CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE SÃO PAULO
DEPARTAMENTO DE SISTEMAS ELETRÔNICOS
ELETRÔNICA INDUSTRIAL

GUILHERME GONÇALVES MIRANDA

TRAÇADOR DE CURVAS PARA TRANSISTORES NMOS

São Paulo
2023

GUILHERME GONÇALVES MIRANDA

TRAÇADOR DE CURVAS PARA TRANSISTORES NMOS

Trabalho de Conclusão de Curso de Graduação em Eletrônica Industrial da Faculdade de Tecnologia de São Paulo como requisito para a obtenção do título de Tecnólogo em Eletrônica Industrial.

Orientador: Prof. Dr. Roberto Katsuhiro Yamamoto

São Paulo

2023

Agradecimentos

Em primeiro lugar, eu agradeço a Deus por me conceder força, foco e fé na conclusão deste TCC.

Aos meus pais, avós e irmã por ter me apoiado sempre e me motivado independentemente da situação.

A minha futura esposa, atual namorada Beatriz por ter tido calma, compreensão e companheirismo.

Ao meu orientador Roberto Katsuhiro Yamamoto, que através de seu vasto conhecimento na área me orientou no TCC.

Aos meus amigos e colegas pela motivação e troca de informações pertinentes da área da eletrônica.

À FATEC-SP e aos professores, por ter moldado o meu perfil acadêmico e profissional buscando a resoluções de problemas não só teóricos como práticos.

"Conheça todas as teorias, domine todas as técnicas, mas ao tocar uma alma humana, seja apenas outra alma humana."

Carl Jung

RESUMO

Transistores de efeito de campo metal-óxido-semicondutor (MOSFET) são componentes semicondutores presentes em diversos circuitos: eletrônicos, analógicos e digitais. Para conhecer as características elétricas deste dispositivo com vasta aplicação, é utilizado um traçador de curvas. Equipamento este que permite traçar a curva de dispositivos eletrônicos, facilitando o conhecimento das características elétricas destes. O traçador de curvas é importante no mundo acadêmico para a prática de laboratório, permitindo comprovação da teoria vista em sala de aula de maneira mais otimizada em relação à prática manual. Pode ser utilizado no mundo corporativo para testagem de pequenas amostras de componentes. Um traçador de transistor MOSFET de baixa potência foi desenvolvido neste trabalho de conclusão de curso. O protótipo é constituído por um Arduino Uno responsável pelo sistema de controle, dois conversores DAC MCP4725 responsáveis pela rampa digital, um *buffer* para isolar o conversor do MOSFET e um medidor de corrente INA 219 de alta precisão, responsável por medir a corrente de dreno do dispositivo. Os dados adquiridos pelo microcontrolador são transmitidos à um *desktop* através da conexão USB e as curvas características são traçadas em um programa desenvolvido em linguagem Processing. Foram levantadas as curvas características dos transistores MOSFET BS170 e 2N7000 e comparadas às curvas ideais do Proteus. Os resultados mostraram o previsto pela teoria.

Palavras-chave: MOSFET, Arduino, curva característica, linguagem processing, MCP4725 amp-op.

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

1.1 Motivação

Para a confirmação da confiabilidade de componentes eletrônicos comprados por um fabricante, é recomendável usar procedimentos de testagem para pequenas amostras, uma maneira bastante usual é através do levantamento da curva característica do dispositivo o que permite saber a características elétricas do dispositivo [1]. Essa análise pode ser feita através de osciloscópio ou por equipamentos específicos para traçar curvas características**.**

Para efetuar a testagem, se feita de maneira manual por um técnico eletrônico, analisa os *datasheets* e parâmetros, acabando por consumir grande parte de seu tempo de trabalho, tornando-o monótono. Buscando a automatização deste processo de testagem e tendo em vista que equipamentos como a SMU (*source measure unit*) [1] estão custando alguns milhares de dólares, uma das motivações foi baratear esse tipo de equipamento como traçador de curvas.

Com este intuito, o projeto pode ser amplamente aplicado no ambiente industrial e ambiente acadêmico, a fim de testar amostras de transistores MOS através de suas curvas caraterísticas.

1.2 Objetivo

O objetivo deste trabalho de conclusão de curso é desenvolver um protótipo de um traçador de curvas para transistores MOSFET tipo-n de baixa potência, utilizando a plataforma Arduino como sistema de controle e a linguagem de programação Processing para o *software* gráfico para visualização das curvas no PC ou *notebook*.

O traçador de curvas é importante pois permite uma análise elétrica do dispositivo e verifica se o transistor está de acordo com as especificações do fabricante.

A fim de provar os conceitos apresentado durante o curso, especialmente na matéria de dispositivos semicondutores que apresentou o dispositivo MOSFET aos alunos.

1.3 Objetivo Específicos

O desenvolvimento do *hardware* e *software* do protótipo é estruturado:

1. Utilizar uma fonte de tensão de 5V/1,5A para alimentar o circuito.

2. Gerar rampa de tensão de Vds e Vgs utilizando dois módulos de conversor digital analógico o MCP4725.
3. Dimensionar um *buffer* de meia onda.
4. Medir a corrente de dreno e tensão Vds utilizando o INA219.
5. Desenvolver um programa em C/C++ para o Arduino Uno.
6. Desenvolver um programa gráfico no *software* Processing.

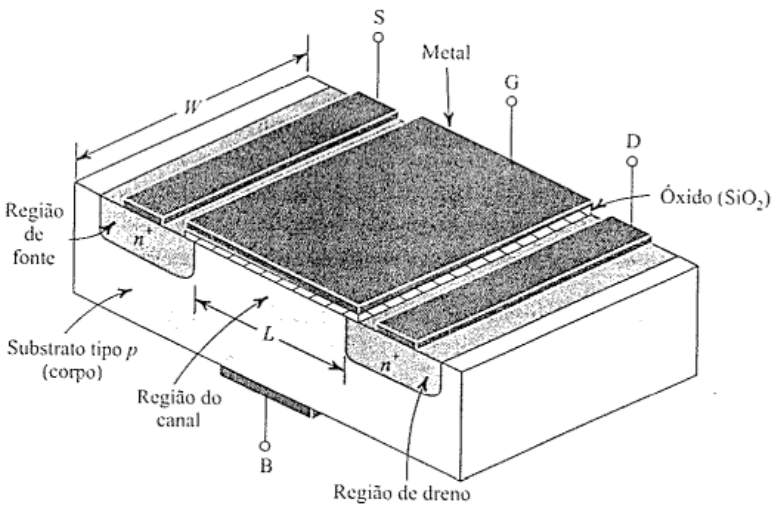
2. Fundamentação Teórica

Esse capítulo visa obter um conhecimento dos fundamentos básicos das características elétricas do NMOSFET tipo enriquecimento.

2.1 Introdução Teórica

Os transistores de efeito de campo MOSFET (transistor de efeito de campo metal-óxido-semicondutor) contêm normalmente três terminais para MOSFET discretos, sendo eles, porta (G), dreno (D), fonte (S) [2]. Apresenta uma fina camada de dióxido de silício entre a porta (G) e o substrato do dispositivo, acima da região do canal. Um metal é depositado sobre a estrutura para realizar os contatos com as regiões de porta, fonte e dreno, como mostra a Figura 1.

Figura 1 - Estrutura interna do NMOSFET tipo enriquecimento



Fonte: Sedra, A.S. (2022)

Seu controle é feito pela tensão de porta e a condução depende unicamente de um portador, elétrons para o canal n e lacunas para o canal p.

A tecnologia MOS (metal-óxido-semicondutor) está presente na eletrônica digital e analógica.

Por consequências das características elétricas e geométricas, o MOS tem alta impedância de entrada, mais estabilidade a diferentes temperaturas, consome menos energia, consome menos que o transistor bipolar de junção (TBJ) [3].

O processo VLSI (integração em larga escala) de CIs é o que permite o avanço tecnológico. O transistor MOS é o mais utilizado devido a facilidade de construção do dispositivo em pequenas dimensões.

Normalmente, para aplicações em circuitos integrados, o MOSFET tem quatro terminais pois o substrato não está conectado com a fonte.[2].

Encontram aplicações em eletrônica de potência, para acionamentos de potência como de motores, conversores de alta potência, e fonte de alta tensão, e em eletrônica digital.

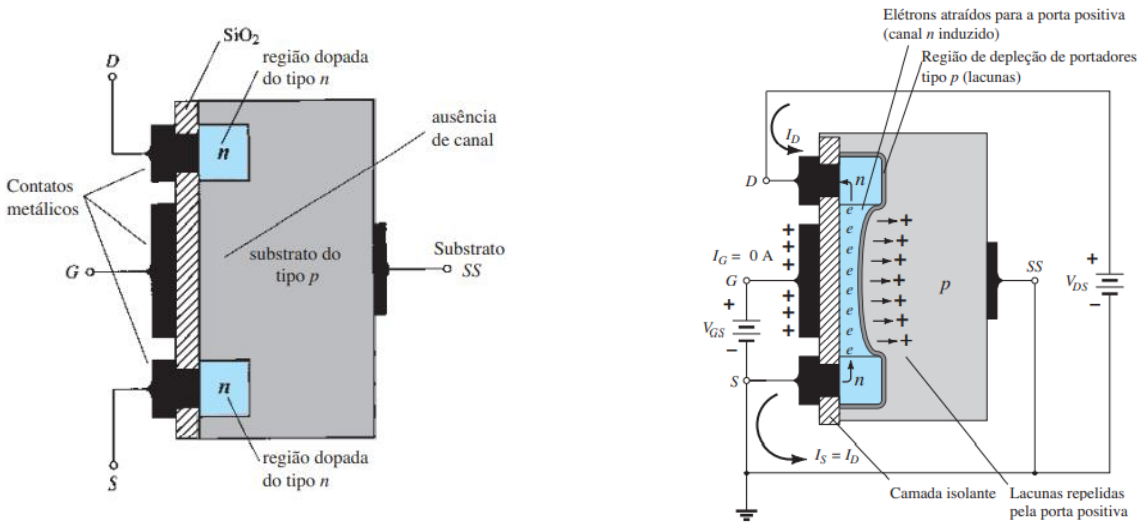
2.1.1 NMOSFET tipo enriquecimento

O dispositivo MOSFET tipo enriquecimento tem como o principal diferencial em relação ao tipo depleção, a falta do canal induzido no substrato entre dreno e fonte, funcionando como uma chave normalmente aberta.

Mesmo com a tensão entre dreno e fonte (Vds) polarizando o dispositivo, a corrente Id inicialmente será nula, até uma determinada tensão de limiar (Vt) positiva entre a porta e fonte (Vgs).

Com o aumento da tensão positiva aplicada na porta, as lacunas são repelidas e os elétrons são atraídos induzindo o canal n entre dreno e fonte. A Figura 2a mostra o NMOSFET sem aplicação de tensão na porta e a Figura 2b mostra a formação do canal n induzido pela aplicação de tensão na porta.

