Laboratório Maker

Da extensão ao osciloscópio

Informações sobre este livro

Este documento é um livro "vivo". Muitas das seções e figuras podem sofrer modificações à medida que a escrita vai se consolidando.

Quando você compra este livro, está comprando na realidade todas as versões durante um ano. Os livros são formatados no formato A5, o que permitirá a impressão e encadernação em papelarias.

Após adquirir uma versão, à cada versão posterior, durante um ano, você receberá um email com a versão mais atualizada.

Você pode influenciar diretamente sobre o conteúdo das versões. Aponte erros que identificou, sugestões de novos assuntos, dúvidas sobre o material escrito, etc.. Suas observações serão analisadas e devidamente incorporadas às novas versões.

Por favor, não copie, não compartilhe sem autorização e não jogue spoilers por aí! Se você curtiu o conteúdo e acha que alguém mais deveria ler, a melhor forma de apoiar é recomendando a compra oficial.

A cópia, distribuição ou compartilhamento não autorizado, seja em formato impresso ou digital, prejudica o trabalho do autor e desvaloriza o esforço dedicado à produção deste conteúdo. Lembre-se: cada cópia não autorizada é um escritor chorando em um canto (e possivelmente planejando vingança literária). Seja legal, apoie o autor!

Copyright @ 2025 por Roberto Tavares

*Histórico de impressões: * Junho 2025 Revisão 0.1*

Todos os direitos reservados e protegidos pela lei 9.610 de 19/02/1998. É proibida a reprodução desta obra, mesmo parcial, em qualquer processo, sem prévia autorização, por escrito, do autor e da editora.

Para interagir com o autor, visite o canal eletrohow, no YouTube, ou a página eletrohow.com.

Laboratório Maker

Da Extensão ao Osciloscópio

Roberto Tavares

Sumário

1. Um novo mundo para os makers 1
1.1. Por que a demanda por esses profissionais vai crescer? 2
1.2. Habilidades essenciais 3
1.3. E onde entra o laboratório nesse cenário? 4
2. Construindo um gerador de pulsos 7
2.1. Aplicações em Laboratórios Eletrônicos 7
2.2. Como Funciona? 8
2.3. Especificações do nosso gerador de pulsos 9
2.4. Material necessário 10
2.5. Comandos de configuração 10
2.6. Comentários sobre o código fonte 10
2.7. Dicas de teste rápido 11
2.8. Código fonte 12
2.9. Como utilizar a interface Pulsar 16
3. Construindo um gerador de palavras 21
3.1. Como é utilizado 21
3.2. Como implementamos 22
3.3. Material necessário 23
3.4. Funcionalidade do sistema 23
3.5. A interface com o usuário 24
3.6. Código fonte 24
3.7. Como utilizar a interface VerbaFlux 27
4. Construindo um analisador lógico 33
4.1. Material necessário 34
4.2. Como é utilizado? 34
4.3. Vantagens em Relação a um Osciloscópio: 36
4.4. Especificações do analisador proposto 37
4.5. Código fonte do Arduino 41
4.6. Como utilizar a interface PulseView 51

1. Um novo mundo para os makers

Um sistema ciberfísico (em inglês, cyber-physical system – CPS) é um sistema que integra dispositivos computacionais e eletromecânicos. Essa é uma área considerada chave de pesquisa pela US National Science Foundation (NSF). O que hoje chamamos de sistemas ciberfísicos é uma evolução do que antes era conhecido como sistemas embarcados. A diferença é que, enquanto os embarcados focavam mais nos elementos computacionais, os sistemas ciberfísicos destacam a interação entre os componentes digitais e físicos.

Um sistema ciberfísico normalmente é composto por uma rede de elementos que interagem entre si, com capacidade de perceber o ambiente por meio de sensores e agir sobre ele usando atuadores. A ponte entre o que é captado e a ação tomada acontece por meio do processamento digital da informação. Se isso te fez pensar em um robô, acertou! O robô é um exemplo clássico de sistema ciberfísico. Os avanços da engenharia, junto com o uso de inteligência artificial, vêm turbinando os sistemas ciberfísicos, deixando-os mais autônomos, eficientes, funcionais, seguros e fáceis de usar.

A demanda por profissionais da área de **Sistemas Ciberfísicos (CPS)** deve crescer bastante nos próximos anos. Isso porque ela é um dos pilares da **Indústria 4.0** e da **transformação digital**, conectando o mundo físico ao digital e criando sistemas inteligentes e autônomos.

