Instituto Mauá de Tecnologia

FLIP-CLOCK

Relatório de Eletrônica Digital e Analógica

Aline Miyuki Arakaki - 21.00252-5

Eliana Wen Teng So - 22.10099-7

Pedro Teodoro Bauke - 22.01668-6

São Caetano Do Sul

2024

1. Introdução

Este projeto tem como objetivo o desenvolvimento de um relógio flip-clock, combinando componentes de eletrônica digital e analógica com fabricação personalizada por impressão 3D.

A proposta visa integrar conceitos teóricos e práticos adquiridos nas disciplinas de Eletrônica Analógica e Eletrônica Digital, do curso de Engenharia de Controle e Automação do 3º ano, proporcionando aos alunos uma experiência completa, que inclui desde o planejamento dos componentes eletrônicos até o controle e automação deles.

A estrutura do relógio e a lógica de controle dos motores de passo foram projetadas para garantir a precisão na movimentação dos dígitos, representando o horário de forma contínua e clara. Além de explorar tecnologias acessíveis como o microcontrolador RP2040 e o driver ULN2003, este projeto busca promover um aprendizado prático sobre o funcionamento e controle de motores de passo, componentes essenciais para o funcionamento do flip-clock.

2. Materiais

Para a construção do projeto foram utilizados materiais próprios e do FabLab do Instituto Mauá de Tecnologia para impressão da estrutura. Os componentes são:

- 1 Microcontrolador RP2040 – WaveShare RP2040 Zero
- 3 motores de passo 28BYJ-48 + 3 Drivers ULN2003
- 3 Micro Switch KW10B
- Protoboard + jumpers
- Impressões 3D

2.1 Motores

Figura 1 – Motores de passo 28BYJ-48

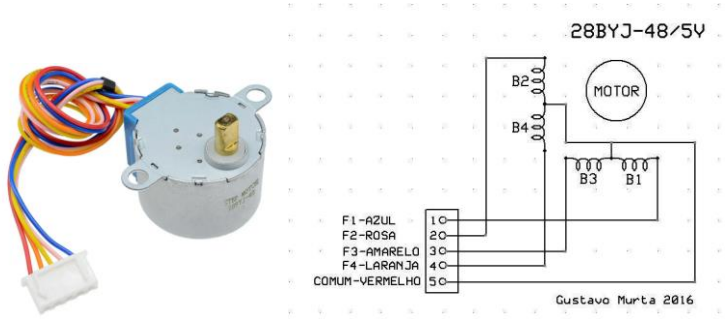
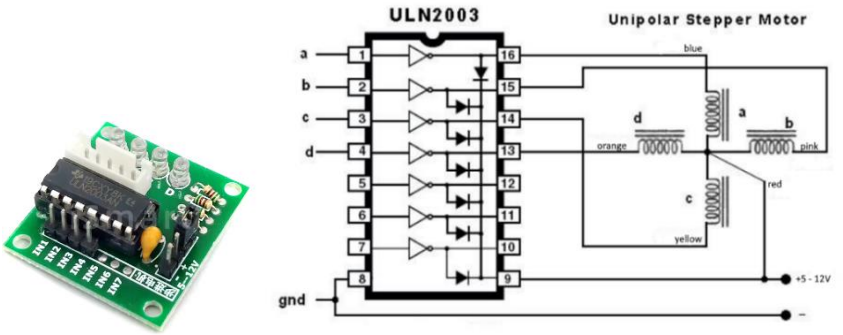


Figura 2 – Driver de motor ULN2003



Optou-se pelo uso de motores de passo, ao invés de servo motores, pois servo motores, como o motor SG90, são construídos para o controle da posição do eixo dentro de um ângulo limitado, fator que impede seu uso nessa aplicação que requer uma movimentação incremental precisa, enquanto se mantem uma rotação continua, característica essa, intrínseca aos motores de passo.