Figura 2 – (a) NMOSFET tipo n (b) Canal induzido



(a) (b)

Fonte: Boylestad, R. L. (2022)

A operação do transistor MOSFET, depende dos valores das tensões aplicadas em seus terminais. Na região de corte onde $V_{GS} \leq V_T$, o MOSFET estará praticamente desativado, não havendo idealmente condução de corrente entre dreno e fonte [6]. Na região de tríodo onde $V_{GS} > V_T$ e $V_{DS} < V_{GS} - V_T$, o transistor passa a conduzir corrente do dreno para a fonte a partir de uma determinada tensão Vgs e opera seguindo a lei de ohm, como um resistor.

Para calcular a corrente de dreno primeiro precisa calcular o parâmetro de transcondutância (K) informação presente no datasheet e depois a corrente de dreno, como mostra as Equações 1 e 2.

$$K = \frac{1}{2}\mu nCox\frac{W}{L} \quad (1)$$

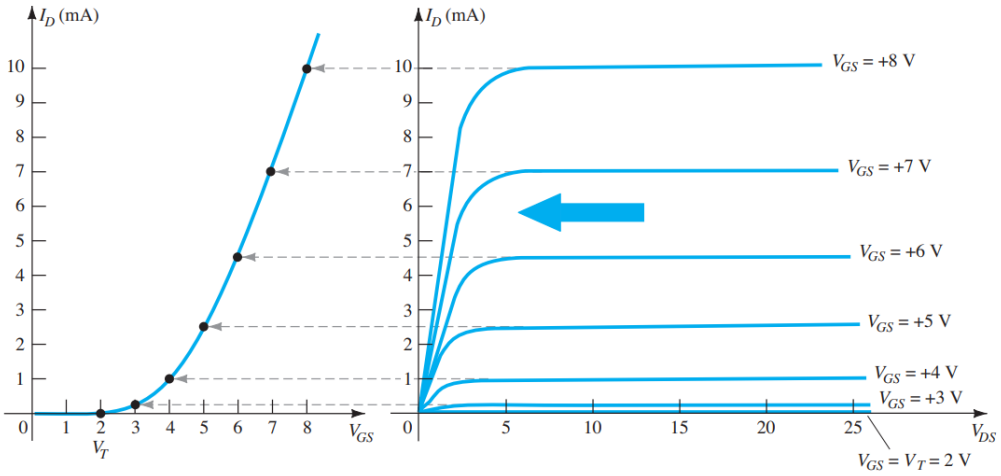
$$Id = K[2(\mathrm{Vgs} - \mathrm{Vt})\mathrm{Vds} - Vds^2\,] \quad (2)$$

Na região de saturação onde $V_{GS} > V_T$ e $V_{DS} > V_{GS} - V_T$, nesta região o MOSFET funcionará com sua capacidade máxima, a corrente de dreno ficará estabilizada, independente dos valores de Vgs. A corrente de dreno é dada pela Equação 3.

$$Id = K(Vgs - Vt)^2 \tag{3}$$

A Figura 3 mostra a curva de transferência I_D x V_{GS} e a curva característica I_D x V_{DS}.

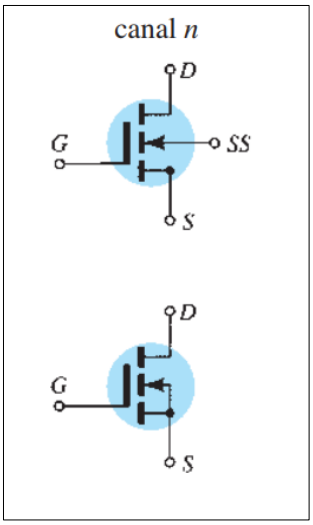
Figura 3 - Curvas de transferência e característica



Fonte: Boylestad, R. L. (2022)

A Figura 4 mostra as simbologias utilizadas para o NMOSFET.

Figura 4 – Simbologias do NMOSFET



Fonte: Boylestad, R. L. (2022)

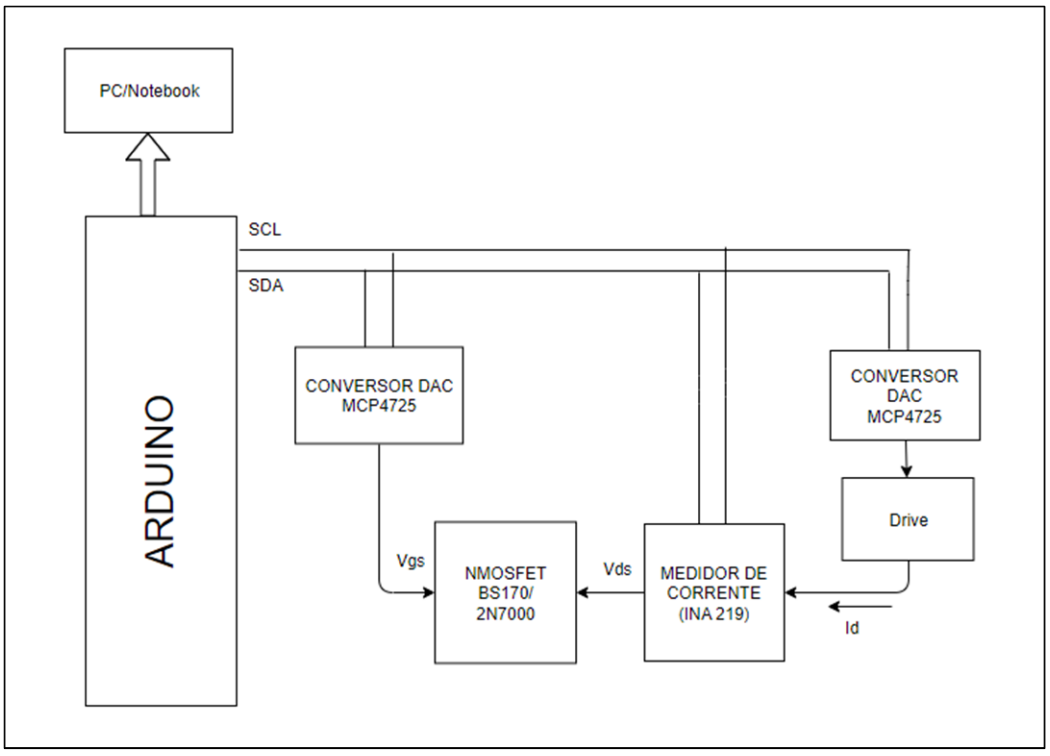
3. DESENVOLVIMENTO

Diante da fundamentação teórica do dispositivo descrita anteriormente, neste capítulo, será abordada a parte responsável pelo desenvolvimento do trabalho, sendo dividido em quatro partes: arquitetura geral do projeto, *hardware* do projeto, obtenção dos parâmetros e fluxogramas e *softwares* utilizados.

3.1 Arquitetura geral do projeto

O diagrama de blocos da Figura 5 ajuda no entendimento do projeto como um todo, principalmente no *hardware* a ser desenvolvido, análogo ao papel do fluxograma que facilita o entendimento do código de programação do projeto.

Figura 5 - Diagrama de blocos do projeto.



Fonte: Próprio autor

3.2 Sistema de Controle

3.2.1 Arduino

Para automatizar o levantamento da curva característica do NMOSFET, foram utilizados dois geradores de rampa para a variação da tensão Vds para cada valor de Vgs e efetuar a medição de Id. Esses sinais foram gerados pelo conversor DAC MCP4725 a partir da sequência binária proveniente do Arduino UNO R3 através da comunicação serial I2C. A medição de corrente foi feita pelo INA219.

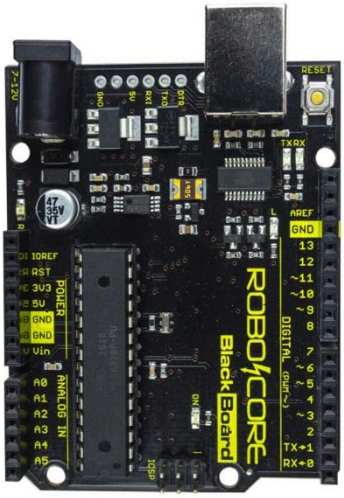
Para o envio das grandezas elétricas medidas utilizou-se a comunicação serial, com uma simples conexão USB 2.0 com o *notebook* ou PC, por fim, gerar o gráfico através de uma plataforma gráfica.

O Arduino é uma plataforma embarcada de fácil implementação criada no ano de 2005. A plataforma é livre, podendo o *hardware* ser alterado, personalizando ou modificado. Com essa filosofia *open-source* facilita o desenvolvimento de protótipos entre hobbystas, iniciantes e projetistas. Como consequência dessa facilidade, encontram-se diversos exemplos na internet e fóruns que ajudam a esclarecer dúvidas.

O microcontrolador utilizado pelo Arduino é um Atmel AVR, responsável por armazenar e executar a lógica de programação em linguagem C/C++ feita pelo usuário da plataforma [4].

Essa programação é feita na IDE do Arduino, um *software* da plataforma onde o usuário cria o seu código baseado em linguagem C/C++ e faz o *upload* para a placa de Arduino através da comunicação serial do cabo USB. A Figura 6 mostra um exemplo de placa de Arduino Uno.

Figura 6 - Arduino Uno



Fonte: https://produto.mercadolivre.com.br/

3.3 Comunicação I2C

O I2C (*Inter-Integrated Circuit*) é uma comunicação criada pela Philips nos anos 90. É uma comunicação serial síncrona que utiliza duas linhas de sinais bidirecional: SDA é responsável pelo envio de dados aos dispositivos conectados ao barramento e SCL é responsável por efetuar o sincronismo entre os dispositivos [5].

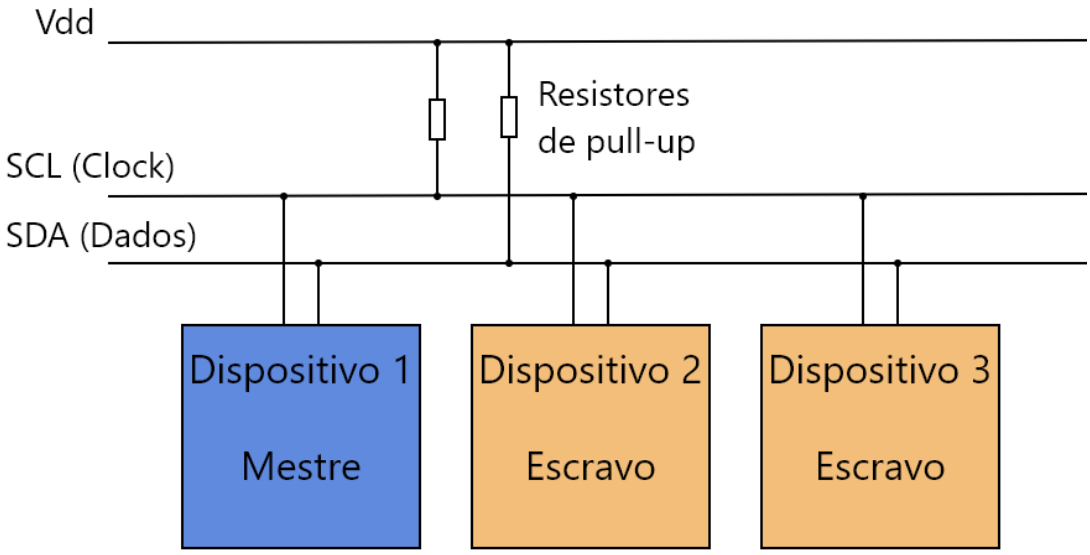
Tem como a principal vantagem, estabelecer a comunicação entre diversos módulos utilizando apenas dois pinos do Arduino A4 pino referente ao SDA e A5 referente ao SCL. O Arduino é o mestre e o demais dispositivos escravos.

