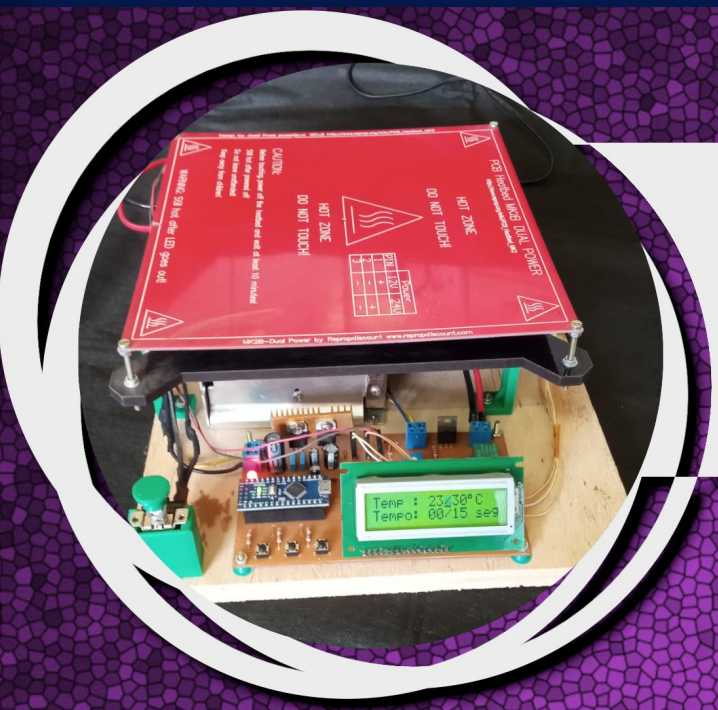
Como Conquistar a Autorizada

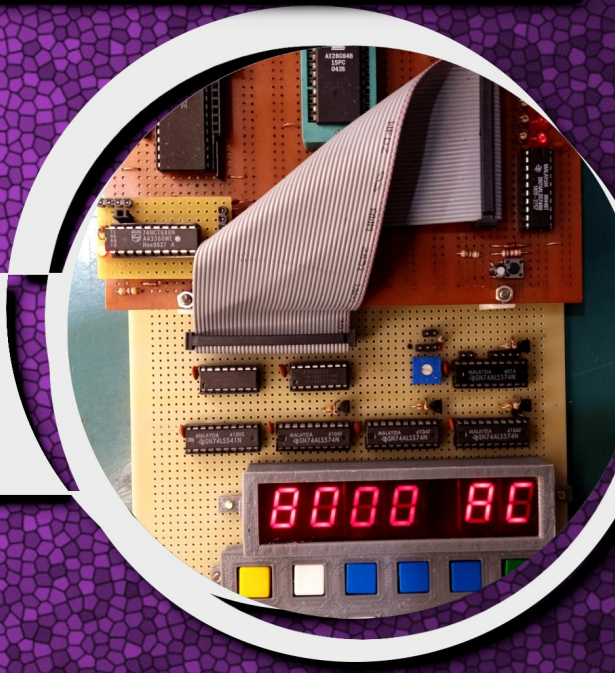
revista
INCBELETRÔNICA

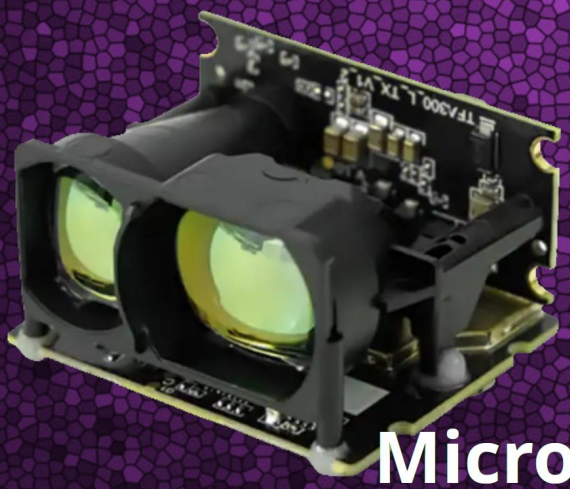
Número 33
mar/abr 2026



HeatBED Arduino

Computador COSMAC - Parte 3



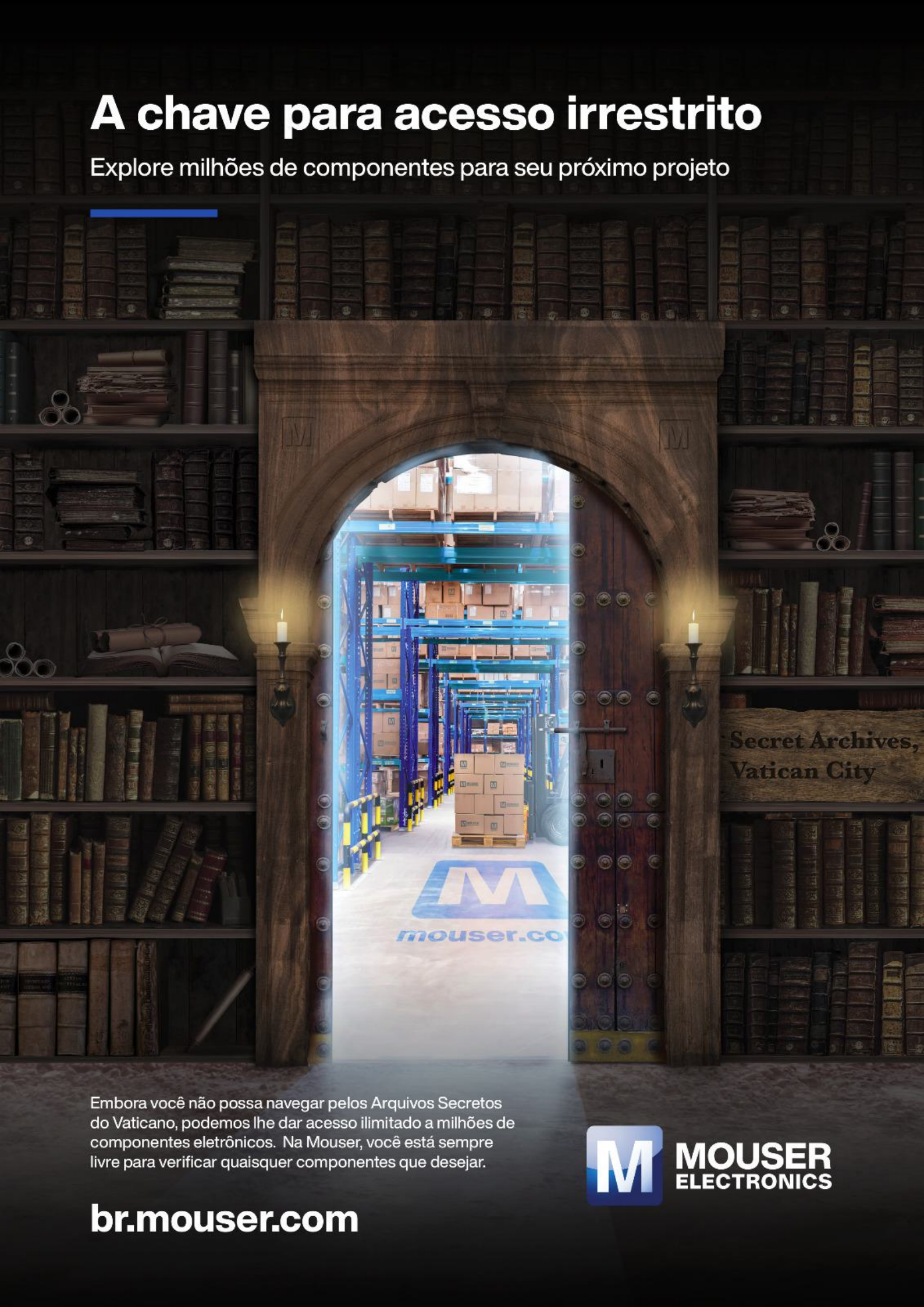


Como Funciona o LiDAR

Microcontrolador Chinês RISC V de 32 bits de Baixo Custo

A chave para acesso irrestrito

Explore milhões de componentes para seu próximo projeto



Embora você não possa navegar pelos Arquivos Secretos do Vaticano, podemos lhe dar acesso ilimitado a milhões de componentes eletrônicos. Na Mouser, você está sempre livre para verificar quaisquer componentes que desejar.

MOUSER ELECTRONICS

br.mouser.com

EXPEDIENTE

Revista INCB Eletrônica
Revista do Instituto Newton C. Braga
Ano 5 - Edição nº 33 - 2026

Editor Administrativo
Newton C. Braga (CEO)

Editor Técnico
MSc. Eng. Prof. Antonio Carlos Gasparetti

Produção
Renato Paiotti

Produção Gráfica
Eng. Vander da Silva Gonçalves

Atendimento ao leitor
leitor@newtoncbraga.com.br

Atendimento ao cliente
publicidade@newtoncbraga.com.br

Jornalista Responsável
Marcelo Lima Braga
MTB 0064610SP

Colaboradores
Alexandre José Nário
Antônio Carlos Gasparetti
João Batista de Sousa
Luis Carlos Burgos
Luiz Henrique Correa Bernardes
Márcio José Soares
Michael A. Shustov
Newton C. Braga
Pedro Bertoletti
Renato Paiotti

ÍNDICE

Nº 33 MARÇO/ABRIL - 2026

MONTAGEM

4 - HEATBED ARDUINO

86 - COMPUTADOR COSMAC - PARTE 3

MICROCONTROLADOR

32 - MICROCONTROLADOR CHINÊS RISC-V DE 32 BITS DE BAIXO CUSTO

64 - USANDO O ACELERÔMETRO MPU6050 COM A RASPBERRY PI PICO 2

CONSERTOS

45 - COMO CONQUISTAR A AUTORIZADA

TECNOLOGIA

52 - COMO FUNCIONA O LIDAR

72 - POLARIZAÇÃO DC COM REALIMENTAÇÃO NEGATIVA

INDÚSTRIA

102 - USE MODBUS TCP COM ESP32-C3

NOTÍCIAS

113 - NOVA TÉCNICA AUXILIA A GERAÇÃO DE ENERGIA OSMÓTICA

115 - SERVICE

EDITORIAL

Antonio Carlos Gasparetti

Bem-vindo à trigésima terceira edição da Revista INCB Eletrônica. Neste número, reafirmamos nosso compromisso de unir a tradição da eletrônica de bancada às tecnologias que estão moldando o futuro industrial e acadêmico. Preparamos um conteúdo que equilibra teoria rigorosa com aplicações práticas imediatas.

Destaques desta edição:

Montagem e Hardware

• Aprimoramento de PCIs com HeatBED Arduino: Este projeto utiliza um Arduino Nano e sensores NTC para automatizar uma "cama quente" com agitação mecânica, o que otimiza a reação do percloreto de ferro para garantir uma corrosão de placas de circuito impresso muito mais rápida, uniforme e com acabamento profissional.

• Arquitetura Profunda com o Computador COSMAC: A terceira parte desta série disseca o funcionamento do microprocessador CDP1802, permitindo que o desenvolvedor domine a lógica de barramentos e registros essencial tanto para a preservação de sistemas clássicos quanto para a compreensão fundamental de arquiteturas de computadores.

Microcontroladores e Sensores

• Explorando o Ecossistema RISC-V com CH32V003: O artigo desbrava o uso deste microprocessador chinês de 32 bits através da IDE MounRiver Studio, fornecendo uma rota de altíssimo desempenho e custo quase zero para projetos de IoT que demandam independência de arquiteturas proprietárias.

• Navegação e Movimento com MPU6050 e Raspberry Pi Pico 2: Este guia técnico ensina a integrar sensores de aceleração e inclinação à nova geração da Raspberry Pi, viabilizando o desenvolvimento de sistemas de estabilização precisos para robótica, drones e wearables.

Tecnologia e Teoria Aplicada

• Estabilidade Total com Realimentação DC Negativa: Essencial para o projetista, este estudo demonstra como a realimentação negativa permite a substituição de componentes por outros de especificações diferentes sem alterar o ponto de operação, garantindo o controle térmico rigoroso e um funcionamento uniforme do circuito durante toda a sua vida útil.

• Visão Computacional e a Tecnologia LiDAR: Uma análise profunda sobre como sistemas de laser e espelhos MEMS realizam o mapeamento tridimensional do ambiente, tecnologia fundamental para quem deseja atuar no desenvolvimento de veículos autônomos e sensoriamento de longa distância.

Prática Profissional e Indústria

• Estratégia para Conquista de Assistência Autorizada: Este roteiro profissional orienta o técnico sobre os passos necessários para formalizar seu negócio e obter o credenciamento de grandes fabricantes, transformando a oficina em um centro de serviço oficial e mais lucrativo.

• Conectividade Industrial com Modbus TCP e ESP32-C3: O artigo mostra como implementar o protocolo Modbus via Wi-Fi, permitindo a integração de dispositivos modernos e acessíveis em redes industriais legadas e sistemas de supervisão SCADA.

Notícias e Inovação

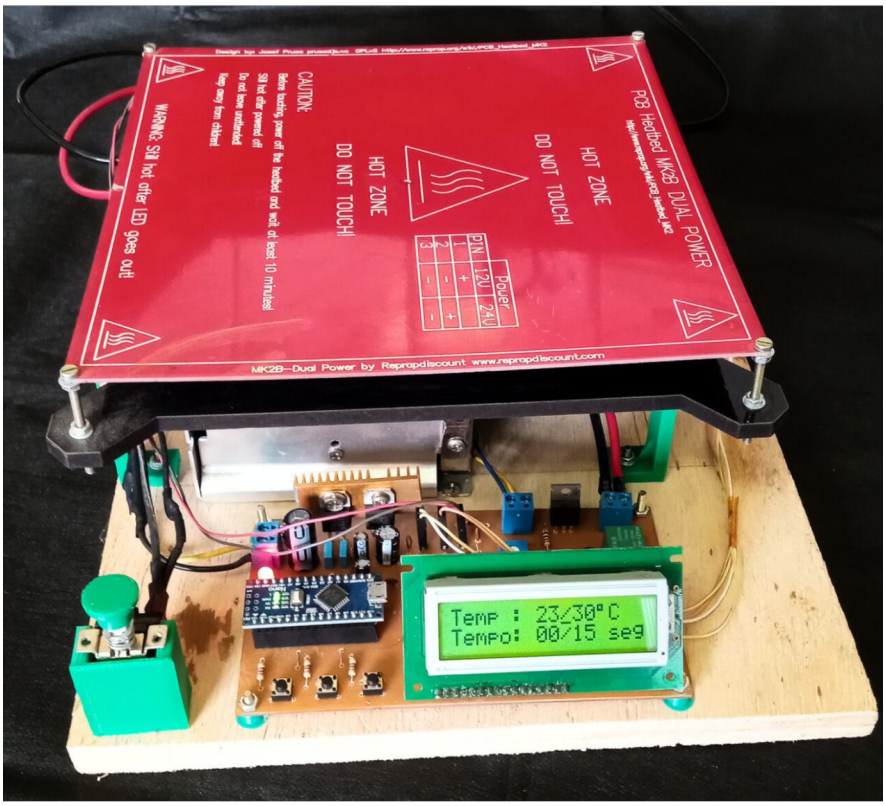
• Avanços na Geração de Energia Osmótica: Uma atualização sobre as novas técnicas que extraem eletricidade do gradiente de salinidade da água, mantendo o leitor informado sobre as fronteiras das fontes de energia renováveis e sustentáveis.

Com esta seleção de artigos, esperamos fornecer as ferramentas necessárias para que você continue inovando em seus projetos e carreira.

Boa leitura e ótimas montagens!

Mais uma edição da Revista INCB Eletrônica chega gratuitamente (ou paga – edição impressa) até você. Como sempre seu conteúdo é atraente para todos os que estão imersos no mundo da eletrônica e precisam estar atualizados com o que há de mais novo nessa tecnologia, desde projetos até artigos teóricos e notícias. Temos mais uma variedade de artigos selecionados por nossa equipe que se preocupa em abranger todos os setores da eletrônica com conteúdo que realmente seja de interesse para você. Sempre lembrando que a revista é uma das mídias que temos para chegar até você e a partir delas muitos links levam a outros canais nossos, como o site, Youtube, livros e muito mais que complementam seu conteúdo.

A Revista é mais uma de nossas mídias, criada justamente visando o público leitor que gosta dessa modalidade de meio de informação. Baixe sua revista, abra uma pasta para formar sua biblioteca e tenha ao seu alcance uma preciosa fonte de informação e conteúdo para seu trabalho, estudo, aula ou consulta.



HeatBED Arduino

Um acessório para ajudá-lo no preparo das suas PCIs

Eng. Márcio José Soares

Durante o processo de confecção de placas de circuito impresso (PCIs) experimentais, é recorrente a necessidade de agitar manualmente a solução de percloreto de ferro a intervalos regulares, com o objetivo de auxiliar na remoção dos resíduos que se formam sobre a superfície da placa ao longo do processo de corrosão. Ademais, é amplamente difundido na literatura técnica e entre praticantes da área que o aquecimento da solução contribui de forma significativa para a eficiência do processo corrosivo. Nesse contexto, propõe-se o desenvolvimento de um sistema automatizado capaz de integrar ambas as funções — de agitação e aquecimento — de maneira controlada e contínua. Tal solução destina-se àqueles que se dedicam à fabricação artesanal de PCIs e que demonstram interesse na automação de processos laboratoriais.

Um pouco de teoria

Primeiro é preciso falar um pouco sobre o que acontece durante o processo chamado de “corrosão” quando o leitor prepara suas PCIs. Lembrando que tratamos aqui do uso da solução mais conhecida que é o Percloreto de Ferro!

O que nós conhecemos como Percloreto de Ferro é na verdade Cloreto de Ferro trivalente (***$FeCl_3$***) que é um oxidante forte. Portanto o processo real que acontece é uma oxidação e não corrosão! Quando este oxidante entra em contato com um metal como o cobre temos algo como:

$2\ FeCl_3 + Cu \rightarrow 2\ FeCl_2 + CuCl_2$

O leitor percebe agora que existe na solução a presença de dois subprodutos novos: o Cloreto de Ferro bivalente (***$FeCl_2$***) que foi reduzido do Cloreto de Ferro trivalente que não é mais um oxidante forte e o Cloreto de Cobre (***$CuCl_2$***).

Como se trata de uma reação do tipo “exotérmica” (que libera calor para o ambiente), mantê-la aquecida irá melhorar a mesma aumentando a energia cinética das moléculas e resultando em colisões mais frequentes e eficazes entre os íons de ferro (***Fe^{3+}***) e a superfície de cobre (***Cu***) da placa aumentando a velocidade da oxidação. Além disso, precisamos lembrar que ao aquecer uma solução diminuímos sua viscosidade o que também melhora a movimentação dos íons (difusão).

Durante essa reação os subprodutos (***$FeCl_2 + CuCl_2$***) formam uma fina camada que tende a ficar sobre a superfície da placa conhecida como camada de passivação, que impede o contato do Cloreto de Ferro trivalente com o Cobre . E a “agitação” ajuda na remoção destes subprodutos “indesejáveis” da superfície da placa. Temos assim a renovação constante do oxidante forte sobre o metal. Chamamos isso de renovação de reagentes!

Um outro detalhe importante é que o movimento diminui significativamente a presença de “manchas” durante a oxidação. O leitor que costuma fazer suas próprias placas já deve ter notado que em determinados momentos a placa parece “corroer” mais em alguns pontos que em outros. Isso é devido à menor concentração dos subprodutos sobre estes pontos em relação aos outros.

Assim, o que para muitos sempre pareceu uma “bobagem” (aquecer e agitar a solução de Percloreto de Ferro) foi aqui explicado quimicamente.

O que ainda falta saber?

Bom, falamos da oxidação e dos subprodutos gerados. Podemos agora compreender o porquê da solução que usamos já há algum tempo parece perder seus efeitos. Temos então uma solução cansada (fraca). Mas é possível recuperarmos a mesma evitando o seu descarte prematuro.

Para recuperar uma solução cansada basta deixá-la em contato com o ar (oxigênio) em uma vasilha grande e rasa (para que um maior volume da solução entre em contato com o ar) fazendo com que o Cloreto de Ferro bivalente volte a ser o Cloreto de Ferro trivalente (oxidante forte). Veja como isso acontece a seguir:

$$2\ FeCl_2 + H_2O + O_2 \rightarrow FeCl_3 + Fe(OH)_3$$

Obs.*: Usar um pequeno compressor de ar pode ajudar muito, tanto na recuperação da solução como também durante a oxidação, já que ele aumenta bastante a presença do oxigênio no processo.*

Agora a solução tem a presença novamente do Cloreto de Ferro trivalente (***$FeCl_3$***) e de um novo subproduto que é o Hidróxido de Ferro trivalente (***$Fe(OH)_3$***) que pode ser reconhecido como aquela borra com cor de ferrugem que se forma no fundo do recipiente.

Para evitar a formação dessa borra e tornar a recuperação ainda mais eficiente, podemos adicionar na solução um pouco de Ácido Clorídrico (***HCl***) e Água Oxigenada (***$H2O_2$***). O ácido dissolverá a borra e com a ajuda do elemento oxidante água oxigenada, o ***$FeCl_2$*** será novamente transformado em ***$FeCl_3$***, deixando a solução límpida e mais potente (oxidante). Veja a seguir:

$$2\ FeCl_2 + 2\ HCl + H_2O_2 \rightarrow 2\ FeCl_3 + 2\ H_2O$$

Porém há um detalhe importante. A recuperação não será total já que o Cloreto de Cobre (***$CuCl_2$***) ainda está presente na solução. E a saturação deste elemento pode prejudicar a ação da solução e o leitor perceberá

isso quando essa mudar sua cor de um marrom-avermelhado para um verde-escuro quase negro. Isso indica que há muito Cloreto de Cobre na solução e é hora de usar mais um truque que irá trocar o cobre por ferro!

Para auxiliar na “troca” do cobre por ferro adicione uma palha de aço nova na solução cansada. Como o ferro é mais reativo, ele vai forçar o cobre a se desprender da solução e a se depositar no fundo da mesma, se apresentando como uma “borra avermelhada” (necessário decantação de pelo menos 48 horas). Usamos aqui a “reatividade dos metais”. Veja a seguir:

$Fe_{(s)} + CuCl_{2(aq)} \rightarrow FeCl_2 + Cu_{(s)}$

s – sólido

aq - aquoso

Agora basta filtrar a solução e teremos removido todo o cobre da mesma. A filtragem pode ser feita com um filtro de café ou um pedaço de pano limpo, por exemplo.

Cuidados: *Não se esqueça que ao manusear materiais químicos você precisa usar os EPI’s (Equipamento de Proteção Individual) adequados como: óculos com proteção lateral, máscara contra gases e luvas de PCV, nitrílica ou neoprene (nunca latex)*.

Dica: *A quantidade estimada, para cada 1000ml de “solução cansada”, é de:*

- *100ml à 150 ml de Ácido Clorídrico;*
- *50ml à 100ml de Água Oxigenada 2 ou 3 volumes.*

Circuito

Na **figura 1** o leitor tem o diagrama elétrico do projeto. U1 é um Arduino Nano utilizado no controle dos demais dispositivos presentes no circuito. Os pinos D4 à D7, D11 e D12 foram utilizados para o controle do LCD1 que é um display de cristal líquido tipo caractere. A função do LCD1 é mostrar as mensagens para o usuário durante a configuração e uso do sistema. P1 auxilia na obtenção do contraste ideal para LCD1.

S1, S2 e S3 são 3 chaves do tipo táctil conectadas aos pinos D8, D9 e D10, respectivamente, e configurados como entradas digitais em U1. A função destas chaves é permitir a configuração do tempo de agitação, a temperatura desejada para a mesa aquecida (ou cama quente) e a própria operação do

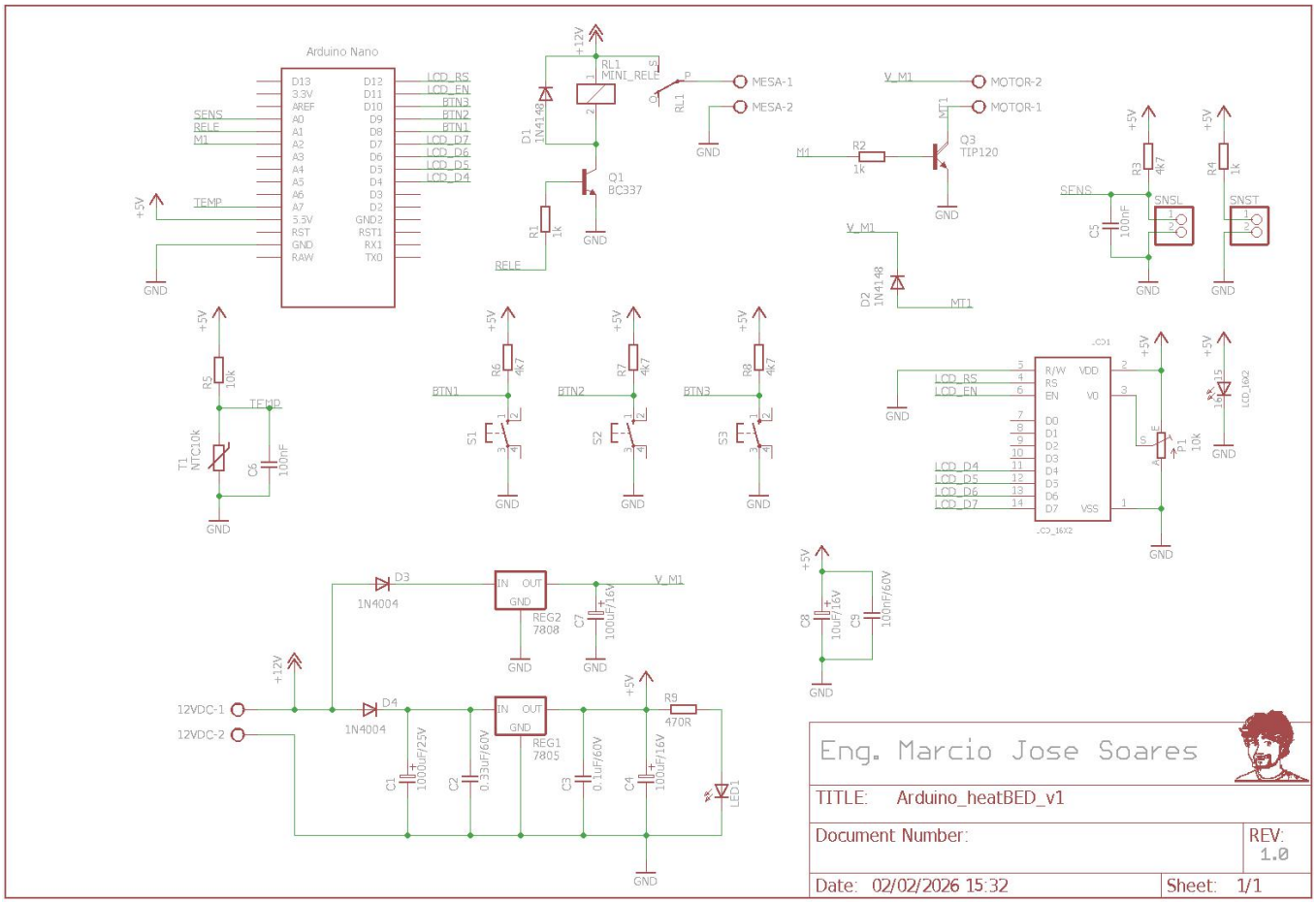


Figura 1 - Diagrama elétrico

sistema. R6, R7 e R8 são resistores de pull-up para as entradas já descritas em U1 garantindo nível lógico HIGH nas mesmas quando a chave conectada não estiver pressionada.

Q1 é um transistor NPN de uso geral que controla o relé RL1, utilizado para ligar/desligar a mesa aquecida. R1 é o resistor de base que ajuda na sua polarização. A outra ponta de R1 está conectada ao pino A1 (digital) de U1. D1 é conhecido como diodo de roda livre ou diodo flyback e serve para proteger os componentes presentes no circuito dos picos de tensão reversa gerados quando a bobina do relé é desligada.

Q2 é um transistor de potência do tipo NPN darlington e é usado no controle do motor M1 que executará a agitação da vasilha (veja mais detalhes a respeito na montagem mecânica). O resistor R2 ajuda na polarização da sua base e está conectado ao pino A2 (digital) de U1. D2 é outro diodo flyback e ajuda na proteção de Q2.

A entrada SNSL conectada a A0 (digital) de U1 e a saída SNST são usadas para conectar um Sensor Óptico IR do tipo barreira que trabalha por interrupção do feixe emitido (SNST ligado ao emissor e SNSL ao receptor). O resistor R3 polariza o receptor do sensor e serve também como resistor de pull-up para a entrada A0, enquanto que R4 é o limitador de corrente para o emissor do sensor.

T1 é um sensor de temperatura do tipo NTC (Negative Temperature Coefficient) de 10k que está conectado fisicamente à mesa aquecida e será usado pelo sistema para controlar a temperatura da mesma. O resistor R5 ajuda na polarização de T1. O nó desta conexão recebe um capacitor C6 para auxiliar na estabilização durante sua leitura via canal analógico A7 de U1.

O circuito conta ainda com uma “fonte regulada” de 5VDC para a parte digital (nível TTL) composta por D4 (que protege contra uma possível inversão na entrada de 12VDC do circuito), os capacitores C1, C2, C3 e C4 que atuam como filtros e REG1 que faz a regulagem em 5VDC. O LED1 juntamente com seu resistor limitador de corrente R9 formam o bloco usado para informar a presença da tensão 5VDC na saída da fonte.

A outra parte da fonte conta com o diodo D3 que tem a mesma função de D4, o regulador REG2 para 8VDC e o capacitor de filtro C7. Os capacitores C8 e C9 são filtros adicionais para o circuito.

Montagem Eletrônica

A **figura 2** mostra o layout para a placa de circuito impresso feito pelo autor e a **figura 3** a placa montada pelo mesmo.

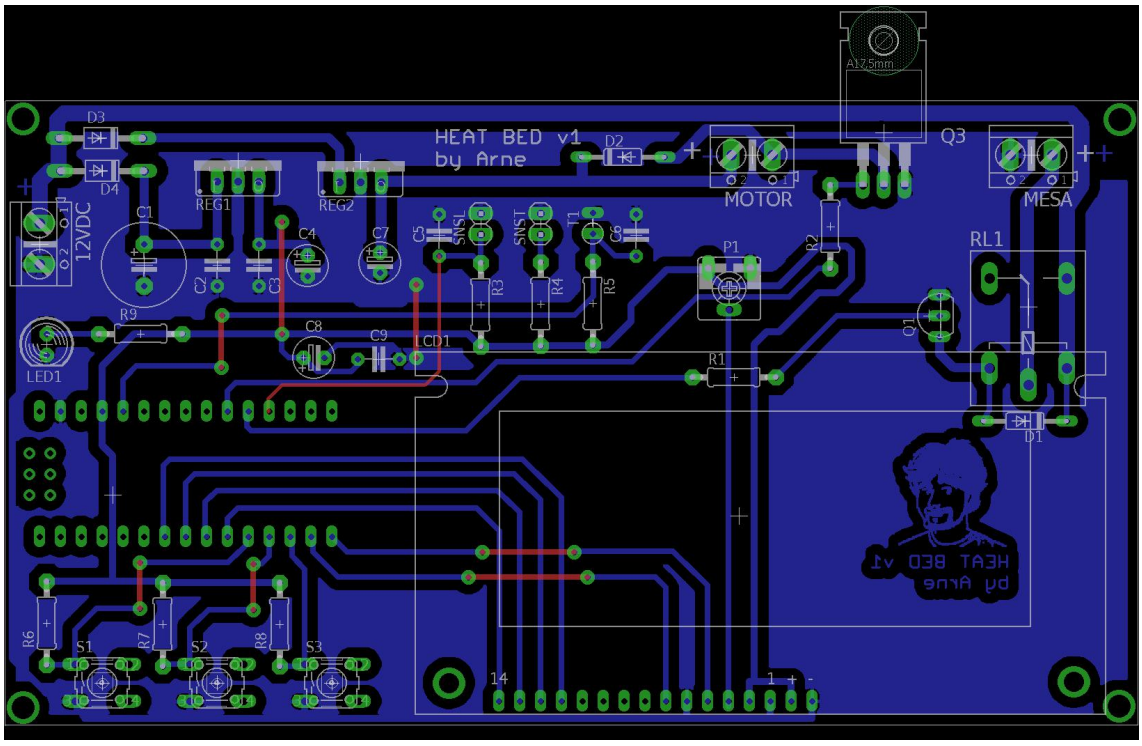


Figura 2 - Layout da placa de circuito impresso.

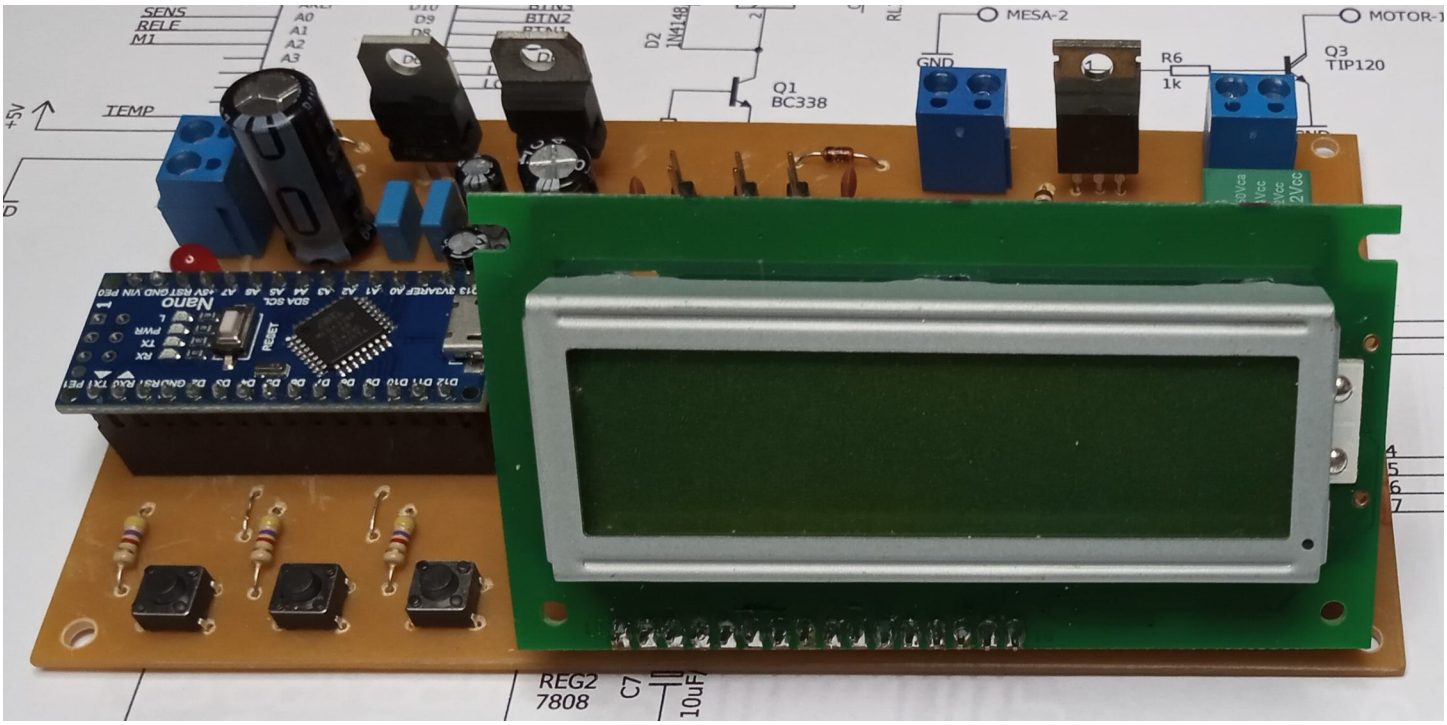


Figura 3 - Placa protótipo montada pelo autor.

