MOSFET no Arduino: O Erro Que Faz o Transistor Virar um Aquecedor!

Descrição

Um MOSFET pode permitir que um Arduino controle motores, solenóides e fitas de LED de alta potência com segurança. Mas existe uma armadilha: se você escolher o MOSFET errado, ele pode não ligar completamente, aquecer demais e até queimar.

Neste tutorial, vamos entender por que o Arduino é um cérebro, mas não é um músculo; por que o MOSFET é uma excelente chave eletrônica; o perigo da tensão de limiar; a importância do MOSFET nível lógico; o papel do resistor de gate; e por que cargas indutivas precisam de diodo de flyback.

Parece magia, mas é tecnologia...

https://youtu.be/xf0o5pNtg98

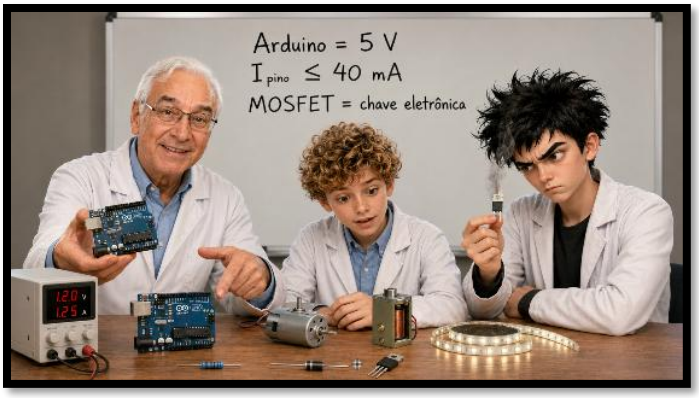
www.bairrospd.com

VISITE O SITE DO PROFESSOR BAIRROS LÁ EM O PDF E MUITO MAIS.
PARA AULAS ONLINE CONTATE VIA SITE.
www.bairrospd.com

https://www.youtube.com/@professorbairros

Sumário

1.1. A pergunta: MOSFET é bom para Arduino?



Arthurzinho olha para o Arduino, olha para o MOSFET e pergunta:

“Professor, o MOSFET é bom mesmo para ligar direto no Arduino?”

Essa pergunta parece simples, mas é uma daquelas que escondem uma armadilha. Sim, um MOSFET pode permitir que um Arduino de 5 volts controle motores, solenóides, fitas de LED e cargas de potência com segurança. Mas existe um detalhe perigoso: se você comprar o tipo errado de MOSFET, a chave não vai apenas falhar. Ela pode aquecer rapidamente, virar um resistor gigante e até queimar.

O Arduino é ótimo para pensar, calcular e comandar. Mas ele não foi feito para entregar potência. Ele é o cérebro do circuito, não o músculo. E é aí que entra o MOSFET: ele fica entre o Arduino delicado de 5 volts e a carga pesada alimentada por uma fonte separada.

1.2. Arduino é cérebro, não é músculo

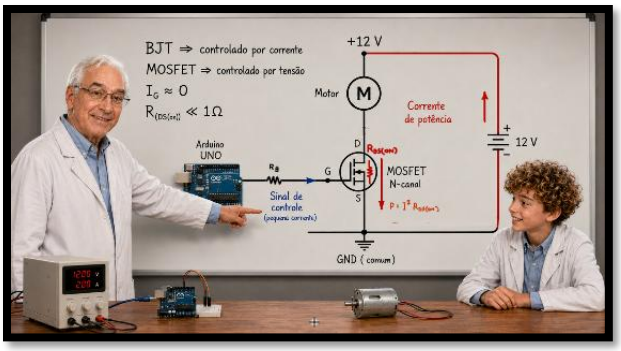
Um pino de saída do Arduino trabalha com 5 volts e pode fornecer, no máximo, algo ao redor de 40 miliamperes. Isso é suficiente para acender um LED indicador, enviar um sinal lógico ou comandar a entrada de outro circuito. Mas não é suficiente para alimentar diretamente um motor de corrente contínua, um solenóide ou uma fita de LED de maior potência.