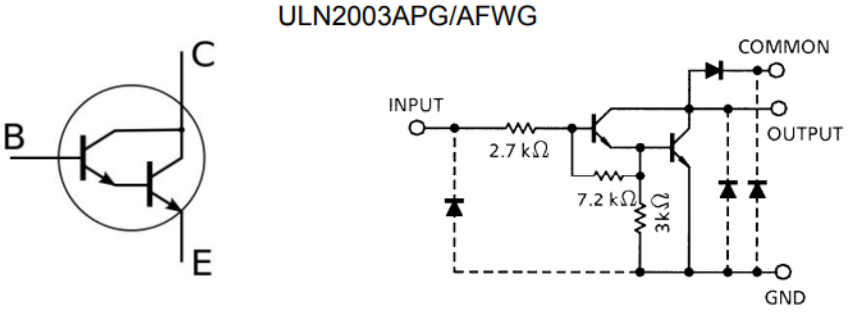
Esses motores, permitem a movimentação incremental exata necessária para alinhar cada dígito na posição correta. No contexto do flip-clock, cada motor de passo é responsável por um dígito (unidades dos minutos, dezenas dos minutos e horas), girando o tambor até a posição desejada para mostrar o horário.

O motor 28BYJ-48 é unipolar, onde na sua construção interna, divide uma bobina em duas metades, isso permite o controle do motor apenas invertendo o campo magnético sem precisar alterar a direção da corrente nas bobinas, ou seja, não é preciso de uma ponte H.

O centro comum das duas metades da bobina é conectado ao VCC, quando utilizado um driver, é possível escolher as metades das bobinas adequadas para que liguem a conexão com o GND fazendo com que a corrente flua em uma direção fixa. Assim, alternando entre as diferentes bobinas, o rotor é "puxado" pelo campo magnético gerado nelas, como é previsto pela Lei de Ampere, para posições especificadas gerando o movimento em passos. Para ter maior resolução e precisão de passos, aumenta-se o número de bobinas, no caso do motor 28BYJ-48 têm-se 2 desses conjuntos de bobinas, 4 fases como representado na Figura 1.

Para controlar os motores foi escolhido o Driver ULN2003, mostrado na Figura 2, que é um circuito integrado que possui, conectada a cada uma das suas entradas, transistores Darlington, que essencialmente é um par de transistores conectados em cascata, a fim de amplificar a corrente de entrada. A Figura 3 apresenta um circuito de um transistor Darlington.

Figura 3 – Transistor Darlington no ULN2003



A base primeiro transistor é o ponto de entrada para o sinal de controle, enquanto o coletor e emissor do segundo transistor compõe os terminais de saída. O circuito tem como principal função amplificar a corrente que chega nos motores, de forma a tornar possível o controle desse por meio de um microcontrolador, que possui baixa corrente nos seus pinos de saída.

Em um sistema de transistores Darlington o beta do transistor, que determina o ganho de corrente entre o base e o emissor é dado pela equação abaixo, onde β_{Q1} representa o beta do primeiro transistor do par, β_{Q2}, do segundo e β_d, o beta da associação deles na configuração citada.

$$I_E = (\beta_d + 1) * I_B$$
$$\beta_d = \beta_{Q1} * \beta_{Q2}$$

O princípio de funcionamento do sistema se da pela aplicação de um sinal alto no pino INPUT, que faz com que o primeiro transistor conduza e gere uma amplificação da corrente de entrada e passe essa corrente para a base do segundo transistor, que por sua vez, entra em condução e amplifique mais uma vez a corrente que flui entre o coletor e emissor do segundo transistor.

Os resistores usados no circuito servem para limitarem a corrente e garantirem que a polarização dos transistores ocorre da forma correta, a fim de ajudar a controlar a corrente que flui pelo circuito. Quando ambos os transistores estão conduzindo, o caminho entre a saída (OUTPUT) e o GND é completado, permitindo que a corrente passe pela carga conectada entre o pino OUTPUT e COMMON.

Por fim, os diodos servem como proteção do circuito, por atuarem como diodos flyback, que tem como intuito impedir uma tensão reversa devido à energia armazenada na indutância, que pode ocorrer quando o circuito possuir uma carga indutiva, como um motor, for desligado. O diodo em paralelo redireciona essa corrente de forma a proteger os elementos mais sensíveis do circuito, como os transistores.

2.2 Microcontrolador

Desenvolvido pela Raspberry Pi Foundation, a família de microcontroladores RP são os microcontroladores que apresentam um baixo custo e alta performance. Optou-se pelo uso do RP2040 pois além de possuir uma arquitetura baseada no processador Dual-Core Arm Cortex M0+, que é capaz de atingir frequências de clock de até 133MHz, ele também apresenta um periférico de extrema importância para a aplicação, um módulo Real Time Clock (RTC).