Comece montando os componentes não polarizados como jumper's (em vermelho na **figura 2**), resistores, capacitores cerâmicos e capacitores de poliéster. Tenha cuidado ao montar os componentes polarizados como diodos, LEDs, transistores, reguladores de tensão e capacitores eletrolíticos.

Para a montagem de U1 é recomendável o uso de barra de pinos tipo fêmea para auxiliar numa eventual extração do componente se e quando for necessário. O mesmo se aplica a LCD1 que poderá ser do tipo 16x2 ou ainda 20x4 (quantidade de colunas x quantidade de linhas). O programa aceita ambos.

As chaves S1, S2 e S3 são do tipo táctil 6x6 mm para montagem em PCI's e a altura do pino de ação nas mesmas é de livre escolha. Essas chaves também podem ser do tipo aérea "normalmente abertas", dependendo exclusivamente da montagem mecânica que o leitor adotará.

O uso de barras de pinos para as conexões do sensor IR também é recomendável, mas o leitor poderá montar o mesmo usando fios soldados diretamente entre o sensor e a placa, sem nenhum problema.

É necessário o uso de dissipadores de calor para REG1 e REG2. Já para Q2 dependerá muito do consumo do motor M1.

Para M1 o autor usou um motor antigo com caixa de redução 1:35 com alimentação de 6 VDC e corrente de trabalho entre 300mA e 500mA máximos. A fonte através de REG2 pode fornecer no máximo 1A. Porém o leitor poderá usar um motor mais moderno como o

conhecido “amarelinho” que pode ser facilmente encontrado no mercado especializado e tem basicamente as mesmas características do motor adotado pelo autor.

O leitor deve ter percebido que REG2 fornece 8VDC e o motor recomendado foi de 6VDC. Note que a queda de tensão de um transistor darlington NPN na sua junção base emissor (VBE) é de aproximados 1,2V à 1,4V. A tensão sobre o motor M1 será então adequada conforme descrito seguir:

$$V_{motor} = VREG2_{Saída} - VBE_{Q2} \simeq 6{,}6V$$

O uso de conectores do tipo KRE (parafusáveis na PCI) pode ajudar na conexão dos itens externos como o motor M1, a mesa aquecida e a entrada da fonte externa de 12VDC. Na sua falta o leitor poderá soldar diretamente os fios na placa.

A fonte externa usada e recomendada pelo autor é do tipo chaveada 12 VDC com 10 A no mínimo. Essa corrente é necessária devido ao consumo da mesa aquecida quando ligada.

Falando sobre a mesa, o autor adaptou ao projeto uma do tipo usada em impressoras 3D, com medidas de 200 x 200 mm, espessura de 1,6 mm, alimentação de 12VDC e temperatura de trabalho de aproximadamente 120°C máximos. Esta mesa pode ser facilmente adquirida no comércio especializado em impressoras 3D e/ou internet. Veja mais detalhes sobre a mesma na lista de materiais.

Montagem mecânica

Todo o projeto foi fixado em uma base de madeira compensado de 25mm de espessura. **A figura 4** mostra as medidas desta base, além de outras medidas para que o leitor possa realizar a sua montagem. Já

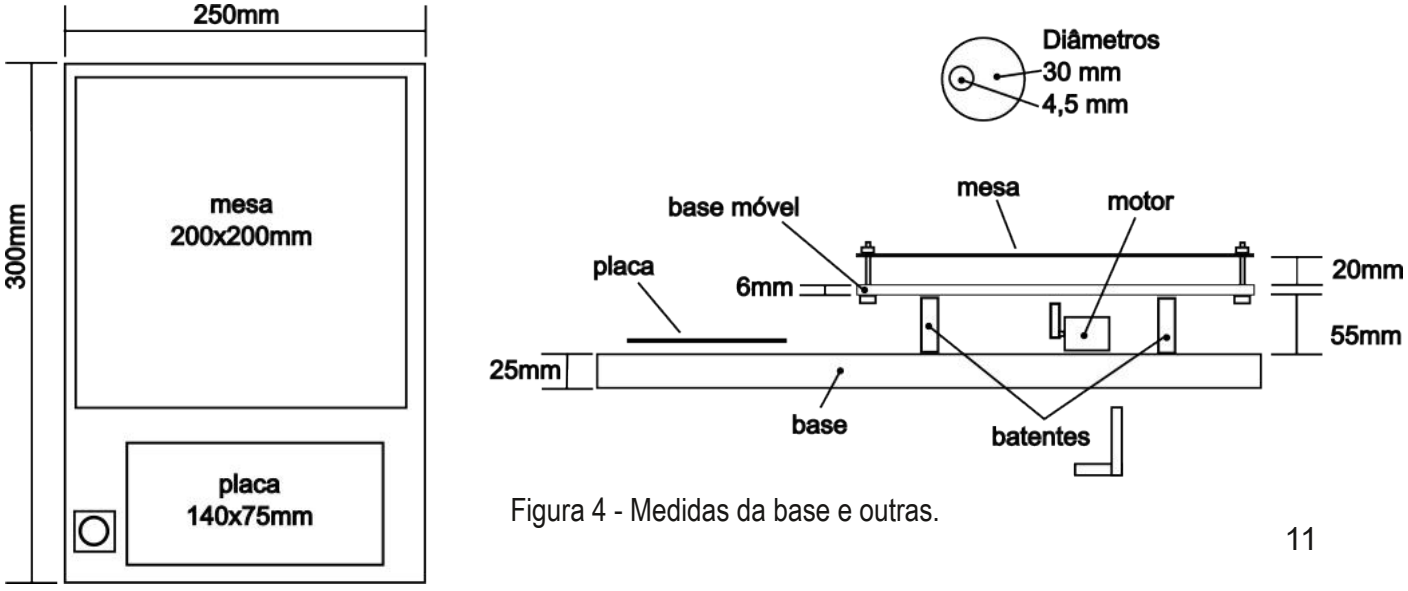


Figura 4 - Medidas da base e outras.

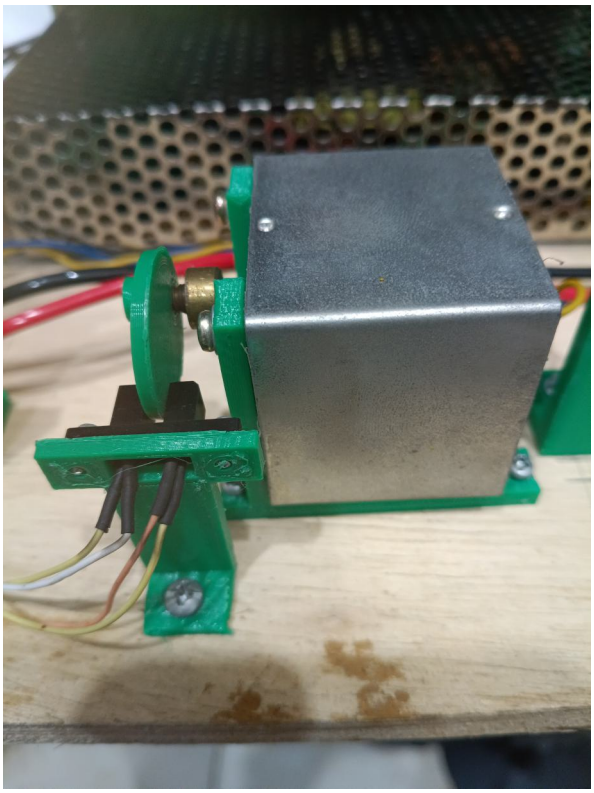


Figura 5 - Fixação do motor e sensor IR.

a **figura 5** detalha a montagem do conjunto motor/sensor IR montado pelo autor.

A mesa aquecida deve ser suportada por uma base móvel de madeira. Para prender a mesa a esta base o leitor deverá utilizar parafusos com 3mm x 45 mm, além de porcas e arruelas para melhor fixação. A mesa não deve encostar de forma alguma a esta base. Veja a **figura 6**.

Para permitir que a mesa seja "agitada" pelo motor, o autor desenhou um conjunto de dobradiças e batentes e os imprimiu em sua impressora 3D. Estas partes garantem a movimentação lateral da base móvel sobre a base, além da sua própria fixação à base através das dobradiças.

Uma circunferência com furo deslocado do centro também foi desenhada e impressa (peça de agitação). É essa peça que permite

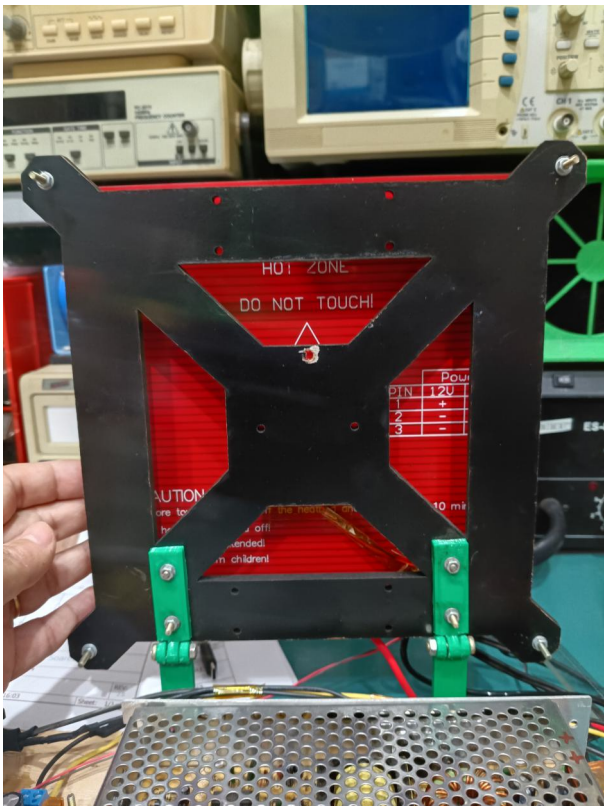


Figura 6A e 6B - Base móvel - mesa aquecida.

que um dos lados da base móvel seja “levantada” quando o motor gira. A montagem precisa garantir que quando o motor girar, a parte alta da peça erga a base móvel da mesa e a parte baixa garanta que a base móvel fique apoiada nos batentes. Veja os detalhes da peça de agitação na **figura 7**.

O leitor que não possuir uma impressora 3D poderá utilizar dobradiças comuns adquiridas no comércio especializado e pequenos pedaços de madeira para fazer a fixação das mesmas à parte móvel e também construir os batentes. Tudo deve ser fixado à base principal de madeira. Use as imagens do artigo como referência para a sua montagem.

O NTC deve ser fixado na parte de baixo da mesa, bem no centro e para isso é necessário utilizar a famosa fita Kapton para alta temperatura e a mesma pode ser facilmente adquirida em lojas especializadas em impressoras 3D e/ou internet.

Foi incluído uma chave liga/desliga no sistema para facilitar o seu uso (figura 8). Essa chave foi retirada de um gabinete de PC antigo fora de uso e a mesma ganhou uma “caixa”, também desenhada pelo autor e impressa em sua impressora 3D. Porém, mais uma vez aqui, o leitor poderá utilizar as peças/partes que tiver em mãos, fazendo uso de sua criatividade.

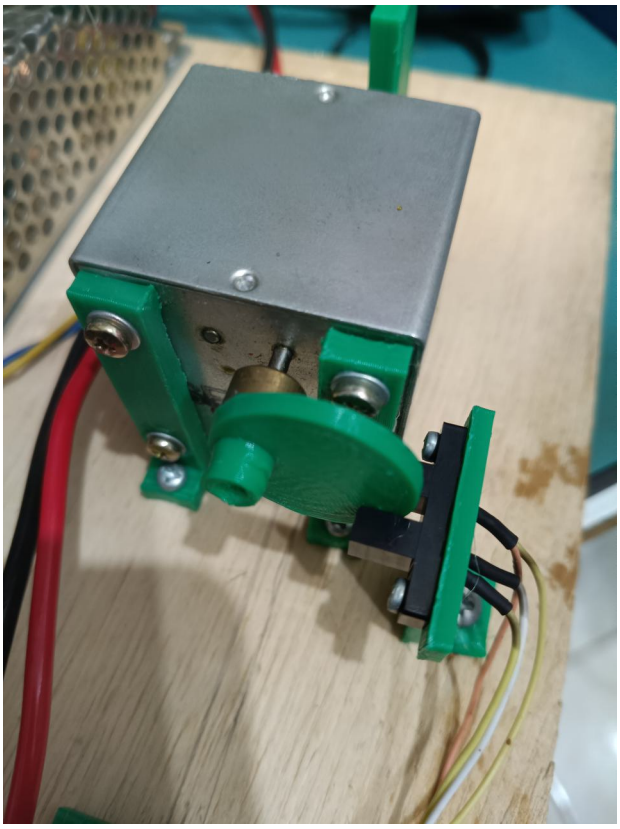
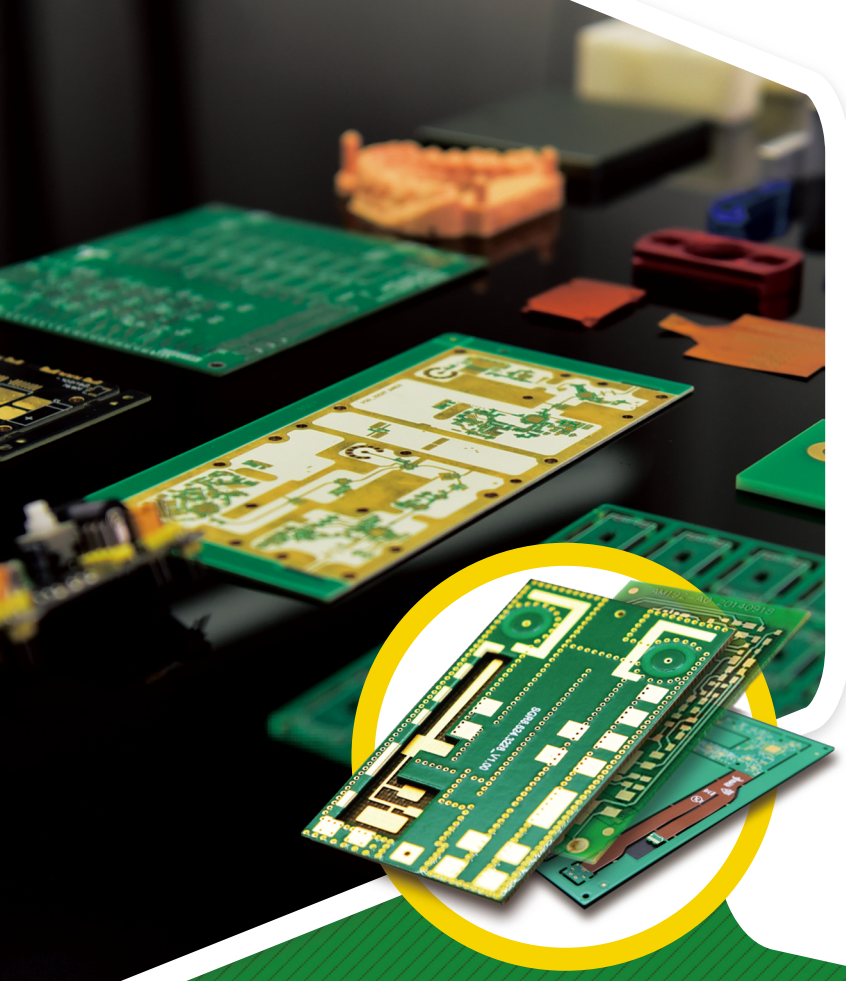


Figura 7 - Peça de agitação.

Figura 8 - Chave liga/desliga do sistema

Programa

A **figura 9** mostra o fluxograma do programa Arduino desenvolvido para U1. Após inserir as bibliotecas necessárias



PCBWay

Seu Parceiro Confiável de Manufatura no Exterior

Experiência sem complicações para todos. Comece sua jornada em PCB com segurança!

Prototipagem de PCB rápida e acessível

Especificação	Preço de Referência	Prazo Padrão	Prazo Expresso
2 camadas – 100 x 100 mm – 10 pcs	US$ 5	24 horas	12 horas
4 camadas – 100 x 100 mm – 5 pcs	US$ 25.97	4-5 dias	24 horas
6 camadas – 100 x 100 mm – 5 pcs	US$ 99.04	5-6 dias	24 horas

Pedido com quantidade mínima

Atendemos pequenos pedidos com PCBs padrão a partir de 5 unidades e outros tipos a partir de 1 unidade. Perfeito para prototipagem e projetos personalizados.

Sistema de cotação instantânea

Basta inserir as especificações no site oficial que o sistema calcula instantaneamente o preço de referência e a data de entrega.

Rigoroso controle de qualidade

Certificações internacionais obtidas, como ISO 9001, ISO 14001, UL e RoHS. A qualidade estável é garantida por meio de um sistema de gestão de qualidade certificado.

Atendimento 24/7

Cada cliente conta com um representante de atendimento exclusivo, e o chat ao vivo também está disponível 24 horas por dia.

www.pcbway.com

service@pcbway.com

(arquivos), preparar as definições e iniciar as variáveis de uso global, o Setup() do Arduino inicializa os pinos de I/O, o LCD e envia as primeiras mensagens para o usuário.

Na sequência em Loop(), as chaves são lidas, assim como a temperatura da mesa e o estado do sensor, e também o LCD é mantido com as mensagens atualizadas. Se uma chave for pressionada o programa analisa qual configuração foi solicitada (tempo de agitação ou temperatura) e abre no LCD um cursor sobre o ponto de acerto. Ao final da configuração desejada o processo é iniciado com o aquecimento da mesa e agitação da mesma (se esta foi configurada). A listagem 1 mostra o programa principal e as listagens 2 e 3 a biblioteca para uso do LCD desenvolvida pelo autor.

O autor não previu nenhuma caixa para a placa, mas o leitor poderá fazê-lo se assim o desejar.

Obs.*: Todos os arquivos STL referentes a esta montagem estão disponíveis no perfil do Thingiverse do autor.*

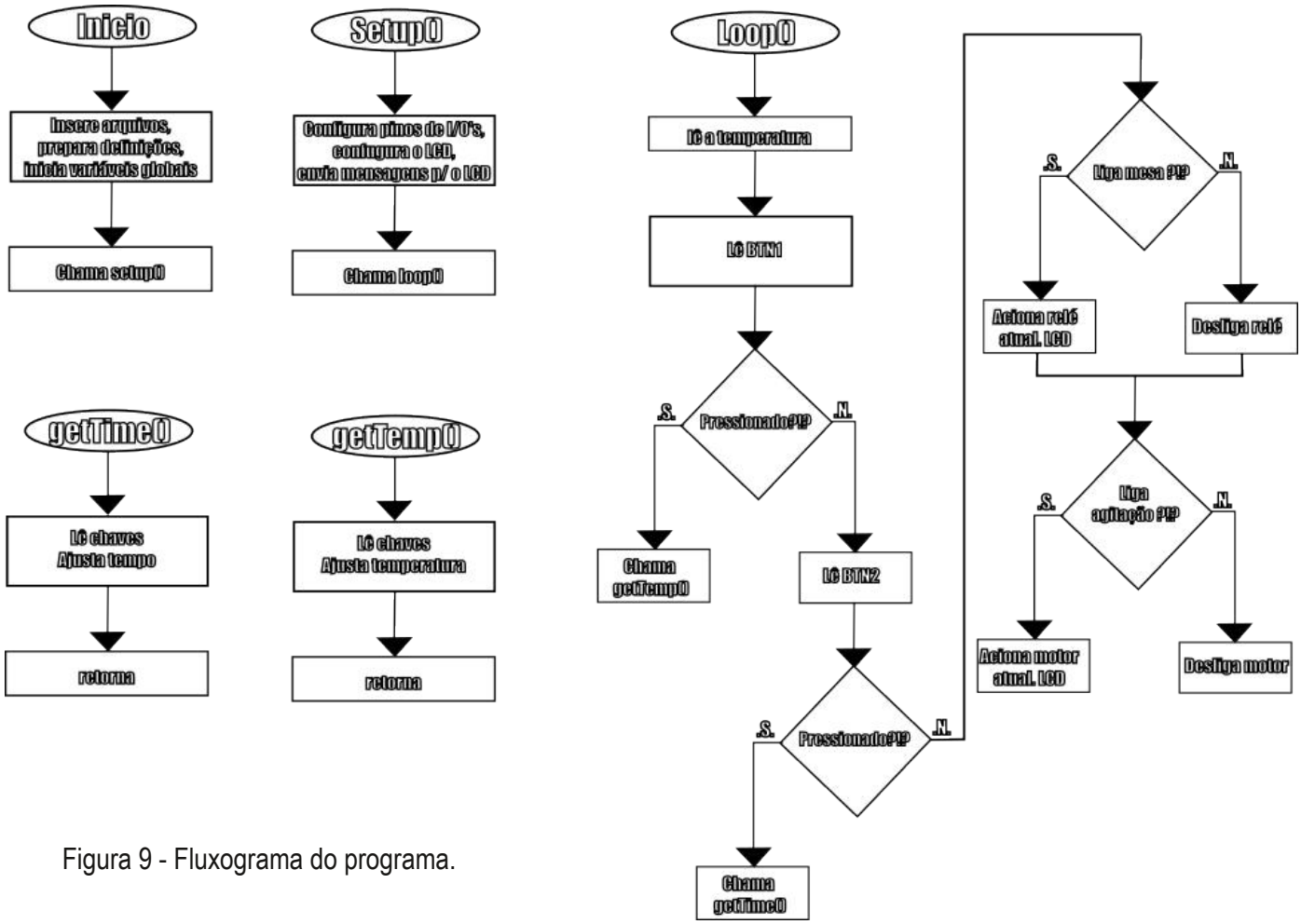


Figura 9 - Fluxograma do programa.

Arquivo heatbed.ino

```
// ********************************************************************************
// Projeto - Heatbed Arduino - Mesa aquecida para auxiliar
//confecção de PCI
// Desenvolvido por: Eng. Márcio José Soares
// versão: 1.0 - 06/11/2023
// Permite controlar o aquecimento de uma "mesa" heatbed MK2 para acelerar o processo de
//oxidação com percloreto de ferro e também fazer uma pequena agitação
// Editor: Kate
// Compilador: arduino-cli (linha de comando)
// Plataforma: Arduino Nano
// Auxiliares: Mesa MK2 e motor DC com caixa de redução
// Pinos para Arduino Nano/Uno:
// Mesa:
//       A0 - Motor DC para agitação
//       A1 - controle de aquecimento da mesa (aquecimento)
//       A7 - sensor NTC
//       LCD
//       4 - LCD D4
//       5 - LCD D5
//       6 - LCD D6
//       7 - LCD D7
//       12 - LCD RS
//       11 - LCD EN
// Botões de setup/controle
//       8 - BTN1
//       9 - BTN2
//       10 - BTN3
// ********************************************************************************
// ********************************************************************************
//       Arquivos incluídos
// ********************************************************************************
extern "C" {
#include "lcd_avr.h" }
// ********************************************************************************
//       Definições do módulo
// ********************************************************************************
#define         BTN3      10                  // configura botões
#define         BTN2      9                   //
#define         BTN1 8
#define         LED       13
#define         M1        16                  // A2
#define         RELE      15                  // A1
#define         SENSOR 14                     // A0
#define         TEMP      7                   // A7
#define         SAMPLES           5
#define         minTemp 30
#define         maxTemp 70
```

```
#define         hysTemp 3
#define         minTime 15
#define         maxTime 90
#ifdef          LCD_20x4
#define         linTemp 2
#define         linTempo 3
#endif
#ifdef          LCD_16x2
#define         linTemp 0
#define         linTempo 1
#endif
#define  __DEBUG__      // libera debug via UART
#define RT0 10000       // valor da resistência do NTC em Ω
#define B 4038          // K obitido do manual 3977 3380 4038
#define VCC 5           // alimentação
#define R 10000         // R=10KΩ
// ***************************************************************************
//                      Define o tipo de ativação do rele - NPN ou PNP
//                      selecionar apenas uma
//
#define                 RELE_NORM // trabalha com acionamento de relés de forma normal - NPN
//#define               RELE_INV // trabalha com acionamento de relés de forma invertida - PNP
#ifdef                  RELE_INV
#define                 myON    0
#define                 myOFF   1
#endif
#ifdef                  RELE_NORM
#define                 myON    1
#define                 myOFF   0
#endif
// ***************************************************************************
//                      Constantes do módulo
// ***************************************************************************
// ***************************************************************************
//                      Instâncias
// ***************************************************************************
// ***************************************************************************
//                      Variávels globais
// ***************************************************************************
float   RT, VR, ln, TX, T0, VRT, Tc;
char    buf[10];
int     amostras[SAMPLES];
uint8_t         liga = 0;
uint8_t         ligaA = 0;
uint8_t         counter = 0;
uint8_t         timePass = 0;
uint8_t         LEDstate = 1;
int     tempUser = minTemp;
int     tempoUser = minTime;
```

```
// ********************************************************************************
//      Declara funções do módulo
// ********************************************************************************
void                        tempRead(void);
void                        getTemp(void);
void                        getTime(void);
// ********************************************************************************
//      Setup
// ********************************************************************************
void setup(void){
        pinMode(BTN1, INPUT);                   // btns
        pinMode(BTN2, INPUT);
        pinMode(BTN3, INPUT);
        pinMode(SENSOR, INPUT);// sensor do motor
        pinMode(LED, OUTPUT);                   // LED
        pinMode(RELE, OUTPUT);    // rele
        pinMode(M1, OUTPUT);      // motor
        digitalWrite(LED, LEDstate);
        digitalWrite(RELE,myOFF);
        digitalWrite(M1,LOW);
        T0 = 25 + 273.15;
        delay(500);
        initLCD(); //usa lib externa
        clearLCD(0);
        delay(1000);
#ifdef LCD_20x4
        pos_LCD(0,5);
        stringR_LCD("Bem-vindo");
        pos_LCD(1,3);
        stringR_LCD("ao laboratorio");
        pos_LCD(2,6);
        stringR_LCD("do Arne");
#endif
#ifdef LCD_16x2
        pos_LCD(0,1);
        stringR_LCD("HeatBED p/ PCI");
        pos_LCD(1,4);
        stringR_LCD("by Arne");
#endif
        delay(3000);
        clearLCD(0);
#ifdef LCD_20x4
        pos_LCD(0,3);
        stringR_LCD("HeatBED p/ PCI");
#endif
        pos_LCD(linTemp,0);
        stringR_LCD("Temp : / ");
        dataC(0xDF);
        dataC('C');
```

```
        pos_LCD(linTemp,10);
        sprintf(buf, "%2d", minTemp);
        stringR_LCD(buf);
        pos_LCD(linTempo,0);
        stringR_LCD("Tempo: 00/ seg");
        pos_LCD(linTempo,10);
        sprintf(buf,"%2d", minTime);
        stringR_LCD(buf);
        tempUser = minTemp;
        digitalWrite(RELE,myON);
        delay(500);
        digitalWrite(RELE,myOFF); }
// ********************************************************************************
//                  Loop
// ********************************************************************************
void                loop(void){
// ********************************************************************************
// Leitura da temperatura
        tempRead(); // lê temperatura
// ********************************************************************************
        //          Controle dos botões de configuração
        if(!digitalRead(BTN1)){         // se botão 1 pressionado
        while(!digitalRead(BTN1));      // enquanto não soltar, fica!
        delay(30);                      // bounce
        getTemp();                      // coleta temperatura do usuário }
        if(!digitalRead(BTN2)){         // se botão 2 pressionado
        while(!digitalRead(BTN2));      // enquanto não soltar, fica!
         delay(30);                     // bounce
        getTime();                      //coleta tempo de agitação }
        // ********************************************************************************
        // Controle da mesa
        if(liga){   // pediu para ligar?!
                    if ((int)Tc <= (tempUser-hysTemp)){
                                digitalWrite(RELE, myON);     //
#ifdef                          LCD_20x4
                                pos_LCD(linTemp,15);
                                dataC('H');
#endif }
                    else if((int)Tc >= tempUser){
                                digitalWrite(RELE,myOFF);     //
#ifdef                          LCD_20x4
                                pos_LCD(linTemp,15);
                                dataC('                       ');
#endif } }
        else {digitalWrite(RELE, myOFF); }
// ********************************************************************************
        //          Controle do motor
        if(ligaA && liga){      // liga o motor?!
        counter++;              // conta tempo a cada 500ms
```

```
                if((counter >= tempoUser*2)){  // tempo de acordo
#ifdef                  LCD_20x4
                        pos_LCD(linTempo,17);
#endif
                counter = 0;
                digitalWrite(M1, HIGH);         // liga motor
                while(digitalRead(SENSOR))  // faz até sensor ser desbloqueado
                                delay(50);
                while(!digitalRead(SENSOR)) // sensor desbloqueado, motor girando
                                delay(50);
                if(digitalRead(SENSOR)){        // se chegou no sensor, desliga M1
                                digitalWrite(M1,LOW);           // desliga motor
                                delay(50); }
                        timePass = 0; }
                else{
                        if(!(counter%2)){               // contou duas vezes
                                timePass++;
                                sprintf(buf, "%02d", timePass);
        pos_LCD(linTempo,7);
        stringR_LCD(buf); } } }
// ***************************************************************************
// Controle do LED Live
        LEDstate ^= 0x01;               // XOR
        digitalWrite(LED, LEDstate);    // troca estado do LED }
// ***************************************************************************
//      Função para ler a temperatura do termistor
//      Entradas - nenhuma
//      Saídas - nenhuma
// ***************************************************************************
void    tempRead(void){
        uint8_t     i;
        float               average = 0;
        for(i=0;i<SAMPLES;i++){
                VRT = analogRead(TEMP); // lê valor analógico
                average += VRT;                         // soma
                delay(100);                             // aguarda }
        VRT = average / SAMPLES;    // pega valor médio
        VRT = (5.00 / 1023.00) * VRT; // converte para tensão
        VR = VCC - VRT;
        RT = VRT / (VR / R);            // resistência RT
        ln = log(RT / RT0);
        TX = (1 / ((ln / B) + (1 / T0)));   // temperatura do termistor
        TX = TX - 273.15; // converte para Celsius
        Tc = truncf(TX * 10.0)/10.0;
        if(Tc < 0)                      // se indicação menor que zero
        Tc = 0.00;                      // mostra zero... moro nos trópicos!!! ;)
        dtostrf(Tc,2, 0, buf);          //double, tamanho, precisão, buffer
        pos_LCD(linTemp,7);
        stringR_LCD(buf); }
```

```
// ***************************************************************************
//      Função para pegar a temperatura do usuário
//      Entradas - nenhuma
//      Saídas - nenhuma
// ***************************************************************************
void getTemp(void) {
        uint8_t     minT = (uint8_t)minTemp;
        uint8_t     dezena = minT/10;
        uint8_t     unidade = minT%10;
        digitalWrite(RELE, myOFF);
        digitalWrite(M1, LOW);
        liga = 0;
        pos_LCD(linTemp,10);
        dataC(dezena + 0x30);
        dataC(unidade + 0x30);
        pos_LCD(linTemp,10);
        while(digitalRead(BTN1)){       // fica na função enquanto não pressionado
        if(!digitalRead(BTN2)){         // BTN2 pressionado?!
        pos_LCD(linTemp,10);
                                while(!digitalRead(BTN2));      // enquanto não soltar, fica!
                                delay(30);                      // bounce
                                dezena    += 1;                 // soma dezena
                                if(dezena >= 10)                // se passar...
                                        dezena = minT/10;
                                dataC(dezena + 0x30);
                                pos_LCD(linTemp,10); }
                        if(!digitalRead(BTN3)){
                                pos_LCD(linTemp,11);
                                while(!digitalRead(BTN3));      // enquanto não soltar, fica!
                                delay(30);                      // bounce
                                unidade += 1;           // soma unidade
                                if(unidade >= 10)               // se passar...
                                        unidade = minT%10;
                                dataC(unidade + 0x30);
                                pos_LCD(linTemp,11); }
                        tempUser = ((dezena * 10) + unidade) * 1.0; }
        if((tempUser <= maxTemp) && (tempUser >= minTemp)){
                        liga = 1; } }
// ***************************************************************************
//                              Função para pegar o tempo do usuário do usuário
//                              Entradas - nenhuma
//                              Saídas - nenhuma
// ***************************************************************************
void getTime(void){
        uint8_t dezena = minTime/10;
        uint8_t unidade = minTime%10;
        while(digitalRead(BTN1)){       // fica na função enquanto não pressionado
        if(!digitalRead(BTN2)){         // BTN2 pressionado?!
        pos_LCD(linTempo,10);
```

```
        while(!digitalRead(BTN2));      // enquanto não soltar, fica!
        delay(30);                      // bounce
        dezena    += 1;                 // soma dezena
        if(dezena >= 10)                // se passar...
        dezena = minTime/10;
        dataC(dezena + 0x30);
        pos_LCD(linTempo,10); }
                if(!digitalRead(BTN3)){
        pos_LCD(linTempo,11);
        while(!digitalRead(BTN3));      // enquanto não soltar, fica!
        delay(30);                      // bounce
        unidade += 1;                   // soma unidade
        if(unidade >= 10)               // se passar...
        if(dezena == 1)
        unidade = minTime%10;
        else
        unidade = 0;
                dataC(unidade + 0x30);
                pos_LCD(linTempo,11); }
                tempoUser = ((dezena * 10) + unidade); }
        if((tempoUser <= maxTime) && (tempoUser >= minTime)){
                ligaA = 1;
                counter = 0;
                timePass = 0; } }
Arquivo lcd_avr.cpp
// ***************************************************************************
//      Funções para uso de LCDs tipo caracter - modo transferência 8 bits
//              Desenvolvido por Márcio José Soares
//              Microcontrolador: AVR
//              Compilador: avr-gcc (GCC) 4.1.0 - Linux
//              Última alteração: 09/08/2007
// ***************************************************************************
#pragma once
// ***************************************************************************
// Arquivos incluídos no módulo
// ***************************************************************************
#include "lcd_avr.h"
#include "Arduino.h"
// ***************************************************************************
//Inicia LCD - configura pinos de I/O
//Entradas - nenhuma
//Saidas - nenhuma
// ***************************************************************************
void initLCD(void){
        //configura pinos de controle como saída
#ifdef                          __USE__RW__
        DDRB    = (1<<Hab) | (1<<RW) | (1<<RS);
#else
        DDRB    = (1<<Hab) | (1<<RS);
```

```
#endif
#ifdef __LCD_4__
        //configura pinos de dados como saída
        DDRD = (1<<LCDD4) | (1<<LCDD5) | (1<<LCDD6) | (1<<LCDD7);
#endif
#ifdef __LCD_8__
        //configura pinos de dados como saída
        DDRD = 0xFF;        //port como saída
#endif
        resLCD;             //configura pinos
#ifdef                      __USE__RW_
        writeLCD;
#endif
        dataLCD;
        config_LCD(); }
// ***************************************************************************
//Pulsa pino E (habilitação) no display
//Entradas - nenhuma
//Saidas - nenhuma
// ***************************************************************************
void strobe(void){
        setLCD;             //pulsa pino de controle habilitação
        resLCD; }
// ***************************************************************************
//Envia um byte para o display - dados ou comando
//Entradas - byte a enviar
//Saidas - nenhuma
// ***************************************************************************
void sendC(byte val){
        delay(1);
        pLCDdt    = val;
        strobe();   // strobe pino E
#if defined(__LCD_4__)              // se modo 4 bits, envia segundo nible
        pLCDdt    = val<<4; // envia nibble menos significativo
        strobe();           // mais um strobe
#endif }
// ***************************************************************************
//Envia dados para o display
//Entradas - dado a enviar
//Saidas - nenhuma
// ***************************************************************************
void dataC(byte val){
#ifdef __USE__RW_
        writeLCD;           //RW em 0
#endif
        dataLCD;            //RS em 1
        sendC(val);         //escreve dado  }
// ***************************************************************************
//Envia comandos para o display
```

```
//Entradas - byte de comando
//Saidas - nenhuma
// ***************************************************************************
void commandC(byte val){
#ifdef __USE__RW_
        writeLCD;               //RW em 0
#endif
        commLCD;                        //RS em 0
        sendC(val);                      //escreve comando }
// ***************************************************************************
//Apaga LCD
//Entradas        - nenhuma
//Saidas           - nenhuma
// ***************************************************************************
void   clearLCD(byte modo){
        byte                    i;
        switch(modo){
                case 0:
                        commandC(0x01);    //apaga todo LCD
                        break;
                case    1:
                        commandC(0x80);    //muda para linha 1
        for         (i=0; i<NRCOLS; i++)            //envia x caracteres 'espaço'
dataC('                                 ');
commandC(0x80);                         //muda     para linha 1 novamente
break;
        case    2:
        commandC(0xC0);                 //muda     para linha 2
        for (i=0; i<NRCOLS; i++) //envia x caracteres 'espaco'
        dataC('                         ');
        commandC(0xC0);                 //muda     para linha 2 novamente
        break;
#ifdef LCD_20x4
        case    3:
        commandC(0x94);                 //muda     para linha 3
        for (i=0; i<NRCOLS; i++) //envia x caracteres 'espaco'
        dataC('                         ');
        commandC(0x94);    //muda para linha 3 novamente
        break;
        case    4:
        commandC(0xD4);                 //muda     para linha 4
        for (i=0; i<NRCOLS; i++) //envia x caracteres 'espaco'
        dataC('                         ');
        commandC(0xD4);                 //muda     para linha 4 novamente
        break;
#endif
        default:                        //se       valor passado não confere
        commandC(0x01);                 //apaga todo LCD
        break; }
```

```
        for (i=0; i<15; i++)
                delay(1);   // espera 15 ms }
// *****************************************************************************
//Posiciona cursor no LCD
//Entradas      - nr da linha (0 a 3) - nr da coluna (0 a NRCOL-1)
//Saidas                                - nenhuma
// *****************************************************************************
void pos_LCD(byte lin, byte col){
byte   my_command = 0;
        if(lin      >= 0 && lin <= 3){
                if(col >= 0 && col <= (NRCOLS-1)){
                        switch(lin){
                                case 0:
                                my_command = 0x80;
                                break;
                                case 1:
                                my_command = 0xC0;
                                break;
                                #ifdef      LCD_20x4
                                case 2:
                                my_command = 0x94;
                                break;
                                case 3:
                                my_command = 0xD4;
                                break;
                                #endif
                                default:
                                my_command = 0x80;}
                        commandC(my_command+col); } } }
// *****************************************************************************
//Muda linha no LCD
//Entradas - nr da linha (1 a 4)
//Saidas - nenhuma
// *****************************************************************************
void mudalinha(byte linha){
        if (linha==1)
                commandC(0x80);             //muda     para linha 1
        if (linha==2)
                commandC(0xC0);             //muda para linha 2
                #ifdef      LCD_20x4
        if (linha==3)
                commandC(0x94);             //muda     para linha 3
        if (linha==4)
                commandC(0xD4);             //muda     para linha 4
                #endif }
// *****************************************************************************
//Inicializa LCD
//Entradas - nenhuma
//Saidas - nenhuma
```

```
// ***************************************************************************
void config_LCD(void){
        byte i;
        for (i=0; i< 100; i++)
                delay(4); // power on em espera de 400ms
        #if defined(__LCD_4__)
        commandC(0x28);                         // modo 4-bits, 2-linha, caracter 5x7
        commandC(0x28);                         // repete comando
        #endif
        #if defined(__LCD_8__)
        commandC(0x38);                 // modo 8-bits, 2-linha, caracter 5x7
        commandC(0x38);   // repete comando
        #endif
        commandC(0x06);   // incrementa caracter a direita, desliga display shift
        commandC(0x0F);   // display ligado, cursor ligado, cursor blink
        clearLCD(0);            // apague display }
// ***************************************************************************
//Envia string para LCD
//Entradas - nenhuma
//Saidas - nenhuma
// ***************************************************************************
void   stringR_LCD(char *msg){
        while(*msg != 0x00){              // faz até encontrar fim da string
                dataC(*msg++);                  // envia caracter por caracter } }
Arquivo lcd_avr.h
/// ***************************************************************************
//      Diretrizes de pré-compilação do módulo lcd_avr.c
//      Desenvolvido por Márcio José Soares
//      Microcontrolador: AVR
//      Compilador: avr-gcc (GCC) 4.1.0 - Linux
//      Última alteração: 09/08/2007
// ***************************************************************************
#ifndef LCD_AVR_H
#define LCD_AVR_H
#ifdef __cplusplus
extern "C" {
#endif
// ***************************************************************************
//      Arquivos incluídos no módulo
// ***************************************************************************
#include <stdio.h>
#include <stdint.h>
#include <avr/io.h>
// ***************************************************************************
//Definições do módulo
// ***************************************************************************
#define byte uint8_t
//#define __USE__RW__  // define se usa pino R/W
// ***************************************************************************
```

```
//Modo de uso do LCD - selecionar apenas um
// ****************************************************************************
#define __LCD_4__         //modo de operação com 4 bits
//#define __LCD_8__       //modo de operação com 8 bits
// ****************************************************************************
//Tipo do LCD - selecionar apenas um
#define           LCD_16x2
//#define         LCD_20x4
// ****************************************************************************
//* Ligação entre as portas do AVR e o LCD
// ****************************************************************************
#define           pLCDdt    PORTD   //porta para dados no LCD
#define           pLCDct    PORTB   //porta para controle do LCD
/// ***************************************************************************
//* Pinos de controle do AVR para o LCD - uso nas macros
// ****************************************************************************
#define Hab PB3               //En em 11
#ifdef __USE__RW__
#define           RW        PB5                 //R/W em 13
#endif
#define           RS        PB4                 //RS em 12
#ifdef __LCD_4__
#define           LCDD4             PD4
#define           LCDD5             PD5
#define           LCDD6             PD6
#define           LCDD7             PD7
#endif
#ifdef __LCD_8__
#define           LCDD0             PD0
#define           LCDD1             PD1
#define           LCDD2             PD2
#define           LCDD3             PD3
#define           LCDD4             PD4
#define           LCDD5             PD5
#define           LCDD6             PD6
#define           LCDD7             PD7
#endif
#ifdef            LCD_20x4
#define           NRCOLS 20
#define           NRLINS 04
#endif
#ifdef            LCD_16x2
#define           NRCOLS 16
#define           NRLINS 02
#endif
// ****************************************************************************
//* Pinos de controle do AVR para o LCD - uso nas macros
// ****************************************************************************
#define           pHab      0x08      //0b00001000        - habilita LCD
```

```
#ifdef __USE__RW__
#define         pRW     0x02    //0b00000010        - escrita(0) ou leitura(1) no LCD
#endif
#define         pRS     0x10    //0b00010000        - comando(0) ou dado(1) no LCD
// ***************************************************************************
//* Macros para uso do LCD
// ***************************************************************************
#define         setLCD          pLCDct |= pHab                  // habilita LCD
#define         resLCD          pLCDct &= ~pHab         // desabilita LCD
#ifdef          __USE__RW__
#define         writeLCD  pLCDct &= ~pRW                        // escreve no LCD
#define         readLCD   pLCDct |= pRW                         // lê do LCD
#endif
#define         commLCD         pLCDct &= ~pRS          // escreve comando no LCD
#define         dataLCD   pLCDct |= pRS                 // escreve dado no LCD
// ***************************************************************************
//* Declara funções do módulo
// ***************************************************************************
void            initLCD(void);
void            strobe(void);
void            sendC(byte val);
void            dataC(byte val);
void            commandC(byte val);
void            clearLCD(byte modo);
void            mudalinha(byte linha);
void            pos_LCD(byte lin, byte col);
void            config_LCD(void);
void            stringR_LCD(char *msg);
#ifdef __cplusplus }
#endif
#endif
```