Outras vantagens:

- Taxas flexíveis de transmissão de dados
- Cada dispositivo do barramento tem seu endereço próprio
- Os dispositivos estabelecem uma comunicação simples Mestre/Escravo

Para a comunicação entre os módulos do circuito, é necessário configurar o endereço independente de cada dispositivo, como ilustra a Figura 7. Estes endereços se encontram na folha de dados do dispositivo.

Figura 7 - Dispositivos conectados ao barramento I2C.



Fonte: https://mundoprojetado.com.br/i2c

Inicialmente, os pinos do barramento SCL e SDA começam em nível alto, por causa do resistor de *pull-up*, norma do próprio protocolo. A comunicação no barramento é iniciada

após o SDA comutar para nível baixo e, após um tempo, o SCL comuta também para nível baixo e, para encerrar a comunicação, o SCL comuta para nível alto e, após um certo tempo, o SDA também comuta para nível alto. Esses dois sinais não podem comutar ao mesmo tempo, como mostra a Figura 8.

Figura 8 - Mostra o início e o fim da comunicação.

Fonte: Adaptado de <https://unelectronica.github.io/I2C_esp8266/>

O limite de dados é de 16 bits e o cabo não pode ultrapassar alguns metros, porque o fio pode ter uma capacitância máxima de 400 pF.

3.4 Conversor digital analógico D/A

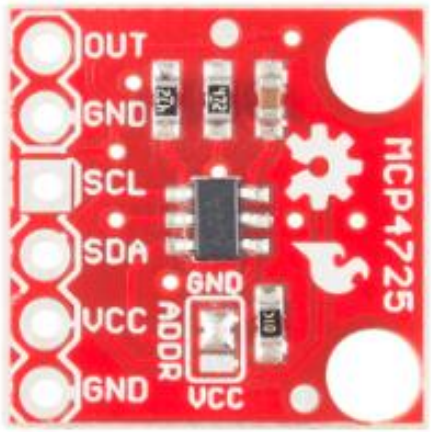
Os sinais digitais assumem valores discretos e limitados pela resolução do dispositivo, enquanto na resolução analógica, entre um valor e outro, existem infinitos valores.

O conversor D/A tem por finalidade, a partir de um sinal com valores estabelecidos em base binária, associar esses valores a faixa de valores analógicos e efetuar essa conversão.

3.4.1 MCP4725

Para traçar a curva característica são necessários diferentes níveis de tensão de Vds para cada valor fixo de Vgs, para obter estes níveis de tensão foram utilizados dois módulos MCP4725 (Figura 9) com DAC (conversor digital-analógico) com uma resolução de 12 bits para gerar um sinal de rampa digital [7]. O sinal digital é enviado via comunicação serial I2C e o módulo faz a conversão destes códigos binários para valores de tensão.

Figura 9 - Módulo do conversor DA



Fonte: https://www.sparkfun.com/products/12918

Pinagem

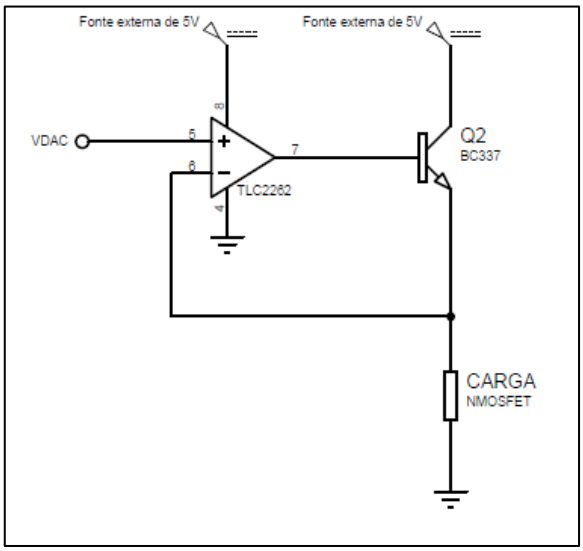
- Out - Saída do sinal
- 2 x Gnd - Terra de referência da alimentação do módulo e do sinal de saída
- SCL - Pino de *clock* da comunicação I2C
- SDA - Pino de dados da comunicação I2C
- VCC- Pino de alimentação do módulo (2,7V – 5V)

3.5 *Buffer* de sinal positivo

Após a realização de testes com o módulo DAC, conectado direto na carga (NMOSFET), o sinal da tensão responsável por polarizar Vds, à medida que o NMOSFET conduz, quando a tensão Vgs atingia Vt, o sinal de saída do DAC caía pela metade, pois o dispositivo estava drenando mais corrente do que o DAC suportava.

Foi necessário adicionar um *buffer* (Figura 10) de sinal positivo [10] entre o DAC responsável pela variação de Vds e o NMOSFET para poder suprir a corrente Id que passa pelo mesmo, pois o DAC tem uma corrente máxima de apenas 25 mA na saída.

Figura 10 - Esquema elétrico do *buffer*.



Fonte: Próprio Autor

Para evitar perdas de tensão no sinal da rampa digital, foi utilizado um amplificador do tipo *rail-to-rail*, o TLC2262[11]. Esse amplificador tem mais eficiência e não precisa de uma tensão simétrica. Foi utilizada uma tensão assimétrica para sua alimentação, com 5 V conectado em sua alimentação feita pelo pino 8 e o GND conectado no pino 4.

O sinal de rampa digital passa pelo seguidor de tensão que isola o DAC do NMOSFET. O transistor será responsável por suprir a corrente, como a corrente do emissor é praticamente a corrente do coletor, sendo assim, a fonte DC externa será responsável por fornecer essa corrente necessária [6].

3.6 Alimentação do protótipo

O Arduino pode ser alimentado de duas formas, como mostra a Figura 11. Por uma fonte externa através de um conector *jack*, é recomendado uma tensão entre 7V até 12V [4]. A outra maneira é uma alimentação simples através de uma conexão USB, a qual foi utilizada neste protótipo.

Figura 11 - Alimentação Arduino Uno

Fonte: Adaptado de https://produto.mercadolivre.com.br/

Para a alimentação dos módulos foi necessária a utilização de uma fonte de alimentação externa. Foi utilizada uma fonte DC chaveada gradeada de 5,6V/1,3A, mostrada na Figura 12. O necessário para suprir a necessidade do protótipo.

O GND do Arduino e da fonte externa foram conectados para terem o mesmo referencial.

Figura 12 - Fonte DC de 5V/1,2A.

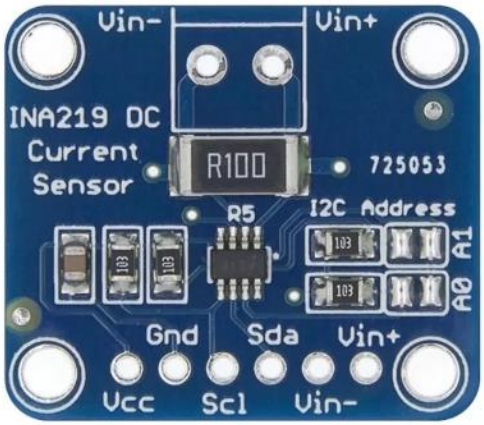
Fonte: Próprio autor

3.7 Medidor de corrente

Para traçar a curva característica é necessário realizar a medição da corrente Id do transistor MOSFET de canal N. Para medir uma corrente na escala de miliampères, foi utilizado um medidor de corrente de alta precisão, o módulo INA 219.

A comunicação é feita através da comunicação serial I2C, através dos pinos SCL e SDA, como mostra a Figura 13.

Figura 13 - Sensor de corrente INA219.



Fonte: https://produto.mercadolivre.com.br/

Para a medição da corrente é necessário colocar o módulo em série com a carga para a corrente passar pelo módulo e ser feita a medição. O método para efetuar a medição é o *high-side* [10], um resistor *shunt* é colocado em série antes da carga e através da queda de tensão sobre o resistor *shunt* determina-se a corrente que passa pelo dispositivo. O resistor *shunt* é de 100 mΩ, e normalmente utiliza-se uma resistência baixa para não interferir na resistência total.

O INA 219 também está sendo usado para medir a tensão de alimentação no barramento, para obter a tensão Vds.