Se fizermos a conta da potência máxima de um pino, temos tensão vezes corrente. Com 5 volts e 40 miliamperes, o pino entrega apenas 0,2 watt.

$$P_{\text{pino}} = 5\ \text{V} \cdot 0{,}04\ \text{A}$$
$$P_{\text{pino}} = 0{,}2\ \text{W}$$

Isso é muito pouco para cargas reais. Por isso, o Arduino não deve alimentar a carga. Ele deve apenas mandar o comando. Quem entrega a corrente pesada é uma fonte externa, e quem deve ligar ou desligar essa corrente é uma chave eletrônica externa. Nesse caso, o MOSFET.

1.3. Por que o MOSFET parece perfeito?



O que torna o MOSFET tão interessante para projetos com Arduino é o mecanismo de entrada. Diferente de um transistor bipolar, que precisa de corrente contínua na base para permanecer ligado, o MOSFET é controlado principalmente por tensão.

O terminal de controle é chamado de gate. Esse gate é eletricamente isolado do canal principal. Então, quando o Arduino aplica 5 volts no gate de um MOSFET adequado, o canal entre dreno e source liga e deixa a corrente passar. Depois de carregado, o gate não consome praticamente corrente alguma em corrente contínua.

$$I_G \approx 0$$

Outro ponto importante é a baixa resistência quando o MOSFET está totalmente ligado. Essa resistência é chamada de resistência entre dreno e source em condução, ou resistência de dreno ligado.

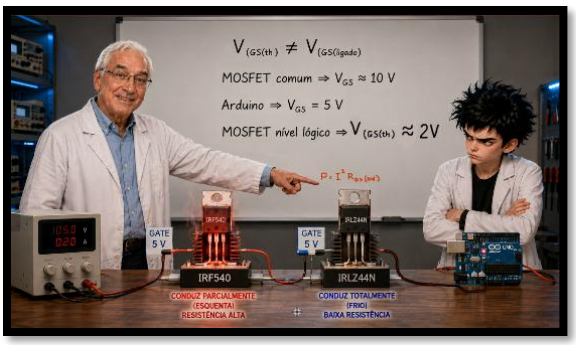
$$R_{DS(on)}$$

Quando o MOSFET está corretamente acionado, essa resistência pode ser muito pequena, frequentemente na casa de milésimos de ohm ou centésimos de ohm. Isso permite chavear correntes altas com pouca dissipação de calor, já que a potência perdida no MOSFET pode ser estimada pela famosa equação I2R, nesse caso o R é a resistência de dreno source ligado.

$$P_{\text{MOSFET}} = I_D^2 \cdot R_{DS(on)}$$

Se a resistência for baixa, o calor também será baixo. Mas aqui aparece a primeira armadilha: essa resistência só fica realmente baixa quando o MOSFET está completamente ligado.

1.4. A grande armadilha: tensão de limiar não é tensão de trabalho



A armadilha mais comum é confundir tensão de limiar, chamada de tensão de threshold nos datasheets, com a tensão de acionamento completo.

$$V_{GS(th)}$$

Muitos iniciantes olham esse valor e pensam: “se o threshold é baixo, então o Arduino consegue ligar o MOSFET”. Só que não é bem assim.

A tensão de limiar é apenas o ponto onde o MOSFET começa a conduzir uma corrente pequena de teste. Ela não significa que o MOSFET já está completamente ligado. Para trabalhar como chave de potência, o MOSFET precisa de uma tensão de gate suficiente para deixar o canal bem ligado e reduzir bastante a resistência, na prática um valor igual ou maior do que o dobro da tensão de threshould.

Por isso, muitos MOSFETs de potência comuns, como o IRF540, precisam de algo em torno de 10 volts no gate para ficarem plenamente ligados. Quando o Arduino aplica apenas 5 volts, esse MOSFET pode ligar só parcialmente. E nesse estado intermediário, ele deixa de se comportar como uma boa chave e passa a se comportar como um resistor.