Teste e uso

Antes de iniciar os testes é altamente recomendável verificar todas as conexões. Esse tempo não será “perdido” e pode ajudar a perceber um erro antes mesmo da famosa “fumaça indicadora de erros”! Seja criterioso em sua verificação! Não tenha pressa! Verifique as trilhas da placa, possíveis trocas de componentes na mesma, inversão no caso dos componentes polarizados, conexões das partes externas, fonte, etc. Após a verificação, grave o programa no Arduino e instale-o na placa.

Ligue o circuito e verifique o display. Após as mensagens de inicialização, será apresentada a mensagem conforme a **figura 10**. É hora de selecionar a configuração desejada para temperatura e o tempo desejado para agitação.

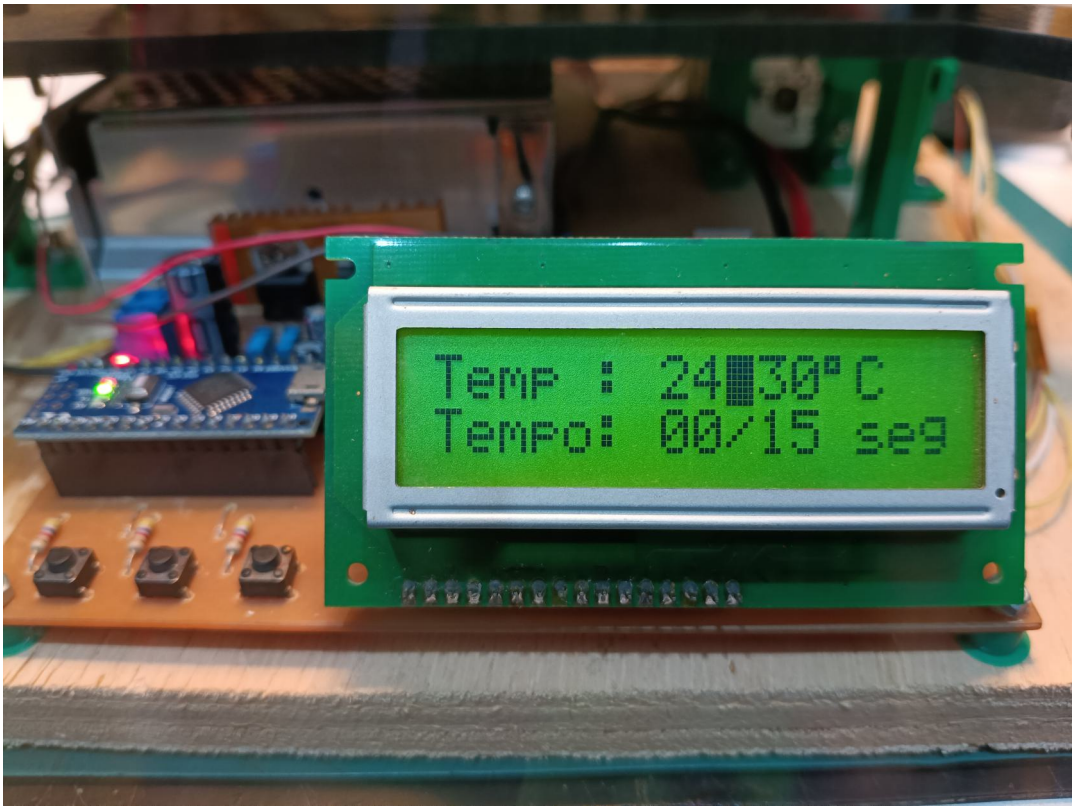


Figura 10 - LCD após inicialização.

Pressione S1 para habilitar a configuração da temperatura. Use S2 e S3 para acertar a dezena e unidade, respectivamente. Após configurar, pressione S1 mais uma vez. Nesse momento o relé será acionado ligando a mesa e o leitor poderá acompanhar a temperatura subindo via LCD até o que foi configurado. A histerese é de 3°C.

Pressione duas vezes S2 para entrar no modo de configuração do tempo (em segundos) para agitação. Aqui novamente S2 acerta a dezena e S3 a unidade. S1 confirma sua escolha. Com isso pronto, o leitor poderá via LCD observar a contagem do tempo que ao chegar no limite estabelecido fará com que o motor realize uma volta completa até chegar no sensor, quando o mesmo para e aguarda um novo período. Essa volta levantará e abaixará lentamente a mesa. Verifique se tudo está ok!

É importante que o leitor ao usar o sistema use a solução apenas até a metade da vasilha, evitando que durante uma agitação a solução seja jogada para fora!!! Outro ponto importante é que o aquecimento da mesa deve iniciar com a solução dentro da vasilha e esta sobre a mesa! Assim a solução vai aquecendo gradativamente. Neste momento nenhuma agitação deve ser realizada. Apenas quando for iniciar realmente a “oxidação” da sua nova PCI é que você deve configurar o tempo de agitação.

Conclusão

Durante o decorrer do artigo o leitor pode ver que aquecer e agitar a solução de Percloreto de Ferro não é uma bobagem, como muitos por aí costumam dizer. A explicação está na química! Além disso, essa montagem permitirá aumentar a eficiência na preparação da sua PCI! Com o sistema proposto tudo ficará mais fácil e melhor controlado. O que o leitor terá de fazer será apenas observar, de tempos em tempos, se a placa já está pronta! Além disso, com um pouco de criatividade e conhecimento, o leitor também poderá alterar o projeto inserindo novos controles. É o que esperamos! Boa montagem e até a próxima!

Lista de materiais

Semicondutores
U1 – Arduino Uno
D1, D2 – 1N4148 – diodo de sinal
D3, D4 – 1N4004 – diodo retificador
LED1 – redondo 5mm, vermelho difuso
Q1 – BC337 – transistor NPN de uso geral
Q2 – TIP120 – transistor NPN darlington de potência
REG1 – regulador de tensão 7805 – 5VDC/1A – TO220
REG1 – regulador de tensão 7808 – 8VDC/1A – TO220

Resistores (5% - 1/8W)
R1, R2 – 1k (marrom, preto, vermelho)
R3 – 4k7 (amarelo, violeta, vermelho)
R4 – 1k (marrom, preto, vermelho)
R5 – 10k (marrom, preto, laranja)
R6, R7, R8 – 4k7 (amarelo, violeta, vermelho)
R9 – 470R (amarelo, violeta, marrom)

Capacitores
C1 – 1000uF/25V – eletrolítico radial
C2 – 0,33uF/60V – poliéster
C3 – 0,1uF/60V – poliéster
C4 – 100uF/16V – eletrolítico radial
C6, C7 – 100nF/60V – cerâmico
C8 – 10uF/16V – eletrolítico radial
C9 – 100nF/60V – cerâmico

Diversos

LCD1 – LCD tipo carácter 16 colunas x 2 linhas c/ backlight (veja texto)

RL1 – mini rele 5V - 1 contato NA-C-NF 10A

SNSL/SNST – sensor óptico infravermelho tipo barreira - LTR9608 ou similar

M1 – motor com caixa de redução 1:50 mín alimentação DC 6V

Mesa - MK2B PCB 12V

Fonte chaveada 12VDC – 10 à 15A

Placa de circuito impresso virgem com 150 x 100 mm, dobradiças, 4 parafusos M3 35mm com porcas e arruelas, 4 parafusos M3 45 mm com porcas e arruelas, base móvel (veja texto), madeira compensado para base (veja texto), fios, solda, ferramentas, etc.

Referências bibliográficas

ALTANA TUBES. Recuperando percloreto de ferro usado. Disponível em: https://www.altanatubes.com.br/webstore/?c=313&t=Recuperando-percloreto-de-ferro-usado&srsltid=AfmBOopBan--YDaUXYgPBhh5eRbOYsBvZK2WFIAcaX9kfdg7_7ksXc0t. Acesso em: 02 fev. 2026.

ARDUINO. Arduino - Home. 2026. Disponível em: https://www.arduino.cc/. Acesso em: 02 fev. 2026.

BRAGA, Newton C. Como fazer uma placa de circuito impresso (ART494). Disponível em: https://www.newtoncbraga.com.br/projetos-educacionais/3612-art494.html. Acesso em: 02 fev. 2026.

BRAGA, Newton C. O Laboratório de Circuito Impresso (ART3269). Disponível em: https://www.newtoncbraga.com.br/?view=article&id=14298:o-laboratorio-de-circuito-impresso-art3269&catid=52. Acesso em: 02 fev. 2026

SOARES, Márcio José. Aprenda como programar em módulos com Arduino. YouTube, 12 set. 2023. Disponível em: https://www.youtube.com/watch?v=N_gOI3wQa3U. Acesso em: 02 fev. 2026.

SOARES, Márcio José. Como fazer em sua casa PCI's com acabamento semiprofissional. Revista INCB Eletrônica, São Paulo, n. 31, p. 28-38, nov./dez. 2025.

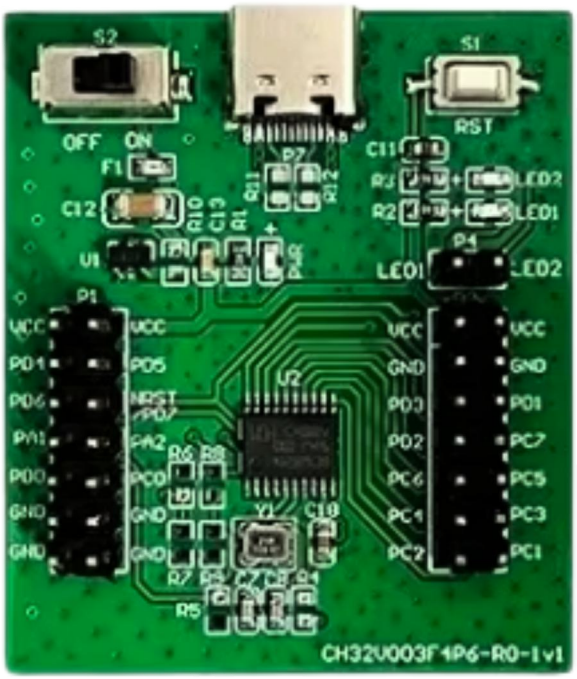
Contatos:

- Página Web – http://www.arnerobotics.com.br

- Instagram - https://www.instagram.com/arnesake/

- YouTube - https://www.youtube.com/c/arnesake

- Thingiverse - https://www.thingiverse.com/arnesake/designs



Microcontrolador Chinês RISC-V de 32 bits de Baixo Custo

Daniel Quispe Marquez

Introdução

O universo dos microcontroladores passou por uma transformação importante nos últimos anos com a consolidação da arquitetura aberta RISC-V International [2]. Diferente das arquiteturas proprietárias tradicionais, a RISC-V permite que fabricantes desenvolvam seus próprios processadores sem pagamento de royalties, estimulando inovação, redução de custos e independência tecnológica.

Dentro desse novo cenário surge a linha de microcontroladores da WCH, que combina a arquitetura RISC-V com baixo custo e excelente disponibilidade no mercado. Um dos representantes mais interessantes dessa família é o CH32V003F4P6, um microcontrolador de 32 bits extremamente compacto, robusto e ideal para aplicações embarcadas de baixo consumo e automação.

Este artigo tem como objetivo servir como um guia completo de introdução para quem deseja começar a trabalhar com essa linha de microcontroladores, abordando desde a arquitetura interna até as ferramentas de desenvolvimento recomendadas.

O que será abordado neste artigo

Ao longo deste conteúdo, o leitor encontrará:

- *Uma explicação clara sobre a arquitetura RISC-V utilizada no CH32V003.*
- *Principais características técnicas do microcontrolador;*
- *Apresentação da IDE oficial MounRiver Studio;*
- *Tipos de gravadores compatíveis e métodos de programação;*
- *A placa de avaliação oficial da WCH;*
- *Configuração inicial do ambiente de desenvolvimento;*
- *Criação de um projeto do zero;*
- *Primeiro exemplo prático: Blink LED;*
- *Estrutura básica de código e organização do projeto.*

O objetivo é proporcionar conhecimento fundamental, permitindo que o leitor tenha segurança para evoluir posteriormente para aplicações mais complexas, como comunicação UART, I2C, PWM, ADC ou integração em sistemas de automação.

Arquitetura RISC-V – História, Estrutura, Pontos Fortes e Fracos

A arquitetura RISC-V é uma Instruction Set Architecture (ISA) aberta baseada no conceito de RISC (Reduced Instruction Set Computer). Ela define como o processador executa instruções, como os registradores funcionam e como o software interage com o hardware — mas não impõe como o chip deve ser fisicamente implementado. Hoje, o padrão é mantido pela RISC-V International, que coordena sua evolução e padronização global. O RISC-V nasceu em 2010 na University of

California, Berkeley, liderado pelo professor David Patterson e sua equipe. O objetivo era criar uma arquitetura moderna que fosse:

- *Simples;*
- *Modular;*
- *Acadêmica e aberta;*
- *Livre de royalties.*

Até então, arquiteturas dominantes como ARM Holdings e Intel eram proprietárias e exigiam pagamento de licenças. A proposta do RISC-V foi diferente: criar um padrão aberto, como o TCP/IP da internet (qualquer empresa poderia usar, modificar e fabricar processadores baseados nele). Em 2015 foi criada a RISC-V Foundation (hoje RISC-V International), consolidando o padrão como especificação industrial global.

Conceito Técnico da Arquitetura RISC-V

RISC (Reduced Instruction Set Computer) tem como princípios:

- *Instruções simples e fixas;*
- *Execução rápida (muitas instruções em 1 ciclo);*
- *Arquitetura load/store;*
- *Uso intenso de registradores;*
- *Hardware mais simples.*

Comparação Geral entre as arquiteturas

Critério	ARM	x86	RISC-V
Aberta	Não	Não	Sim
Royalty	Sim	Sim	Não
Modular	Parcial	Não	Sim
Ecossistema	Maduro	Muito Maduro	Crescendo

O microcontrolador CH32V003F4P6

O microcontrolador CH32V003F4P6 que aqui estamos estudando é fabricado pela empresa Chinesa WCH (https://www.wch-ic.com/). Recomendo visitar o site da empresa para conhecer melhor os modelos de microcontrolador bem como toda linha de produtos e softwares, **figura 1**.

KIT DE MICROCONTROLADOR HOLTEK

- Alimentação de 7 à 15V
- Display de 7 segmentos
- Conversor USB serial

- Alimentação de 7 à 15V
- Display de 7 segmentos
- Conversor USB serial
- Conector para módulo micro Holtek HT66F0185
- Portas de expansão
- Oito Leds de alto brilho
- Um buzzer ativo de 5V
- Um display Oled com SSD1307
- Quatro chaves de pressão
- Regulador interno de 5V e outro de 3,3V

Rua Ezequiel Freire 192, Sala 503. São Paulo - SP
11 98647-0924 www.qsptec.com.br

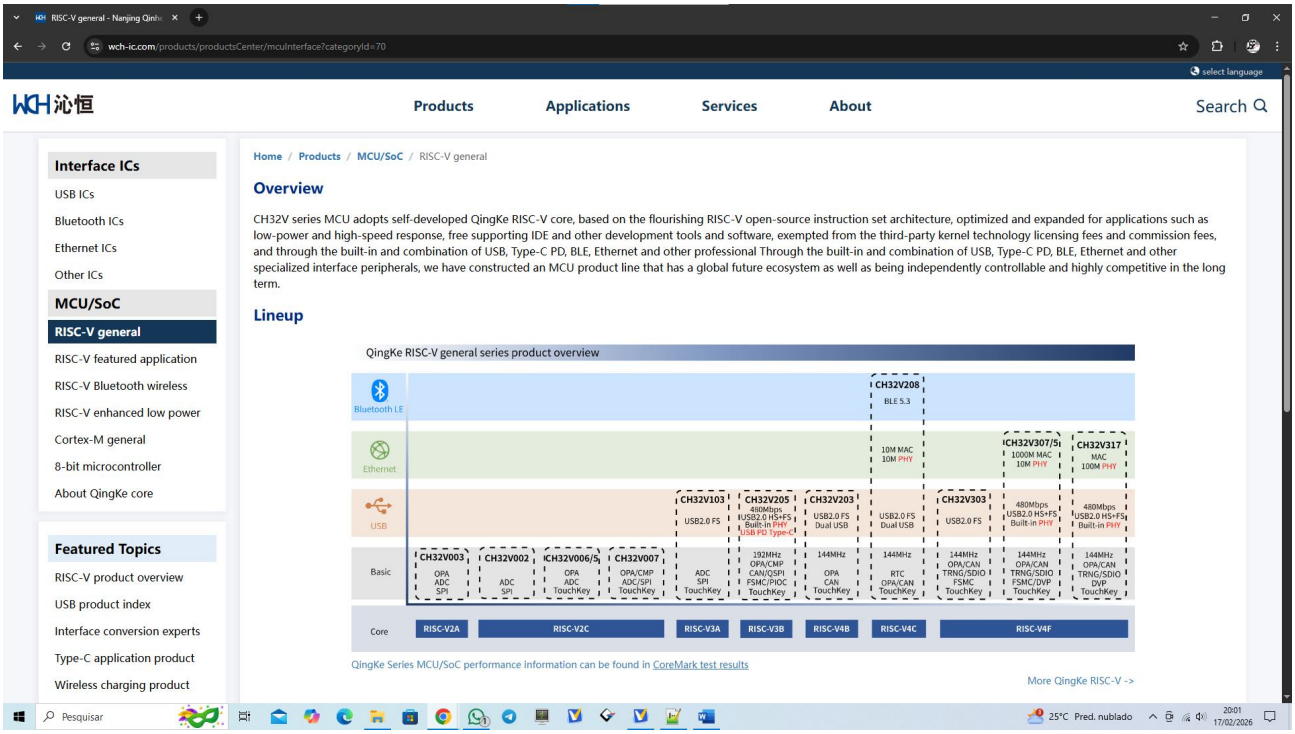


Figura 1 - Página do Microcontrolador CH32V003F4P6.

Este modelo CH32V003 é um microcontrolador de 32 bits de propósito geral e de extremo baixo custo e com o núcleo RISC-V2A. Abaixo segue suas principais características:

- *Processador QingKe 32 bits RISC-V2A, com suporte a aninhamento de interrupções em 2 níveis.*
- *Frequência principal do sistema de até 48 MHz*;
- *2 KB de SRAM, 16 KB de Flash*;
- *Tensão de alimentação: 3,3 V / 5 V*;
- *Múltiplos modos de baixo consumo: Sleep e Standby*;
- *Reset por ligamento/desligamento (Power-on/Power-off Reset) e monitor de tensão programável*;
- *1 controlador DMA de uso geral com 1 canal*;
- *1 conjunto de amplificadores operacionais (OPAs)*;
- *1 conversor A/D de 10 bits (ADC)*;
- *1 temporizador avançado de 16 bits e 1 temporizador de uso geral de 16 bits*;
- *2 watchdog timers e 1 temporizador base de tempo do sistema de 32 bits*;
- *1 interface USART, 1 interface I2C, 1 interface SPI*;
- *18 portas de I/O, podendo ser mapeadas para 1 interrupção externa*;
- *ID único do chip de 96 bits*;
- *Interface de depuração serial de 1 fio (1-wire)*;
- *Encapsulamentos disponíveis: TSSOP20, QFN20, SOP16, SOP8.*

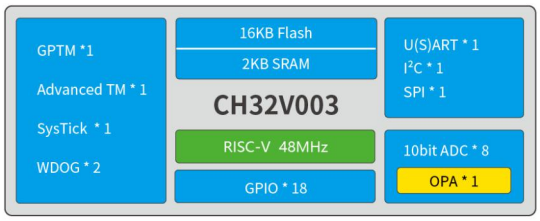


Figura 2 - Diagrama de Blocos.

Part NO.	Freq	Flash	SRAM	GPIO	Timer				ADC (10bit) Unit/CH	OPA	U(S)ART	I²C	SPI	VDD	Package
					Adv (16bit)	GP (16bit)	WDOG	SysTick (32bit)							
CH32V003J4M6	48MHz	16K	2K	6	1	1	2	√	1/6	1	1	1	-	3.3/5.0	SOP8
CH32V003A4M6	48MHz	16K	2K	14	1	1	2	√	1/6	1	1	1	-	3.3/5.0	SOP16
CH32V003F4P6	48MHz	16K	2K	18	1	1	2	√	1/8	1	1	1	1	3.3/5.0	TSSOP20
CH32V003F4U6	48MHz	16K	2K	18	1	1	2	√	1/8	1	1	1	1	3.3/5.0	QFN20

Figura 3 - Tabela de Comparação dos Modelos.

MounRiver Studio

A MounRiver Studio (MRS) é a IDE oficial da WCH para desenvolvimento com seus microcontroladores baseados em RISC-V e ARM. No contexto da linha CH32V (como o CH32V003), ela é hoje a principal ferramenta para criação, compilação e gravação de firmware.

O que é a MounRiver Studio? É um ambiente de desenvolvimento integrado (IDE) baseado no Eclipse CDT, já configurado com:

- *Toolchain GCC para RISC-V.*
- *Debugger integrado.*
- *Programador embutido.*
- *Exemplos oficiais da WCH.*
- *Biblioteca de periféricos (Standard Peripheral Library).*

Ela foi pensada para simplificar a entrada no ecossistema CH32 (**figura 4**).

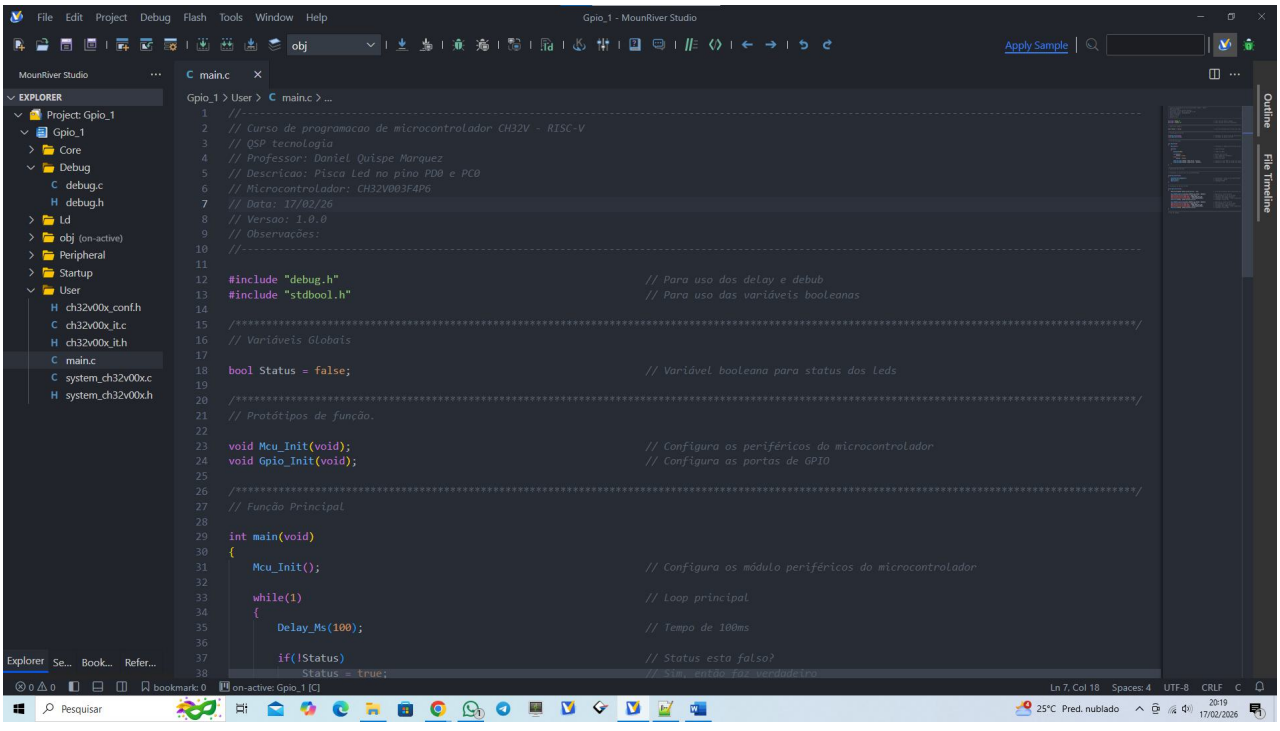


Figura 4

Ferramentas de hardware – Gravadores

O microcontrolador possui gravação In-Circuit e podemos utilizar um gravador de muito baixo custo da própria fabricante WCH. O nome deste gravador é WCH-LinkE (figura 5). ele é um adaptador USB que funciona tanto como programador quanto como debugger para chips RISC-V e ARM da WCH.

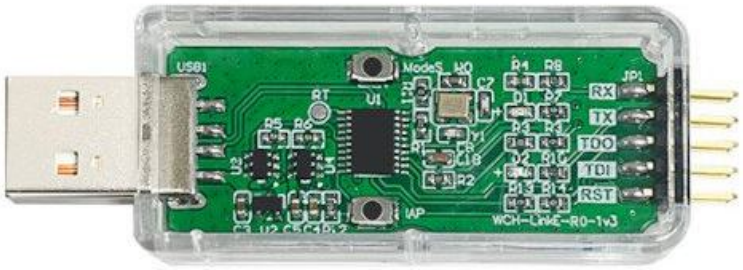


Figura 5 - WCH-LinkE

O que ele faz

- *Permite gravar firmware no microcontrolador diretamente a partir da IDE (MounRiver Studio) ou ferramentas compatíveis, como utilitários de linha de comando.*
- *Funciona em modo RISC-V para a linha CH32V.*
- *Pode alimentar o alvo com 3,3 V ou 5 V.*
- *Suporta debug (depuração) com breakpoints, leitura de registradores, etc.*
- *Tem incorporado um conversor USB para USART criando uma porta COM virtual (VCP).*

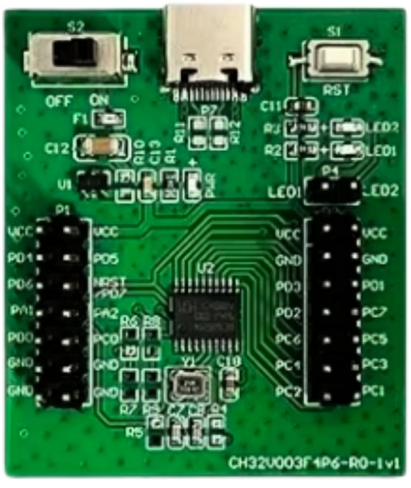


Figura 6

Placa de avaliação da WCH

É uma plaquinha de baixo custo que contém um microcontrolador CH32V003F4P6, três LEDs, uma porta USB tipo C, uma chave, um botão de reset, alguns componentes e duas barras de pinos para acesso aos IOs do microcontrolador (**figura 6**).