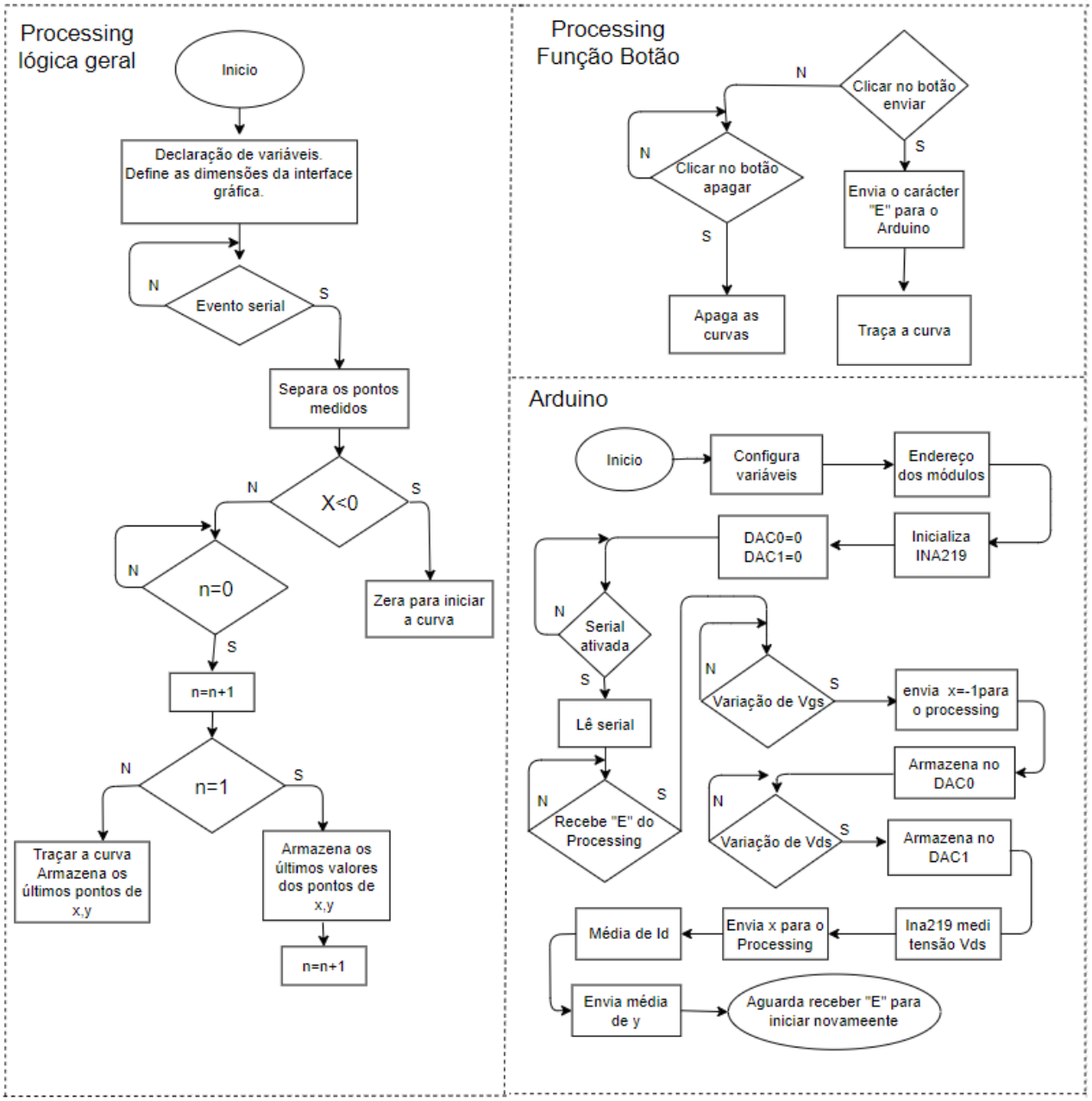
O módulo mede tensões de até 26 Vdc e corrente máxima de 3,4 A. Pode ser alimentado com 3,3 V a 5 V. Neste protótipo, o módulo foi alimentado por uma fonte chaveada externa de 5V/1,2A.

Caso necessário o endereço utilizado para reconhecer o dispositivo no barramento I2C, pode ser modificado, fazendo um *jumper* em A1 e ou A0, podendo assumir quatro endereços, se for necessária a utilização de 4 dispositivos.

3.8 Fluxograma

O fluxograma da Figura 14 ajuda na descrição do funcionamento geral da programação feita neste protótipo.

Figura 14 - Fluxograma geral do projeto.



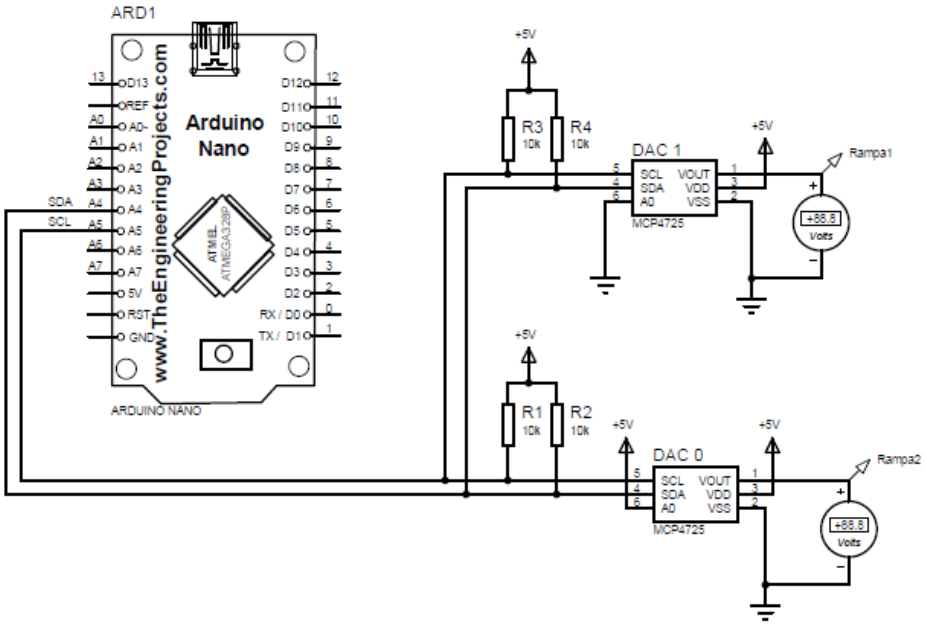
Fonte: Próprio Autor

4. RESULTADOS E DISCUSSÕES

4.1 Gerador de sinais

Para obter a curva característica do transistor foi feita a variação em rampa de tensão (dente de serra) em Vds para cada valor fixo de Vgs. Foram utilizados dois conversores DACs MCP4725 de 12 bits, tendo uma resolução de 4095, como mostra a Figura 15.

Figura 15 - Gerador de rampa.



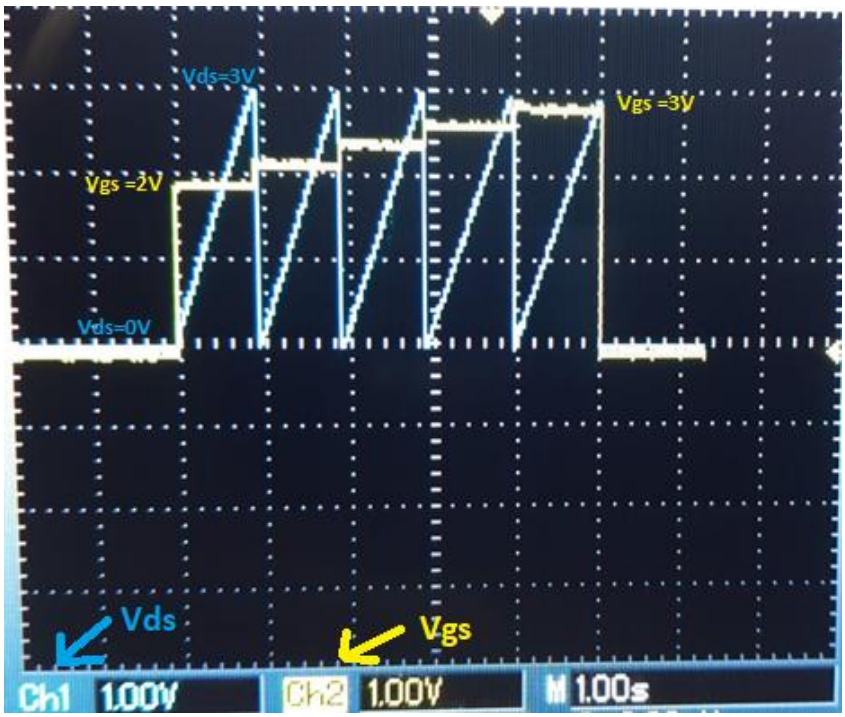
Fonte: Próprio Autor

A comunicação serial I2C é feita através de sequências digitais de 12 bits pelos pinos A4 (transmissão de dados) e A5 (sincronismo).

Para o Arduino reconhecer os dois módulos de forma simultânea são necessários endereços diferentes. Neste projeto foram utilizados A0 (0x60) e A0 (0x61).

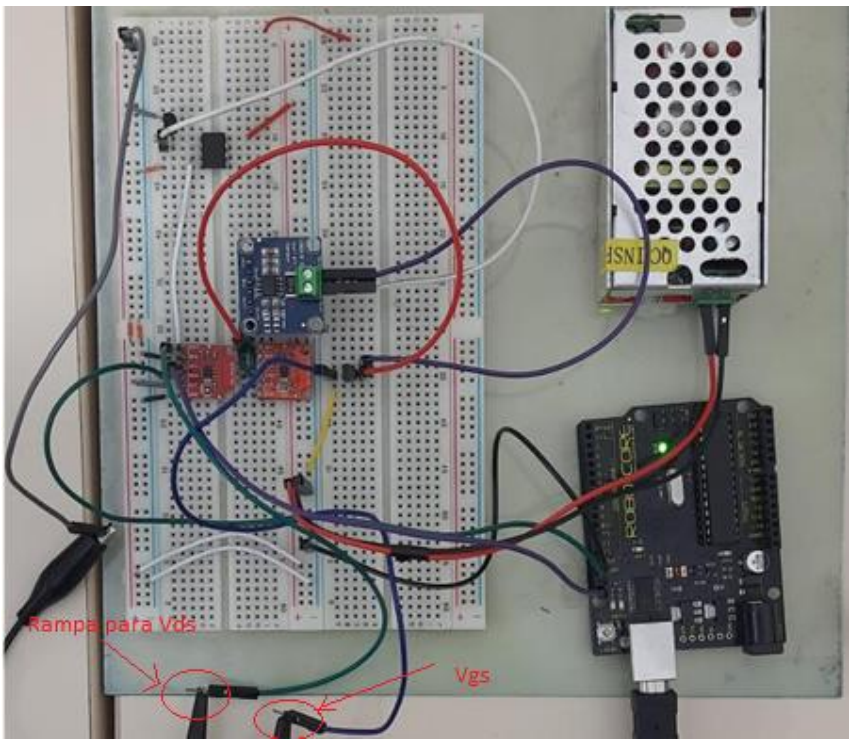
A Figura 16 mostra como é feito o sincronismo do sinal. Para a obtenção deste sinal foi feita a programação com base na biblioteca Adafruit que disponibiliza as funções básicas e um exemplo de código de um sinal triangular [8]. A Figura 17 mostra o ponto medido no protótipo.

Figura 16 – Formas de onda de Vgs e Vds.



Fonte: Próprio Autor

Figura 17 - Pontos de medição de Vgs e Vds.



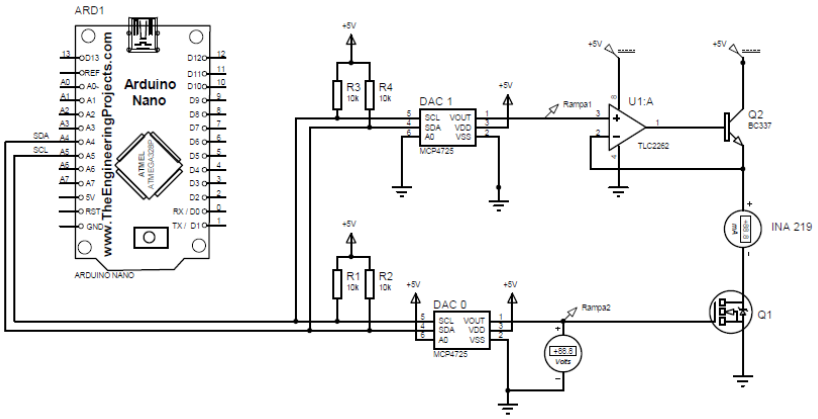
Fonte: Próprio Autor

4.2 Hardware

Todo o *hardware* da Figura18 foi desenvolvido no Proteus antes de ser testado na bancada.