1.1. Por que a demanda por esses profissionais vai crescer?

Vários fatores estão puxando esse aumento:

- **Indústria 4.0 e Automação:** Setores como manufatura, logística e energia estão cada vez mais buscando otimizar processos, aumentar a produtividade e reduzir falhas. Isso passa pelo uso crescente de CPS como robôs colaborativos, gêmeos digitais e dispositivos IoT industriais.
- **Cidades Inteligentes e Veículos Autônomos:** Esses sistemas precisam lidar com enormes volumes de dados em tempo real. Para isso, vão precisar de gente especializada para projetar, implementar e manter tudo funcionando.
- **Cibersegurança:** À medida que sistemas operacionais (OT) e de TI ficam mais conectados, a segurança cibernética vira um ponto crítico. Proteger infraestruturas críticas contra ataques é cada vez mais importante — e exige especialistas em segurança de OT.
- **Inteligência Artificial (IA) e Machine Learning:** Integrar IA e aprendizado de máquina aos CPS permite decisões autônomas e mais eficientes. Isso aumenta a procura por profissionais que dominem essas tecnologias.
- **5G:** A chegada do 5G nas indústrias, por causa da baixa latência e alta confiabilidade, vai exigir profissionais que saibam lidar com o controle e a segurança dessas redes.

- **Sustentabilidade:** A busca por práticas mais verdes e economia circular também se apoia na automação e em sistemas ciberfísicos para ser viável.

1.2. Habilidades essenciais

Pra dar conta dessa demanda, quem quiser trabalhar com sistemas ciberfísicos vai precisar de uma boa mistura de **hard skills e soft skills**:

- **Hard Skills:**
- **Engenharia de Automação e Controle:** Ter domínio de sistemas de controle, automação industrial e robótica.
- **Programação:** Mandar bem em linguagens como Python, C++, Java, entre outras.
- **Redes e Comunicação:** Entender de redes industriais, protocolos e infraestrutura de rede.
- **Cibersegurança:** Saber proteger sistemas operacionais, redes e dispositivos IoT contra ataques cibernéticos.
- **Análise de Dados e Inteligência Artificial:** Saber coletar, analisar e tirar conclusões de grandes volumes de dados, além de aplicar modelos de IA e Machine Learning.
- **Computação em Nuvem (Cloud) e Internet das Coisas (IoT):** Conhecer como essas tecnologias funcionam na prática.

- **Mecatrônica e Eletrônica:** Ter noções sólidas dos componentes físicos que fazem o sistema funcionar.
- **Gêmeos Digitais (Digital Twins):** Saber trabalhar com simulações e modelos virtuais de sistemas físicos.
- **Soft Skills:**
- **Pensamento Crítico e Resolução de Problemas:** Pra encarar os desafios técnicos com criatividade.
- **Adaptabilidade e Aprendizado Contínuo:** A tecnologia muda rápido. Estar sempre aprendendo é essencial.
- **Trabalho em Equipe e Comunicação:** Saber se comunicar e colaborar com gente de várias áreas.
- **Criatividade e Inovação:** Pensar fora da caixa para criar soluções novas e eficazes.

Resumindo: o campo de Sistemas Ciberfísicos está crescendo rápido e oferecendo um monte de oportunidades. Quem estiver disposto a se capacitar e evoluir junto com a tecnologia vai ter muito espaço nesse mercado.

1.3. E onde entra o laboratório nesse cenário?

Ter acesso a um bom laboratório de medição e

geração de sinais elétricos faz toda a diferença pra quem desenvolve sistemas ciberfísicos. Ser capaz de gerar sinais de teste e medir o que está acontecendo no sistema que está sendo criado (ou consertado) é essencial pra ganhar tempo e precisão no desenvolvimento.

Só que tem um detalhe: montar um laboratório completo custa caro. A boa notícia é que hoje, com microcontroladores e circuitos integrados multifuncionais e de baixo custo, dá pra montar instrumentos que antes só existiam em laboratórios de universidades ou centros de pesquisa.

A tecnologia embarcada está mudando tudo. Agora é possível criar dispositivos cada vez mais inteligentes, acessíveis e conectados – mesmo com poucos recursos.

Esses dispositivos estão em tudo quanto é lugar: equipamentos médicos, sistemas automotivos, casas inteligentes, agricultura de precisão… e claro, no universo maker. E se você está lendo este livro, provavelmente já está começando a fazer parte desse mundo.