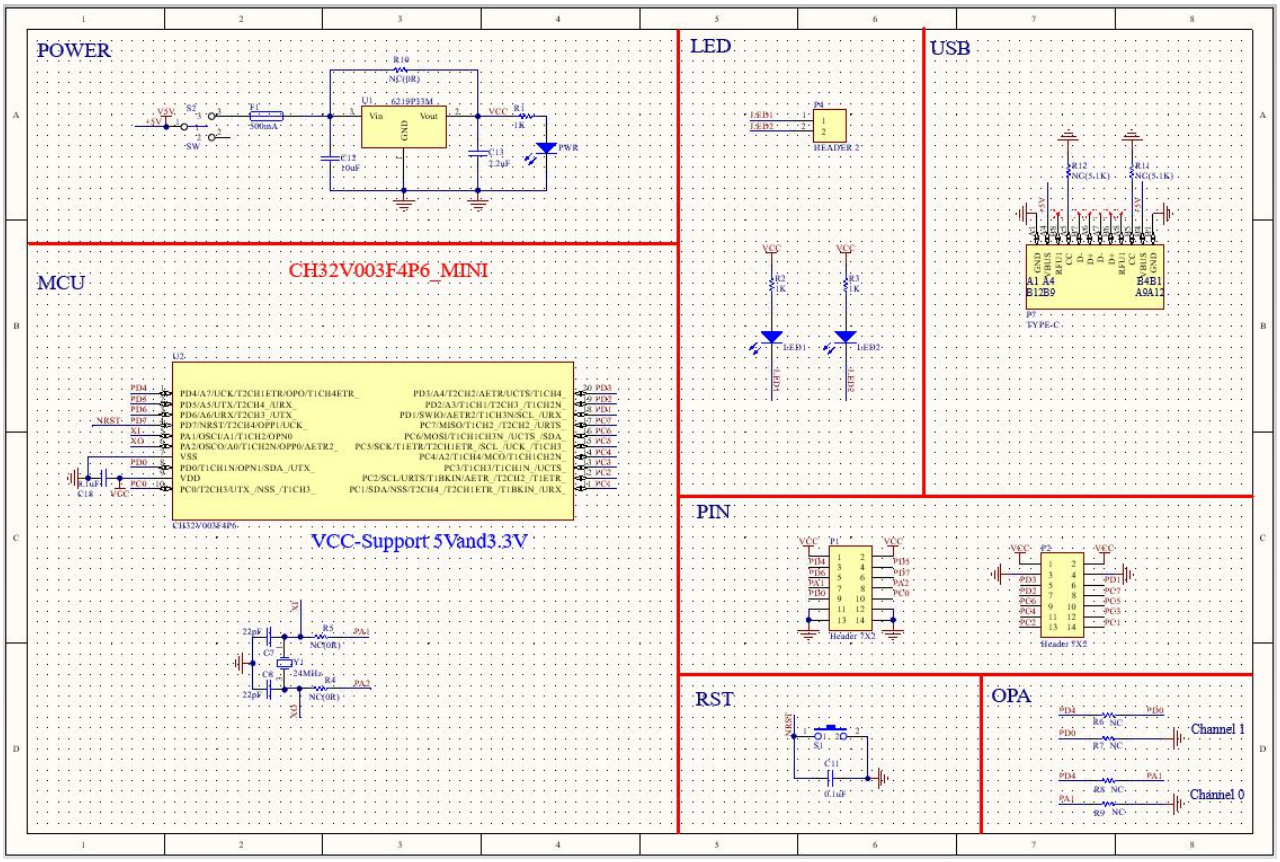


Figura 7 - Esquemário completo.

Na **figura 7** podemos verificar seu esquemático completo para referência.

Criando um projeto na IDE MounRiver Studio

Para se criar um novo projeto na IDE, é muito fácil e quem já está habituado com outras ferramentas não terá dificuldades. Siga os passos a seguir para a criação do projeto com a seleção do microcontrolador e das ferramentas de gravação. Vá em File -> New -> MounRiver Project, como mostra a imagem da **figura 8**.

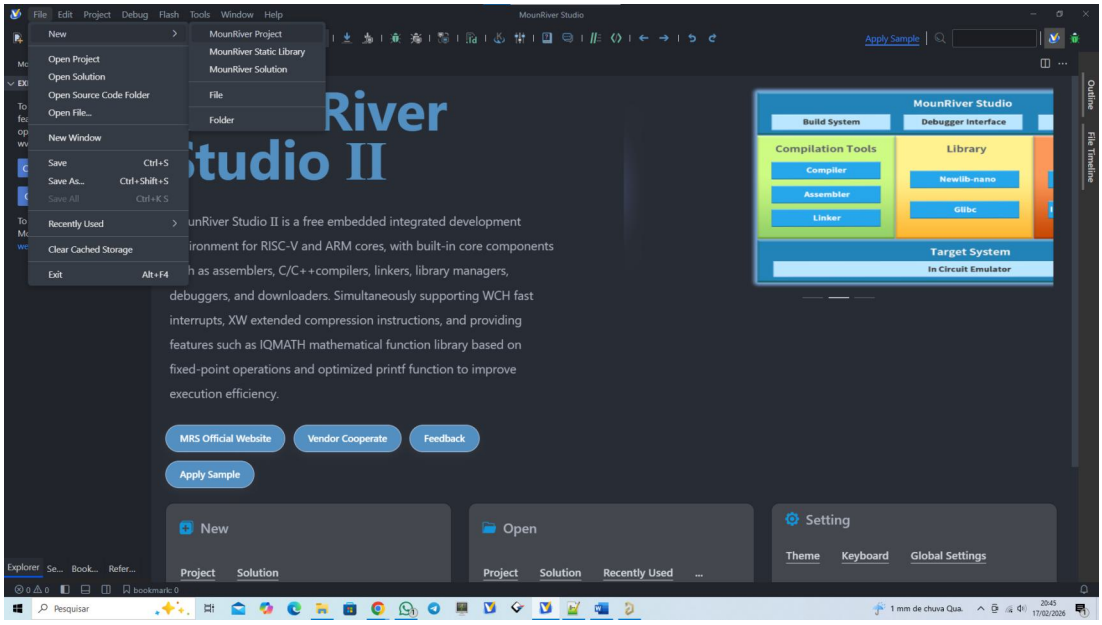


Figura 8

Na sequência, dê um nome para este projeto, escolha um local para salvar, em MCU Core, selecione apenas RISC-V, no campo de busca de família de microcontroladores, digite CH32V003 e depois selecione o microcontrolador que no meu caso é o CH32V003F4P6, pressione Finish e você terá criado o seu primeiro projeto (**figura 9**).

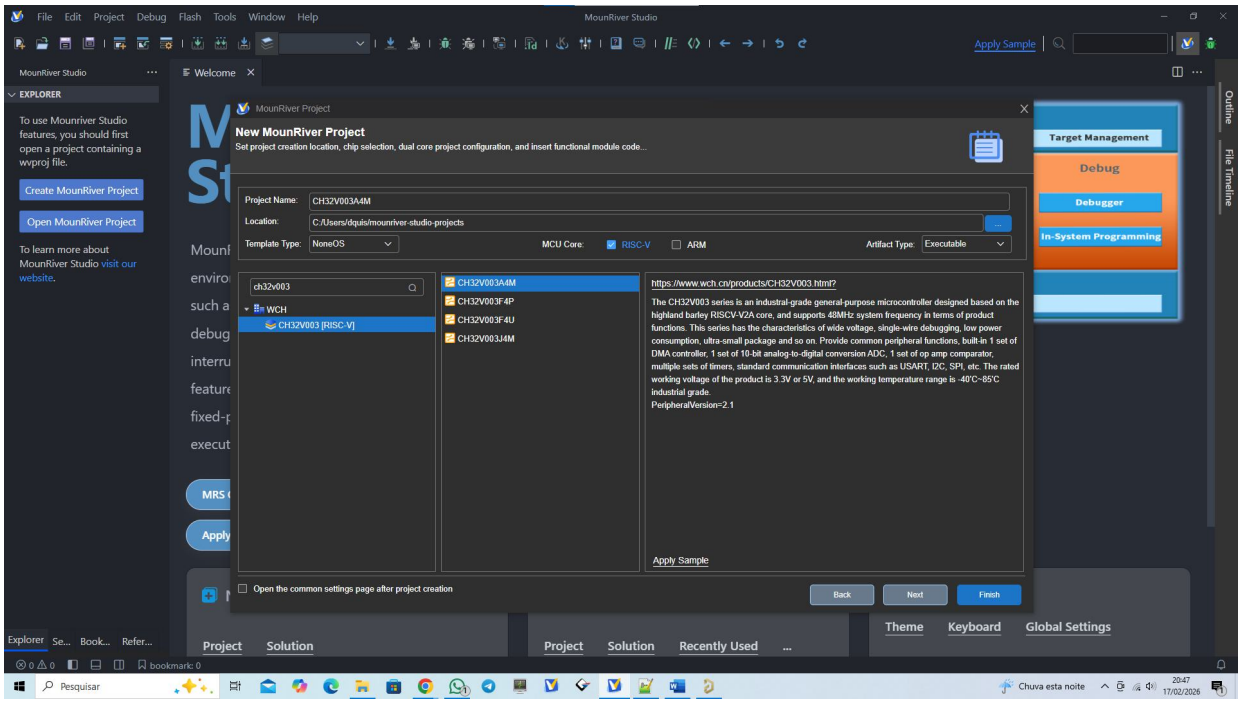


Figura 9

Na barra lateral esquerda (Workspace) abra a pasta User e depois clique em Main. Vamos digitar o código abaixo que faz piscar dois leds da placa de avaliação, um o pino PC0 e outro no pino PD0.

```
//---------------------------------------------------------------------------------------------------
// Curso de programação de microcontrolador CH32V - RISC-V
// QSP tecnologia
// Professor: Daniel Quispe Marquez
// Descrição: Pisca Led no pino PD0 e PC0
// Microcontrolador: CH32V003F4P6
// Data: 17/02/26
// Versão: 1.0.0
// Observações:
//---------------------------------------------------------------------------------------------
#include "debug.h"                    // Para uso dos delay e debub
#include "stdbool.h"                  // Para uso das variáveis booleanas
```

```
/*****************************************************************************************/
// Variáveis Globais
bool Status = false;                        // Variável booleana para status dos leds
/*****************************************************************************************/
// Protótipos de função.
void Mcu_Init(void);                        // Configura os periféricos do microcontrolador
void Gpio_Init(void);                       // Configura as portas de GPIO
/*****************************************************************************************/
// Função Principal
int main(void)
{
   Mcu_Init();                          // Configura os módulo periféricos do microcontrolador
   while(1)                               // Loop principal
   {
      Delay_Ms(100);                       // Tempo de 100ms
      if(!Status)                         // Status esta falso?
         Status = true;                    // Sim, então faz verdadeiro
      else                                // Caso contrário
         Status = false;                   // Faça ele falso
GPIO_WriteBit(GPIOD, GPIO_Pin_0, Status); // Atualiza o pino PD0 de acordo com Status
GPIO_WriteBit(GPIOC, GPIO_Pin_0, !Status);// Atualiza o pino PC0 de acordo com Status
   } }
/*****************************************************************************************/
// Definição das funções
//----------------------------------------------------------------------------------------
// Configura os periféricos do microcontrolador
void Mcu_Init(void)
{
   SystemCoreClockUpdate();                  // Inicializa o clock do microcontrolador
   Delay_Init();                          // Inicializa o delay
   Gpio_Init();                           // Configura GPIO
}
//----------------------------------------------------------------------------------------
// Configura as portas de GPIO
void Gpio_Init(void)
{
 GPIO_InitTypeDef GPIO_InitStructure = {0};  // Cria uma estrutura para inicializar as portas
// de GPIO
 RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);    // Habilita o clock do
//portc
 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;        // Vou configurar o bit 0 do GPIO
 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;              // Configura para modo
//push-pull
```

```
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;   // Configura para 50Mhz
//- Alta frequência
GPIO_Init(GPIOC, &GPIO_InitStructure);   // Configura o pino PC0
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);  // Habilita o clock do
//portd
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;   // Vou configurar o bit 0 do GPIO
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;  // Configura para modo push-pull
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;  // Configura para 50Mhz - Alta
//frequência
GPIO_Init(GPIOD, &GPIO_InitStructure);   // Configura o pino PD0
}
/*********************************************************************************/
// Fim do código
```

Fazendo o download do firmware no microcontrolador

Para gravar o arquivo firmware no microcontrolador, vamos conectar o WCH-LinkE na placa de avaliação usando três JUMPERS, +5V, GND e SWDIO (Veja essas siglas da placa do gravador). O sinal de dados é bidirecional e é conectado no pino PD1 (SWDIO) do microcontrolador, veja a imagem abaixo. Conecte o gravador em uma porta USB disponível do seu PC e pressione SHIFT F8 na IDE MounRiver Studio. Vai aparecer uma janela como a da imagem abaixo. Clique nos botões Query para procurar o microcontrolador e o núcleo. Repare que em MCU Type é mostrado qual o microcontrolador que está conectado em sua aplicação, depois clique em Apply e em seguida em Close (**figura 10**).

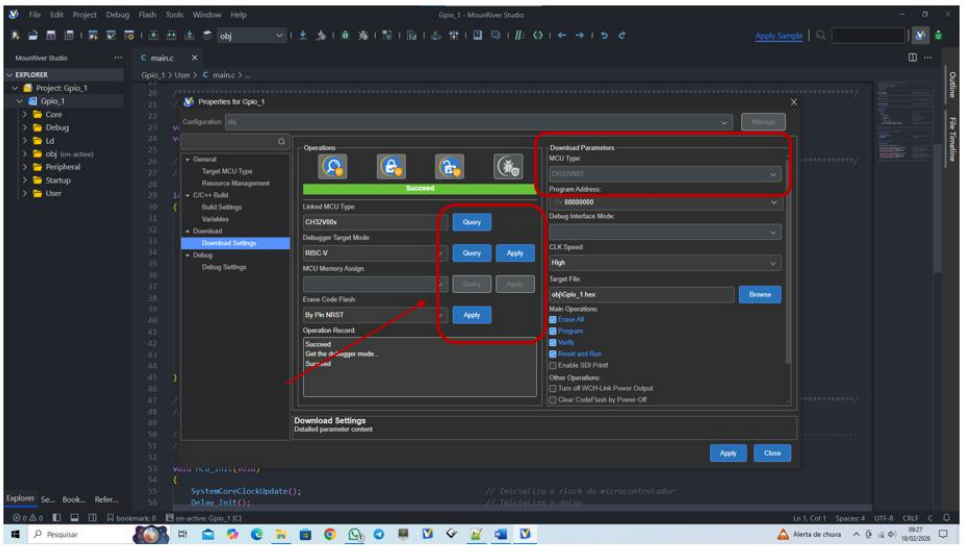


Figura 10

Agora vamos fazer o download do firmware em nossa placa de avaliação clicando em no menu FLASH -> DOWNLOAD ou simplesmente F8. A imagem da **figura 11** mostra o processo de gravação. Inicia a operação da Flash, ou seja, da gravação, primeiro apaga o conteúdo, faz a gravação do arquivo hexadecimal, verifica com o buffer de saída, reseta o microcontrolador e pronto, já deve estar rodando sua aplicação na placa.

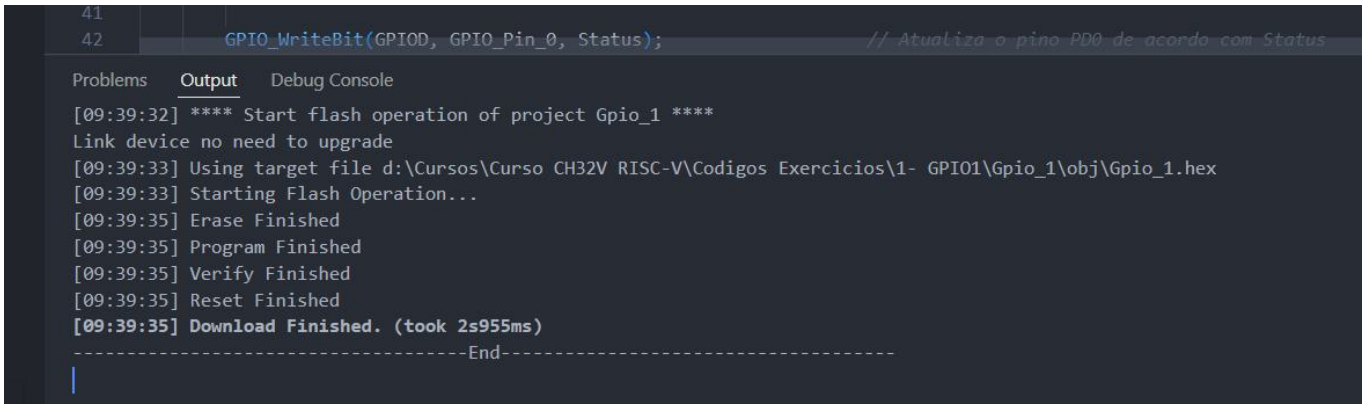


Figura 11

Placa de avaliação com o gravador WCH-LinkE rodando o firmware de teste

Podemos visualizar na **figura 12**, a placa de avaliação junto com o gravador rodando o código acima. Para ver um vídeo do funcionamento bem como uma explicação passo a passo do código, visite nosso canal no Youtube e vá na playlist sobre microcontroladores WCH RISC-V

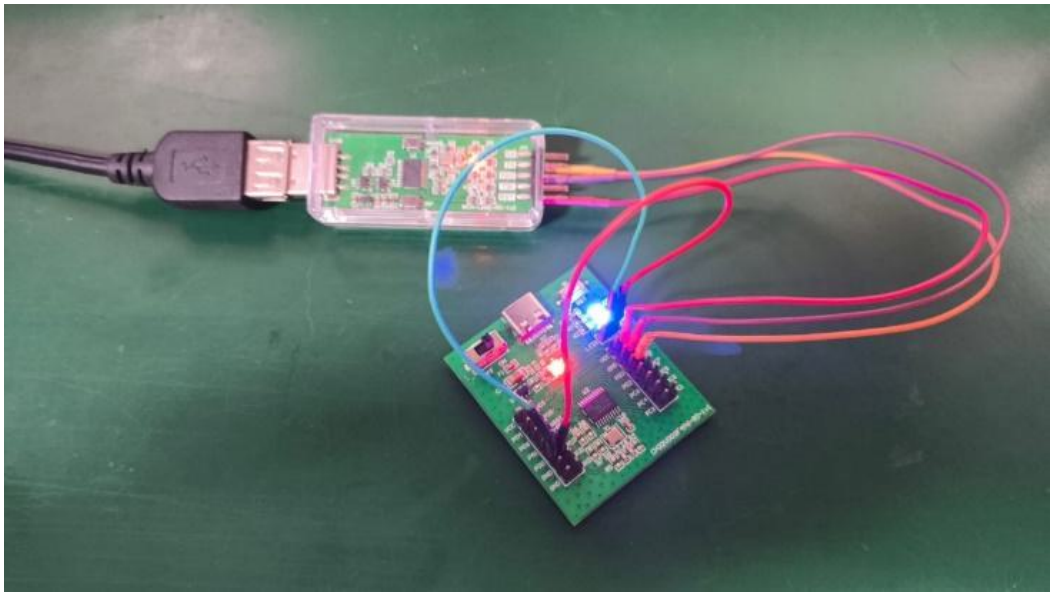


Figura 12

Contatos:

Canal no Youtube: https://www.youtube.com/@qsptecnologia

Site da nossa Empresa: www.qsptec.com.br

Nossa loja: www.robomakers.com.br

Conclusão

Podemos dizer que temos uma excelente opção para projetos eletrônicos de baixo custo com tecnologia de 32 bits. Ferramentas de software gratuitas e de hardware de baixo custo bem como microcontroladores muito acessíveis inclusive para produtos sensíveis a custo. Então está esperando o que para dar os primeiros passos nesta linha de microcontroladores RISC-V.

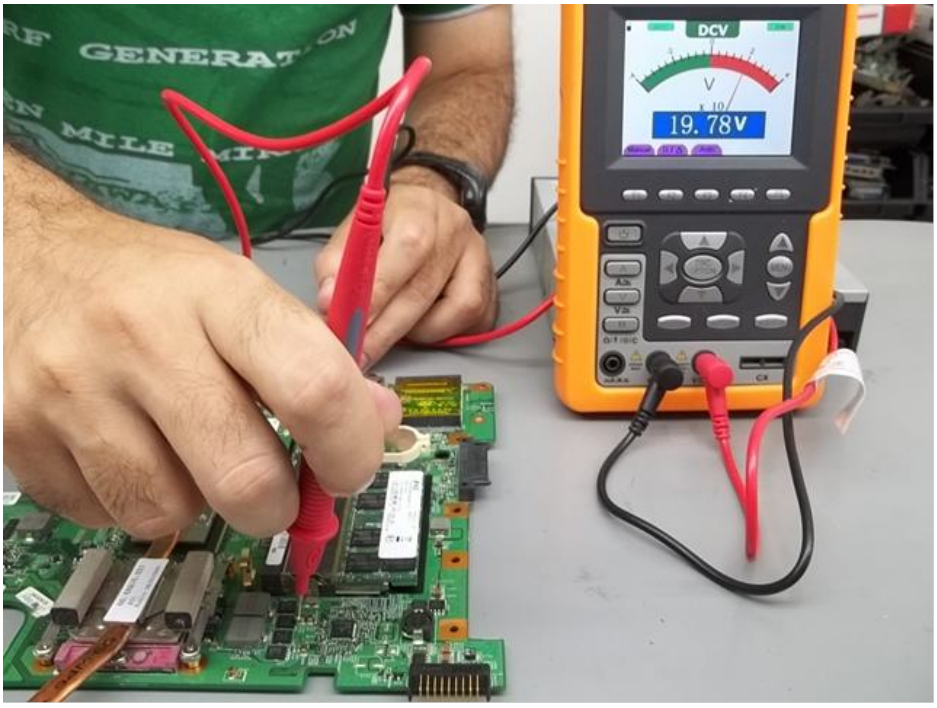
Obs. *Em breve teremos um curso On-line sobre esta linha de microcontroladores, sempre esteja conectado em nosso canal no Youtube*

Referências

[1] ARM - https://pt.wikipedia.org/wiki/Arm_Holdings

[2] RISC-V - https://riscv.org/

[3] Mbed - https://os.mbed.com/blog/



Como Conquistar a Autorizada

Como Abrir Sua Própria Oficina

João Batista de Sousa

Abrir o próprio negócio e ser dono de si mesmo é o sonho de muitos técnicos. Muitos ficam imaginando que isso nunca será possível, pois acham que a primeira condição seria ter um grande capital, sem o qual nada daria certo.

É claro que sem dinheiro não dá para fazer nada. No entanto, com um pouco de dinheiro, muita garra e bom conhecimento das atividades que se vai exercer, torna-se possível montar uma oficina técnica ou qualquer outro negócio comercial.

Crédito: *Luiz Carlos Burgos Artigos técnicos para jornal eletrônica em foco (https://www.aeletronicaemfoco.com.br).*

O ramo da assistência técnica de rádio, som, TV, DVD, ... depende basicamente de um técnico bem preparado, que execute o serviço de forma rápida e correta.

As orientações para abrir uma oficina técnica podem ser descritas em seis partes sucessivas:

1) A ideia

É aqui que começa a nascer a empresa, a oficina de assistência técnica.

Você começa a ponderar os prós e contras e decide assumir os riscos para tocar sua ideia avante.

Saiba arriscar conscientemente. Esteja preparado para enfrentar desafios e ousar na execução de um novo empreendimento, escolher os melhores caminhos, baseando-se naquilo que você conhece. Jamais inicie um empreendimento sobre o qual você não conheça nada.

2) Escolha do local

A escolha do ponto é determinante para o seu sucesso, e a especialização no ramo ajuda a tornar o negócio altamente competitivo. O faturamento mensal varia muito de região para região e também de acordo com o porte da oficina.

À medida que o negócio for crescendo, você poderá entrar em contato com as fábricas e indústrias para solicitar seu aval para operar como oficina autorizada. Esta marca que lhe concede ser uma autorizada será como uma bandeira, que o identificará como uma assistência confiável e respeitável.

3) Formalização jurídica

Após a ideia e a escolha de um bom local, vem a formalização jurídica da empresa. A sua oficina deve estar legalizada para que você possa trabalhar tranquilo, inclusive com a perspectiva de tornar-se uma autorizada de alguma fábrica futuramente.

4) Fluxo de caixa

Uma vez formalizada juridicamente a empresa, é necessário passar para a terceira parte, o fluxo de caixa.

É necessário fazer um estudo para traçar um quadro objetivo:

- Cálculo do investimento;
- Cálculo dos salários e encargos dos funcionários;
- Estimativa das rendas e dos custos gerais;
- Apuração dos resultados;
- Projeção do fluxo de caixa e do capital de giro;
- Cálculo dos preços de serviços.

5) Contratações

De início, talvez você mesmo faça todos os serviços técnicos. Mas, à medida que a oficina for crescendo, será necessário contratar pessoas competentes para lhe assessorar.

Ao fazer as contratações, consulte seu contador sobre os encargos trabalhistas e sobre a legislação do trabalho. Ser um empregador é uma coisa muito séria e você precisa estar preparado e consciente.

6) Organização da oficina

É necessário que o principiante se acostume desde o início a ter a ideia de ser ordeiro, organizado, visão 5S, visando obter uma condição de vida mais aceitável entre os próprios elementos integrantes da classe.

Assim sendo, a oficina deve ter uma boa aparência e uma boa organização, a fim de proporcionar um ambiente agradável aos clientes e aos próprios técnicos que trabalham nela.

Técnicas de atendimentos

O atendimento, uma etapa fundamental para o sucesso da sua oficina, inicia na recepção do cliente e/ou no atendimento pelo telefone. Um bom atendimento pode ser o diferencial para a conquista do cliente e, também, para a fidelização dele à sua oficina. Sugerimos, para isso, que adote as seguintes regras básicas:

- Use linguagem clara;
- Fale com entusiasmo;
- Olhe nos olhos do cliente (isso transmite segurança);
- Escute o cliente;
- Nunca interrompa a fala do cliente;
- Nunca diga que o cliente está errado.

A abordagem deve iniciar sempre com um sorriso, demonstrando, assim, simpatia e alegria por receber o cliente.

Lembre sempre que o cliente procura uma oficina porque tem um aparelho com defeito. Isso já é um fator desfavorável para o técnico, pois o cliente está inquieto, triste e preocupado.

Jamais despreze ou inferiorize o aparelho do cliente. A sobrevivência da empresa de assistência técnica depende dele. Portanto, um aparelho velho e defeituoso é tão importante e valioso quanto aquele último modelo que está nas vitrines das lojas.

Em 50% dos equipamentos que entram em uma oficina para conserto, o diagnóstico começa com as informações do cliente. Por isso, ao receber um aparelho, converse com o proprietário e colha o máximo de informações de como o problema ocorreu.

E, ao iniciar o conserto de qualquer aparelho, observe as seguintes dicas. Elas vão ajudá-lo muito.

Quando houver evidências de queda, observe se há componentes soltos ou faltando, partes carbonizadas e/ou deterioradas.

Faça uma medição de componentes a frio, ou seja, com o aparelho desligado. Lembre-se de que, para que se possa fazer uma medição correta, a maioria dos componentes deve ser removida do circuito.

Inicialmente, ligue o aparelho em uma tomada que tenha algum dispositivo de segurança, como, por exemplo, lâmpada em série com a alimentação da tomada.

De posse do diagrama, faça as medições de tensão dos circuitos.

Ao ligar o aparelho, procure descobrir se existem componentes superaquecendo, liberando fumaça e/ou cheiro de queimado.

Fique atento aos barulhos, ruídos ou estalidos que vêm do circuito ou estão presentes nos alto-falantes. Nos cinescópios é possível observar uma falta ou a presença de imagem deformada. Isso ajuda muito para saber em que circuito se deve trabalhar.

Substitua, quando possível, os componentes pelos originais. Se isso não for possível, use somente os equivalentes. Nas associações de resistores ou capacitores, fique atento para que a soma total seja exatamente igual ao componente original.

Mantenha uma caderneta (ou banco de dados) com relatórios dos reparos já executados. Recomendo as fichas de service das antigas revistas Saber Eletrônica e revistas INCB ELETRÔNICA.

Essas informações aliadas a um lugar arejado, bem ventilado, bem iluminado, ferramentas adequadas, uma boa bancada de trabalho, tempo disponível e conhecimentos técnicos certamente são

ingredientes para o sucesso no diagnóstico do defeito do aparelho.

A manutenção, seja ela de equipamento eletrônico ou de veículo, deve obedecer a uma linha lógica de raciocínio que venha a apresentar uma conduta para a pesquisa de defeito. O êxito da manutenção somente é alcançado quando a pesquisa é dirigida com objetividade e, para isso, necessário que haja uma sequência lógica de passos em sua execução.

Mesmo o técnico experiente usa o esquema elétrico. A inspeção de alguns componentes, durante o processo de comprovação de seu funcionamento, exigirá, por exemplo, o uso de um multímetro ou de outra ferramenta especial, assim como o conhecimento e o domínio de termos técnicos. Portanto, o uso do esquema elétrico e do caderno de manutenção facilita a identificação do problema e permite chegar a um diagnóstico do defeito, bastando, para isso, seguir as orientações de pesquisa, item por item, à medida que for sendo inspecionado cada componente.

Na verdade, a manutenção, seja ela qual for, exige que se siga, passo a passo, cada circuito relacionado ao problema apresentado, verificando o estado de funcionamento das peças ou o fluxo de corrente. Se o circuito for constituído de componentes elétricos, testam-se pontos de tensão e fluxo de corrente. O importante é que se realize a pesquisa até chegar às possíveis causas e, então, faça-se o diagnóstico do defeito.

Muitas vezes, a sequência de manutenção é confundida com o diagnóstico. A primeira é o procedimento de atuação, ou seja, é o processo de análise do defeito, e o segundo é o resultado final do processo de pesquisa do defeito.

Tanto na manutenção preventiva quanto na corretiva, seja ela elétrica, eletrônica, ou até mesmo mecânica, de motocicleta ou de automóvel, considera-se “sequência” toda a rotina de tarefas que deve ser seguida durante a pesquisa do defeito, a fim de chegar a um diagnóstico para posterior substituição da peça ou do componente defeituoso, cumprindo as normas de testes estabelecidas pelo fabricante. É importante que o técnico construa uma linha de trabalho e, para isso, é necessário que ele tenha em mente a função de cada etapa.

Organizar uma sequência de ações que devem ser realizadas em uma manutenção corretiva é o que chamamos de otimização do trabalho.

Durante a vivência profissional, é possível constatar muitos fatos curiosos, dentre eles uma característica que evidencia a pressa em obter a solução. Por exemplo, ao concluir um determinado conserto e constatar que o aparelho não funciona corretamente ou continua com problema, a reação imediata de um técnico com pressa é a de abandonar tudo ou recorrer ao auxílio de alguém mais experiente, ao invés de verificar ao menos a existência das condições básicas para o funcionamento do equipamento.

O que queremos mostrar é que, antes de sair fazendo leituras e substituições de componentes, é importante que se construa uma sequência de ações baseada na comparação entre as informações do cliente e aquilo que aprendemos durante o desenvolvimento do nosso aprendizado. Somente dessa forma evitaremos investimentos desnecessários em peças e tempo extra de mão de obra.

O procedimento, a conduta lógica e o raciocínio são os fatores básicos para uma decisão acertada. Por isso devemos sempre checar todas as alternativas de manutenção. O procedimento racional e lógico para pesquisa de um defeito deve ser iniciado pelos sintomas do defeito.

Não existe crescimento sem problemas e obstáculos. Eles existem, e somente aqueles que persistirem irão atingir seus objetivos. Tenha persistência e nunca se esqueça do valor de uma grande ideia. Confie em si mesmo para definir sua conduta. Tenha coragem de arriscar, pois isso é essencial em manutenção!

Credenciamento como assistência técnica autorizada

As indústrias exigem que as assistências técnicas estejam aptas para atender aos clientes com um determinado padrão de qualidade. Para isso, é necessário observar os seguintes requisitos:

- Ter toda a documentação jurídica da empresa em dia;
- Ter um bom relacionamento e um bom conceito junto aos lojistas e clientes da região;
- Contar com bancadas e instrumentos bem organizados;
- Possuir um local adequado para a recepção dos clientes, para armazenar os aparelhos e para o estoque de componentes;

- Participar de atualizações técnicas e treinamentos ministrados pelo fabricante;
- Utilizar nos serviços realizados somente peças originais de fábrica;
- Ter uma boa formação técnica;
- Participar de testes teóricos e práticos.

As vantagens de ser um posto autorizado são: cursos de reparação gratuitos, suporte técnico constante, material promocional grátis, divulgação do endereço no catálogo das indústrias, formulários padronizados, assessoria jurídica e técnica por parte dos inspetores técnicos.

Sugerimos aos técnicos interessados que elaborem um currículo e façam uma apresentação da empresa e enviem para o departamento de assistência técnica das indústrias.

Bibliografia

Centro Profissional de Educação a Distância. Eletrônica Básica. Consertos em geral Módulo I. Páginas 2, 3, 4, 5, 6, 7.

Como Funciona o LiDAR

Newton C. Braga

Sistemas de localização usando sinais eletromagnéticos ou acústicos como o RADAR e o SONAR são bem conhecidos de todos que trabalham com eletrônica. No entanto, um sistema que cada vez encontra mais aplicações práticas e que não tem sido abordado de forma mais profunda nas publicações técnicas é o LiDAR. Utilizando pulsos de LASER, este sistema de curto alcance é um poderoso sensor de obstáculos à curta distância com aplicações inimagináveis. Como funciona o LiDAR é o que explicaremos nesse artigo dando suas principais aplicações e como obtê-lo na prática para seus projetos. LiDAR é a abreviação de Light Detection and Ranging ou Detecção e Alcance (Medida de Distância) usando Luz [1][2].

Figura 1 - O Princípio de operação do LiDAR

O princípio de funcionamento é o mesmo do RADAR. Um sinal eletromagnético é emitido em direção à área que se deseja escanear e pela reflexão dos sinais pode-se ter uma imagem em 3D dela ou simplesmente acusar objetos que nela estejam presentes. A diferença é que no Radar são utilizadas ondas de rádio, ondas muito curtas da faixa de micro-ondas de acordo com as dimensões dos objetos que devem ser detectados, enquanto no LiDAR usamos luz, que é uma radiação eletromagnética de comprimento muito menor [2].

Temos no site um artigo que mostra como funciona o RADAR e que pode ser usado como comparativo para os que desejam saber mais. Ele está em: https://www.newtoncbraga.com.br/como-funciona/10739-como-funciona-o-radar-art154.html

No LiDAR, um Laser infravermelho é utilizado para emitir pulsos numa taxa que depende da aplicação, e esses pulsos refletidos em objetos são detectados por um sensor [1]. Dependendo da varredura feita e da taxa de pulsos, pode-se ter uma imagem 3D do local escaneado, conforme podemos ver pela **figura 2**.

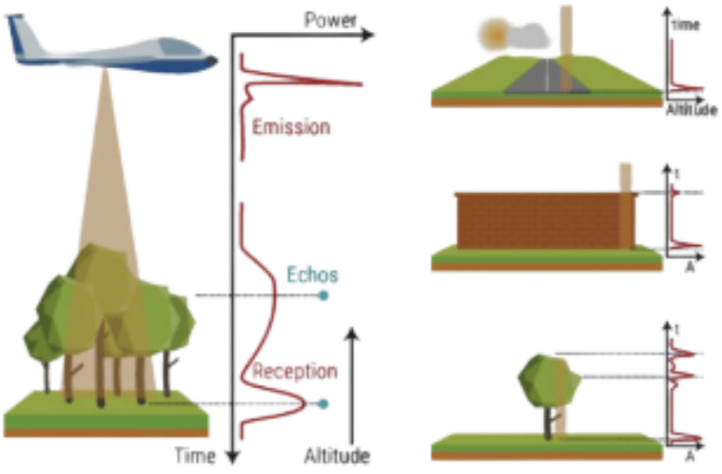


Figura 2 - Diferententes taxas de reflexão possibilitam o levantamento de imagens 3D.

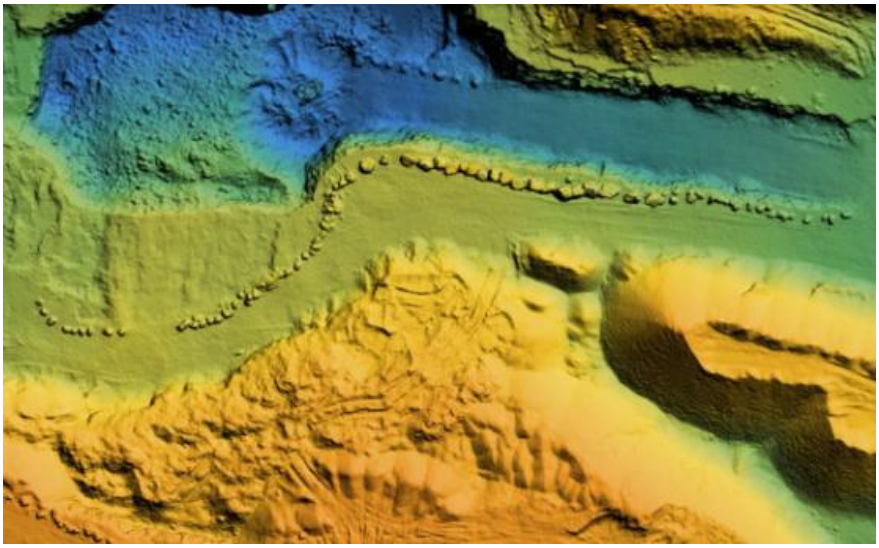


Figura 3 - Uma imagem de relevo em 3D.