Descrição do circuito: o Arduino é responsável por enviar os dados digitais para os dois DACs (conversor analógico digital) que fazem a rampa digital como mostrado na Figura 16. O *buffer* faz a isolação entre o DAC 1 e o MOSFET.

Figura 18 - Esquema elétrico do *hardware*.



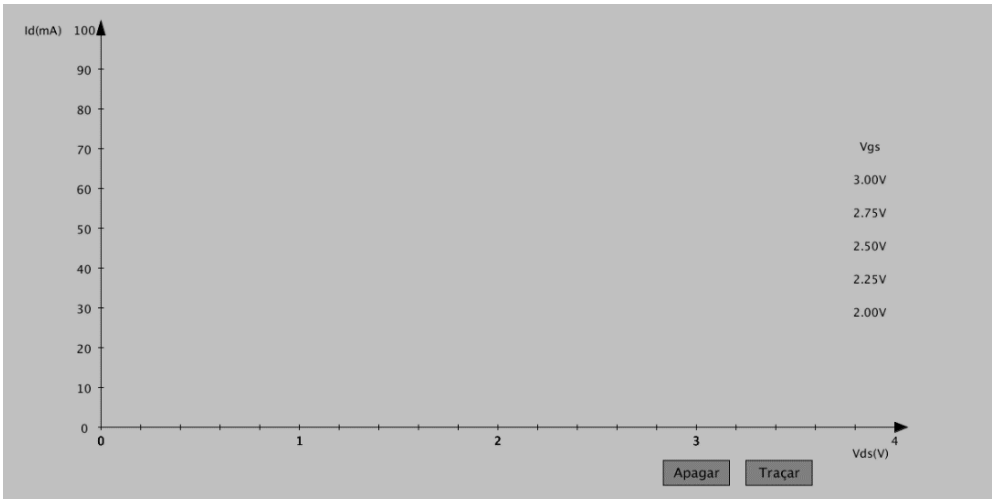
Fonte: Próprio Autor

4.3 Interface Gráfica

Para criar a interface gráfica, primeiro, é necessário criar uma janela feita com os dimensionamentos escolhendo a altura e largura da janela desejada. Com base nesses valores são dimensionados os eixos do gráfico de acordo com a escalas das medidas.

Para a interface, mostrada na Figura 19, ficar intuitivo, foi inserido dois botões. O botão “Traçar”, responsável por iniciar a comunicação com o Arduino e traçar a curva e o botão “Apagar” que, uma vez pressionado, apaga todas as curvas mostradas, para reiniciar o traçado de uma nova curva.

Figura 19 - Interface gráfica feita no Processing.



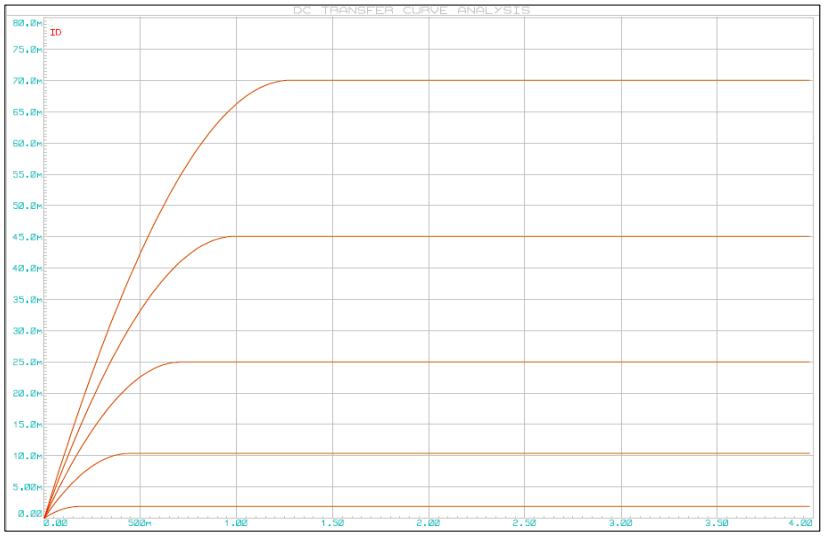
Fonte: Próprio Autor

4.4 Curva característica

Para validação das medidas utilizou-se o método comparativo. No primeiro momento buscou-se utilizar o *datasheet* do próprio fabricante para fazer a comparação com a curva característica do dispositivo, porém o *datasheet* do NMOSFET BS170 (Anexo A) e 2N700 (Anexo B) mostra a curva característica obtida com um sinal pulsado de Vgs menor ou igual a 300 µs e um ciclo de trabalho menor ou igual a 2% como mostram as Figuras A-1 e B-1 dos Anexos. Portanto, os dados obtidos neste protótipo não podem ser comparados diretamente com o manual do fabricante do dispositivo.

Por esse motivo, foi escolhido comparar a curva traçada pelo protótipo desenvolvido com a curva do dispositivo obtida no *software* de simulação Proteus 8.13 (Figura 20) que teoricamente simula os valores ideais previsto no *datasheet* dos dispositivos.

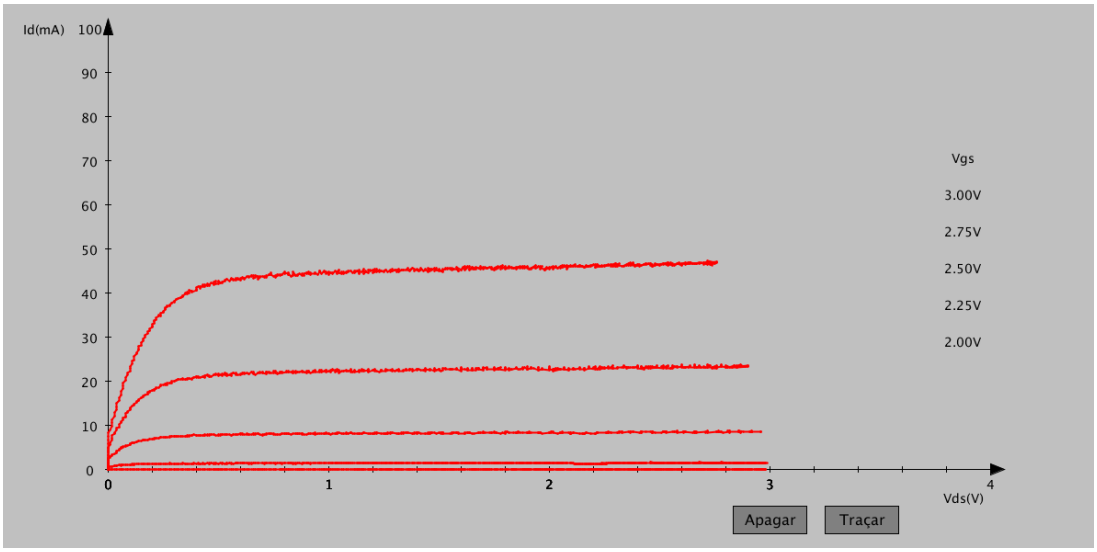
Figura 20 - Curva caraterística do BS170 traçada no *software* Proteus.



Fonte: Próprio Autor

Nota-se que na primeira curva, para um valor abaixo da tensão de limiar Vt até 2 V, idealmente com esse valor de tensão de acordo com o *datasheet*, o transitor praticamente não conduz. Por isso a corrente fica praticamente nula, como mostra a Figura 21. Após a tensão Vt, o transistor começa a conduzir.

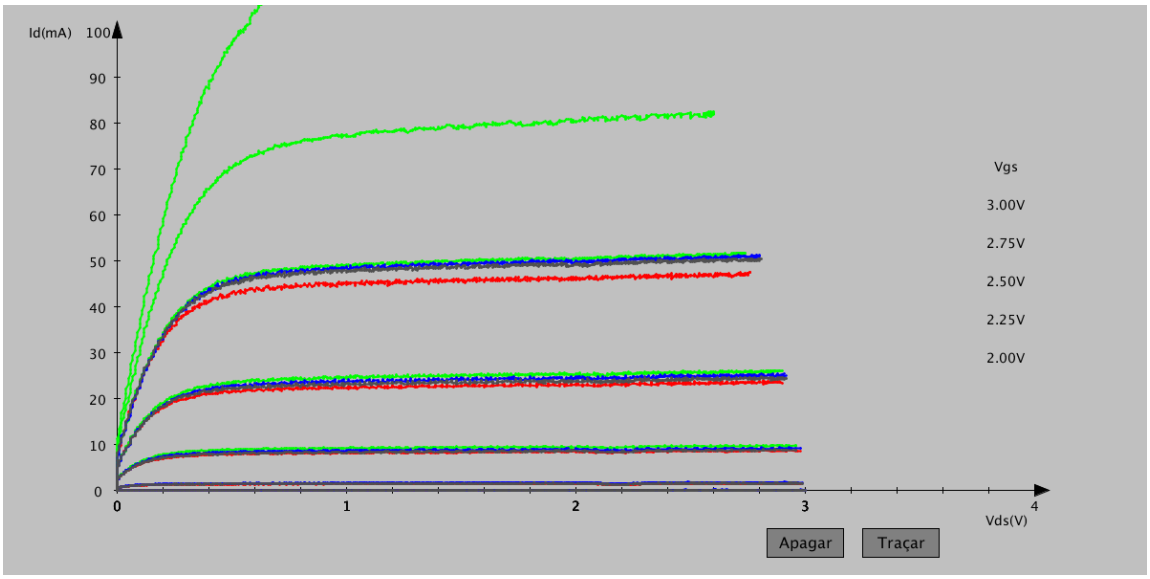
Figura 21 – Curva característica do BS170 traçada pelo protótipo



Fonte: Próprio Autor

Para demonstrar que o mesmo componente eletrônico do mesmo código e fabricante difere entre si, tendo por consequência outras caracteristicas elétricas, foram obtidas as curvas caracteristicas de quatro transistores BS170 do mesmo código e fabricante (Figura 22).

Figura 22 - Comparação entre quatro BS170 do mesmo código e fabricante.