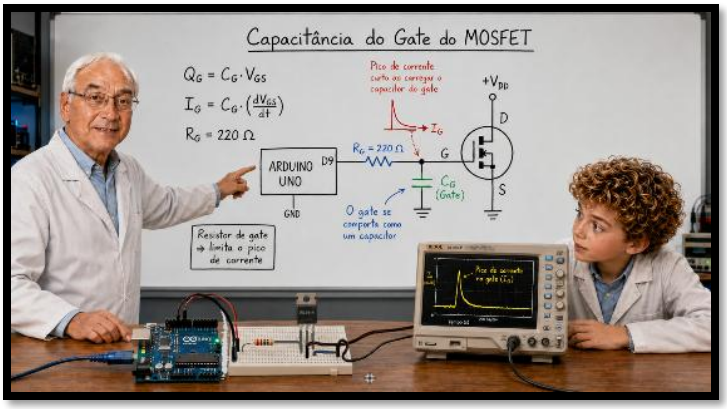
A potência perdida cresce rapidamente e é função da corrente de dreno ao quadrado multiplicado pela resistência de dreno source ligado do MOSFET.

$$P_{\text{perda}} = I_D^2 \cdot R_{DS(on)}$$

Se a corrente do motor for alta e a resistência do MOSFET ficar alta porque ele não ligou completamente, o resultado é aquecimento. Muito aquecimento.

Por isso, com Arduino, a escolha correta é um MOSFET de nível lógico, como o IRLZ44N ou outro modelo cujo datasheet especifique tensão de threshould da ordem de 2 volts no gate.

1.5. O gate é isolado, mas não é mágico: ele é um capacitor



O segundo problema é a capacitância do gate. O gate do MOSFET é isolado eletricamente, e por isso ele se comporta como um pequeno capacitor. Então, embora em corrente contínua ele consuma praticamente nada, no instante da comutação ele precisa ser carregado e descarregado.

A carga do gate pode ser representada por pelo produto da capacitância do gate multiplicado pela tensão gate source.

$$Q_G = C_G \cdot V_{GS}$$

E a corrente necessária para carregar essa capacitância depende da velocidade de variação da tensão, como mostra a equação, a corrente é igual a capacitância do gate que multiplica a taxa de variação do gate source em relação ao tempo é esse dv/dt da equação.

$$I_G = C_G \cdot \frac{dV_{GS}}{dt}$$

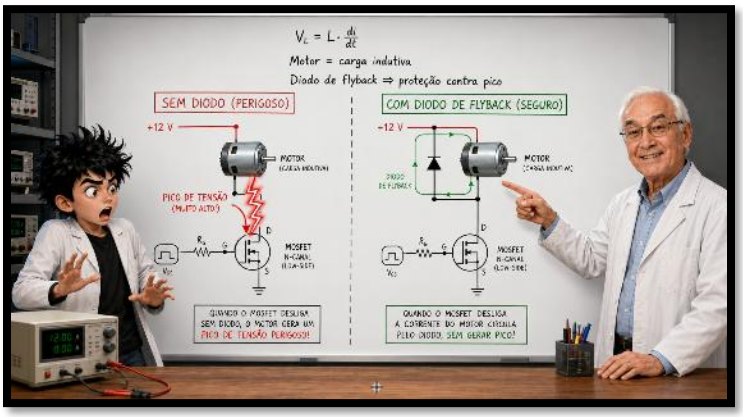
Quando o pino do Arduino muda rapidamente de zero para 5 volts, o gate inicialmente parece quase um curto momentâneo, puxando um pico rápido de corrente. Esse pico pode ultrapassar o limite seguro do pino do Arduino, mesmo que dure pouco tempo.