Na **figura 3** temos um exemplo de imagem tridimensional obtida num levantamento topográfico.

Na prática, um sistema de detecção LiDAR é formado por 3 componentes. O primeiro componente tem por base um LASER que opera na região do infravermelho (dependendo da aplicação) e que está acoplado a um sistema de varredura ou escaneamento. Basicamente temos 3 comprimentos de onda mais usados: 905 nm para aplicações de consumo, automotivas e drones [3]. Os sensores são semicondutores comuns. Outro comprimento de onda é o de 1550 nm, utilizado em sistemas de longo alcance [3]. Finalmente, temos o 532 nm, que já cai na faixa da luz visível (verde) e que é usado em aplicações batimétricas que veremos mais adiante neste artigo [2][3]. A taxa de repetição dos pulsos varia conforme o número de pontos de imagem desejado e a taxa de varredura ou escaneamento varia de 1 Hz a 100 Hz conforme a definição de imagem também.

Para fazer a varredura do feixe existem 3 tecnologias possíveis.

As duas primeiras envolvem recursos mecânicos como o uso de espelhos giratórios e espelhos oscilantes. Essas tecnologias são usadas em veículos terrestres e voadores como drones. A **figura 4** nos mostra os princípios de funcionamento da varredura mecânica mais utilizados.

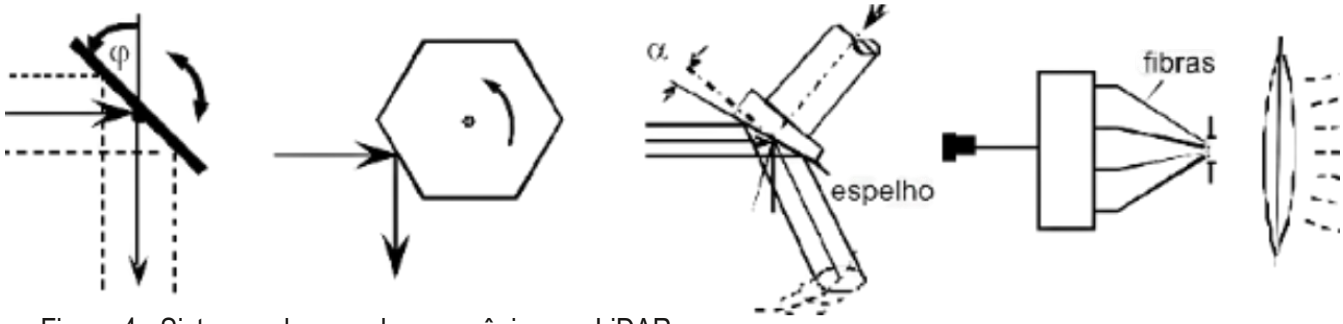


Figura 4 - Sistemas de varredura mecânica em LiDAR

No entanto, as tecnologias de semicondutores utilizando dispositivos MEMS (Micro-Electro-Mechanical Systems) de micro-espelhos também são usadas [3]. Esse sistema pode ser formado por micro-espelhos que são controlados por um microcontrolador que determina a área a ser explorada,conforme mostra a **figura 5**.

Num sistema de levantamento topográfico, por exemplo, esta área pode abranger vários quilômetros enquanto num sistema de exploração de objetos próximos pode variar entre alguns centímetros e alguns metros. Por exemplo, num sistema explorador de área para detectar vagas num estacionamento, a área explorada é de alguns metros apenas [5], conforme mostra a **figura 6**.

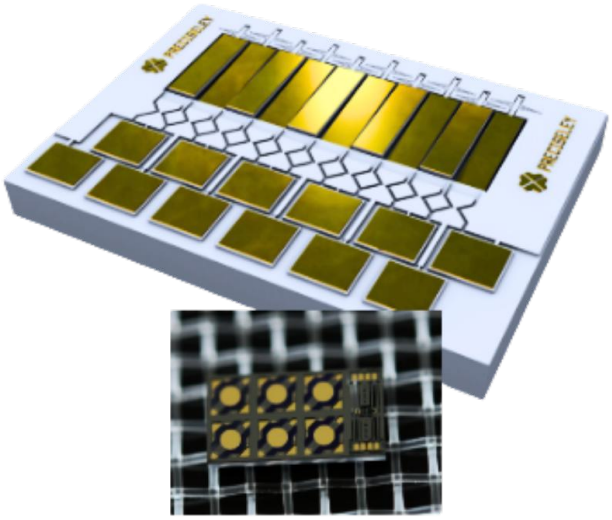


Figura 5 - Arrays de micro espelhos da Precisely Mircotechnology Corp..

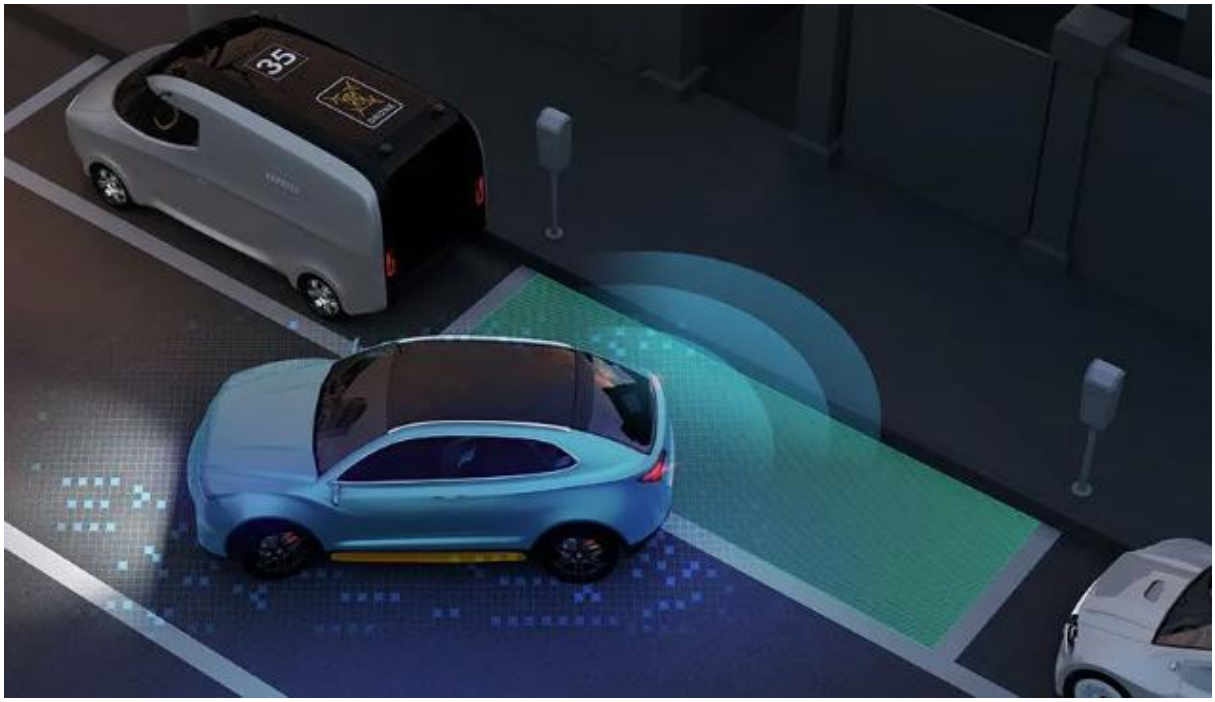


Figura 6 - Sistema de detecção de vagas e estacionamento autônomo usando LiDAR.

Figura 7 - Sensores comuns para aplicação em LiDAR.

O segundo elemento do sistema é um sensor que detecta os pulsos refletidos. Normalmente é usado um fotossensor rápido como os fotodiodos para poder detectar o tempo de ida e retorno do sinal, necessário para se determinar a distância do ponto em que ele refletiu [1]. A **figura 7** mostra alguns deles.

Os sinais obtidos do sensor são enviados ao terceiro elemento do circuito que é um microcontrolador que analisa o padrão de retorno levando em conta o tempo de ida e vinda do pulso e sua posição no escaneamento que ele também controla de modo a haver o sincronismo dos dados. Desta forma é possível formar uma imagem de milhões ou bilhões de pontos em disposição 3D do local escaneado, conforme mostra a **figura 8**.

Os sistemas que processam os dados obtidos pelo sistema sensor podem ter os mais diversos graus de complexidades. Alguns usam algoritmos simples quando se deseja coletar poucas informações do

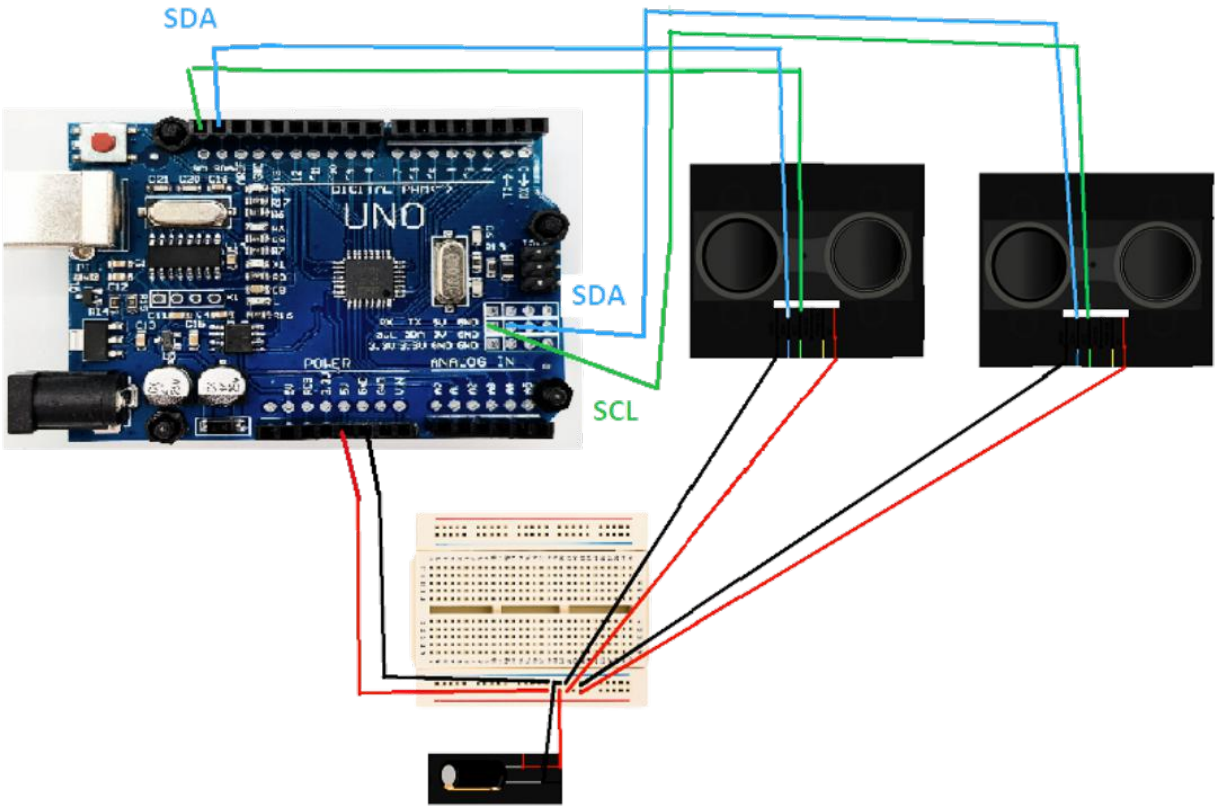


Figura 8 - Um sistema LiDAR com Arduino para aplicações em Robótica e controle.

sistema como a abertura e fechamento de uma porta de elevador ou de entrada ou extremamente complexo quando se deseja fazer o levantamento topográfico de uma região.

Recursos de inteligência artificial podem ser agregados caso o processamento das informações exija algum tipo de resposta em tempo real ou a identificação de objetos e até mesmo a leitura de etiquetas com códigos.

Lembramos que os leitores de QRcode e de códigos de barra são sistemas que fazem uso deste tipo de sensoriamento.

Figura 9 - Na aerofotogrametria são tiradas processadas para se obter imagens 3D

Tipos

Podemos classificar os sistemas LiDAR em dois grupos conforme o uso.

No primeiro grupo podemos colocar os tipos de longo alcance como os que são usados em drones, helicópteros e aviões para sensoriamento a longa distância. São utilizados em levantamento topográficos 3D, mapeamentos aéreos substituindo o recurso da fotogrametria em que são usadas câmeras fotográficas que tiravam fotos sequenciais em uma varredura que geravam um mosaico de fotos que então eram analisadas. A **figura 9** mostra uma aplicação com câmera em aerofotogrametria.

Um tipo de LiDAR importante que deve ser citado é o Batimétrico, que consegue fazer o levantamento da superfície do fundo do mar usando laser verde [2]. O laser verde, diferentemente do laser infravermelho, consegue penetrar em águas rasas e refletir no fundo [2].. A **figura 10** mostra como ele funciona.

Desta forma, com sua utilização é possível fazer o levantamento tipográfico do fundo do mar nas regiões costeiras.

Outra vantagem está no seu uso no espaço, como feito pela NASA e outros centros de pesquisas espaciais [4].. Além de ser utilizado para navegação espacial ele pode ser usado para o levantamento

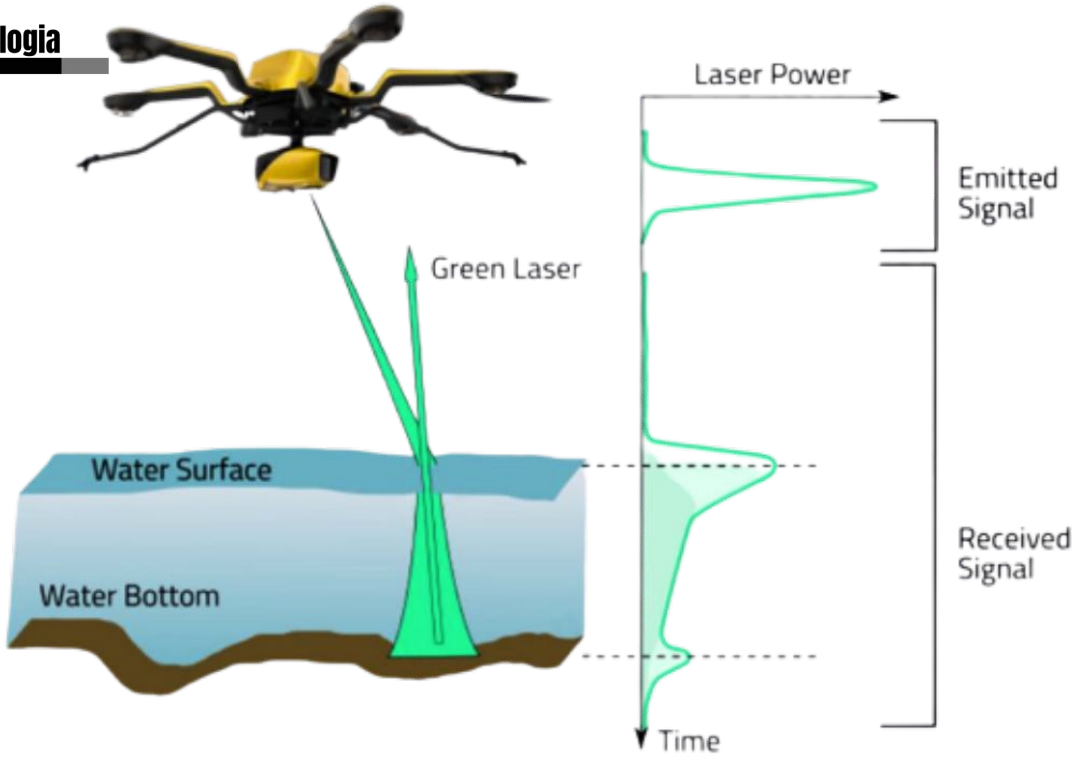


Figura 10 - O LiDAR batimétrico usa laser verde peça sua capacidade em penetrar na água.

topográfico da superfície de corpos celestes próximos como asteroides e cometas. A **figura 11** mostra sua aplicação no veículo explorador de Marte Perseverance.

Ainda no espaço podemos citar o controle de veículos autônomos como as sondas utilizadas em Marte, tanto no solo como no helicóptero utilizado nessa expedição [4]..

No solo temos diversas aplicações importantes, na verdade a maioria que podem até servir de inspiração para que nossos seguidores incluam este tipo de sensor em seu próximo projeto.

Assim temos o grupo dos LiDAR terrestres que operam no solo quer seja de modo fixo (LiDAR estático ou fazendo parte da IoT) quer seja acoplado a veículos (LiDAR móvel ou embarcado)

Por exemplo, nas aplicações fixas o LiDAR pode ser fixado a uma aplicação para fazer a exploração constante de uma determinada área. Um exemplo interessante é o monitoramento de áreas de risco, vulcões ativos, áreas sujeitas a inundações e projetos de construção. A **figura 12** ilustra seu uso.

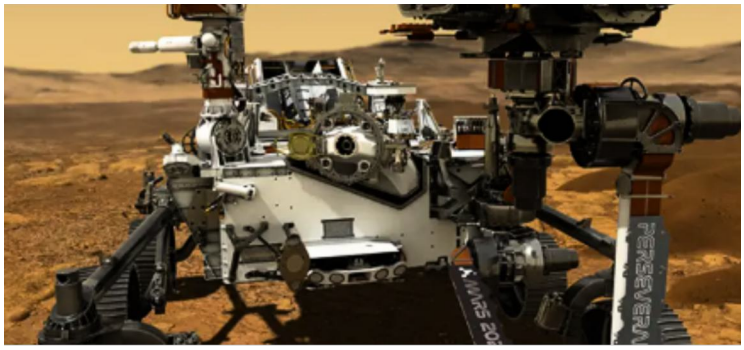


Figura 11 - Sonda em Marte usando LiDAR.

Para as aplicações móveis, um dos principais grupos em desenvolvimento é a dos veículos autônomos, onde o LiDAR pode detectar vagas de estacionamento para uma manobra

Figura 12 - LiDAR no monitoramento de obras.

automática [5]. E, em trânsito, ele pode detectar objetos em tempo real com grande segurança [5]. Na agricultura, podem fazer levantamentos de terreno, estimar a biomassa das coberturas vegetais, detectar propriedades do solo como a umidade, exposição solar e também pilotar de modo automático veículos agrícolas [1].

Em geração de energia ele pode ser utilizado para avaliar ventos para usinas eólicas, exploração de petróleo e outras fontes de energia tanto naturais como artificiais. Também serve para prospecção e controle de minas e pedreiras.

No lar e nas aplicações em ambientes fechados podemos utilizar o LiDAR em games, realidade virtual e automatismos inteligentes. Portas automáticas, eletrodomésticos inteligentes, presença de objetos ou quedas podem ser detectadas com sensores LiDAR,

Finalmente temos algumas aplicações importantes que envolvem recursos médicos, fitness, esportes e a própria meteorologia fazendo o levantamento de padrões de nuvens e outros parâmetros que auxiliam nas previsões meteorológicas.

Vantagens

Uma primeira vantagem do sistema está na sua imunidade a ruídos e interferência, o que não ocorre com os sistemas baseados em ondas de rádio ou sons. Outra vantagem está na possibilidade de detectar objetos de dimensões muito pequenas, o que não ocorre com o

RADAR e o SONAR, em que as dimensões estão agregadas ao comprimento de onda que é muito maior. Temos ainda o fato de que o infravermelho reflete bem em objetos que não refletem radiações como a luz visível. A grande intensidade do laser infravermelho e o fato de que ele não é visível tornam imperceptível sua presença, e ele não incomoda ou apresenta perigo para pessoas ou animais. É um sistema compacto, de baixo custo e de fácil implementação [1][3].No mercado

São muitas as empresas de semicondutores que fabricam os sistema LiDAR e os disponibilizam através de grandes distribuidoras, como a MOUSER ELECTRONICS (www.mouser.com)

No catálogo da Mouser encontramos alguns exemplos de sensores que podem ser obtidos, a maioria com a disponibilização de datasheets e até mesmo de placas de avaliação e desenvolvimento, facilitando assim a criação de um projeto.

Sensor LiDAR de Ponto Único Leve TFA300-L da DFRobot

O sensor LiDAR (Detecção e Alcance a Laser) de ponto único leve TFA300-L da DFRobot oferece medições de distância rápidas e precisas de até 290 m, com conectividade de protocolo duplo. Este sensor funciona com base no princípio de Tempo de Voo (ToF) e fornece medições de distância precisas. O sensor LiDAR TFA300-L apresenta uma frequência excepcional de até 10.000 Hz. Este sensor foi projetado especificamente para aplicações que exigem posicionamento rápido e preciso. As aplicações típicas incluem desvio de obstáculos em drones de alta velocidade, controle de altitude estável e rastreamento inteligente de alvos em sistemas optoeletrônicos.

Acesse: https://br.mouser.com/new/dfrobot/dfrobot-tfa300-l-lidar-sensor/

Figura 13 - O LiDAR TFA300-L da DFRobot

Módulos de Sensor LiDAR Benewake TFA300

Os módulos de sensor LiDAR Benewake TFA300 são LiDARs de ponto único de médio a longo alcance que apresentam uma alta frequência de até 10.000 Hz. Esses módulos estão disponíveis em duas versões: o ultraleve TFA300-L sem caixa e o TFA300 com caixa de proteção IP67. Os sensores LiDAR TFA300 apresentam um design compacto e leve e oferecem suporte à comunicação dupla por meio de interfaces UART e CAN. Os múltiplos modos de operação integrados permitem que os usuários alterem os parâmetros e a configuração para atender a diferentes aplicações.

Os módulos de sensor LiDAR TFA300 são usados em aplicações como gimbals eletro-ópticos/infravermelhos (EO/IR), drones de busca e salvamento (SAR) e drones de corrida FPV (visão em primeira pessoa) de altíssima velocidade.

Acesse: https://br.mouser.com/new/benewake/benewake-tfa300-lidar-sensor-modules/

Figura 14 - LiDAR Benewake TFA300

Telêmetro a Laser LightWare LiDAR GRF-500

O telêmetro a laser LightWare LiDAR GRF-500 é um dispositivo compacto e ultraleve com alcance de 500 m (1640 pés), otimizado para baixo consumo de energia e fácil integração com estabilizadores EO/IR e sistemas de medição de distância. O LightWare LiDAR GRF-500 utiliza o princípio do tempo de voo para medir a distância até objetos. O sistema emite uma rápida sucessão de pulsos de laser que refletem no objeto alvo até que o receptor atinja o tempo de emissão de laser designado. O GRF-500 utiliza tecnologia laser de 905 nm, garantindo desempenho otimizado e atendendo aos padrões de segurança ocular da classe 1M.

Acesse: https://br.mouser.com/new/lightware-lidar/lightware-grf-500-laser-rangefinder/

Figura 15 - Telêmetro LightWare LiDar GRF-500

Sensor de Distância LiDAR de Longo Alcance Seeed Studio TF03-180 (180m)

O Sensor de Distância LiDAR de Longo Alcance Seeed Studio TF03-180 (180m) é um sensor de distância de longo alcance de nível industrial. O TF03-180 possui um alcance máximo de detecção de até 180m e uma taxa de quadros ajustável com um máximo de 1KHz. Com um algoritmo de compensação integrado para brilho externo e outras interferências, o TF03-180 pode operar em ambientes com forte luminosidade, chuva, neblina e neve. Possui também uma caixa de liga de alumínio com resistência à água e poeira IP67. Vários modos de operação integrados estão incluídos para alterar parâmetros e configurações para atender a diferentes aplicações.

Acesse: https://br.mouser.com/new/seeed-studio/seeed-tf03-180-lidar-sensor/

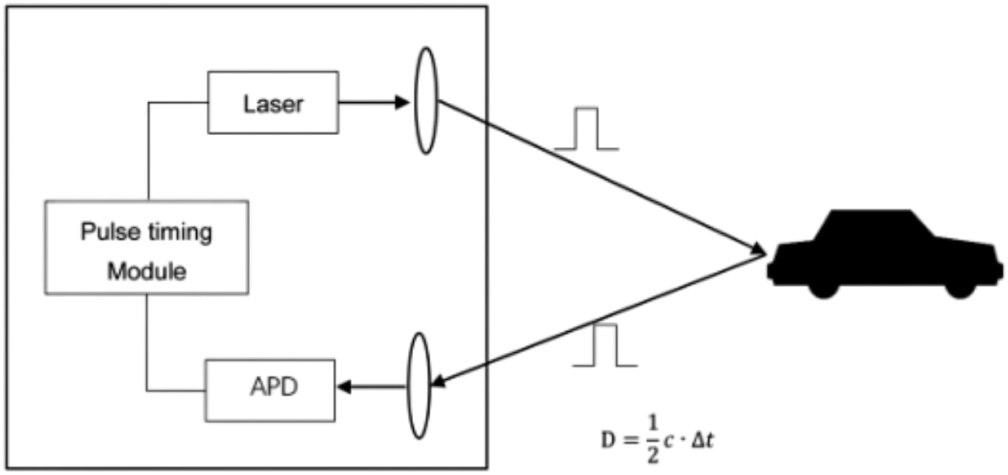
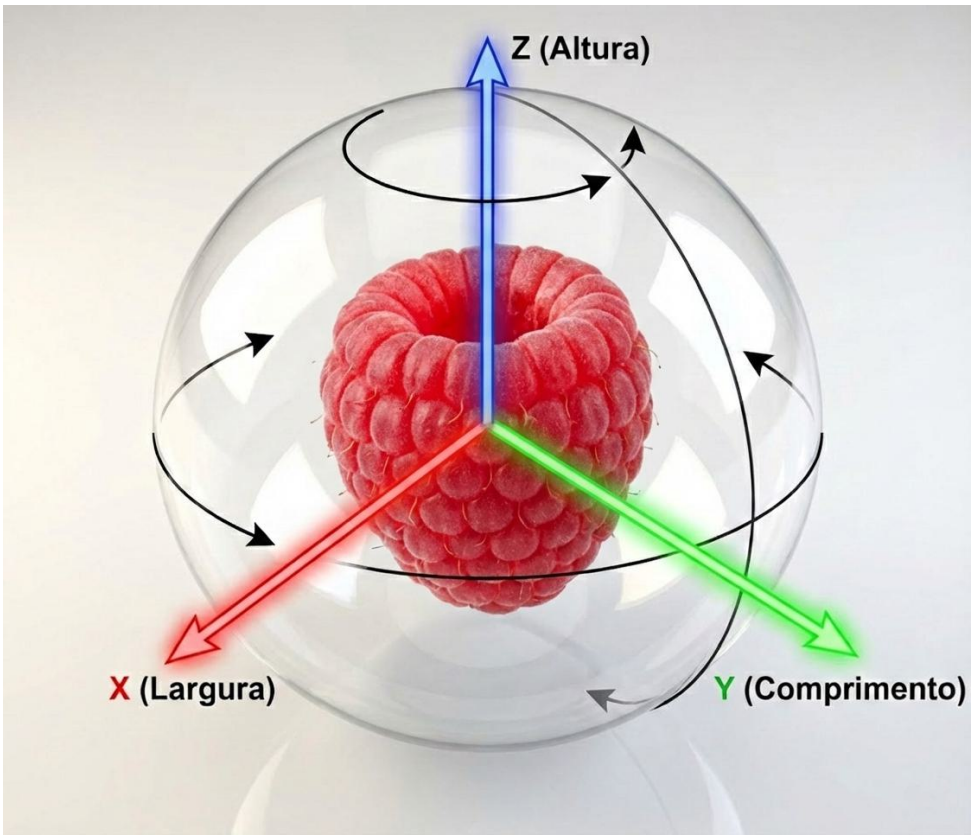


Figura 16 - Sensor de distância Seed Studio TF03-180

Bibliografia

- [1] SHAN, Jie; TOTH, Charles K. Topographic Laser Scanning: Principles and Processing. CRC Press, 2018. (Referência essencial para compreensão dos princípios de Tempo de Voo, hardware e processamento de nuvens de pontos 3D e aplicações agrícolas).
- [2] NOAA (National Oceanic and Atmospheric Administration). What is LIDAR?. National Ocean Service. Disponível em: https://oceanservice.noaa.gov/facts/lidar.html. (Fonte oficial governamental explicando o uso prático e a diferença física entre lasers infravermelhos para topografia e lasers verdes para batimetria).
- [3] SCHWARZ, B. "LIDAR: Mapping the world in 3D". Nature Photonics, v. 4, n. 7, p. 429-430, 2010. (Artigo de revisão detalhando os comprimentos de onda de 905nm, 1550nm e as tecnologias de escaneamento mecânico e estado sólido/MEMS).
- [4] NASA. Mars 2020 Perseverance Rover - Instruments. Disponível em: https://mars.nasa.gov/mars2020/spacecraft/instruments/. (Documentação oficial sobre o uso da tecnologia LiDAR em sondas espaciais e navegação autônoma fora da Terra).
- [5] THRUN, Sebastian; BURGARD, Wolfram; FOX, Dieter. Probabilistic Robotics. MIT Press, 2005. (Literatura fundamental sobre a integração de sensores LiDAR em veículos autônomos e robótica para localização e mapeamento simultâneos - SLAM).
- [6] MOUSER ELECTRONICS. Catálogo de Sensores Ópticos e LiDAR. Disponível em: https://br.mouser.com/. (Portal técnico da distribuidora referenciada no texto para consulta de datasheets e especificações dos componentes TF03-180, TFA300 e GRF-500).



Usando o Acelerômetro MPU6050 com a Raspberry Pi Pico 2

Luiz Henrique Correa Bernardes
Renato Paiotti

Neste artigo temos o uso do integrado MPU6050 através do módulo GY-521, que é um acelerômetro muito comum no mercado e que pode ser utilizado em diversos projetos que precisam de estabilidade através de dados como velocidade e posição.

O MPU6050

O Circuito Integrado MPU6050 foi desenvolvido pela InvenSense (atual TDK) para equipar smartphones, pois ele possui 6 eixos de rastreamento, o acelerômetro nos 3 eixos e um giroscópio de 3 eixos, tudo num único integrado. A TDK já tem atualmente uma versão mais moderna deste CI, é o ICM-4270-P[1], mas é fácil encontrar no mercado o MPU-6050.

O GY-521

Este é o módulo que auxilia o uso do MPU6050 para quem está na fase de protótipo ou em trabalhos makers. Como é fabricado por diversas empresas, as diferenças aparecem, mas na maioria dos casos respeitam a disposição dos pinos como apresentado na **figura 1**.

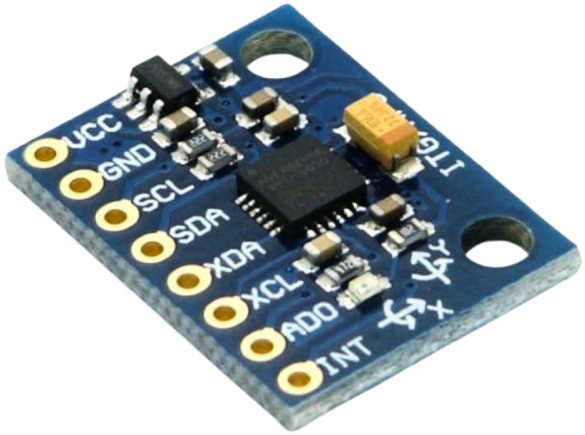


Figura 1 - O MPU6050 no módulo.

Por ser uma placa que utiliza o protocolo I2C, ela já vem com um endereço padrão, o 0x68HEX, mas é possível usar um outro acelerômetro ou módulo com o endereço 0x69HEX (pino AD0 ao VCC), ou outros sensores usando os pinos XDA e XCL como auxiliares na comunicação I2C. A seguir temos as especificações de cada pino do módulo.

Alimentação e Terra

VCC: É o pino de entrada de energia. O módulo GY-521 possui um regulador de tensão interno, o que permite alimentá-lo com 3.3V ou 5V com segurança.

GND: Pino de aterramento (0V). Deve ser conectado ao GND do seu microcontrolador.

Comunicação I2C (Principal)

SCL (Serial Clock): Fornece o sinal de clock para a comunicação I2C. Ele sincroniza a transferência de dados entre o sensor e o processador.

SDA (Serial Data): É a linha por onde os dados de movimento (aceleração e rotação) são enviados de fato.

I2C Auxiliar (Master)

Estes pinos permitem que o MPU-6050 atue como um "mestre" para outros sensores externos, como um magnetômetro (bússola), sem sobrecarregar o processador principal:

XDA (Auxiliary Data): Linha de dados para o barramento I2C auxiliar.

XCL (Auxiliary Clock): Linha de clock para o barramento I2C auxiliar.

Configuração e Controle

AD0 (Address Select): Define o endereço I2C do módulo.

Se estiver desconectado ou no GND, o endereço padrão será 0x68.

Se estiver conectado ao VCC, o endereço muda para 0x69. Isso permite usar dois módulos MPU-6050 no mesmo projeto.

INT (Interrupt): Pino de interrupção. Ele avisa ao seu microcontrolador quando novos dados estão prontos para serem lidos ou quando um evento específico (como uma queda livre ou movimento brusco) foi detectado.

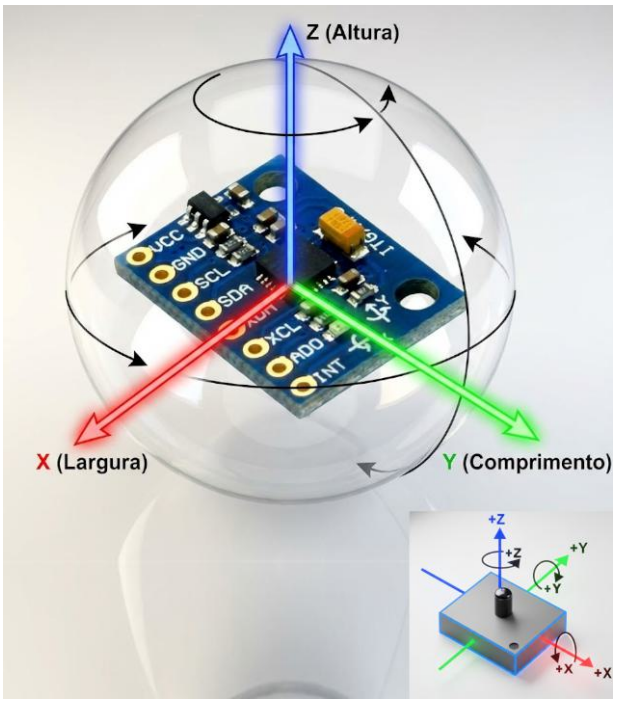


Figura 2 - Os Movimentos aplicados no módulo.

O Funcionamento do Módulo

Quando o MPU 6050 é inicializado , ele começa a detectar qualquer tipo de movimento e converter tanto o deslocamento entre os eixos X, Y e Z, como também a velocidade que este movimento é feito, convertendo essas informações em dados e transmitindo para a Pi Pico quando estes dados são solicitados.

Na **figura 2** temos um gráfico de como funciona fisicamente estes movimentos.

Conectando na Pi Pico 2

Na **figura 3** temos o esquema de ligação do módulo com a Pi Pico, lembrando aqui que a fonte de energia vem via USB o qual estamos subindo o código. Em caso de funcionamento fora do ambiente de desenvolvimento , uma fonte externa é necessária.