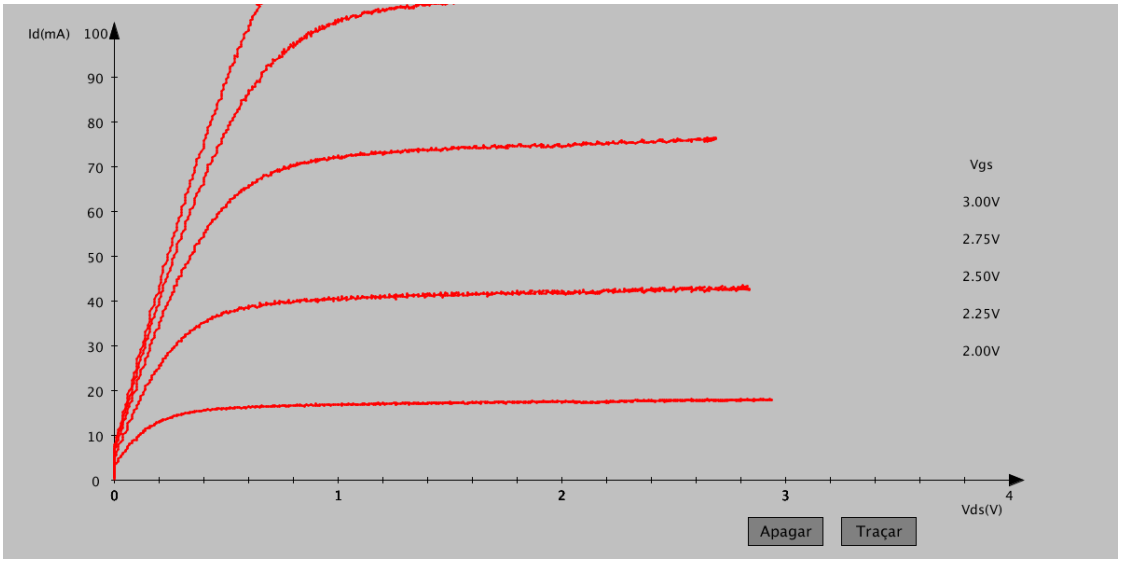


Fonte: Próprio Autor

O datasheet informa que, assim como o BS170, o 2N7000 começa a conduzir com uma tensão de limiar (Vt) maior que 2V.

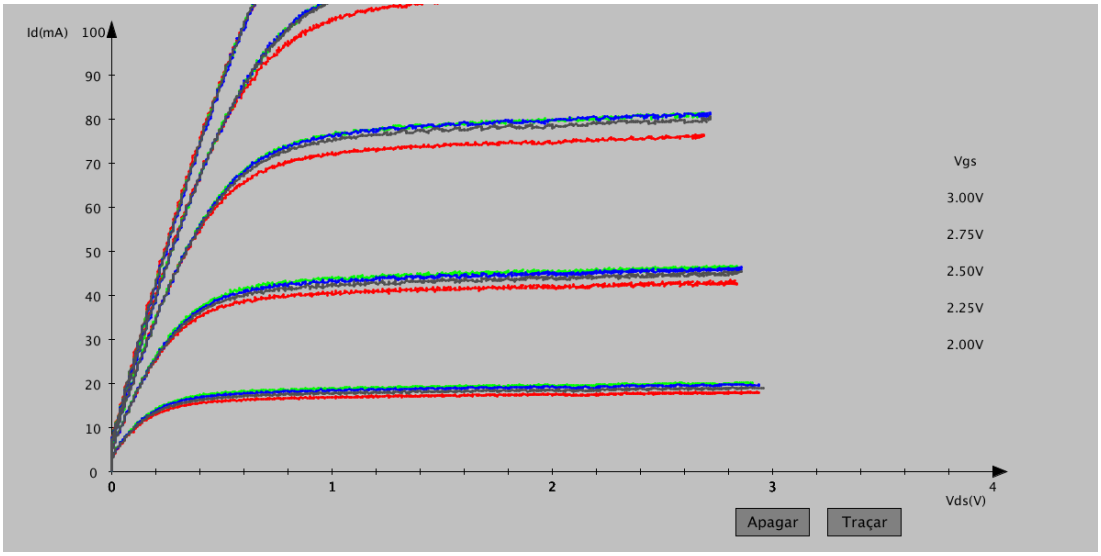
As duas últimas curvas não estão de acordo com a dimensão da interface gráfica, por este motivo não aparecem no gráfico da Figura 23. A Figura 24 mostra as curvas características de quatro transistores 2N7000 do mesmo código e fabricante.

Figura 23 - Curva característica do 2N7000 traçada pelo protótipo.



Fonte: Próprio Autor

Figura 24 - Comparação entre quatro 2N7000 do mesmo código e fabricante.

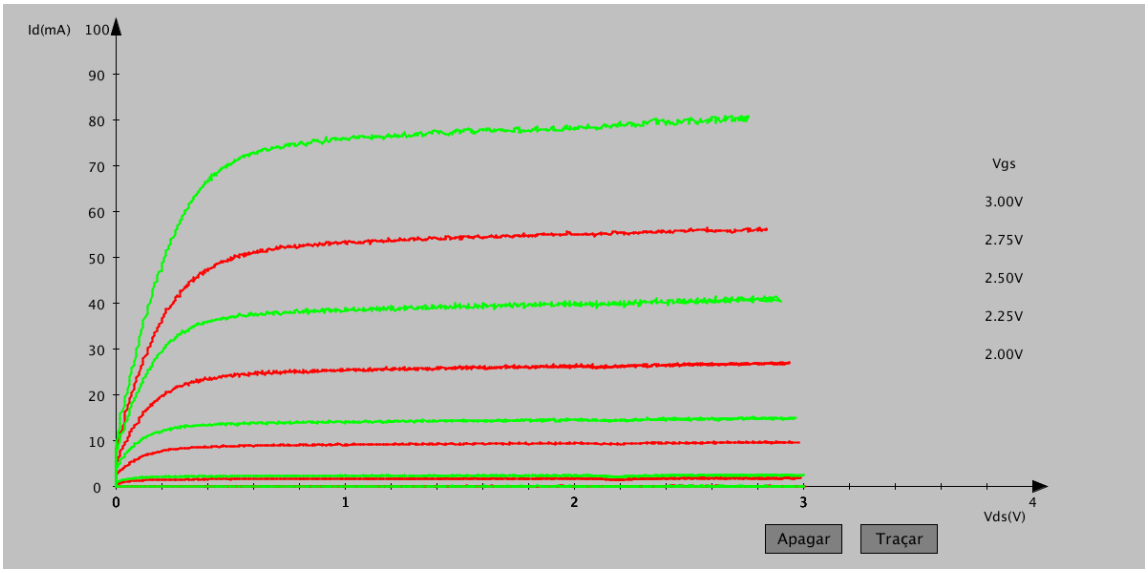


Fonte: Próprio Autor

As curvas traçadas anteriormente evidenciaram uma leve divergência. Esta divergência entre as medidas é um ponto esperado, pois tratam-se de experiências ideais e empíricas, que ainda assim apresentaram o mesmo comportamento.

Para demonstrar a diferença prática entre fabricantes do mesmo componente, as curvas do dispositivo BS170 da Farchild foram traçadas com as de outro fabricante, mostradas na Figura 25.

Figura 25 – Transistor BS170 da Farchild em verde e de outro fabricante em vermelho.

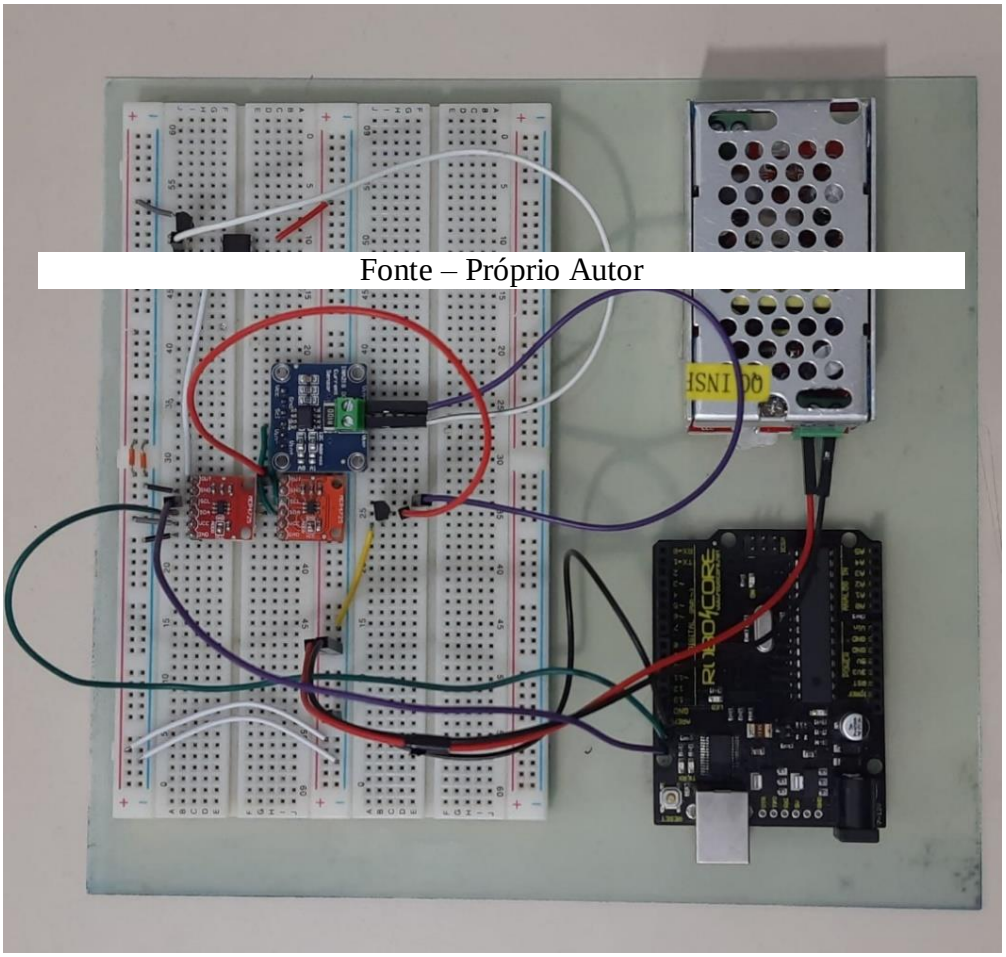


Fonte: Próprio Autor

4.5 Protótipo

A *protoboard* com os três módulos e o *buffer* do protótipo, o Arduino Uno e a fonte chaveada de 5V/1,2A foi posta sobre uma base isolante de fibra de vidro (Figura 26) para facilitar no manuseio e evitar desconexões dos fios ou até mesmo mau contato.

Figura 26 - Protótipo montado.



Fonte – Próprio Autor

5. CONCLUSÃO

O desenvolvimento deste trabalho de conclusão de curso comprovou conceitos básicos sobre as características de transistores MOS apresentados teoricamente na disciplina de Dispositivos Semicondutores e Eletrônica 1. Possibilitou a construção de um protótipo que replica as funcionalidades de um traçador de curvas, equipamento que extrai curvas características de transistores. Neste caso, o trabalho se restringe a efetuar a leitura de transistores MOSFET. Em comparação a um produto comercial, tem como vantagem o seu baixo custo.

O protótipo é constituído por um Arduino Uno responsável pelo sistema de controle que envia sinais digitais para os dois conversores DAC MCP4725, responsáveis pela rampa digital para o Vgs e Vds do transistor, um *buffer* para isolar o conversor do MOSFET e um medidor de corrente INA 219 de alta precisão conectado em série com o circuito, responsável por medir a corrente de dreno do dispositivo.

