up to do. In the end, it "kills itself" by cutting off the power through the MOSFET.

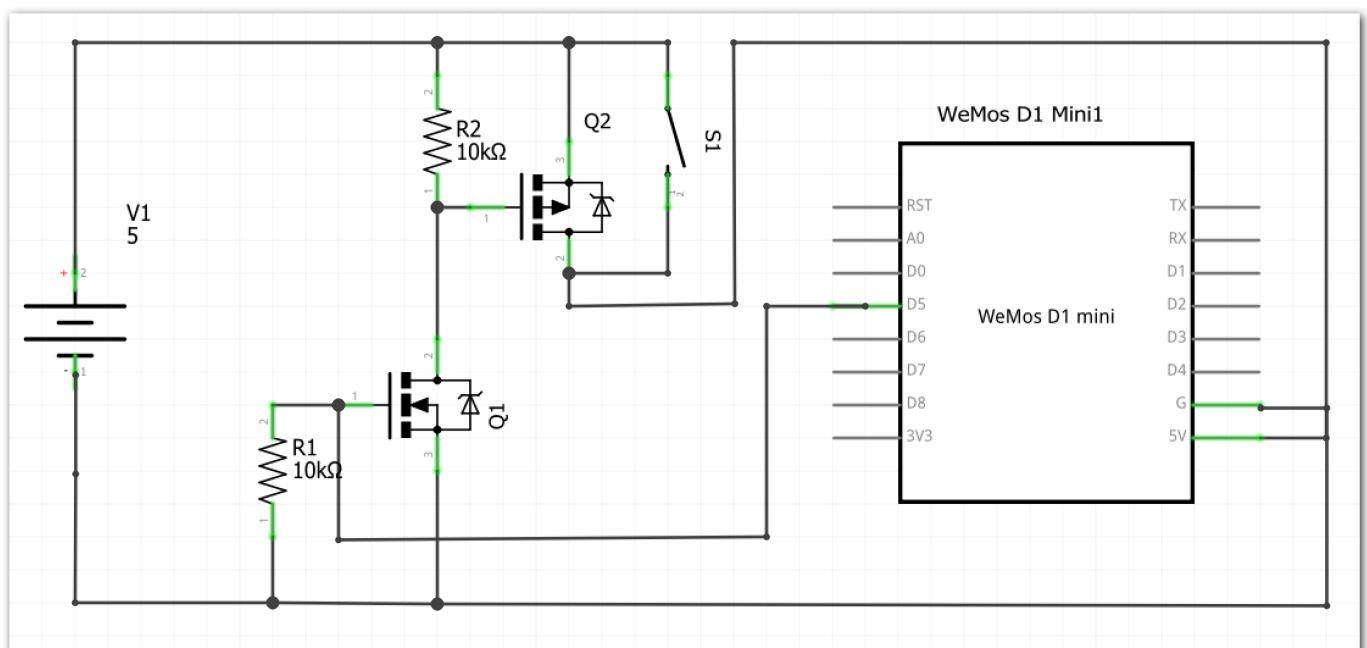
So I bought a P-channel MOSFET ([IRF9Z34N](#) - probably not ideal with a  $V_{gs(th)}$  max of -4V, but it was the best through-hole part I was able to find). And I built the following circuit:



When I press the switch, the D1 powers up correctly, pulls the MOSFET gate pin low, and goes through all of its code. In the end, it pulls the gate pin high, powering itself off.

However, something is wrong. When powering itself off (after setting D5 to HIGH), I would have expected the gate voltage to go to +5V through the pull-up resistor. Instead, it stays at around 1.25V - enough to keep the MOSFET partially open, with enough current going through it to keep the MCU in some weird, not-alive-but-not-dead-either, state (the MCU doesn't receive enough current to keep running its code, but it receives enough to prevent it from fully resetting and starting over again when I press the switch).

Based on the suggestion received from Brian Drummond on my original question, I built a dual-MOSFET arrangement:

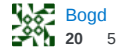


If it helps, [here](#) is the schematic for the Wemos D1 Mini. Note that I am powering it through the 5V line, which then goes through a power regulator.

mosfet low-power esp8266

edited May 5 at 20:08

asked May 5 at 19:20



Bogd  
20 5

I suspect we can see the likely cause already... However, to be sure: What is the part number of the P-channel MOSFET that you're using? I didn't see that in the question. A link to its datasheet would be helpful. – [SamGibson](#) May 5 at 19:42

Sorry about that - I wanted to copy/paste the P/N (IRF9Z34N) right before I mentioned the Vgs(th), but somehow the name didn't make it through. I edited the question to add it. – [Bogd](#) May 5 at 20:09

Two MOSFETs and two resistors is far from needlessly complicated in my book :) – [anrieff](#) May 5 at 20:31

@anrieff - twice as complicated as the original :) . And trying to simplify it is proving to be a learning experience in itself :P – [Bogd](#) May 5 at 20:37

True, but keep in mind that in EE, being overzealous on minimizing partcount is rarely a good strategy. If you *really* wanted a single-part switch, that could be a LDO (with Enable pin) or a SSR for the pass element. – [anrieff](#) May 5 at 21:08

## 1 Answer

As per your comments, let's discuss why the single PMOS solution doesn't work.

All I/O pins of modern microcontrollers use [protection diodes](#) to, well, protect sensitive internal circuitry from fault conditions - e.g. if you externally drive 5V or -5V onto a pin of a MCU, which is powered by 3.3V. This could also happen due to ESD, and without such protection, devices are easily destroyed.

However, this has two consequences:

1. If a microcontroller chip is turned off (0V on its power (Vdd) pin), putting a sufficient voltage to any I/O pin, will cause current to "overflow" to the Vdd pin, and the chip will spring to life, as if you've powered it normally
2. Again, in a turned off MCU, if you have a pull-up resistor to an active supply, small current will flow into the chip, try to power it, but as the current is insufficient, it will hang into this "neither on, neither off" state you're seeing.

A simple rule of the thumb there is that, if a MCU is controlling something, and it is expected the MCU to be off or otherwise disabled for some time (e.g. in sleep mode, for MCUs that don't keep their I/Os driven during sleep), it is best if the controlled thing requires an "active on" signal, and there's a pull-down resistor. No resistor to the gate of a MOSFET would be bad, pull-up is worse, pull-down works.

Note that this kind of precautions aren't required if the MCU is expected to be always active. It is not the best practice, as resistors cost pennies, but you *could* omit the pull-down resistor in such a case.

For your configuration, the two-MOSFET solution is about the best it can possibly get.

Another scenario to explore, which can bring the same functionality without any MOSFETs, is to use the sleep/reset/wake-on-change functionality of the MCU (I'm not proficient with the Wemos, but I bet it has something like it). When you want to "turn off", you simply go to sleep. The MCU will still consume *some* current, but that would be too small to matter. And your switch can be wired to a reset pin of the MCU, or, if you want to keep the processor state - to a pin, on which you've configured interrupt/wake-on-change.

answered May 6 at 9:43



anrieff  
1,282 8 21

Thank you for the very detailed reply - it definitely answers the "why" :) . To address your final suggestion, the ESP8266 has a nice deep sleep option, and I did try that initially. However, I was unable to design a circuit that would generate the reset pulse for waking up the device. I will try asking another question to get some pointers. Once again, thank you! – [Bogd](#) May 6 at 14:05