2. Construindo um gerador de pulsos

Os geradores de pulsos são equipamentos eletrônicos muito importantes nos laboratórios de eletrônica. Desempenham um papel crucial em testes, medições e controle de sistemas. Compreender o que são e para que servem é essencial para estudantes, técnicos e engenheiros que trabalham com circuitos e dispositivos eletrônicos.

Um gerador de pulsos é definido como um dispositivo eletrônico projetado para produzir uma sequência de pulsos elétricos. Esses pulsos são caracterizados por serem sinais que alternam rapidamente entre dois níveis de tensão definidos, tipicamente um nível baixo (muitas vezes zero volts) e um nível alto (geralmente 5 ou 3.3 volts). Um bom gerador deve permitir a configuração da frequência de repetição do pulso e a largura, ou seja, o tempo alto do pulso.

2.1. Aplicações em Laboratórios Eletrônicos

A utilidade dos geradores de pulsos em laboratórios eletrônicos é vasta. Uma das aplicações mais comuns é no teste e depuração de circuitos digitais e analógicos. Utilizamos os pulsos gerados para estimular circuitos sob teste e observar sua

resposta, verificando se operam conforme o esperado. Por exemplo, podem ser usados para fornecer sinais de clock para circuitos digitais, testar a velocidade de resposta de transistores ou amplificadores, ou simular sinais de sensores. Além disso, são ferramentas indispensáveis em sistemas de comunicação para gerar sinais de teste que simulam dados transmitidos ou para modular informações. Em equipamentos de medição, os pulsos podem servir como base de tempo ou como sinais de disparo para sincronizar medições.

2.2. Como Funciona?

Até alguns anos atrás os geradores de pulso eram construídos baseados em componentes passivos, como resistores e capacitores, e componentes ativos, como transistores ou circuitos integrados especializados. A interação desses componentes permite criar um ciclo de carga e descarga ou chaveamento que resulta na forma de onda pulsada na saída. Parâmetros cruciais do pulso, como a frequência e a largura do pulso e, em alguns casos, a amplitude (o valor máximo da tensão do pulso) podiam ser ajustados pelo usuário.

Desde circuitos simples baseados no Ci 555 até circuitos bem complexos com grande excursão dos parâmetros de configuração, eram equipamentos que podiam assumir valores bastantes elevados.

Com o aparecimento dos microcontroladores, uma opção totalmente nova surgiu. Geração de pulsos desde os poucos Hz a dezenas de Mhz se tornou

possível sem o uso de chaves seletoras, alta precisão sem a presença de componentes especiais. E tudo isto a custo bem baixo. Foi mais uma área onde os microcontroladores marcaram fortemente a sua presença.

Nós vamos projetar um gerador de pulsos baseado na placa DoiTKit v1.2 , com o uso do microcontrolador ESP32. Mas como utilizamos a IDE do Arduino, é fácil migrar do ESP32 para o Arduino UNO ou outro Arduino. O único cuidado é verificar os aspectos de velocidade de clock e ajustar o programa no que for necessário.

2.3. Especificações do nosso gerador de pulsos

O gerador aqui descrito terá uma largura de pulso ajustável entre 1 e 99%, com resolução de oito bits. A frequência poderá ser ajustada entre 10Hz e 30Mhz. A saída será de 0/3.3 volts, diretamente do pino D18 da placa DoItKit v1.1. A largura de pulso e a frequência serão controlados via interface serial.

Muitos destes parâmetros podem ser alterados no firmware. Ficará fácil de ver como após estudarmos o script apresentado.

2.4. Material necessário

```
Placa DoitKit ESP32 V 1
Cabo USB A/B mini
Gabinete
Cabos de ligação e conectores para estes cabos
```

2.5. Comandos de configuração

Os comandos são recebidos pela interface serial a 115200, 8N1.

Função	mensagem	retorno	obs
Ajusta largura	1,nn\n	$1,LP nn\n	nn= largura entre 1 e 99
Ajusta frequência	2,nnnn\n	$2,%d\n	nnnn = frequência em Hz

2.6. Comentários sobre o código fonte

A geração do pulso em si é realizada pela função de alto nível ledcWrite(pwmPin, dutyValue); . Esta função envia ao pino "pwmPin" uma onda retangular de largura de pulso proporcional ao "dutyValue".

O software possui duas tarefas principais. Uma deles gerencia os comandos advindos da porta serial, a outra gera os pulsos tendo como

parâmetros variáveis que serão alteradas através dos comandos advindos da porta serial.