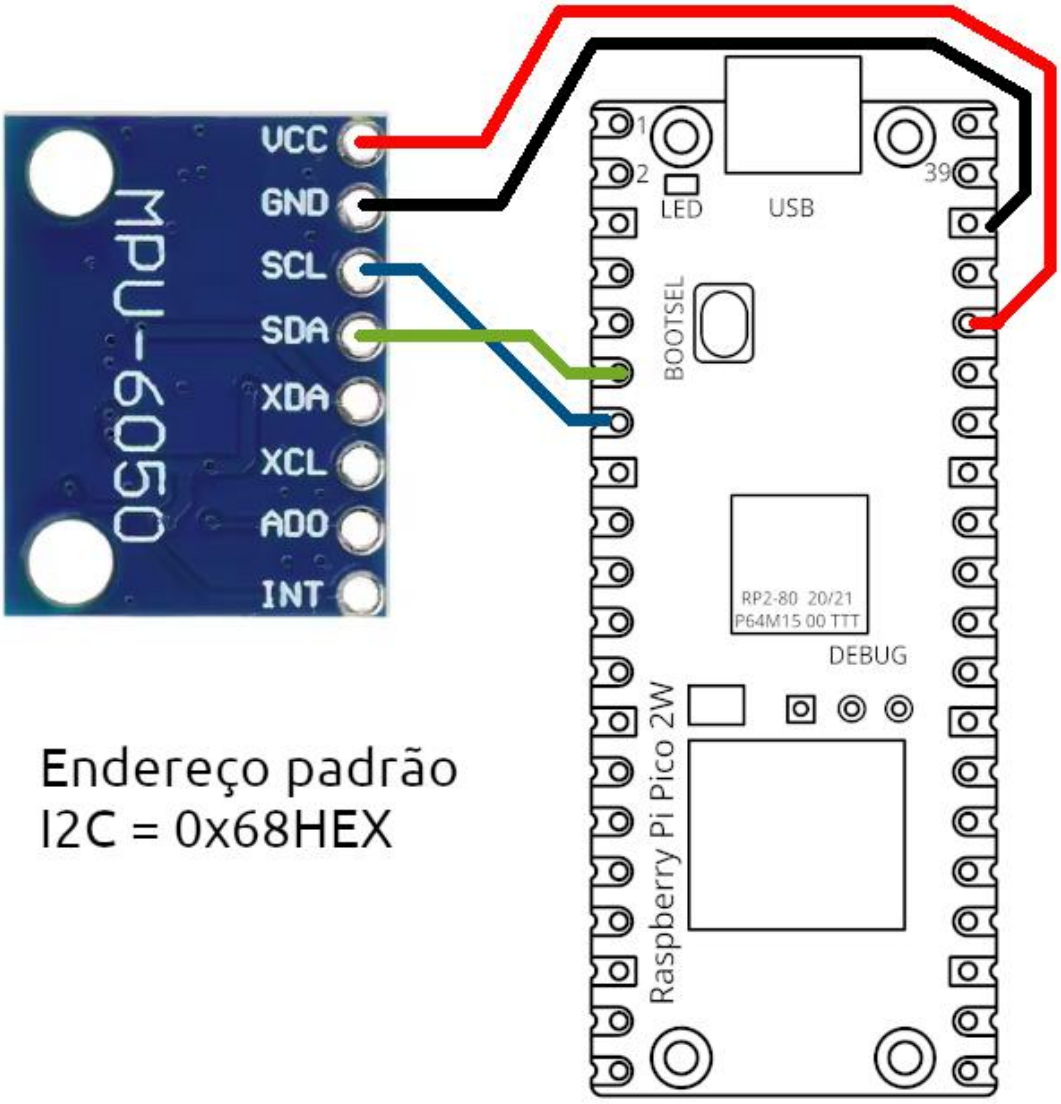


Figura 3 - Esquema Elétrico.

Código-Fonte

Vamos começar com um código de teste, uma função que captura os dados sem tratamento nenhum e mostrá-lo no shell da IDE do Thonny.

```
from machine import Pin, I2C
from time import sleep
#atribuimos o endereço em hexadecimal do MPU
MPU_ADDR = 0x68
i2c = I2C(0, scl=Pin(5), sda=Pin(4), freq=400000)
# Acorda o MPU6050
i2c.writeto_mem(MPU_ADDR, 0x6B, b'\x00')
# função a ser chamada
def read_raw(reg):
    high = i2c.readfrom_mem(MPU_ADDR, reg, 1)[0]
    low = i2c.readfrom_mem(MPU_ADDR, reg+1, 1)[0]
    value = (high << 8) | low
    if value > 32767:
        value -= 65536
    return value
while True:
    ax = read_raw(0x3B) / 16384
    ay = read_raw(0x3D) / 16384
    az = read_raw(0x3F) / 16384
    gx = read_raw(0x43) / 131
    gy = read_raw(0x45) / 131
    gz = read_raw(0x47) / 131
    print("Aceleração (g):", ax, ay, az)
    print("Giroscópio (°/s):", gx, gy, gz)
    print("-----")
    sleep(0.5)
```

Agora vamos utilizar o módulo para detectarmos o movimento de algo onde ele é aplicado.

```
from machine import Pin, I2C
from time import sleep
import math
# I2C nos pinos 4 (SDA) e 5 (SCL)
i2c = I2C(0, scl=Pin(5), sda=Pin(4), freq=100000)
addr = 0x68
# Acorda o MPU6050
```

```
i2c.writeto_mem(addr, 0x6B, b'\x00')
def read_raw(reg):
    high = i2c.readfrom_mem(addr, reg, 1)[0]
    low = i2c.readfrom_mem(addr, reg+1, 1)[0]
    value = (high << 8) | low
    if value > 32767:
        value -= 65536
    return value
def get_angles():
    ax = read_raw(0x3B) / 16384
    ay = read_raw(0x3D) / 16384
    az = read_raw(0x3F) / 16384

    # Cálculo dos ângulos
    pitch = math.degrees(math.atan2(ax, math.sqrt(ay*ay + az*az)))
    roll  = math.degrees(math.atan2(ay, math.sqrt(ax*ax + az*az)))
    return pitch, roll
def detect_gesture(pitch, roll):
    # Ajuste fino dos limites conforme seu uso
    if roll > 25:
        return "Inclinado para a direita"
    elif roll < -25:
        return "Inclinado para a esquerda"
    elif pitch > 25:
         return "Levantando a mão"
    elif pitch < -25:
        return "Baixando a mão"
    else:
        return "Neutro"
print("Controle gestual iniciado...")
while True:
    pitch, roll = get_angles()
    gesto = detect_gesture(pitch, roll)
    print(f"Pitch={pitch:.1f}  Roll={roll:.1f}  ->  Gesto: {gesto}")
    sleep(0.2)
```

E para finalizar, vamos utilizar o sensor como uma ferramenta, o “nível”.

```
from machine import Pin, I2C
from time import sleep
import math
```

```
# I2C0 nos pinos 4 (SDA) e 5 (SCL)
i2c = I2C(0, scl=Pin(5), sda=Pin(4), freq=100000)
addr = 0x68
# Acorda o MPU6050
i2c.writeto_mem(addr, 0x6B, b'\x00')
def read_raw(reg):
    high = i2c.readfrom_mem(addr, reg, 1)[0]
    low = i2c.readfrom_mem(addr, reg+1, 1)[0]
    value = (high << 8) | low
    if value > 32767:
        value -= 65536
    return value
def get_angles():
    ax = read_raw(0x3B) / 16384
    ay = read_raw(0x3D) / 16384
    az = read_raw(0x3F) / 16384
    pitch = math.degrees(math.atan2(ax, math.sqrt(ay*ay + az*az)))
    roll  = math.degrees(math.atan2(ay, math.sqrt(ax*ax + az*az)))
    return pitch, roll
def nivel_status(pitch, roll, limite=3):
    # limite = tolerância em graus para considerar "nivelado"
    if abs(pitch) < limite and abs(roll) < limite:
        return "NIVELADO"
    status = []
    if pitch > limite:
        status.append("Frente Alta")
    elif pitch < -limite:
        status.append("Frente Baixa")
    if roll > limite:
        status.append("Direita Alta")
    elif roll < -limite:
        status.append("Esquerda Alta")
    return " | ".join(status)
print("Nível digital iniciado...")
while True:
    pitch, roll = get_angles()
    estado = nivel_status(pitch, roll)
    print(f"Pitch={pitch:6.2f}°  Roll={roll:6.2f}°  ->  {estado}")
    sleep(0.2)
```

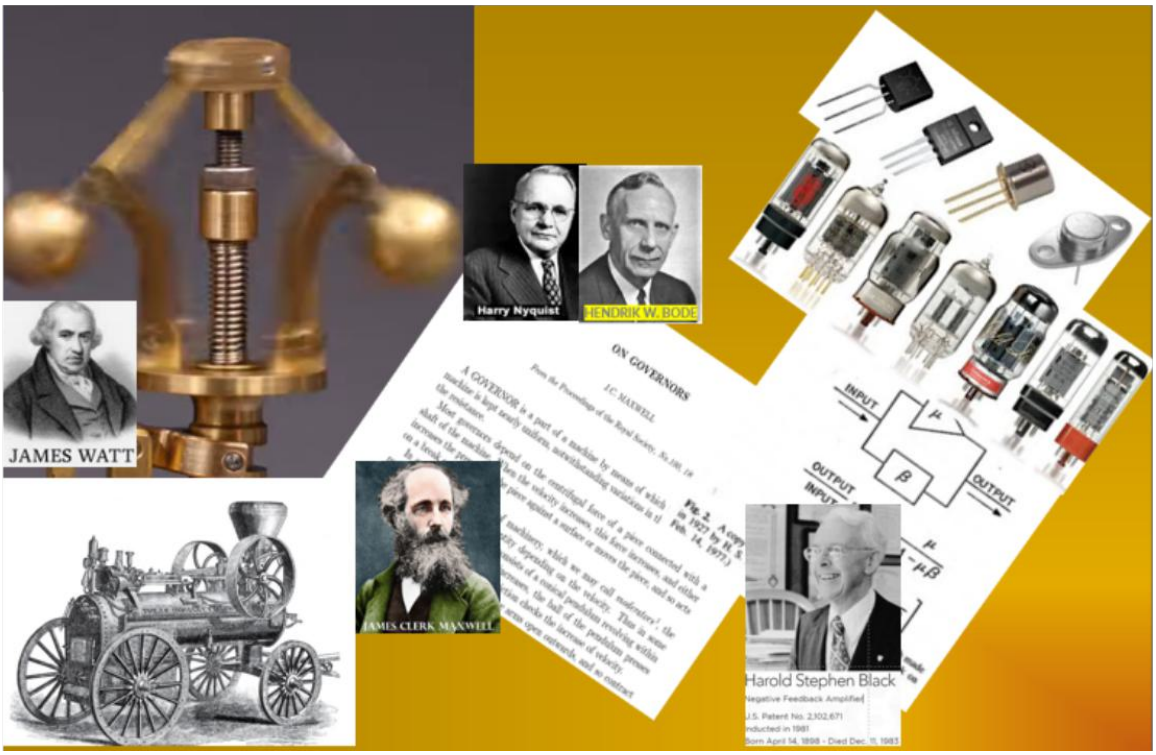
Conclusão

Atualmente, o uso de giroscópios e acelerômetros é comum em aplicações onde manter o alinhamento é fundamental, como em máquinas de precisão, drones, patinetes e dispositivos médicos. Existem diversos tipos de sensores, e o preço aumenta conforme a velocidade de processamento e a precisão. No entanto, para iniciar este módulo, o custo é acessível, permitindo realizar testes e experimentos sem grandes prejuízos.

Referências bibliográficas

ICM-42670
https://invensense.tdk.com/download-pdf/icm-42670-p-datasheet/

Série - A Saga de Usar a Pi Pico
https://www.youtube.com/watch?v=YUYZi53PirM&list=PLUg1G7GdWdJxOumCzW3H2cqWJY3a1-kVM



Polarização DC com Realimentação Negativa

MSc.Eng.Prof. Antonio Carlos Gasparetti

1 - Apresentação

Neste artigo vamos analisar a realimentação negativa em DC e como ela controla a polarização e a estabilidade dos circuitos eletrônicos. Serão apresentados exemplos para BJTs, FETs e válvulas termiônicas.

2 - Introdução

Projetistas de circuitos eletrônicos frequentemente se deparam com algumas questões técnicas importantes. Uma delas é por que o circuito permanece estável em sua polarização, mesmo com as variações de temperatura? E por que é possível substituir um transistor por outro, ou mesmo trocar uma válvula termiônica em um circuito de equipamento, com o

código de fabricante idêntico, sabendo que suas características não são exatamente iguais entre um componente e outro e mesmo assim o circuito continua funcionando de forma equivalente? A técnica utilizada vem da teoria de controle automático por malha fechada de realimentação, onde o conceito é manter um sistema estável de forma que as variações das características da planta mantenham o sistema dentro do valor ajustado pelo ponto de operação projetado (set point). Como curiosidade, um dos primeiros sistemas de controle automático foi o regulador centrífugo de velocidade usado em máquinas a vapor, desenvolvido por James Watt por volta de 1788. A primeira análise matemática de estabilidade foi feita por James Clerk Maxwell em 1868 no artigo “On Governors”. Esse trabalho é considerado o nascimento da teoria de controle. Harold Stephen Black, em 1927 nos Laboratórios Bell, concebeu o amplificador com realimentação negativa, seguido pela formalização matemática e análise de estabilidade por Harry Nyquist e Hendrik Wade Bode na década de 1930. [7][8][9]

3 - Controle com Realimentação Negativa em Malha Fechada

Um sistema de controle em malha fechada, como mostrado na **figura1**, utiliza a saída do próprio sistema para ajustar continuamente seu comportamento. A saída medida é comparada com um valor de referência, gerando um erro que é utilizado para corrigir o sinal aplicado à planta. Esse mecanismo permite reduzir erros, aumentar a estabilidade e tornar o sistema menos sensível a variações internas ou perturbações externas [7] [8] [9] [10].

As principais variáveis em um sistema de controle são:

- r(t) – sinal de referência (valor desejado)
- y(t) – saída do sistema
- e(t) – erro entre referência e saída
- u(t) – sinal de controle aplicado à planta

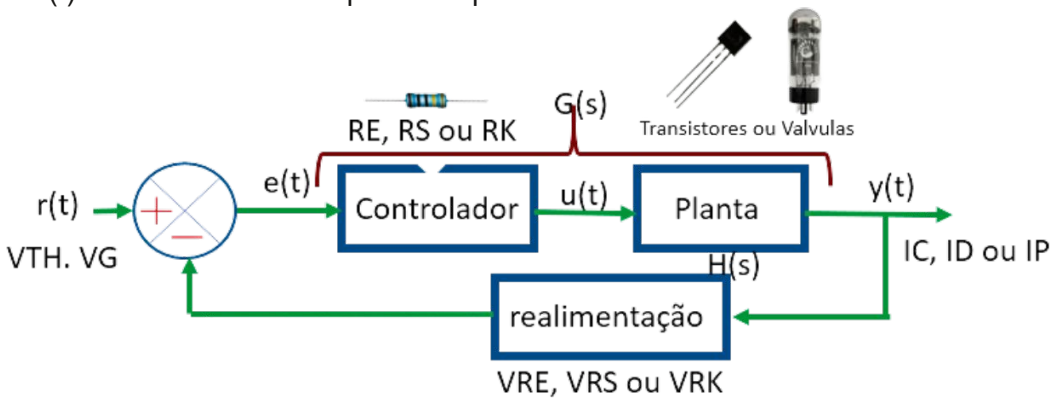


Figura 1 - Modelo geral de um sistema de controle em malha fechada.

O erro é definido por: e(t)=r(t)−y(t).

Com rede de realimentação H(s), no domínio de Laplace:

$$E(s) = R(s) - H(s)Y(s) \quad \text{eq.1}$$

Função de transferência

Se G(s) representa o caminho direto (controlador + planta):

$$Y(s) = G(s)E(s) \quad \text{eq.2}$$

Substituindo:

Logo, a função de transferência em malha fechada é:

$$T(s) = \frac{Y(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)} \quad \text{eq.3}$$

O termo G(s)H(s) é o ganho de malha, que determina estabilidade e sensibilidade do sistema [7][9].

4 - Analise da realimentação na polarização de um BJT por resistor de emissor

Considere o circuito da **figura 2**, de polarização com divisor de tensão substituído pelo equivalente de Thévenin.

4.1 - Equação de Malha DC

Considere o circuito de polarização com divisor de tensão substituído pelo equivalente de Thévenin [1], [2], [3]:

1. R1, R2→ divisor de tensão da base
2. RE → resistor de emissor (realimentação DC)
3. VCC → alimentação
4. VBE → queda base-emissor

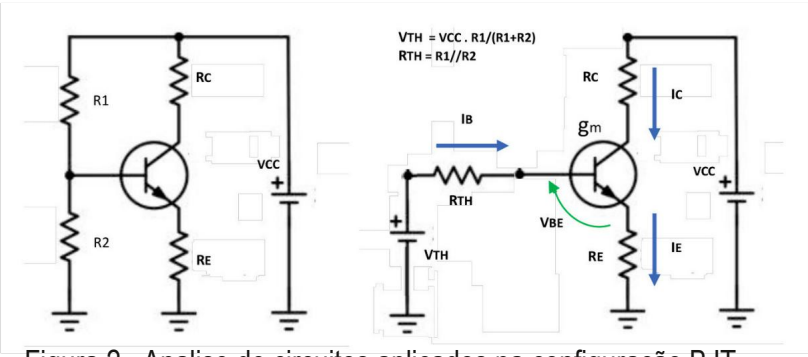


Figura 2 - Analise de circuitos aplicados na configuração BJT

Símbolo	Significado	Unidade
I_C	Corrente de coletor	A
I_E	Corrente de emissor	A
V_{BE}	Tensão base-emissor	V
R_E	Resistor de emissor	Ω
V_{TH}	Tensão equivalente na base	V
g_m	Transcondutância	S
V_T	Tensão térmica (~25mV)	V

Tabela 1 - Variáveis empregadas na análise

Equivalente de Thévenin do divisor de tensão na base do transistor:

$$V_{TH} = V_{CC}\frac{R_2}{R_1 + R_2}$$

$$R_{TH} = \frac{R_1 R_2}{R_1 + R_2}$$

eq.4 e eq.5

A equação da malha base–emissor é:

$V_{TH} - I_B R_{TH} - V_{BE} - I_E R_E = 0$

como

$I_E \approx (\beta + 1) I_B$

então

$V_{TH} = I_B R_{TH} + V_{BE} + (\beta + 1) I_B R_E$

Logo

$$I_B = \frac{V_{TH} - V_{BE}}{R_{TH} + (\beta + 1) R_E}$$

$$I_C = \beta I_B$$

$$I_C = \beta \frac{V_{TH} - V_{BE}}{R_{TH} + (\beta + 1) R_E}$$

eq.6

Essa é a equação de polarização do transistor.

4.2 Equação de Controle

Interpretação como sistema de controle:

- entrada (referência) → VTH
- saída → corrente IC
- realimentação → tensão gerada em RE

A tensão no emissor é:

$$V_E = I_E R_E$$

Essa tensão reduz a tensão efetiva de entrada, caracterizando realimentação negativa.

Erro do sistema o quanto precisa ser corrigido:

$$e = V_{TH} - V_E$$

A planta (transistor) transforma erro em corrente:

$$I_C = G(s)\, e$$

A rede de realimentação gera:

$$V_E = H(s) I_C$$

Substituindo no erro:

$$e = V_{TH} - H(s) I_C$$

Substituição na planta

$$I_C = G(s)[V_{TH} - H(s) I_C]$$

$$I_C = G(s) V_{TH} - G(s) H(s) I_C$$

$$I_C + G(s) H(s) I_C = G(s) V_{TH}$$

$$I_C [1 + G(s) H(s)] = G(s) V_{TH}$$

Função de transferência equivalente

$$\boxed{\frac{I_C(s)}{V_{TH}(s)} = \frac{G(s)}{1 + G(s) H(s)}} \qquad \boxed{\frac{I_C}{V_{TH}} = \frac{g_m}{1 + g_m R_E}}$$

eq.7

G(s) → ganho do transistor (transcondutância - gm)

H(s) → rede de realimentação associada ao resistor RE

Essas expressões correspondem diretamente à forma clássica de *realimentação negativa em malha fechada*.

4.3 Exemplo prático - Estabilidade Térmica (BJT)

Vamos comparar duas situações:

1. Sem resistor de emissor RE
2. Com resistor de emissor RE (realimentação negativa)

Dados do circuito:

- VTH = 2.0 V
- RE =1kΩ
- VT =25mV (25°C)
- Corrente nominal ≈ 1–2 mA

Relações utilizadas:

$$g_m = \frac{I_C}{V_T}$$

$$I_C = g_m V_{BE}$$

$$I_C = g_m(V_{TH} - I_C R_E)$$

4.3.1 Caso 1 — Circuito sem RE - **sem realimentação**.

$$I_C = g_m V_{BE}$$

Assumindo inicialmente:

$$V_{BE} \approx 0.7V$$

$$g_m = \frac{1mA}{25mV} = 0.04S$$

Então

$$I_C = 0.04 \times 0.7$$

$$I_C = 28mA$$

Variação com temperatura

O VBE do silício varia aproximadamente:

$$-2mV/^\circ C$$

Se a temperatura subir **25°C → 75°C**:

$$\Delta V_{BE} \approx -100mV$$

Novo valor de VBE para 75°C:

$$V_{BE} \approx 0.6V$$

Para ΔT = 30°C o resultado é:

Temperatura	V_{BE}	I_C
25°C	0.7 V	28 mA
75°C	0.6 V	24 mA

Tabela 2 - Variação de VBE e IC em função da variação de temperatura no dispositivo

Conforme a **tabela 2**, grande variação → polarização instável. Além disso, na prática pode ocorrer fuga térmica ("thermal runaway"), processo no qual o aumento da temperatura provoca aumento da corrente, e esse aumento de corrente gera ainda mais aquecimento, criando um ciclo de realimentação positiva térmica que pode levar à destruição do dispositivo.

Conclusão: o resistor de emissor RE é necessário para manter a estabilidade térmica do circuito. Mesmo com variações de temperatura, RE mantém o ponto de operação estável. [1], [6]

4.3.2 Caso 2 — Circuito **com RE**

Equação de controle:

$$I_C = \frac{g_m}{1 + g_m R_E} V_{TH}$$

Variação de temperatura

Se a temperatura variar, **gm** variará.

Comparação final

$$g_m = 0.02S$$

$$I_C = \frac{0.02}{21} \times 2$$

$$I_C \approx 1.90mA$$

$$g_m = 0.06S$$

$$I_C = \frac{0.06}{61} \times 2$$

$$I_C \approx 1.97mA$$

Situação	Corrente mínima	Corrente máxima	Variação absoluta	Variação %
Sem R_E	24 mA	28 mA	4 mA	**15.4 %**
Com R_E	1.90 mA	1.97 mA	0.07 mA	**3.6 %**

Tabela 3 - Análise final da resposta do circuito com e sem RE.

Conclusão. A **tabela 3** mostra como RE é necessário para manter a estabilidade térmica do circuito. Mesmo variando a temperatura, RE mantem o **circuito sob controle**.

4.4 Exemplo clássico – **variação de β com e sem RE**

Considere dois transistores 1 e 2, do mesmo modelo, porém com dispersão de fabricação no ganho de corrente, transistores do mesmo modelo podem ter:

$$50 \leq \beta \leq 200$$

Dados do circuito exemplo:

- Corrente de base fixa: $I_B = 10\,\mu A$
- $V_{TH} = 2V$
- $R_E = 1k\Omega$

Circuito sem resistor de emissor:

A corrente de coletor depende diretamente de β:

$$I_C = \beta I_B$$

Transistor 1

$$\beta = 50$$

$$I_C = 50 \times 10\mu A$$

$$\boxed{I_C = 0.5mA}$$

Transistor 2

$$\beta = 200$$

$$I_C = 200 \times 10\mu A$$

$$\boxed{I_C = 2mA}$$

Circuito com resistor de emissor:

$$I_C = \frac{g_m V_{TH}}{1 + g_m R_E} \qquad g_m R_E \gg 1 \qquad I_C \approx \frac{V_{TH}}{R_E}$$

$$I_C \approx \frac{2V}{1k\Omega} \qquad \boxed{I_C \approx 2mA}$$

Situação	β mínimo	β máximo	Corrente mínima	Corrente máxima	Variação %
Sem R_E	50	200	0.5 mA	2 mA	≈120%
Com R_E	50	200	1.96 mA	1.99 mA	≈1.5%

Tabela 4 - Análise final da resposta do circuito com e sem RE trocando o dispositivo

Conclusão

Conforme verificamos na tabela 4, sem resistor de emissor, a corrente depende diretamente de β. Portanto, trocar o transistor altera fortemente a polarização. Com resistor de emissor ocorre realimentação negativa em DC e a corrente passa a depender principalmente da tensão de polarização e do resistor RE.

$$I_C \approx \frac{V_{TH}}{R_E}$$

Assim, mesmo trocando o transistor, o ponto de operação permanece praticamente constante.

4.5 JFET – Estabilidade de polarização

Considere o circuito JFET a seguir.

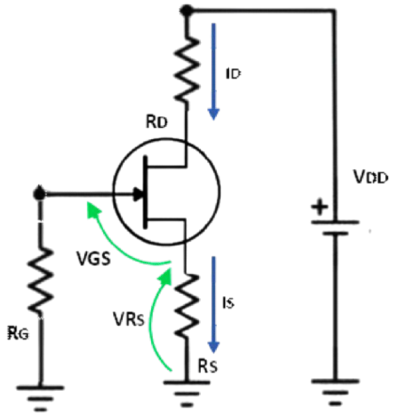


Figura 3

4.5.1 JFET com resistor de source RS.

Analise do JFET tem por referências [3], [6].

Variáveis principais:

- V_G – tensão de gate (definida pelo divisor)
- $V_S = I_D R_S$
- $V_{GS} = V_G - V_S$

Equação de polarização $V_{GS} = V_G - I_D R_S$

Modelo do JFET: $I_D = g_m V_{GS}$

$$I_D = g_m(V_G - I_D R_S)$$

$$I_D = g_m V_{GS}$$

$$I_D + g_m R_S I_D = g_m V_G$$

$$I_D(1 + g_m R_S) = g_m V_G$$

$$\boxed{I_D = \frac{g_m}{1 + g_m R_S} V_G}$$

Eq. 8

$$\frac{\text{Saída}}{\text{Entrada}} = \frac{G}{1 + GH} \qquad G = g_m \qquad H = R_S$$

$$V_G = 0 \Rightarrow I_D = \frac{g_m}{1 + g_m R_S} \times 0 \Rightarrow I_D = 0$$

Isso mostra que **o modelo linear não determina o ponto DC**.

$$I_D = I_{DSS}\left(1 - \frac{V_{GS}}{V_P}\right)^2 \qquad V_{GS} = -I_D R_S$$

Equação não linear que determina o ponto de polarização DC para o JFET

4.5.2 Exemplo numérico – efeito da temperatura em JFET com auto-polarização (VG=0V)

Considere um JFET com resistor de source RS. Dados do circuito e equação de controle:

$$I_{DSS}(25^{\circ}C) = 10\,mA$$

$$V_P = -4V$$

$$R_S = 1k\Omega$$

$$V_{GS} = -I_D R_S$$

$$I_D = I_{DSS}\left(1 - \frac{V_{GS}}{V_P}\right)^2 \quad \Rightarrow \quad I_D = I_{DSS}\left(1 - \frac{-I_D R_S}{V_P}\right)^2$$

Dados do circuito	Temperatura 25°C	Temperatura 75°C
$I_D \approx 3mA$ $V_S = I_D R_S$ $V_S = 3V$ $V_{GS} = -3V$ $V_P = -4V$ $R_S = 1k\Omega$	$I_{DSS}(25^{\circ}C) = 10\,mA$ $I_D = 10\left(1 - \frac{-3}{-4}\right)^2$ $I_D = 10(0.25)^2$ $I_D \approx 2.5mA$	I_{DSS} (75°C) $= 13\,mA$ $I_D = 13\left(1 - \frac{-3}{-4}\right)^2$ $I_D = 13(1 - 250I_D)^2$ $I_D \approx 2.7mA$
$V_{GS} = -I_D R_S$	$V_{GS} = -2.5V$	$V_{GS} = -2.7V$

Tabela comparativa

Temperatura	Variação Temp.	I_{DSS}	Variação I_{DSS}	I_D	Variação I_D
25 °C	—	10 mA	—	2.5 mA	—
75 °C	+200 %	13 mA	+30 %	2.7 mA	+8 %

Tabela 5 - Variação da temperatura e de IDSS e o efeito de RS na estabilização Térmica

Conclusão: Examinando a tabela 5, mesmo com cerca de 30% de variação no parâmetro IDSS do transistor, a corrente do circuito varia apenas aproximadamente 8%, mostrando o efeito estabilizador da auto-polarização.

5.0 Análise da válvula termiônica (triodo) com auto-polarização

Considere um tríodo de pequeno sinal [5].

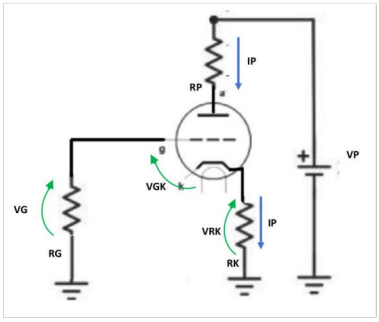


Figura 4

A análise segue a mesma lógica usada para o JFET, mas usando os parâmetros típicos das válvulas em autopolarização.

Elementos principais:

- Resistor de cátodo RK
- Resistor de grade RG

Como a corrente de grade é praticamente nula:

$$V_G \approx 0 \qquad I_G \approx 0$$

5.1 Equações de polarização

Tensão no cátodo:

$$V_K = I_P R_K$$

Tensão grade-cátodo:

$$V_{GK} = V_G - V_K$$

Como VG≈0V

$$V_{GK} = -I_P R_K$$

Modelo aproximado da válvula

Para pequenas variações pode-se usar o modelo de transcondutância:

$$I_P \approx g_m V_{GK}$$

Substituindo:

$$I_P = g_m(-I_P R_K)$$

Assim como no JFET, esse modelo não define sozinho o ponto DC, que vem das curvas da válvula. A válvula tem uma equação característica própria, e é ela que determina o ponto de operação DC. O modelo linear com gm serve apenas para pequenas variações em torno do ponto de operação. Para um triodo, a equação clássica usada para modelagem é a equação de Child–Langmuir modificada (modelo de Langmuir):

$$I_P = K\left(V_{GK} + \frac{V_P}{\mu}\right)^{3/2}$$

onde:

- I_P → corrente de placa
- V_{GK} → tensão grade-cátodo
- V_P → tensão placa-cátodo
- μ → fator de amplificação da válvula
- K → constante do dispositivo

Eq.9

Essa equação define o modelo físico da válvula.

5.2 Inserindo a auto-polarização no circuito da válvula termiônica

No circuito com resistor de cátodo:

$$V_G \approx 0$$

$$V_K = I_P R_K$$

$$V_{GK} = V_G - V_K$$

$$V_{GK} = -I_P R_K$$

$$I_P = K\left(V_{GK} + \frac{V_P}{\mu}\right)^{3/2} \Rightarrow \boxed{I_P = K\left(-I_P R_K + \frac{V_P}{\mu}\right)^{3/2}}$$

onde:

- I_P → corrente de placa
- V_{GK} → tensão grade-cátodo
- V_P → tensão placa-cátodo
- μ → fator de amplificação da válvula
- K → constante do dispositivo

5.3 Exemplos de variação de temperatura e variação de características de uma válvula do mesmo tipo e código de fabricação.

Considere um tríodo pequeno.

$V_P = 200V \quad \mu = 20 \quad K = 0.002 \quad R_K = 1k\Omega \Rightarrow I_P \approx 4mA$

$K \uparrow 30\%$

$K = 0.002 \Rightarrow K = 0.0026$

$V_{GK} = 0 \quad I_P = K\left(\frac{V_P}{\mu}\right)^{3/2} \quad I_P = 0.002(10)^{3/2} \quad I_P \approx 4mA$

$I_P \approx 4mA \Rightarrow I_P \approx 5.2mA \quad +30\%$

$V_K = 5.2V \quad V_{GK} = -5.2V \quad I_P \approx 5.2mA \Rightarrow I_P \approx 4.4mA$ 15,4 %. $I_P \approx 4mA$ 10 %.

Caso 2 — Troca da válvula (mesmo modelo)

É comum duas válvulas do mesmo tipo terem diferenças grandes.

Vamos analisar trocando a válvula A pela válvula B

$K = 0.002$ $K = 0.0028$

Sem RK

Valvula A *Valvula B*

$I_P \approx 4mA$ $I_P \approx 5.6mA$ $+40\%$

Com RK

$I_P \approx 4mA$ $I_P \approx 4.5mA$ +12.5 %

5.3.1 Tabela comparativa

Situação	Corrente inicial	Corrente final	Variação
Sem R_K temperatura	4 mA	5.2 mA	+30 %
Com R_K temperatura	4 mA	4.4 mA	+10 %
Sem R_K troca válvula	4 mA	5.6 mA	+40 %
Com R_K troca válvula	4 mA	4.5 mA	+12.5 %

Tabela 6 - O efeito de RK na estabilização do circuito

Interpretação física

Como podemos verificar na tabela 6, quanto mais IP aumenta, mais VK aumenta, tornando VGK mais negativo, o que reduz automaticamente a condução da válvula, para os casos de troca do mesmo tipo e código do dispositivo com características diferentes quanto com relação a temperatura.

$$\uparrow V_K = \uparrow I_P R_K \qquad \uparrow -V_{GK} = \uparrow I_P R_K$$

Isso reduz automaticamente a condução da válvula. Esse mecanismo é realimentação negativa DC.

Conclusão geral

As análises mostram um princípio fundamental da eletrônica analógica:

A estabilidade da polarização não deve depender do dispositivo, mas do circuito.

O uso de resistores de auto-polarização:

- reduz a sensibilidade a temperatura
- permite trocar dispositivos do mesmo tipo
- evita instabilidade térmica
- estabiliza o ponto de operação

Esse princípio é um exemplo clássico de realimentação negativa aplicada à polarização DC.

Referências

[1] Sedra, A. S.; Smith, K. C. Microelectronic Circuits. 7. ed. Oxford: Oxford University Press, 2015.

[2] Millman, J.; Halkias, C. Integrated Electronics: Analog and Digital Circuits and Systems. New York: McGraw-Hill, 1972.

[3] Boylestad, R. L.; Nashelsky, L. Electronic Devices and Circuit Theory. 11. ed. Boston: Pearson, 2013.

[4] Gray, P. R.; Hurst, P. J.; Lewis, S. H.; Meyer, R. G. Analysis and Design of Analog Integrated Circuits. 5. ed. New York: Wiley, 2009.

[5] Langford-Smith, F. Radiotron Designer's Handbook. 4. ed. Sydney: Wireless Press, 1953.

[6] Horowitz, P.; Hill, W. The Art of Electronics. 3. ed. Cambridge: Cambridge University Press, 2015.

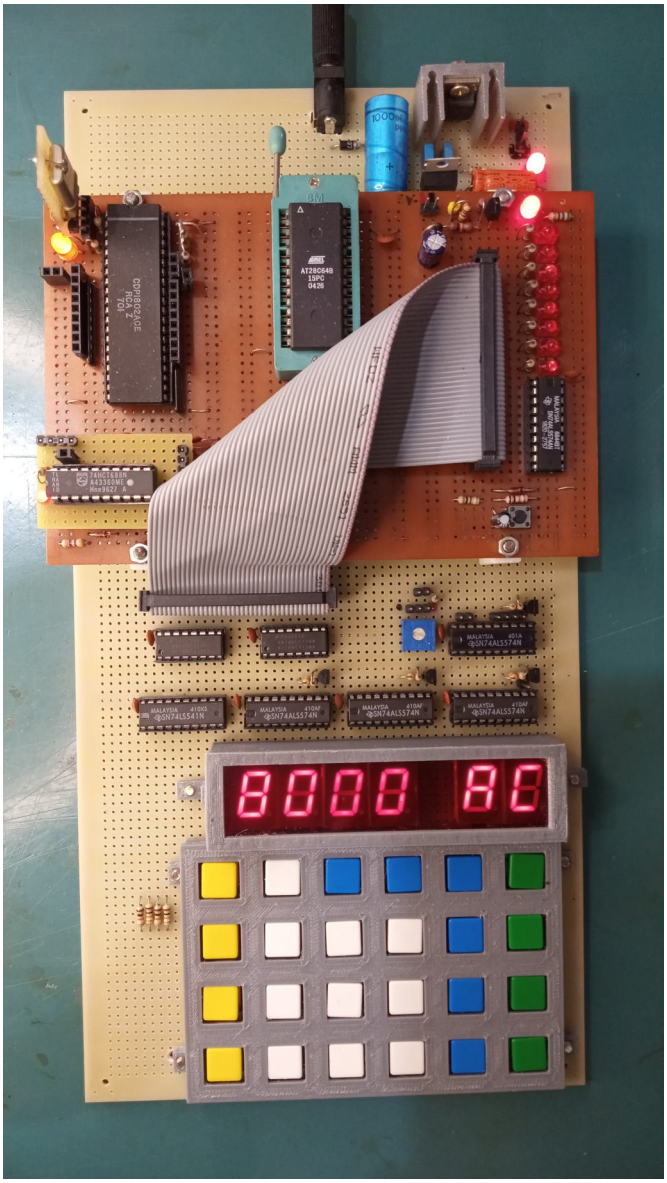
[7] Ogata, K. Modern Control Engineering. 5. ed. Upper Saddle River: Prentice Hall, 2010.