Os dados adquiridos pelo microcontrolador são transmitidos a um *desktop* através da conexão USB e as curvas características são traçadas em um programa desenvolvido em linguagem Processing, responsável pela interface gráfica.

A montagem de todo o protótipo foi feita sobre uma base isolante de fibra de vidro para facilitar o manuseio. As conexões do circuito foram feitas em uma *protoboard*.

As curvas características dos transistores MOSFET de BS170 e 2N7000 foram traçadas e comparadas às curvas ideais do Proteus, o que evidenciou uma leve divergência. Esta divergência entre as medidas é um ponto esperado, pois trata-se de experiências ideais e empíricas, que ainda assim apresentaram o mesmo comportamento.

Para evidenciar a divergência de características elétricas entre componentes eletrônicos do mesmo código e fabricante, foram traçadas as curvas de 4 componentes, submetidos à mesma rampa de tensão. Os resultados mostraram o previsto pela teoria.

5.1 Sugestão de trabalhos futuros

- Implementar outros transistores PMOS, bipolar PNP, NPN.

- Colocar o protótipo dentro de uma caixinha. Para o projeto ficar mais compacto é possível trocar o Arduino Uno para o Arduino NANO e trocar a fonte externa para uma bateria. E, traçar as curvas em um *display*, visando um equipamento portátil.

REFERÊNCIAS BIBLIOGRÁFICAS

1 - BRAGA, N. C. **Traçador de Curvas (INS573)**. Disponível em: <https://www.newtoncbraga.com.br/index.php/instrumentacao/108-artigos-diversos/9297-tracador-de-curvas-ins573>. Acesso em: 17/03/2023.

2 - **Mosfet: Funcionamento, Tipos e Aplicações**. Disponível em: <https://flaviobabos.com.br/mosfet/>. Acesso: 20/02/2023.

3 - ALBERT PAUL MALVINO. **Eletrônica**. São Paulo: Pearson Makron Books, 2007.

4 - SOUZA, F. **Arduino UNO**. Disponível em: <https://embarcados.com.br/arduino-uno/>. Acesso em: 05/04/2023.

5 - **I2C - Mundo Projetado**. Disponível em:<https://mundoprojetado.com.br/i2c/>. Acesso em: 03/05/2023.

6 - BOYLESTAD, R. L.; NASHELSKY, L. **Dispositivos eletrônicos e teoria de circuitos**. São Paulo: Pearson Education Do Brasil, 11ª. edição, 2013.

7 - **MCP4725 pdf, MCP4725 Description, MCP4725 Datasheet, MCP4725 view ALL DATASHEET**. Disponível em: <https://pdf1.alldatasheet.com/datasheet-pdf/view/233449/MICROCHIP/MCP4725.html>. Acesso em: 17/03/2023.

8 - **Adafruit_MCP4725**. Disponível em: <https://github.com/adafruit/Adafruit_MCP4725>. Acesso em: 17/03/ 2023.

9 - SACCO, F. **Buffers e Seguidores de Tensão**. Disponível em: <https://embarcados.com.br/buffers-e-seguidores-de-tensao/>. Acesso:16/02/2023

10 - LOCATELLI, C. **INA219 como utilizar o módulo**. Disponível em: <https://curtocircuito.com.br/blog/Categoria%20Arduino/ina219-como-usar>. Acesso em: 17 maio. 2023.

11 - **BS170 pdf, BS170 Description, BS170 Datasheet, BS170 view ALLDATASHEET.** Disponível em: <https://pdf1.alldatasheet.com/datasheet-pdf/view/50804/FAIRCHILD/BS170.html>. Acesso em: 16/03/ 2023.

12 - **TLC2262CP pdf, TLC2262CP Description, TLC2262CP Datasheet, TLC2262CP view ALLDATASHEET**. Disponível em: <https://pdf1.alldatasheet.com/datasheet-pdf/view/184073/TI/TLC2262CP.html>. Acesso em: 22/02/ 2023.

13 - **2N7000 pdf, 2N7000 Description, 2N7000 Datasheet, 2N7000 view ALLDATA SHEET** Disponível em: <https://pdf1.alldatasheet.com/datasheet-pdf/view/50067/FAIRCHILD/2N7000.html>. Acesso em: 12/06/2023.

APÊNDICE A – Código do Arduino

```
#include <Wire.h>
#include <Adafruit_INA219.h>
#include <Adafruit_MCP4725.h>
Adafruit_INA219 ina219;
Adafruit_MCP4725 dac0;
Adafruit_MCP4725 dac1;
char comando;
int n=0,counter2;
int i=0;
float media,soma;
float current_mA = 0;
float Busvoltage = 0;
float Vds =0;
float Vgs=0;

void setup(void)
{
  Serial.begin(9600);
  Serial.flush();

  while (!Serial)
   {
       delay(1);
   }
  dac0.begin(0x60);//A0 conectado ao GND
  dac1.begin(0x61);//A0 conectado ao vcc
  if (! ina219.begin())
     { // inicializa o INA219.
         while (1)
          {
           delay(10);
          }
     }
  ina219.setCalibration_16V_400mA();
  //seta o medidor INA219 para alta precisão
}

void loop(void)
{
  uint32_t counter;
  uint32_t counter1;
  float current_mA=0;
  float Busvoltage=0;
  float Vds=0;
  float Vgs=0;
  counter1=0;
  dac1.setVoltage(counter1, false);
  counter=0;
```

```
dac0.setVoltage(counter, false);
if(Serial.available()>0)
 {
  comando= Serial.read();
  if(comando=='E')
   {
     for (counter=1638; counter<=2458; counter+=820/4)
     // variação de VGS = 2V, 2.25V, 2.5V, 2.75V e 3V
     {
      dac0.setVoltage(counter, false);
       Serial.print(-1.0); Serial.print(",");
       Vgs= float(2+(counter-1638)/820.0);
       Serial.print(Vgs);
       Serial.println(",");
      for (counter1=0; counter1<=2458; counter1+=4)
      //variação de VDS
        {
         dac1.setVoltage(counter1, false);
         Busvoltage = ina219.getBusVoltage_V();
         Serial.print( Busvoltage );
         Serial.print(",");
         soma=0;
         for(i=0;i<10;i++)
           {
            current_mA = ina219.getCurrent_mA();
            soma += current_mA;
           }
         media = soma/10;
         Serial.print(media);
         Serial.println(",");
      }
    }
  }
 }
}
```

Fonte: Modificado pelo Autor [8]

APÊNDICE B – Código da interface em Processing

```
import processing.serial.*;
Serial port;
int LARG_TELA=1200,
    ALT_TELA=600,
    X0_BOTAO=900,
    Y0_BOTAO=550,
    LARG_BOTAO=80,
    ALT_BOTAO=30,
    X1_BOTAO=800,
    Y1_BOTAO=550,
    LARG_BOTAO1=80,
    ALT_BOTAO1=30,
    X0_GRAF=int(0.1*LARG_TELA),//10% da largura de tela
    Y0_GRAF=int(0.85*ALT_TELA),//85% da altura de tela
    LARG_GRAF=int(0.8*LARG_TELA),//80% da largura ddo gráfico
    ALT_GRAF=int(0.8*ALT_TELA);//80% altura do grafico
float x,y,Xm,Ym,Xm0,Ym0,soma,Vgs, Ult_x,Ult_y;
int n, A, Npontos=10, n_curva=0, n_cor=0;

void setup()
{
  size(LARG_TELA,ALT_TELA); // janela do aplicativo
  port = new Serial(this,"COM7",9600);//porta serial a ser lida
  port.bufferUntil('\n');//faz a leitura até o fim de linha
  TelaGrafica();// função tela gráfica
}

void draw()
{
}

void serialEvent(Serial port)//função da serial
{
  String texto;//sequencia de caracteres
  texto=port.readStringUntil('\n');//leitura até o fim da linha
  String[] dados=split(texto,',');
  //separa os valores ([0],[1]) ponto a ponto
  x=float(dados[0]);// valor de tensão Vds medido pelo ina219
  y=float(dados[1]);// valor de corrente Id medida pelo INA 219
  if(x<0)//aviso do arduino a cada curva
    n=0;
  else// senão mostra apenas os pontos
   {
     if(n==0)//depraza o primeiro valor
          n++;
        else if(n==1)// despreza o último ponto
```

```
          n++;
        else
         {
          line(X0_GRAF+int(map(Ult_x,0,4,0,LARG_GRAF)),
               Y0_GRAF-int(map (Ult_y,0,100,0,ALT_GRAF)),
               X0_GRAF+int(map(x,0,4,0,LARG_GRAF)),
               Y0_GRAF-int(map(y,0,100,0,ALT_GRAF)));
          Ult_x = x; // armazena o último valor de x
          Ult_y = y; // armazena o último valor de y
          strokeWeight(2);//define a largura dos pontos
         }
     }
}

void mouseMoved()
{
  if(mouseX>=X0_BOTAO && mouseX<=X0_BOTAO+LARG_BOTAO &&
     mouseY>=Y0_BOTAO && mouseY<=Y0_BOTAO+ALT_BOTAO)
    cursor(HAND);
  else if(mouseX>=X1_BOTAO && mouseX<=X1_BOTAO+LARG_BOTAO1 &&
          mouseY>=Y1_BOTAO && mouseY<=Y1_BOTAO+ALT_BOTAO1)
         cursor(HAND);
       else
          cursor(ARROW);
}

void mouseClicked()
{
  if(mouseX>=X0_BOTAO && mouseX<=X0_BOTAO+LARG_BOTAO &&
     mouseY>=Y0_BOTAO && mouseY<=Y0_BOTAO+ALT_BOTAO)
    {
      port.write('E');
      //altera a cor da curva depois de cinco curvas traçadas
      switch(n_cor)
       {
         case 0: stroke(255,0,0); break;
         case 1: stroke(0,255,0); break;
         case 2: stroke(0,0,255); break;
         case 3: stroke(100,50,50); break;
         case 4: stroke(80,80,80); break;
         case 5: stroke(20,20,20); break;
       }
      stroke(0,50,50);//altera a cor da curva
      n_cor++;
    }
   else if(mouseX>=X1_BOTAO && mouseX<=X1_BOTAO+LARG_BOTAO1
           && mouseY>=Y1_BOTAO && mouseY<=Y1_BOTAO+ALT_BOTAO1)
     {
      n_curva=0;
      n_cor=0;
      TelaGrafica();
```

```
    }
}

void TelaGrafica()
{
  background(192);
  stroke(0,0,0);
  strokeWeight(1);
  fill(128);//define a cor do botão
  rect(X0_BOTAO,Y0_BOTAO,LARG_BOTAO,ALT_BOTAO); //botão Traçar
  rect(X0_BOTAO-100,Y0_BOTAO,LARG_BOTAO,ALT_BOTAO);//botão Apagar
  textSize(16);//define o tamanho da fonte
  textAlign(CENTER, CENTER);//define o alinhamento do texto
  fill(0);//define a cor
  text("Traçar",X0_BOTAO+LARG_BOTAO/2,Y0_BOTAO+ALT_BOTAO/2-2);
  text("Apagar",X1_BOTAO+LARG_BOTAO1/2,Y1_BOTAO+ALT_BOTAO1/2-2);
  textSize(14);//define o tamanho da fonte
  line(X0_GRAF,Y0_GRAF,X0_GRAF,Y0_GRAF-ALT_GRAF);
  line(X0_GRAF,Y0_GRAF,X0_GRAF+LARG_GRAF,Y0_GRAF);
  triangle(X0_GRAF - 5, 75/2, X0_GRAF,-Y0_GRAF+(Y0_GRAF+20) ,
           X0_GRAF +5, 75/2);
  triangle(LARG_GRAF+120, Y0_GRAF+6, LARG_GRAF+120, Y0_GRAF-6,
           LARG_GRAF+135, Y0_GRAF );
  text("Id(mA)",  X0_GRAF-70,Y0_GRAF - ALT_GRAF );
  text("Vds(V)",  X0_GRAF + LARG_GRAF -30 ,Y0_GRAF +30);
  text("Vgs",  X0_GRAF + LARG_GRAF -30 ,Y0_GRAF - 340);
  text("3.00V",  X0_GRAF + LARG_GRAF -30 ,Y0_GRAF -300);
  text("2.75V",  X0_GRAF + LARG_GRAF -30 ,Y0_GRAF -260);
  text("2.50V", X0_GRAF + LARG_GRAF -30 ,Y0_GRAF -220);
  text("2.25V", X0_GRAF + LARG_GRAF -30 ,Y0_GRAF -180);
  text("2.00V", X0_GRAF + LARG_GRAF -30 ,Y0_GRAF -140);
  for(int i=0;i<=3;i++)
    for(int j=0; j<=4; j++)
      {
       line(X0_GRAF+int(LARG_GRAF/20)*(5*i+j),Y0_GRAF-3,
            X0_GRAF+int(LARG_GRAF/20)*(5*i+j),Y0_GRAF+3);
       text(i,X0_GRAF+int(LARG_GRAF/20)*5*i,Y0_GRAF+15);
      }
       line(X0_GRAF+int(LARG_GRAF/20)*20,Y0_GRAF-3,
            X0_GRAF+int(LARG_GRAF/20)*20,Y0_GRAF+3);
       text(4,X0_GRAF+int(LARG_GRAF/20)*20,Y0_GRAF+15);
  for(int i=0;i<=10;i++)
    {
     line(X0_GRAF+3,Y0_GRAF-int(ALT_GRAF/10)*i,X0_GRAF-3,
          Y0_GRAF-int(ALT_GRAF/10)*i);
     text(i*10,X0_GRAF-20,Y0_GRAF-int(ALT_GRAF/10)*i);
    }
}
```