No bloco "setup", configuramos a porta serial, op vetor de chamadas de função e valores default para a largura e a frequência.

No bloco "loop", verificamos se tem caracteres na porta serial, monta a mensagem caso exista caractere. Após o recebimento do caractere "\n" envia a mensagem para ser decodificada. A decodificação altera os parâmetros da geração de pulsos.

Com estas informações e acompanhando os comentários ao longo do código, você identifica toda a funcionalidade do mesmo. Caso tenha alguma dúvida acesse o canal "Eletrohow" no Youtube, ou mais diretamente em https://youtu.be/8X9EpqxRG-s , e exponha a sua dúvida.

2.7. Dicas de teste rápido

Você pode utilizar qualquer emulador de terminais para testar o firmware, ou mesmo desenvolver uma interface específica para usar com o gerador de pulsos. Nossa sugestão é o terminal YAST, apresentado em https://youtu.be/d6SzgJReQwA e cujo download pode ser realizado em https://eletrohow.com/seleciona_softwares. Ou utilizar a interface Pulsar, também disponível em https://eletrohow.com/seleciona_softwares. O uso desta interface é descrita logo após a apresentação do código fonte do ESP32.

2.8. Código fonte

```
/*
 * Nome do projeto: geradorPulsos
 * Versão: 0.3 - 20/05/2025
 * Descrição: Geração de PWM com API 3.x do ESP32
(canal automático)
 */

#include <Arduino.h>

// ============ CONFIGURAÇÃO ============
// Serial
#define MAX_LEN_ENTRADA     64  //tamanho do buffer
Rs232C de entrada
#define NUMERO_MAXIMO_FUNCOES    5  // numero maximo
de funcoes
#define MAX_LEN_SAIDA     16    // tamnho do buffer
RS232C de saida

//Variáveis  globais
int  chLido;    // contem o caracter lido na serial
int cIndexEscrita; // conta o numero de caracteres
recebidos

char chParam[3];    // buffer para armazenamento do
parametro associado ao parametro

char chCmd[2];      // buffer para o armazenamento do
comando completo

char chSaida[MAX_LEN_SAIDA];
char chPDU[MAX_LEN_ENTRADA];

//vetor de funcoes que implementam os comandos
void (*func[NUMERO_MAXIMO_FUNCOES])();
```

```
// PWM

//Variáveis  globais

const uint8_t pwmPin = 18;           // Pino de saída
PWM
uint32_t pwmFrequency = 1000;  // Frequência inicial
do PWM em Hz
uint8_t pwmResolution = 8;     // Resolução inicial
em bits (1 a 16)
float dutyCyclePercent = 50.0; // Duty cycle inicial
(%)
uint32_t dutyValue;

int pwmChannel; // Canal atribuído automaticamente
pela API 3.x


int  modo_operacao = 0;    // para uso futuro quando
tivermos mais de um instrumento


void setup() {
  Serial.begin(115200, SERIAL_8N1);
  delay(500);

  //inicializacao do vetor de chamada de funcoes
  func[0]=  seleciona_pwm;
  func[1] = ajusta_largura_pulso;
  func[2] = ajusta_frequencia;


  // Validação e cálculo do duty cycle
  float dc = constrain(dutyCyclePercent, 1.0, 99.0);
  dutyValue = (uint32_t)(dc * ((1 << pwmResolution) -
1) / 100.0);

  // Configura o PWM e obtém o canal automaticamente
```

```
  pwmChannel = ledcAttach(pwmPin, pwmFrequency,
pwmResolution);


  Serial.print("PWM iniciado no pino ");
  Serial.print(pwmPin);
  Serial.print(" (canal ");
  Serial.print(pwmChannel);
  Serial.print(") com ");
  Serial.print(dc);
  Serial.println("% de duty cycle.");
}

void loop()
{

chLido= Serial.read();

 if(chLido != -1)
  {
  if(chLido != 0x0d)  //joga fora os 0x0d caso
existam
    {
    chPDU[cIndexEscrita]= (byte) chLido;

    if(chLido == 0x0A)  // delimitador de final de
comando detectdo
      {
      chPDU[cIndexEscrita]=0;
      processaCmd();  //ao final do comando chama a
decodificacao
      cIndexEscrita=0;  //prepara para um novo
comando
      }
    else
      {
      ++cIndexEscrita;
      cIndexEscrita &= (MAX_LEN_ENTRADA -1);
      }
    }
```