[8] Dorf, R. C.; Bishop, R. H. Modern Control Systems. 13. ed. Boston: Pearson, 2017.

[9] Franklin, G. F.; Powell, J. D.; Emami-Naeini, A. Feedback Control of Dynamic Systems. 8. ed. Boston: Pearson, 2019.

[10] Nise, N. S. Control Systems Engineering. 8. ed. Hoboken: Wiley, 2020.

Computador COSMAC Parte 3



Eng. Márcio José Soares

Esta é a terceira parte do projeto COSMAC-ELF que iniciamos na edição nr 31. Se você ainda não viu a primeira e a segunda parte, vale a pena o download das revistas para a leitura. É de graça! Nesta terceira parte vamos apresentar o circuito com os 6 displays de 7 segmentos cada. Se você curte montagens e circuitos digitais/ microprocessados no estilo “old time” essa série de artigos foi feita na medida para você! Enjoy!

A proposta

A proposta deste artigo é permitir ao leitor continuar a montagem da placa de acessórios, proposta na edição passada, que ampliará os recursos do COSMAC-ELF CPU apresentado na primeira parte desta saga. Agora com o display nos aproximamos do programa monitor a ser apresentado na próxima edição.

Um pouco de teoria – Displays de 7 segmentos

Os displays de sete segmentos são compostos basicamente de LED's dispostos de maneira a formar um "dígito". De acordo com que são "acesos" ou "apagados" é possível demonstrar os números de "0 a 9" e até mesmo algumas letras de forma bem rudimentar. Na **figura 1** é possível observar a posição de cada segmento, sua nomenclatura, assim como sua ligação através dos pinos. Existem muitos outros modelos, com pinagens diferentes e até mesmo com mais segmentos. E antes de usá-los é recomendável uma consulta ao datasheet do fabricante[1] [8].

Além dos tipos, tamanhos, cores e "pinagens" diferentes temos também que observar a ligação do pino "comum". Ele pode ser do tipo "anodo comum" ou "catodo comum". A escolha de um ou outro depende do projeto, mas um não substitui o outro. A **figura 2** demonstra os dois tipos de ligação.

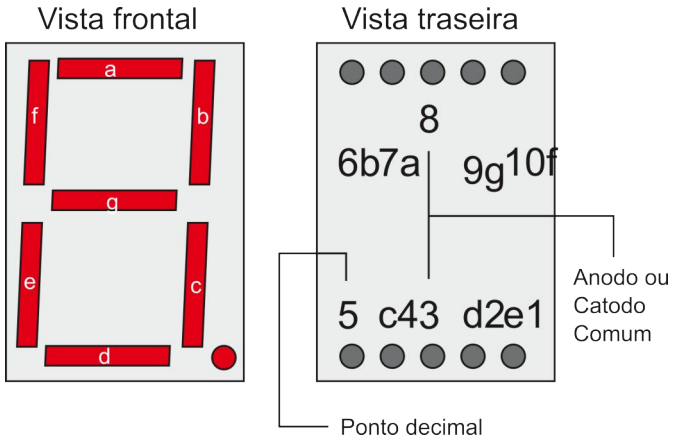


Figura 1 - Visão frontal e traseira de um display de 7 segmentos

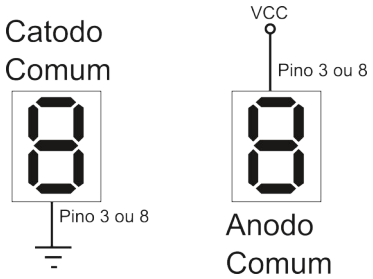


Figura 2 - Ligação dos displays de catodo ou anodo comum

Como podemos notar, precisamos de sete pinos de I/O (ou oito, se considerarmos o ponto) no circuito para ligar um display do tipo. Mas e se precisarmos controlar mais que um?! Como fazer?! Usar uma quantidade ainda maior de pinos de I/O?!? Isso seria viável?!? A resposta é simples: podemos aqui também usar o recurso da "varredura".

A varredura não serve apenas para teclados matriciais. Podemos utilizá-la também com displays, porém de uma forma diferente. Na **figura 3**, temos a demonstração de como isso pode ser implementado. Ligamos todas as linhas de A à G (segmentos) juntas (A com A, B com B e assim sucessivamente). Assim quando enviamos um valor para um display, estamos enviando para todos, pois formamos um bus (barramento) para as linhas A-G.

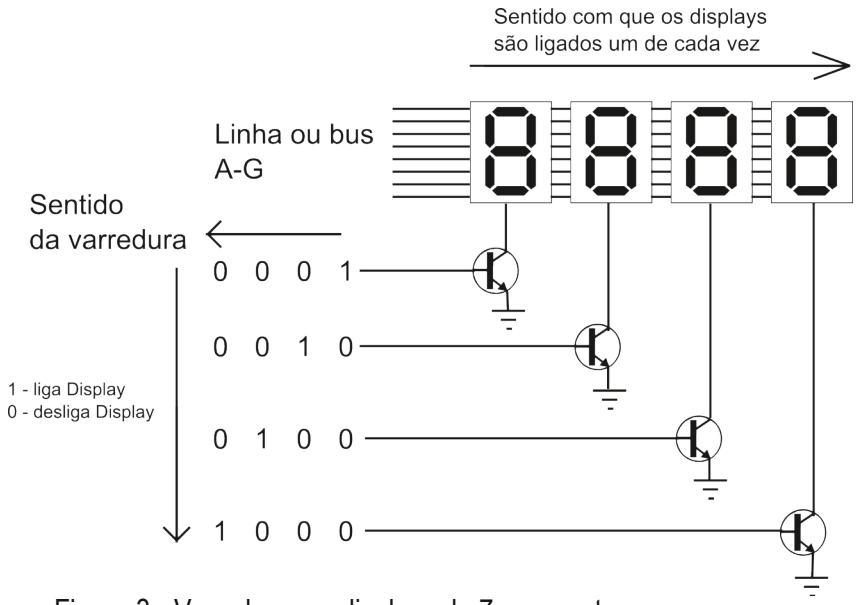


Figura 3 - Varredura em displays de 7 segmentos

O que determinará qual display mostrará um valor qualquer será o controle no anodo/catodo. Ao inserir um byte nas linhas A-G, estando todos os anodos (ou catodos) despolarizados, nenhum segmento de nenhum display será ligado. Porém se "ligarmos" um dos displays através de seu anodo (ou catodo) o byte presente nas linhas A-G será apresentado por este display na posição desejada. Para mostrar um outro byte em outro display, basta trocar o byte na linha A-G e "ligar" apenas o display desejado. Estamos assim realizando uma varredura,

usando mais de um display e apenas um “bus” de A-G. O ponto mais importante aqui é a velocidade com que isso deve ser feito, pois devido à “persistência retiniana” não perceberemos que os bytes estão sendo trocados e muito menos que a posição dos mesmos está variando. Em média podemos usar tempos entre 5 ms (0,005s) à 30 ms (0,03s), que serão mais que suficientes para “enganar” nossa visão [2].

Teremos assim a impressão de estarmos vendo um dado com um número de dígitos todos mostrados ao mesmo tempo. No caso deste projeto em específico serão 4 dígitos para o endereço selecionado da memória e mais 2 dígitos para o byte presente no referido endereço.

Circuitos da placa de acessórios

Circuito dos displays

Na **figura 4** o leitor pode ver o circuito do display da placa COSMAC-ELF. Ele possui 6 displays de 7 segmentos do tipo catodo comum, sendo 4 displays dedicados a mostrar um endereço de memória qualquer (ou conteúdo de um registrador de 16 bits) e mais 2 displays para mostrar o byte presente no endereço ou ainda o nr do registrador selecionado (além de outras implementações futuras).

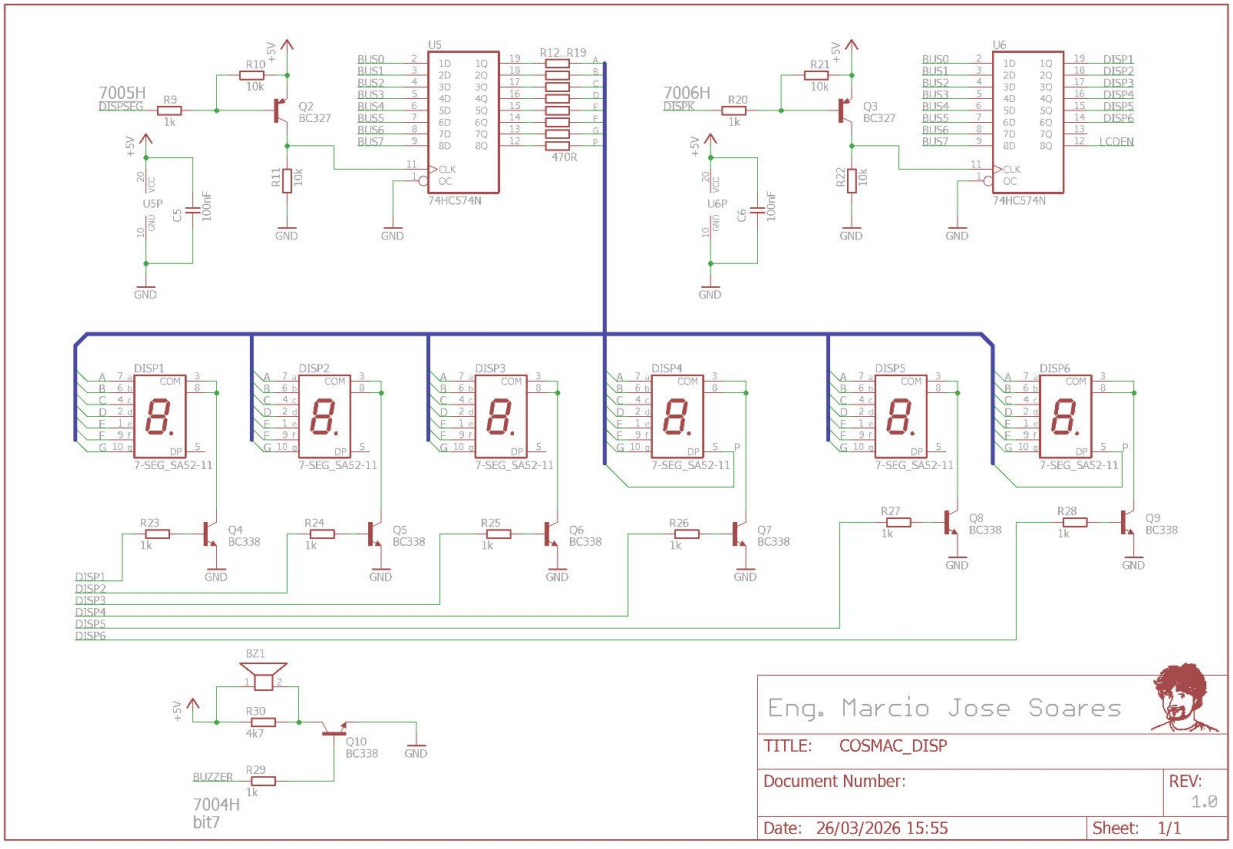


Figura 4 - Circuito do display da placa COMAC_ELFO

U5 é um Octal D Lath com saída 3-state utilizado na transferência do byte presente no barramento de dados do COSMAC-ELF para o bus “A-G+Ponto” dos displays. Essa transferência ocorre quando Q2 é devidamente polarizado através de U1 (circuito do teclado - edição número 32) com endereço de saída 7005H.

U6 é do mesmo tipo que U5, porém utilizado na transferência do byte presente no barramento do COSMAC-ELF para a polarização dos transistores Q4 à Q9 que controlam os catodos dos displays.

R9, R10, R11 são utilizados na polarização de Q2, assim como R20,R21 e R22 na polarização de Q3. Os resistores R12 à R19 são resistores limitadores de corrente para os segmentos A-G e “ponto” dos displays. Os resistores R23 à R28 auxiliam no controle da corrente de base necessária para o chaveamento dos transistores Q4 à Q9, respectivamente.

Os capacitores C5 e C6 servem como filtro para os Cis U5 e U6, respectivamente.

Temos ainda a presença de um circuito buzzer que servirá para usos futuros do programa monitor (que será apresentado na próxima edição). Este pequeno circuito é composto por BZ1, seu transistor de controle Q10 e os resistores R29 que auxilia no controle da corrente de base para o chaveamento do transistor e R30 que atua como carga no coletor. O buzzer é acionado via endereço 7004H (o mesmo usado no teclado) porém considerando apenas o bit7 deste.

Montagem

Na **figura 5** o leitor pode ver a montagem feita pelo autor em uma placa do tipo padrão com 300 x 150 mm. Todas as conexões foram feitas com fio wire warpping soldados ponto a ponto. Apesar de parecer um tanto “confusa”, essa técnica permite montagens de circuitos complexos em um espaço menor se comparado a uma montagem em matriz de contatos, por exemplo, e com a vantagem de não sofrer com o “fantasma” do mau contato.

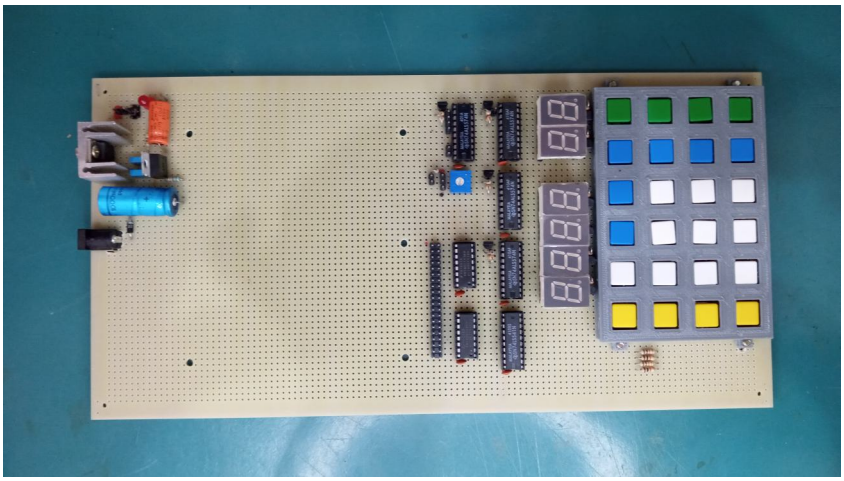


Figura 5 - Placa de acessório montada pelo autor.

Caso você ainda não tenha iniciado ainda a sua montagem, você poderá montar as partes apresentadas nas edições anteriores (nr 31 e nr 32) juntamente com a parte descrita neste artigo na mesma placa.

É altamente recomendado o uso de soquetes para todos os CI's, inclusive para os displays de 7 segmentos. Neste caso será necessário adaptá-los a partir de soquetes de 24 ou 32 pinos do tipo large. Cuidado para não inverter os componentes polarizados como CI's. diodos, transistores, capacitores eletrolíticos, além dos próprios displays.

Seja qual for o tipo de montagem escolhida pelo leitor a recomendação é sempre verificar todas as conexões durante e após a montagem. Confundir um pino ou um lado inteiro de um CI é bastante comum, já que a placa fica literalmente de "cabeça para baixo" durante a sua montagem. Verifique quantas vezes puder antes de ligar o circuito.

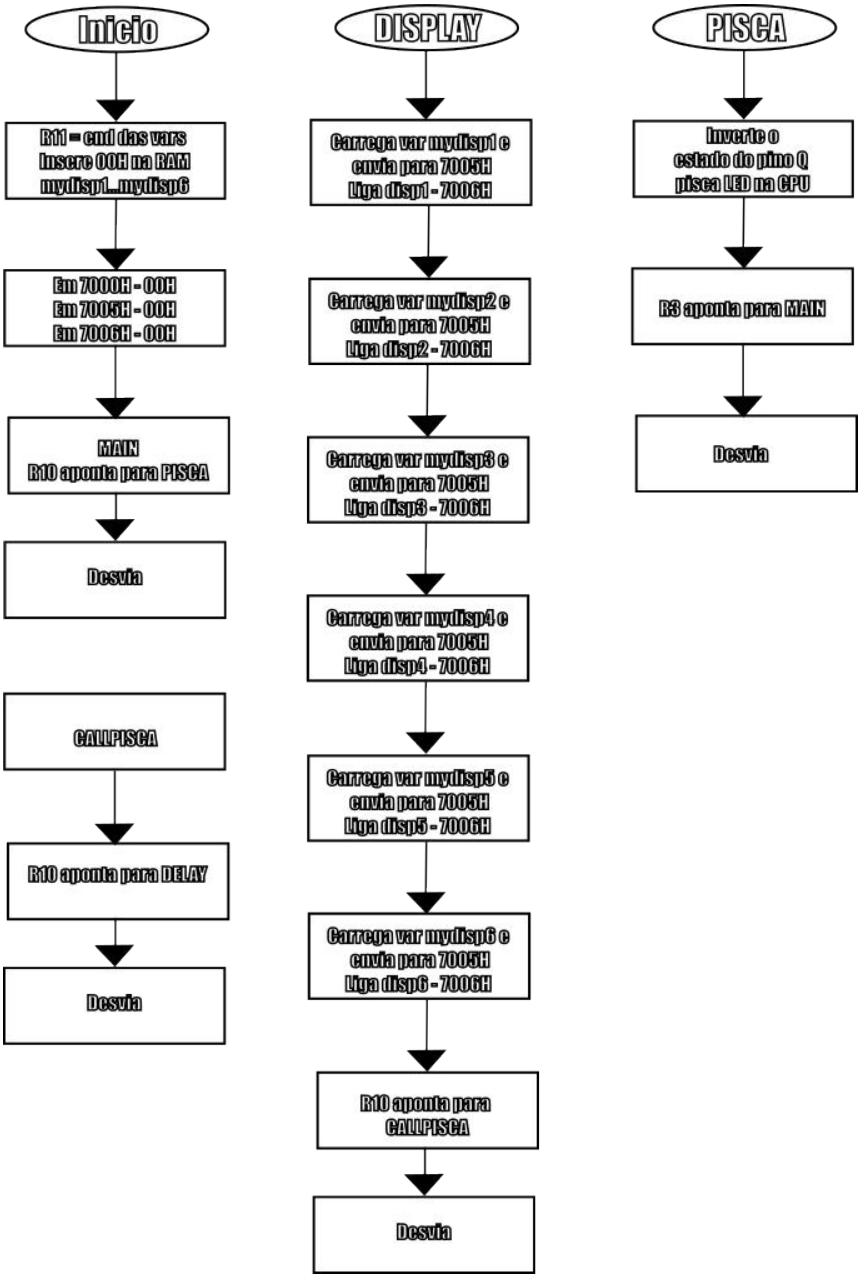


Figura 6 - Fluxograma programa de testes teclados

Programa

Na **figura 6** temos um fluxograma de um programa para os testes do teclado.

O programa inicia os displays apresentando nos mesmos o valor "8000H" nos 4 primeiros displays (endereço base da RAM) e ACH nos dois últimos (byte hipotético localizado no endereço). O fluxograma deste programa está presente na figura 6 e

sua listagem em “Listagem 1”.

Listagem 1 – programa teste para os displays

```
                        ;*****************************************************************************
;               Prog disp_test.asm - Teste dos displays da placa COSMA by Arne
;               Desenvolvido por: Eng. Márcio José Soares
;                       versão 1.0: 31/10/2025
;                       Compilador: a18
;       Plataforma: placa CPU COSMAC + placa extensora COSMAC by Arne (home built)
;*****************************************************************************
;                       Define CPU
CPU                      1802
;*****************************************************************************
;                       Define registradores
R0                      EQU     0
R1                      EQU     1
R2                      EQU     2
R3                      EQU     3
R4                      EQU     4
R5                      EQU     5
R6                      EQU     6
R7                      EQU     7
R8                      EQU     8
R9                      EQU     9
R10                     EQU     10
R11                     EQU     11
R12                     EQU     12
R13                     EQU     13
R14                     EQU     14
R15                     EQU     15
;*****************************************************************************
;                       Define endereços das PORTAS de I/O
SLED                            EQU 7000H
DISPSEG                 EQU 7005H
DISPKS                          EQU 7006H
;*****************************************************************************
;                       Define posições na RAM
TAMDATA                 EQU 31H ; RAM para dados internos 49 bytes máximos!
TAMBUFFER               EQU 80H ; buffer de 128 bytes
RAMTOP      EQU 0FFFFH ; tamanho da memória 6116
RAMPRG      EQU 8000H ; endereço inicial do programa do          usuário
POSDATA                 EQU ((RAMTOP) - (TAMDATA+TAMBUFFER+1));
```

```
;*******************************************************************************
;                       Endereços na RAM
mydisp1                 EQU POSDATA ; displays
mydisp2                 EQU POSDATA+1
mydisp3                 EQU POSDATA+2
mydisp4                 EQU POSDATA+3
mydisp5                 EQU POSDATA+4
mydisp6                 EQU POSDATA+5
myCatodos               EQU POSDATA+6 ; controle dos catodos
;*******************************************************************************
;                       Endereços na ROM
STACK_H                 EQU 80H ; endereço
STACK_L                 EQU 64H
DISPLAY_H               EQU 00H
DISPLAY_L               EQU 2FH
;*******************************************************************************
;                       Endereço onde tudo começa!!!
ORG                             #0000
BR                      START
;*******************************************************************************
;               sub-rotina de inicio
START:          OAD R11,mydisp1 ; em R11, endereço da RAM para display 1
SEX                     R11
LDI                     08H ; carrega D com 8
STR                     R11 ; guarda
LOAD            R11,mydisp2 ; em R11, endereço da RAM para display 2
SEX                     R11
LDI                     00H ; carrega D com 0
STR                     R11 ; guarda
LOAD            R11,mydisp3 ; em R11, endereço da RAM para display 3
SEX                     R11
LDI                     00H ; carrega D com 0
STR                     R11 ; guarda
LOAD            R11,mydisp4 ; em R11, endereço da RAM para display 4
SEX                     R11
LDI                     00H ; carrega D com 0
STR                     R11 ; guarda
LOAD            R11,mydisp5 ; em R11, endereço da RAM para display 5
SEX                     R11
LDI                     0AH ; carrega D com A
STR                     R11 ; guarda
LOAD            R11,mydisp6 ; em R11, endereço da RAM para display 6
```

```
SEX                 R11
LDI                 0CH ; carrega D com C
STR                 R11 ; guarda
LDI                 00H ; em D 00H
PHI                 R4 ; carrega no MSB de R4 conteúdo de D
LDI                 01H
PLO                 R4 ; no LSB de R4 01H - disp1 ligado
LOAD      R7,DISPKS ; R7 aponta p/ endereço 7006H - saída catodos dos displays
SEX                 R7
GLO                 R4 ; recupera LSB em D
STR                 R7 ; envia p/ catodos dos segmentos dos displays
LOAD      R6,DISPSEG ; R6 aponta p/ end. 7005H - saída p/ segmentos dos displays
SEX                 R6
GHI                 R4 ; recupera MSB de R4 em D
STR                 R6 ; envia p/ segmentos dos displays
LOAD       R5,SLED ; R5 aponta p/ endereço 7000H
SEX                 R5
GLO                 R4 ; recupera LSB em D
STR                 R5 ; envia p/ LEDs
;*******************************************************************************
;                   sub-rotina principal
MAIN:     LOAD R10,DISPLAY ; aponta para subrotina display
SEX                 R10 ; x = 10
SEP                 R10 ; desvia
CALLPISCA:  LOAD R10,PISCA ; aponta para subrotina teclado
SEX                 R10 ; x = 10
SEP                 R10 ; desvia
BR                  MAIN ; volta - laço eterno!
IDL                 ; break se chegar aqui
;*******************************************************************************
;                   sub-rotina DISPLAY - mostra valores nos displays
DISPLAY:            LOAD R11,mydisp1 ; pega endereço
SEX                 R11 ; x=11
LDN                 R11 ; lê conteúdo do endereço em D
LOAD      R8,_7SEGDATA ; R8 contem agora o endereço da tabela
GLO                 R8 ; recupera conteúdo
ADD                 ; soma
PLO                 R8 ; guarda na parte baixa de R8
SEX                 R8 ; indice agora em R8
LDXA                        ; em D o byte presente na tabela
PHI                 R4 ; coloca D na parte alta de R4
LOAD      R6,DISPSEG ; R6 aponta p/ end. 7005H - saída p/ segmentos dos displays
```

```
SEX             R6 ; x=6
GHI             R4 ; recupera byte lido em _7SEGDATA
STR             R6 ; envia p/ os segmentos
LOAD    R5,SLED ; R5 aponta p/ endereço 7000H
PLO             R11
SEX             R5 ; x=5
STR             R5 ; envia p/ os LEDs também
LOAD    R7,DISPKS ; R7 aponta p/ endereço 7006H - saída catodos dos displays
SEX             R7 ; x=7
GLO             R4 ; recupera byte inserido no LSB
STR             R7 ; liga disp
LDI             00H ; prepara para temporizar!
PHI             R9
LDI             96H ; carga imediata de 100 em D
PLO             R9 ; coloca D no LSB de R9
LOOP1:  DEC R9 ; decrementa R9
GLO             R9 ; coloca em D a parte MSB de R9
BNZ             LOOP1 ; desvia se não for zero
LOAD    R11,mydisp2 ; pega endereço
SEX             R11 ; x=11
LDN             R11 ; lê conteúdo do endereço em D
LOAD    R8,_7SEGDATA ; R8 contem agora o endereço da tabela
GLO             R8 ; recupera conteúdo
ADD             ; soma
PLO             R8 ; guarda na parte baixa de R8
SEX             R8 ; indice agora em R8
LDXA            ; em D o byte presente na tabela
PHI             R4 ; coloca D na parte alta de R4
LOAD    R6,DISPSEG ; R6 aponta p/ end. 7005H - saída p/ segmentos dos displays
SEX             R6 ; x=6
GHI             R4 ; recupera byte lido em _7SEGDATA
STR             R6 ; envia p/ os segmentos
LOAD    R5,SLED ; R5 aponta p/ endereço 7000H
SEX             R5 ; x=5
STR             R5 ; envia p/ os LEDs também
LOAD    R7,DISPKS ; R7 aponta p/ endereço 7006H - saída catodos dos displays
LDI             02H ; manda para segundo display
SEX             R7 ; x=7
STR             R7 ; liga disp
LDI             00H ; prepara para temporizar!
PHI             R9
LDI             96H ; carga imediata de 100 em D
```

```
PLO             R9 ; coloca D no LSB de R9

LOOP2:                  DEC R9 ; decrementa R9
GLO             R9 ; colaca em D a parte MSB de R9
BNZ             LOOP2 ; desvia se não for zero
LOAD                    R11,mydisp3 ; pega endereço
SEX             R11 ; x=11
LDN             R11 ; lê conteúdo do endereço em D
LOAD            R8,_7SEGDATA ; R8 contem agora o endereço da tabela
GLO             R8 ; recupera conteúdo
ADD             ; soma
PLO             R8 ; guarda na parte baixa de R8
SEX             R8 ; indice agora em R8
LDXA                    ; em D o byte presente na tabela
PHI             R4 ; coloca D na parte alta de R4
LOAD    R6,DISPSEG ; R6 aponta p/ end. 7005H - saída p/ segmentos dos displays
SEX             R6 ; x=6
GHI             R4 ; recupera byte lido em _7SEGDATA
STR             R6 ; envia p/ os segmentos
LOAD                    R5,SLED ; R5 aponta p/ endereço 7000H
SEX             R5 ; x=5
STR             R5 ; envia p/ os LEDs também
LOAD    R7,DISPKS ; R7 aponta p/ endereço 7006H - saída catodos dos displays
LDI             04H ; manda para terceiro display
SEX             R7 ; x=7
STR             R7 ; liga disp
LDI             00H ; prepara para temporizar!
PHI             R9
LDI             96H ; carga imediata de 100 em D
PLO             R9 ; coloca D no LSB de R9
LOOP3:                  DEC R9 ; decrementa R9
GLO             R9 ; colaca em D a parte MSB de R9
BNZ             LOOP3 ; desvia se não for zero
LOAD                    R11,mydisp4 ; pega endereço
SEX             R11 ; x=11
LDN             R11 ; lê conteúdo do endereço em D
LOAD                    R8,_7SEGDATA ; R8 contem agora o endereço da tabela
GLO             R8 ; recupera conteúdo
ADD             ; soma
PLO             R8 ; guarda na parte baixa de R8
SEX             R8 ; indice agora em R8
LDXA                    ; em D o byte presente na tabela
```

```
PHI             R4 ; coloca D na parte alta de R4
LOAD    R6,DISPSEG ; R6 aponta p/ end. 7005H - saída p/ segmentos dos displays
SEX             R6 ; x=6
GHI             R4 ; recupera byte lido em _7SEGDATA
STR             R6 ; envia p/ os segmentos
LOAD                    R5,SLED ; R5 aponta p/ endereço 7000H
SEX             R5 ; x=5
STR             R5 ; envia p/ os LEDs também
LOAD    R7,DISPKS ; R7 aponta p/ end. 7006H - saída catodos dos displays
LDI             08H ; manda para quarto display
SEX             R7 ; x=7
STR             R7 ; liga disp
LDI             00H ; prepara para temporizar!
PHI             R9
LDI             96H ; carga imediata de 100 em D
PLO             R9 ; coloca D no LSB de R9
LOOP4:                  DEC R9 ; decrementa R9
GLO             R9 ; colaca em D a parte MSB de R9
BNZ             LOOP4 ; desvia se não for zero
LOAD                    R11,mydisp5 ; pega endereço
SEX             R11 ; x=11
LDN             R11 ; lê conteúdo do endereço em D
LOAD                    R8,_7SEGDATA ; R8 contem agora o endereço da tabela
GLO             R8 ; recupera conteúdo
ADD             ; soma
PLO             R8 ; guarda na parte baixa de R8
SEX             R8 ; indice agora em R8
LDXA                    ; em D o byte presente na tabela
PHI     R4 ; coloca D na parte alta de R4
LOAD    R6,DISPSEG ; R6 aponta p/ end. 7005H - saída p/ segmentos dos displays
SEX             R6 ; x=6
GHI             R4 ; recupera byte lido em _7SEGDATA
STR             R6 ; envia p/ os segmentos
LOAD                    R5,SLED ; R5 aponta p/ endereço 7000H
SEX             R5 ; x=5
STR             R5 ; envia p/ os LEDs também
LOAD    R7,DISPKS ; R7 aponta p/ end. 7006H - saída catodos dos displays
LDI             10H ; manda para quinto display
SEX             R7 ; x=7
STR             R7 ; liga disp
LDI             00H ; prepara para temporizar!
PHI             R9
```

```
LDI             96H ; carga imediata de 100 em D
PLO             R9 ; coloca D no LSB de R9
LOOP5:                  DEC R9 ; decrementa R9
GLO             R9 ; coloca em D a parte MSB de R9
BNZ             LOOP5 ; desvia se não for zero
LOAD        R11,mydisp6 ; pega endereço
SEX             R11 ; x=11
LDN             R11 ; lê conteúdo do endereço em D
LOAD        R8,_7SEGDATA ; R8 contem agora o endereço da tabela
GLO             R8 ; recupera conteúdo
ADD             ; soma
PLO             R8 ; guarda na parte baixa de R8
SEX             R8 ; indice agora em R8
LDXA                    ; em D o byte presente na tabela
PHI             R4 ; coloca D na parte alta de R4
LOAD        R6,DISPSEG ; R6 aponta p/ end. 7005H - saída p/ segmentos dos displays
SEX             R6 ; x=6
GHI             R4 ; recupera byte lido em _7SEGDATA
STR             R6 ; envia p/ os segmentos
LOAD                    R5,SLED ; R5 aponta p/ endereço 7000H
SEX             R5 ; x=5
STR             R5 ; envia p/ os LEDs também
LOAD        R7,DISPKS ; R7 aponta p/ end. 7006H - saída catodos dos displays
LDI             20H ; manda para sexto display
SEX             R7 ; x=7
STR             R7 ; liga disp
LDI             00H ; prepara para temporizar!
PHI             R9
LDI             96H ; carga imediata de 100 em D
PLO             R9 ; coloca D no LSB de R9
LOOP6:                  DEC R9 ; decrementa R9
GLO             R9 ; colaca em D a parte MSB de R9
BNZ             LOOP6 ; desvia se não for zero
EXIT_D:         LDI 01H
PLO             R4
LOAD                    R3,CALLPISCA
SEX             R3
SEP             R3
IDL             ; break se chegar aqui
;*******************************************************************************
;           sub-rotina TECLADO - faz o tratamento do teclado - a implementar!
TECLADO:        NOP
```

```
    LSQ                         ; se Q = 1 pula as duas próximas
instruções
    SEQ                 ; liga Q
    SKP                 ; pula a próxima instrução
    REQ                         ; desliga Q
    LOAD                        R3,MAIN
    SEX                 R3
    SEP                 R3
    IDL                 ; break se chegar aqui
    ;***************************************************************************
    ;                   Constantes inseridas na ROM
    ;
    _7SEGDATA:          DB 03FH,06H,5BH,4FH,66H,6DH,7DH,07FH
    DB                  7FH,67H,77H,7CH,39H,5EH,78H,71H
    DB                   00H
    ;***************************************************************************
    ;                   Fim
    END
```

Teste e uso

Não é possível testar a placa de acessórios e consequentemente qualquer parte sua sem a presença da placa ou parte CPU.

Após “montar” (compilar) o programa (os links das ferramentas necessárias estão no primeiro artigo desta série – edição nr 31), basta gravar uma EEPROM e instalá-la na placa e alimentar o circuito para confirmar o funcionamento do mesmo. Qualquer discrepância do exemplificado no fluxograma exigirá uma nova verificação nas conexões e na montagem (compilação) do programa.

Conclusão

Sabemos que esta não é uma montagem simples se comparada com outras montagens. Por isso a mesma foi dividida em partes para que o leitor tenha tempo de conseguir os componentes e também possa ir conhecendo aos poucos tanto a CPU CDP1802 como o hardware do nosso COSMAC-ELF. Estamos nos aproximando do final da nossa saga. Na próxima edição

Contatos:

- Página Web – http://www.arnerobotics.com.br

- Instagram - https://www.instagram.com/arnesake/

- YouTube - https://www.youtube.com/c/arnesake

- Thingiverse - https://www.thingiverse.com/arnesake/designs

traremos o programa monitor que permitirá fazer todo o gerenciamento da placa. Boa montagem e até a próxima parte desta saga!