ANEXO A – *Datasheet* do transistor BS170

Figura A-1. Datasheet BS170

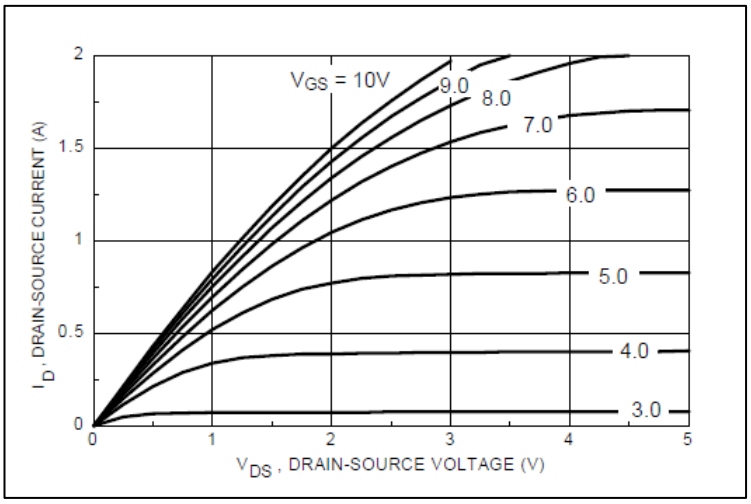
Electrical Characteristics (T_A = 25°C unless otherwise noted)

Symbol	Parameter	Conditions	Type	Min	Typ	Max	Units
OFF CHARACTERISTICS							
BV_{DSS}	Drain-Source Breakdown Voltage	V_{GS} = 0 V, I_D = 100 µA	All	60			V
I_{DSS}	Zero Gate Voltage Drain Current	V_{DS} = 25 V, V_{GS} = 0 V	All			0.5	µA
I_{GSSF}	Gate - Body Leakage, Forward	V_{GS} = 15 V, V_{DS} = 0 V	All			10	nA
ON CHARACTERISTICS (Note 1)							
$V_{GS(th)}$	Gate Threshold Voltage	V_{DS} = V_{GS}, I_D = 1 mA	All	0.8	2.1	3	V
$R_{DS(ON)}$	Static Drain-Source On-Resistance	V_{GS} = 10 V, I_D = 200 mA	All		1.2	5	Ω
g_{FS}	Forward Transconductance	V_{DS} = 10 V, I_D = 200 mA	BS170		320		mS
		$V_{DS} \geq 2\ V_{DS(on)}$, I_D = 200 mA	MMBF170		320		
DYNAMIC CHARACTERISTICS							
C_{iss}	Input Capacitance	V_{DS} = 10 V, V_{GS} = 0 V, f = 1.0 MHz	All		24	40	pF
C_{oss}	Output Capacitance		All		17	30	pF
C_{rss}	Reverse Transfer Capacitance		All		7	10	pF
SWITCHING CHARACTERISTICS (Note 1)							
t_{on}	Turn-On Time	V_{DD} = 25 V, I_D = 200 m A, V_{GS} = 10 V, R_{GEN} = 25 Ω	BS170			10	ns
		V_{DD} = 25 V, I_D = 500 mA, V_{GS} = 10 V, R_{GEN} = 50 Ω	MMBF170			10	
t_{off}	Turn-Off Time	V_{DD} = 25 V, I_D = 200 m A, V_{GS} = 10 V, R_{GEN} = 25 Ω	BS170			10	ns
		V_{DD} = 25 V, I_D = 500 mA, V_{GS} = 10 V, R_{GEN} = 50 Ω	MMBF170			10	

Note:
1. Pulse Test: Pulse Width ≤ 300µs, Duty Cycle ≤ 2.0%.

Fonte – Adaptado de: https://pdf1.alldatasheet.com/datasheet-pdf/view/233449/MICROCHIP/MCP4725.html

Figura A-2. Curva característica do *datasheet*.



Fonte – Adaptado de: https://pdf1.alldatasheet.com/datasheet-pdf/view/233449/MICROCHIP/MCP4725.html

ANEXO B – *Datasheet* do transistor 2N7000

Figura B-1. Datasheet do 2N7000

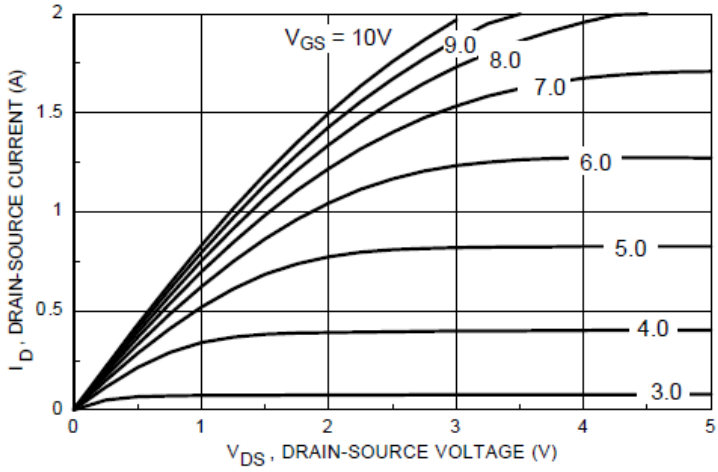
Electrical Characteristics T_A = 25°C unless otherwise noted

Symbol	Parameter	Conditions	Type	Min	Typ	Max	Units
ON CHARACTERISTICS Continued (Note 1)							
$I_{D(ON)}$	On-State Drain Current	V_{GS} = 4.5 V, V_{DS} = 10 V	2N7000	75	600		mA
		V_{GS} = 10 V, $V_{DS} \geq 2\ V_{DS(on)}$	2N7002	500	2700		
		V_{GS} = 10 V, $V_{DS} \geq 2\ V_{DS(on)}$	NDS7002A	500	2700		
g_{FS}	Forward Transconductance	V_{DS} = 10 V, I_D = 200 mA	2N7000	100	320		mS
		$V_{DS} \geq 2\ V_{DS(on)}$, I_D = 200 mA	2N7002	80	320		
		$V_{DS} \geq 2\ V_{DS(on)}$, I_D = 200 mA	NDS7002A	80	320		
DYNAMIC CHARACTERISTICS							
C_{iss}	Input Capacitance	V_{DS} = 25 V, V_{GS} = 0 V, f = 1.0 MHz	All		20	50	pF
C_{oss}	Output Capacitance		All		11	25	pF
C_{rss}	Reverse Transfer Capacitance		All		4	5	pF
t_{on}	Turn-On Time	V_{DD} = 15 V, R_L = 25 Ω, I_D = 500 mA, V_{GS} = 10 V, R_{GEN} = 25	2N7000			10	ns
		V_{DD} = 30 V, R_L = 150 Ω, I_D = 200 mA, V_{GS} = 10 V, R_{GEN} = 25 Ω	2N7002 NDS7002A			20	
t_{off}	Turn-Off Time	V_{DD} = 15 V, R_L = 25 Ω, I_D = 500 mA, V_{GS} = 10 V, R_{GEN} = 25	2N7000			10	ns
		V_{DD} = 30 V, R_L = 150 Ω, I_D = 200 mA, V_{GS} = 10 V, R_{GEN} = 25 Ω	2N7002 NDS7002A			20	
DRAIN-SOURCE DIODE CHARACTERISTICS AND MAXIMUM RATINGS							
I_S	Maximum Continuous Drain-Source Diode Forward Current		2N7002			115	mA
			NDS7002A			280	
I_{SM}	Maximum Pulsed Drain-Source Diode Forward Current		2N7002			0.8	A
			NDS7002A			1.5	
V_{SD}	Drain-Source Diode Forward Voltage	V_{GS} = 0 V, I_S = 115 mA (Note 1)	2N7002		0.88	1.5	V
		V_{GS} = 0 V, I_S = 400 mA (Note 1)	NDS7002A		0.88	1.2	

Note:
1. Pulse Test: Pulse Width ≤ 300μs, Duty Cycle ≤ 2.0%.

Fonte – Adaptado de: https://pdf1.alldatasheet.com/datasheet-pdf/view/50067/FAIRCHILD/2N7000.html

Figura B-2. Curva característica do *datasheet*.



Fonte – Adaptado de: https://pdf1.alldatasheet.com/datasheet-pdf/view/50067/FAIRCHILD/2N7000.html