É por isso que colocamos um resistor entre o Arduino e o gate do MOSFET. Um valor comum para projetos simples é 220 ohms.

$$R_G = 220\Omega$$

Esse resistor limita o pico de corrente, reduz o estresse no pino do Arduino e ajuda a suavizar a comutação. Ele não está ali para “polarizar” o MOSFET como em um transistor bipolar. Ele está ali principalmente para controlar a corrente de carga e descarga da capacitância do gate.

1.6. Motor não perdoa: precisa de diodo de flyback



O terceiro cuidado é com os picos de tensão. Motores, relés, solenóides e bobinas são cargas indutivas. Isso significa que elas armazenam energia em um campo magnético enquanto a corrente circula.

Quando o MOSFET desliga, a corrente da bobina tenta continuar circulando. Como a corrente muda muito rapidamente, a tensão na indutância pode subir bastante, veja a equação a tensão na indutância é igual a indutância que multiplica a taxa de variação da corrente em relação a tensão é aquele di/dt da equação.

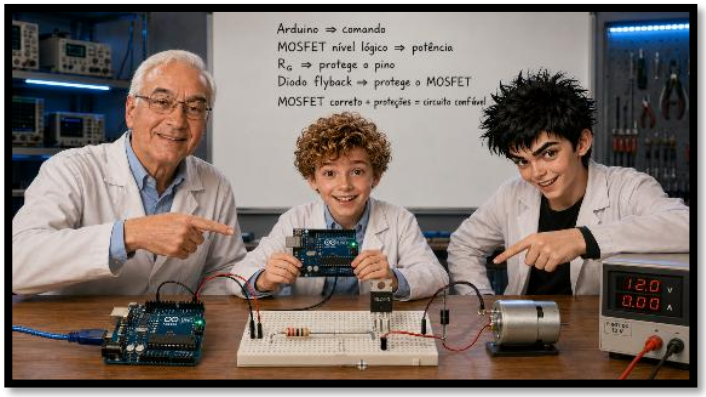
$$V_L = L \cdot \frac{di}{dt}$$

Esse pico de tensão pode chegar a dezenas ou centenas de volts, mesmo em circuitos alimentados por apenas 12 volts ou 5 Volts. E se não houver um caminho seguro para essa energia dissipar, o pico de tensão pode destruir o MOSFET e até o microcontrolador.

A solução clássica é colocar um diodo de flyback em paralelo com a carga indutiva, em português um diodo supressor. No motor DC, o diodo fica em paralelo com o motor, polarizado de forma reversa durante o funcionamento normal. Quando o MOSFET desliga, o diodo conduz a corrente de retorno e protege o transistor.

Então, para motor, relé ou solenóide, a regra é simples: MOSFET sem diodo supressor é estar pedindo para queimar.

1.7. Conclusão: MOSFET é ótimo, mas tem que respeitar as armadilhas



Então, o MOSFET é bom para Arduino? Sim. Na verdade, ele é uma das melhores opções para fazer o Arduino controlar cargas de maior potência. Mas ele precisa ser usado corretamente.

Primeiro: o Arduino não deve alimentar motor, solenóide ou fita de LED de potência diretamente. Ele deve apenas mandar o sinal de comando.

Segundo: o MOSFET precisa ser de nível lógico, para ligar corretamente com 5 volts ou 3,3 volts no gate. Não basta olhar a tensão de limiar. É preciso olhar no datasheet se a resistência em condução é baixa na tensão que o Arduino consegue fornecer.

Terceiro: o gate do MOSFET se comporta como um capacitor. Por isso, o resistor de gate ajuda a limitar o pico de corrente no pino do Arduino.

Quarto: se a carga for indutiva, como motor, relé ou solenóide, o diodo de flyback é obrigatório para proteger o MOSFET e o Arduino contra picos de tensão.

Negativino olha para o circuito funcionando e conclui:

Negativino:
“Então o MOSFET não queimou porque era ruim... queimou porque escolheram ele no escuro!”

E é exatamente essa a ideia. O MOSFET é excelente, mas não é magia é tecnologia.

Bom-proveito!