Lista de materiais

Semicondutores

U5, U6 - 74HC574 - Octal D Lath com saída 3-state
Q2, Q3 – BC327 – transistor PNP de uso geral
Q4 à Q9 – BC338 – transistor NPN de uso geral
Q10 – BC338 - transistor NPN de uso geral

Resistores (1/8W – 5%)
R9 – 1k (marrom, preto, vermelho)
R10, R11 – 10k (marrom, preto, laranja)
R12..R19 – 470R (amarelo, violeta, marrom)
R20 - 1k (marrom, preto, vermelho)
R21, R22 – 10k (marrom, preto, laranja)
R23..R28 – 1k (marrom, preto, vermelho)
R29 – 1k (marrom, preto, vermelho)
R30 – 4k7 (amarelo, violeta, vermelho)

Capacitores
C5, C6 – 100nF/60V (cerâmicos)

Diversos
6 - displays catodo comum (HS-5101AS ou similar)
2 - soquetes para CIs com 20 pinos
2 - soquetes para CIs com 32 pinos (uso com displays - veja texto)
1 - placa padrão 300 x 150 mm (montagem anterior ou outra)
1 – buzzer 5VDC com oscilador interno
Solda, fio wire wripping, etc

Referências bibliográficas

[1] BRAGA, Newton C. Curso de Eletrônica Digital – Display Experimental (ART2558)

. Instituto Newton C. Braga. São Paulo, [s.d]. Disponível em: https://www.newtoncbraga.com.br/projetos-educacionais/10985-display-experimental-art2558.html

[2] BRAGA, Newton C. Curso de Eltrônica Digital – Multiplexadores, Demultiplexadores, Decodificadores e Displays (CUR5012). Instituto Newton C. Braga. São Paulo, [s.d]. Disponível em: https://www.newtoncbraga.com.br/eletronica-digital/16357-curso-de-eletronica-eletronica-digital-multiplexadores-demultiplexadores-decodificadores-e-displays-cur5012.html

[3] SOARES, Márcio José. Controle de Displays de LEDs Utilizando o PIC16F628A (MIC548). Revista Mecatrônica Jovem (ou Instituto Newton C. Braga), São Paulo, [s.d.]. Disponível em: www.newtoncbraga.com.br. Acesso em: 15 jan. 2026.

[4] SOARES, Márcio José. Display de 7 segmentos com Raspberry Pi Pico. Arne Robotics, São Paulo, [s.d.]. Disponível em: http://www.arnerobotics.com.br/eletronica/raspBPi_pico_disp7seg.htm. Acesso em: 15 jan. 2026.

[5] SOARES, Márcio José. Microcontroladores PIC - Displays de sete segmentos. Arne Robotics, São Paulo, [s.d.]. Disponível em: http://www.arnerobotics.com.br/eletronica/Microcontrolador_PIC_pratica_4.htm. Acesso em: 15 jan. 2026.

[6] TEXAS INSTRUMENTS. SN54HC574, SN74HC574 octal edge-triggered D-type flip-flops with 3-state outputs. Dallas, Dez. 1982. Revisado em: Mai. 2022. Disponível em: https://www.ti.com/lit/ds/symlink/sn74hc574.pdf. Acesso em: 15 jan. 2026.

[7] TEXAS INSTRUMENTS. SNx4HC541 octal buffers and line drivers with 3-state outputs. Dallas, [s.d.]. Disponível em: https://www.ti.com/lit/ds/symlink/sn74hc541.pdf. Acesso em: 15 jan. 2026.

[8] XLITX. LED 7-Segment Display, China, [s.d.]. Disponível em: https://www.xlitx.com/datasheet/5101AS.pdf

Use MODBUS TCP com ESP32-C3

Pedro Bertoletti

Introdução

A capacidade de dispositivos de diferentes funcionalidades, marcas e modelos trocarem informações de forma confiável e padronizada é um dos principais fatores que torna a automação industrial confiável e escalável. Desde o final da década de 1970, o nome "MODBUS" é muito comum quando se fala dessa troca de informações entre dispositivos industriais.

MODBUS atingiu um patamar tão grande de aceitação e popularidade no meio industrial que, ao se desenvolver um produto com suporte à esta comunicação (em qualquer uma de suas variantes, como o MODBUS RTU e o MODBUS TCP), já se dá como garantida

sua capacidade de integração com CLPs e demais controladores em ambiente industrial.

Neste artigo, será abordado o MODBUS TCP, variante do MODBUS que funciona sob o protocolo TCP/IP seja via Internet ou Wi-Fi. Ainda, é mostrado como fazer um servidor MODBUS TCP com ESP32-C3, o qual você pode usar para validar a comunicação MODBUS TCP na sua própria rede doméstica e aí mesmo na sua bancada.

O que é MODBUS TCP?

Para entender o MODBUS TCP de forma direta, imagine-o como uma linguagem universal que permite que dispositivos industriais conversem entre si em rede (nesse caso, Ethernet ou Wi-Fi). Nesta linguagem, um dispositivo somente responde quando requisitado: ou seja, não há "tráfego espontâneo", apenas sob demanda. No MODBUS TCP existem dois tipos de elementos: os servidores (dispositivos que ficam no aguardo de requisições e as respondem) e clientes (dispositivos que geram a requisição e aguardam pela resposta).

Criado originalmente para comunicações seriais, o MODBUS evoluiu para a versão TCP para aproveitar a velocidade e a infraestrutura das redes Ethernet e Wi-Fi, além das vantagens do próprio TCP/IP como por exemplo: garantia de recepção dos pacotes, retransmissão automática de pacotes que falharam e verificação de integridade dos dados.

Observe na **figura 1** o fluxo de dados entre um cliente e um servidor MODBUS TCP.

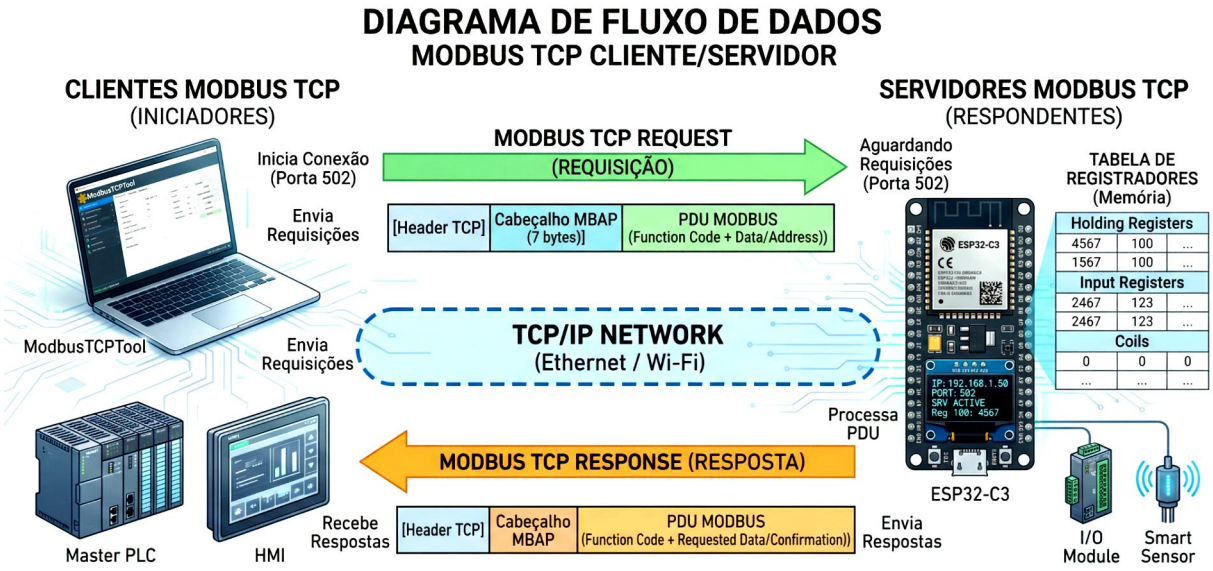


Figura 1 - Diagrama de fluxo de dados entre um cliente e um servidor MODBUS TCP

Em termos simples, o MODBUS TCP é a mensagem original do MODBUS "embrulhada" em um envelope de rede moderno. A grande diferença aqui é que, enquanto as versões seriais (como o RTU) precisam de endereços físicos de escravos e verificar a integridade de cada mensagem, o MODBUS TCP utiliza o cabeçalho (chamado MBAP, Modbus Application Protocol). Esse cabeçalho de 7 bytes não considera endereço do dispositivo-alvo nem verificação de integridade, uma vez que o TCP/IP já garante que a mensagem chegue inteira e ao destino correto através do endereço IP.

A organização dos dados dentro de uma comunicação MODBUS TCP é feita através de quatro tabelas de memória fundamentais:

• *Coils (Bobinas): valores de 1 bit, leitura e escrita (ex: acionar um relé).*

• *Discrete Inputs: valores de 1 bit, apenas leitura (ex: sensor de fim de curso).*

• *Holding Registers: valores de 16 bits, leitura e escrita (ex: setpoint de temperatura).*

• *Input Registers: valores de 16 bits, apenas leitura (ex: leitura de um sensor analógico).*

Diferenças e semelhanças entre MODBUS TCP e RTU com RS485

Atualmente, existem diversas variantes do MODBUS, dentre as quais duas são as principais: MODBUS RTU e MODBUS TCP. O primeiro, utiliza comunicação serial para trafegar dados (RS485, RS232, etc.), enquanto o segundo faz uso de interfaces de comunicação que comportam o TCP/IP (Ehernet e Wi-Fi).

Como semelhanças, ambas variantes de MODBUS compartilham o mesmo PDU (parte em verde na **figura 2**). Ou seja, os mesmos códigos de função (códigos de 1 byte que indicam a ação da mensagem, que pode ser leitura de um holding register, escrita de um coil, etc.) e a localização do payload são idênticos. Isso significa que, independentemente do meio físico ser um par de fios trançados (RS485) ou um cabo de rede (Ethernet), os PDUs do MODBUS TCP e RTU são idênticos. Ainda, seja no MODBUS TCP ou RTU, toda informação é organizada em um ou mais registros endereçados, onde cada registro tem sempre 16 bits (2 bytes) de tamanho. Logo, toda

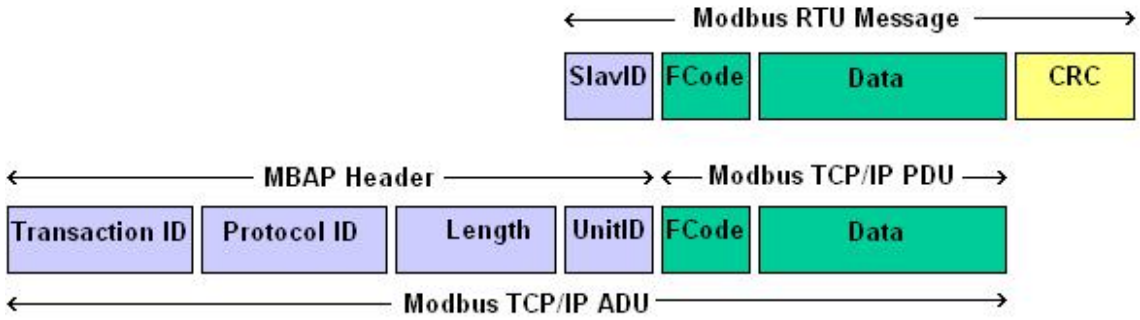


Figura 2 - Datagramas de mensagens MODBUS TCP e RTU *(figura extraída de: https://www.simplymodbus.ca/TCP.htm)*

comunicação MODBUS envolve que, nos dois lados (mestre escravo ou cliente e servidor), seja conhecido quais registros devam ser lidos e escritos e seus respectivos endereços.

A lógica de endereçamento de memória (começando geralmente em 0 ou 1) permanece constante, facilitando a vida do desenvolvedor que já está acostumado com o ambiente industrial.

As diferenças, porém, começam na forma como esses dados são transportados e identificados. No MODBUS RTU, a comunicação é baseada em tempo (timing) e exige um intervalo de silêncio para identificar o fim de uma mensagem e um tempo de tolerância para receber a resposta. No MODBUS TCP, essa responsabilidade é do protocolo TCP, que usa o cabeçalho MBAP (**figura 2**) para definir o tamanho da mensagem, eliminando a dependência de temporização rigorosa.

No MODBUS TCP não é explicitado no datagrama o endereço ao qual a mensagem se destina nem o verificador de integridade (CRC) da mensagem. No lugar destes, é usado o endereço IP como endereço de destino e a própria verificação de integridade e retransmissão de pacotes do TCP/IP ao invés do CRC.

Outra diferença importante é na sincronicidade de comunicação. No MODBUS RTU quem envia a requisição deve sempre esperar pela resposta (ou seja, comunicação síncrona), enquanto no MODBUS TCP existe o Transaction ID, um identificador único por requisição. Dessa forma, no MODBUS TCP, quem faz a requisição não precisa esperar: ele envia e, quando vier uma resposta, ele verifica o Transaction ID para saber a qual requisição aquela resposta pertence. Isso abre possibilidade de maior velocidade, uma vez que a comunicação se torna assíncrona.

Ainda, pelo campo Protocol ID, é possível criar segmentações de grupos de dispositivos no MODBUS TCP. Um grupo de dispositivos responde à mensagens apenas com Protocol ID 0x0000, outro grupo responde apenas ao Protocol ID 0x0001, e assim por diante. Isso permite maior controle da comunicação.

Funções MODBUS

Para que o cliente e o servidor se entendam no MODBUS TCP, eles utilizam Códigos de Função. Esses códigos determinam qual ação será realizada sobre determinados registros. Estes códigos entrar como FCode, conforme mostrado nos datagramas da **figura 2.**

As funções MODBUS mais utilizadas são:

• Função 01 (Read Coils): utilizada para ler o estado de uma ou mais saídas digitais.

• Função 02 (Read Discrete Inputs): utilizada para ler entradas digitais que não podem ser alteradas pelo cliente (apenas leitura).

• Função 03 (Read Holding Registers): a função mais comum do MODBUS, usada para ler valores de 16 bits que podem representar variáveis de processo ou configurações.

• Função 04 (Read Input Registers): utilizada para ler dados analógicos de sensores (apenas leitura).

• Função 05 (Write Single Coil): envia um comando para ligar ou desligar uma única bobina (saída).

• Função 06 (Write Single Register): escreve um valor de 16 bits em um único registrador de retenção.

• Função 15 (Write Multiple Coils): permite acionar vários relés ou saídas de uma só vez.

• Função 16 (Write Multiple Registers): permite escrever um bloco de dados (ex: uma configuração complexa) de uma única vez / numa única mensagem, otimizando o tráfego de rede.

Por que o MODBUS TCP é muito usado na indústria?

A dominância do MODBUS TCP se deve à sua versatilidade. Como ele roda sobre Ethernet ou Wi-Fi, não há necessidade de passar cabos especiais e caros de comunicação serial por toda a fábrica; basta conectar o dispositivo ao switch mais próximo (ou, se for Wi-Fi, ao roteador / AP mais próximo) e usar a própria infraestrutura de rede existente. Como as redes Ethernet e Wi-Fi hoje são muito populares

(incluindo em chão de fábrica) e baratas, isso reduz muito o custo de instalação e tempo de manutenção.

Além disso, o MODBUS TCP resolve o problema do gargalo de dados. No MODBUS RTU com RS485, por exemplo, o Mestre precisa fazer um pooling geral: ele precisa perguntar para cada Escravo, um por um, "você tem dados?" e aguardar a resposta de cada um em seguida. Isso gera um atraso (latência) considerável. No Modbus TCP, a rede é muito mais rápida e permite que vários sistemas (como um supervisório e um computador) acessem o mesmo sensor ao mesmo tempo, algo impossível na versão serial. A facilidade de diagnóstico também é um fator: qualquer técnico com um notebook e conhecimentos básicos de rede pode usar ferramentas gratuitas para verificar se os dados estão chegando corretamente, sem precisar de equipamentos especializados.

Principais casos de Uso Comum do MODBUS TCP

Um dos maiores casos de uso é a medição de grandezas elétricas. Multimedidores de grandezas elétricas expõem centenas de parâmetros (tensão, corrente, harmônicas, fator de potência, histórico de consumo de energia, etc.) via Modbus TCP. Um exemplo é o PC3220, da Siemens, visto na **figura 3**. Um sistema central (supervisório, por exemplo) lê esses dados periodicamente para garantir que a rede elétrica (e o consumo dos equipamentos) está dentro dos limites operacionais aceitáveis.

Outro cenário comum é a automação de sistemas de refrigeração (HVAC). Em grandes prédios, os chillers e fan-coils utilizam Modbus TCP para reportar temperaturas e receber ordens de ligar/desligar de um sistema central de automação predial. Isso garante eficiência energética, gerenciamento do sistema e conforto térmico de forma automatizada.

Figura 3 - Multimedidor de grandezas elétricas PAC 3220, do fabricante Siemens.

ESP32-C3 com OLED 0.42" como MODBUS TCP Server

O uso do ESP32-C3 como um servidor MODBUS TCP é perfeito para você, aí mesmo na sua casa, testar o MODBUS TCP. Afinal, com um devkit pequeno, uma rede Wi-Fi e um computador na mesma rede você pode ter uma comunicação MODBUS TCP estabelecida em poucos minutos.

Como devkit para isso, será abordado o ESP32-C3 com OLED 0.42”, conforme motrado na **figura 4**. Ele pode ser encontrado facilmente (e por baixos preços) em diversas lojas de componentes eletrônicos, ou até mesmo no Mercado Livre (link de referência: https://meli.la/1BGkYb8

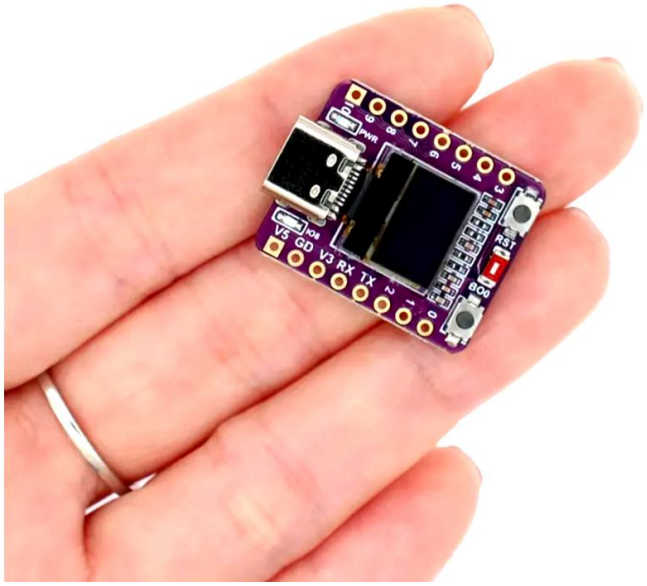


Figura 4 - ESP32-C3 com OLED 0.42"

Exemplo Prático: Fluxo de Operação entre ESP32-C3 e ModbusTCPTool

O exemplo prático deste artigo consiste em transformar o ESP32-C3 em um MODBUS TCP server, capaz de ler do ESP32-C3 os dados de temperatura do chip, RSSI da comunicação Wi-Fi e um contador.

Abaixo, segue uma explicação do que o projeto faz:

1. O projeto utiliza o WI-Fi do próprio ESP32-C3 para trafegar as mensagens MODBUS TCP. Para facilitar a localização do dispositivo na rede, o projeto utiliza IP estático (aqui, referido como 192.168.68.210), garantindo que o cliente MODBUS TCP (um programa no computador, abordado mais a frente) sempre saiba exatamente em qual IP o ESP32-C3 está.

2. O display OLED do devkit é configurado para mostrar o último octeto do IP (neste caso, para o IP 192.168.68.210, corresponde ao número 210) logo após a conexão ser estabelecida. Dessa forma, o OLED funciona como uma “etiqueta digital” de identificação.

3. O programa organiza a exposição de dados através de três Holding Registers (Hregs) principais:

- Registro 100 (REG_TEMP): armazena a temperatura interna do chip ESP32-C3;

- Registro 101 (REG_COUNTER): contém um contador iniciado em zero e incrementado uma vez por segundo;

- Registro 102 (REG_RSSI): contém a potência de recepção do Wi-Fi (RSSI) no ESP32-C3.

4. No loop principal, o programa processa as requisições recebidas (via Wi-Fi, na porta 502), enviando os dados (leituras mais atuais de temperatura do chip, RSSI da comunicação Wi-Fi e do contador) sempre que solicitado. Ainda, a cada segundo, o contador e as leituras de RSSI e temperatura são atualizados nos seus respectivos registros.

No computador, instale a ferramenta ModbusTCPTool (disponível na loja de aplicativos oficial do Windows, a Microsoft Store) e configure a leitura dos registros (100, 101 102) no IP do devkit (nesse caso, 192.168.68.210) conforme destacado em vermelho na figura 5. Ainda na **figura 5**, na área destacada em amarelo, estão as leituras dos dados (temperatura do chip, RSSI da comunicação Wi-Fi e do contador).

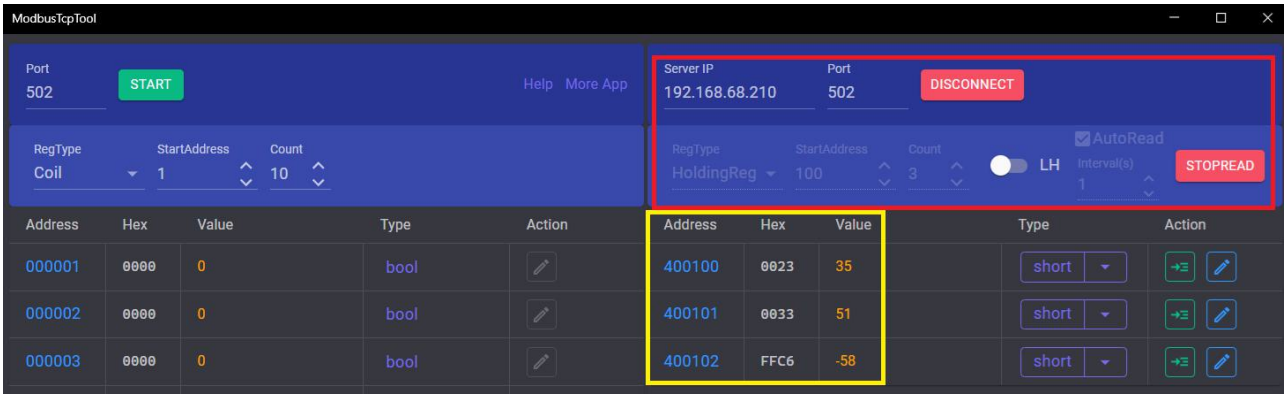


Figura 5 - Ferramenta ModbusTCPTool em operação, lendo o ESP32-C3

Código-fonte

O código-fonte do projeto pode ser lido abaixo.

Antes de utilizar, não se esqueça de:

• Instalar a versão mais atual da biblioteca u8g2 (via gerenciador de bibliotecas da Arduino IDE);

• Substituir o SSID e senha da sua rede Wi-FI em ssid e password;

• Caso sua subnet for diferente de 192.168.68.X, ajuste-a em: local_IP e gateway.

```
#include <WiFi.h>
#include <ModbusIP_ESP8266.h>
#include <U8g2lib.h>
#include <Wire.h>
// --- CONFIGURAÇÃO DE REDE ---
const char* ssid = "Coloque_aqui_o_SSID_da_sua_rede";
const char* password = "Coloque_aqui_a_senha_da_sua_rede";
const int X_ADDR = 210;
IPAddress local_IP(192, 168, 68, X_ADDR);
IPAddress gateway(192, 168, 68, 1);
IPAddress subnet(255, 255, 255, 0);
// Objeto do display
U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, U8X8_PIN_NONE, 6, 5);
// --- MAPEAMENTO MODBUS ---
const int REG_TEMP    = 100; // Temperatura Interna
const int REG_COUNTER = 101; // Contador Sequencial
const int REG_RSSI    = 102; // Sinal Wi-Fi
ModbusIP mb;
// Variáveis de controle
unsigned long lastTick = 0;
uint16_t counter = 0;
void setup()
{
  int width = 72;
  int height = 40;
  int xOffset = 30; // = (132-w)/2
  int yOffset = 12; // = (64-h)/2
  Serial.begin(115200);
  WiFi.config(local_IP, gateway, subnet);
  WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) { delay(500); Serial.print(".");}
Serial.printf("\nESP32 [%d] Conectado! IP: %s\n", X_ADDR, WiFi.localIP().toString().c_str());
  mb.server();
  // Adiciona os 3 registros de retenção
  mb.addHreg(REG_TEMP);
  mb.addHreg(REG_COUNTER);
  mb.addHreg(REG_RSSI);
  // Escreve final do IP no display
  u8g2.begin();
  u8g2.setContrast(255); // set contrast to maximum
  u8g2.setBusClock(400000); //400kHz I2C
  u8g2.setFont(u8g2_font_ncenB10_tr);
```

```
  u8g2.clearBuffer(); // clear the internal memory
  u8g2.setCursor(xOffset+5, yOffset+40);
  u8g2.printf("IP:  %d", X_ADDR);
  u8g2.sendBuffer(); // transfer internal memory to the display  }
void loop()
{   mb.task();
  // 1. Atualiza contador a cada 1 segundo (sem travar o Modbus)
  if (millis() - lastTick >= 1000) {
    lastTick = millis();
    counter++;
    mb.Hreg(REG_COUNTER, counter); }
  // 2. Lê Temperatura Interna (Arredondado para Int)
  // O ESP32-C3 tem sensor interno acessível por temperatureRead()
  float temp = temperatureRead();
  mb.Hreg(REG_TEMP, (int)temp);
  // 3. Atualiza RSSI (Nível de sinal Wi-Fi)
  // RSSI é retornado como long negativo. Cast para int16_t funciona no Modbus
  int16_t rssi = (int16_t)WiFi.RSSI();
  mb.Hreg(REG_RSSI, rssi);
  delay(10); }
```

Conclusão

O MODBUS TCP é a referência atual em comunicação de dispositivos em meio industrial, pois une a simplicidade do MODBUS original à infraestrutura popular e barata das redes Ethernet e Wi-Fi. Sua arquitetura aberta e o uso do cabeçalho MBAP eliminam as barreiras físicas e as restrições de tempo dos barramentos seriais, permitindo uma integração vertical direta entre o chão de fábrica e sistemas de gestão em larga escala de forma confiável e sem custos de licenciamento.

O projeto prático com o ESP32-C3 demonstra como transformar um microcontrolador de baixo custo em um servidor industrial funcional, expondo dados vitais de telemetria como temperatura interna, força do sinal Wi-Fi (RSSI) e um contador através de registradores de retenção (Holding Registers). Com a identificação visual do endereço IP no display OLED, esta implementação serve de base sólida para que o leitor desenvolva seus próprios sensores inteligentes, gateways de telemetria ou sistemas de monitoramento remoto de ativos preparados para os desafios da Indústria 4.0.

Referências

Protocolo Modbus: Fundamentos e Aplicações - acessado 31/03/2026 https://embarcados.com.br/protocolo-modbus/

Modbus TCP/IP - acessado 31/03/2026 - https://www.simplymodbus.ca/TCP.htm

Manual: Power Monitoring Device 7KM PAC3120 and PAC3220 - acessado em 31/03/2026 https://cache.industry.siemens.com/dl/files/307/109767307/att_1298442/v1/MAN_L1V30425208F_RS-AA_002_en_en-US.pdf

Nova Técnica Auxilia a Geração de Energia Osmótica

Luiza Campos

A energia osmótica é uma fonte de energia sustentável capaz de gerar eletricidade a partir da diferença de salinidade entre a água do mar e a água dos rios.

Para aumentar a eficiência desse processo, pesquisadores da Escola Politécnica Federal de Lausanne (EPFL), na Suíça, desenvolveram uma técnica de lubrificação aplicada às membranas utilizadas nesses sistemas.

O método consiste em lubrificar os nanoporos presentes na membrana seletiva, que atua como um filtro, com minúsculas bolhas formadas por moléculas de lipídios. Esse processo reduz o atrito na passagem dos íons e facilita o fluxo através da membrana.

A professora Aleksandra Radenovic, pesquisadora da EPFL, responsável pela pesquisa sobre energia osmótica, explica que o estudo buscou unir duas estratégias já utilizadas na coleta desse tipo de energia, a alta porosidade das membranas de polímero e a precisão dos dispositivos nanofluídicos.

Segundo a cientista, essa combinação permite controlar melhor os nanoporos responsáveis pela passagem dos íons, aumentando a eficiência do processo de geração de energia.

Nos testes realizados em condições que simulam as concentrações naturais de sal da água do mar e da água de rios, a membrana lubrificada apresentou uma densidade de potência de aproximadamente 15 watts por metro quadrado.

Além disso, os pesquisadores acreditam que a tecnologia poderá ser adaptada para diferentes tipos de membranas no futuro, o que amplia as possibilidades de aplicação da descoberta em projetos de produção de energia sustentável.

A expectativa do grupo de pesquisa é que a técnica possa contribuir para o desenvolvimento de sistemas mais eficientes de geração de energia osmótica.

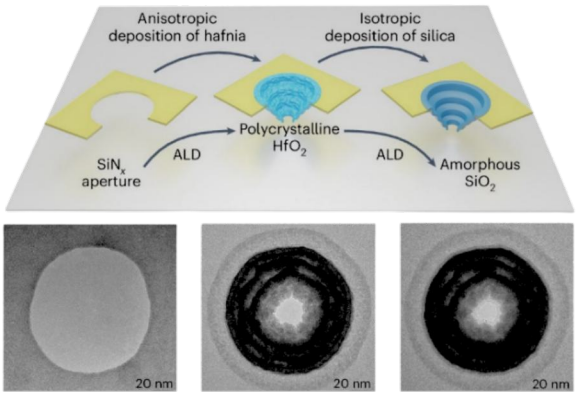


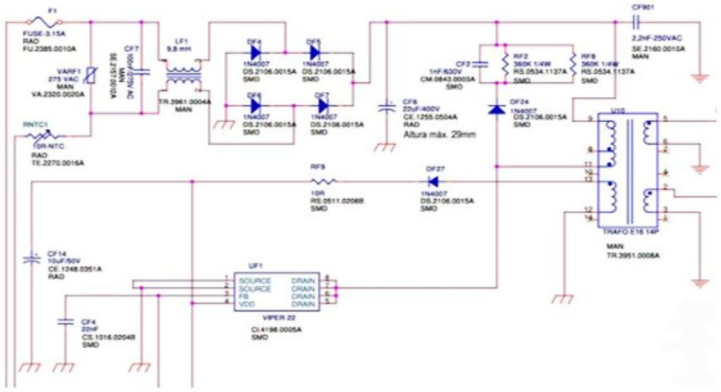
Figura 1 - Processo de lubrificação dos nanoporos [Imagem: Yunfei Teng et al. – 10.1038/s41560-026-01976-0]

Fonte: https://actu.epfl.ch/news/slippery-ions-create-a-smoother-path-to-blue-ene-2/#:~:text=For%20their%20study%2C%20the%20scientists,(hydrophilic)%20heads%20pointing%20outward.

Artigo no site: https://www.newtoncbraga.com.br/meio-ambiente-e-saude/4717-el047.html

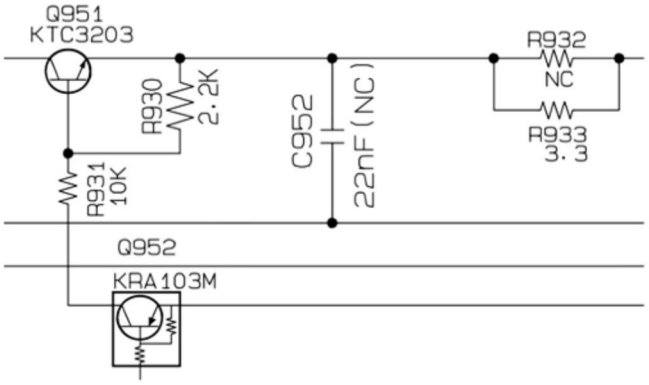
Ficha: 129	**Defeito:** Não Liga (totalmente inoperante)
Marca: Century	**Aparelho/Modelo:** Receptor Satélite Nano Box
Service	**Autor:** Alexandre J. Nário

Comecei medindo a tensão sobre o capacitor de filtragem da rede CF6 (22uF/400V) que estava presente e sem ripple. O próximo passo foi verificar a alimentação do integrado oscilador/ controlador PWM UF1 (Viper22). A tensão no terminal 4 estava zerada. A tensão que alimenta o integrado UF1 é proveniente do transformador chopper U10. A partir daí ficou fácil solucionar o problema: bastou segui a linha +B responsável por alimentar o integrado UF1 para encontrar o resistor RF9 (10 ohms) aberto e o diodo DF27 (1N4007) em curto. Fiz as substituições dos componentes defeituosos e liguei o aparelho numa lâmpada em série (100W) para evitar riscos de queimas desnecessárias de componentes caso ainda haja um eventual curto no aparelho. O receptor voltou a funcionar em perfeitas condições.



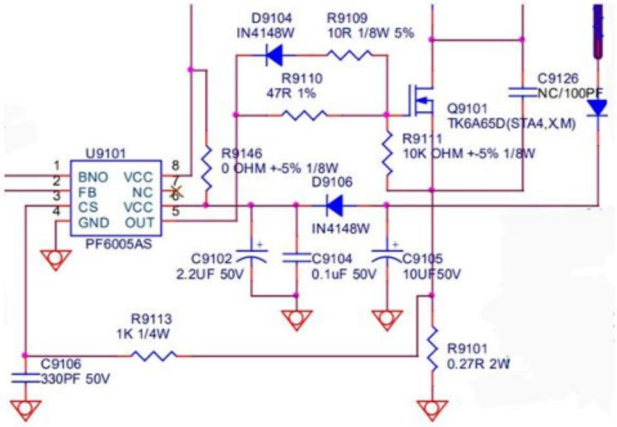
Ficha: 131	**Defeito:** Display apagado
Marca: LG	**Aparelho/Modelo:** Mini System CM4320
Service	**Autor:** Alexandre J. Nário

Quando o Mini System era ligado, todas as funções funcionavam corretamente, porém o display permanecia totalmente apagado. Desmontei o aparelho e verifiquei que as tensões FL+ e FL- que alimentam o filamento do display estavam ausentes na PCI frontal. Passei a analisar a placa da fonte de alimentação. No conector CN901, nos pinos 1 e 2, a tensão também estava ausente. Testando os componentes associados aos pinos 1 e 2, do conector CN901, responsáveis por fornecer a tensão que alimenta o display na placa do painel frontal, a partir do transformador chopper T901, encontrei, na base do transistor Q951, o resistor R931 (10KΩ) aberto. Bastou fazer a sua troca para o display voltar a acender normalmente.



Ficha: 130	**Defeito:** Não Liga (LED stand by apagado)
Marca: Philips	**Aparelho/Modelo:** TV LED 32PHG5102
Service	**Autor:** Alexandre J. Nário

A total inoperância do televisor normalmente é causada por problemas no primário da fonte chaveada. Caso não ocorra oscilação no primário, não haverá indução no transformador chopper para o secundário fornecer as tensões necessárias para o restante do aparelho. Quanto ao defeito, no primário da fonte, testei os componentes ativos que circulam correntes intensas. Nada de anormal foi encontrado. Ao verificar os componentes de alimentação, acoplamento e desacoplamento do integrado U9101, encontrei o resistor R9101 (0,27Ω/2W) aberto. Realizei a sua troca por outro equivalente e, ao ligar o aparelho, o funcionamento foi restabelecido.



Ficha: 132	**Defeito:** Não Liga (LED stand by apagado)
Marca: AOC	**Aparelho/Modelo:** TV LCD LE40D1442
Service	**Autor:** Alexandre J. Nário

O televisor totalmente inoperante e com o LED Power (stand by) apagado é um forte indício de problemas na fonte chaveada. Medindo as tensões na saída da fonte, confirmei que estavam todas ausentes. Como o primário da fonte não oscilava, minhas suspeitas recaíram sobre a existência de algum curto no circuito secundário. Com o aparelho desligado da rede elétrica, testei os componentes associados ao integrado IC9104 e encontrei o diodo D9108 (SK510C) em curto. Esse componente trata-se de um diodo ultra rápido, muito comum em circuitos de alta frequência como as fontes chaveadas. Na impossibilidade de adquirir o componente original, fiz a sua substituição por um diodo MUR1620CT que tem características semelhantes conforme informações dos respectivos datasheets. Ao ligar o televisor, o seu funcionamento foi restabelecido.