Foi utilizada a placa de desenvolvimento Waveshare RP2040 Zero, representada na Figura 4, por ela apresentar um custo menor que a placa oficial da Raspberry e por possuir dimensões menores.

Figura 4 – Waveshare RP2040 Zero

2.3 Micro Switch

O micro switch, representado na Figura 5, é um interruptor com 3 terminais: o comum, normalmente aberto “NO” e normalmente fechado “NC”. O seu funcionamento consiste em conectar um fio ao comum e outro em qualquer um dos outros terminais. No momento que a haste é pressionada, caso esteja conectado o fio no normalmente aberto, será fechado a ligação e servirá como uma chave fechada, mas caso esteja conectado o fio no terminal normalmente fechado, a conexão será aberta.

É necessário utilizar essas chaves nos tambores de cada dígito para que haja um dígito de referência a partir do relevo da face dos tambores que será explicado com mais detalhes ao decorrer deste relatório.

Figura 5 – Micro Switch KW10B

3. Desenvolvimento

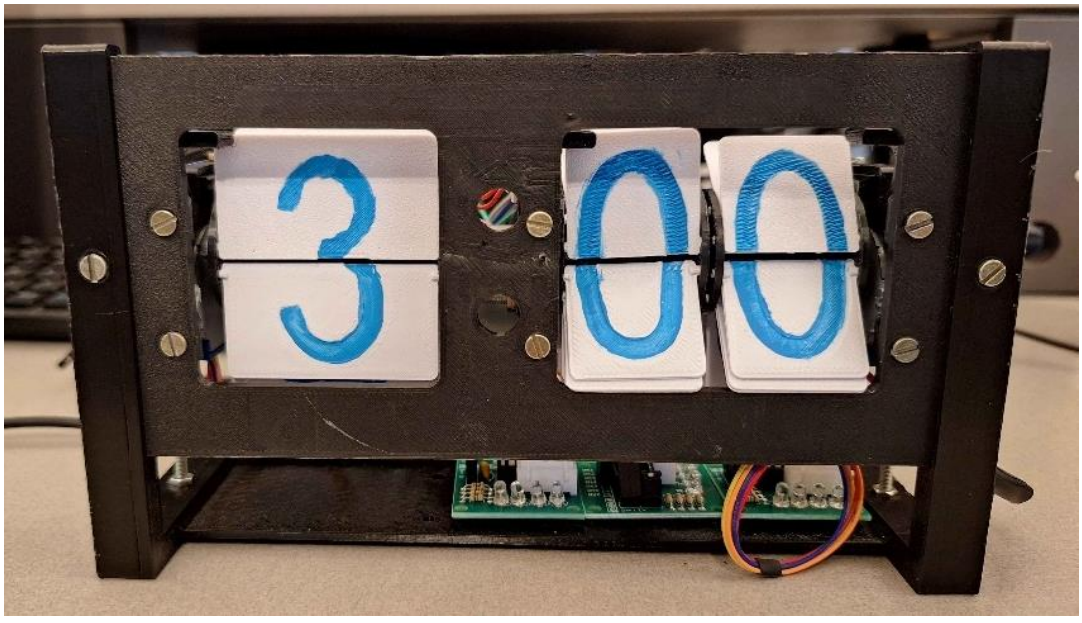
O projeto de forma geral foi inspirado por um repositório no site *github.com* que apresentava a documentação de um outro projeto de flip-clock. A estrutura e o mecanismo dos motores foram aproveitados para este projeto, e o código disponibilizado serviu como inspiração.

3.1 Estrutura

A estrutura principal é composta por duas chapas impressas, uma servindo de base e a outra como a frente do relógio. Essas peças são ligadas por dois suportes em formato de "L".

Na parte traseira foram anexados mancais para os motores de passo, onde cada motor é responsável por um digito, sendo eles, as unidades dos minutos, as dezenas dos minutos e as horas. Acoplado a cada motor existe um tambor que segura as abas que compõem cada digito. A Figura 6 apresenta a montagem final do projeto.

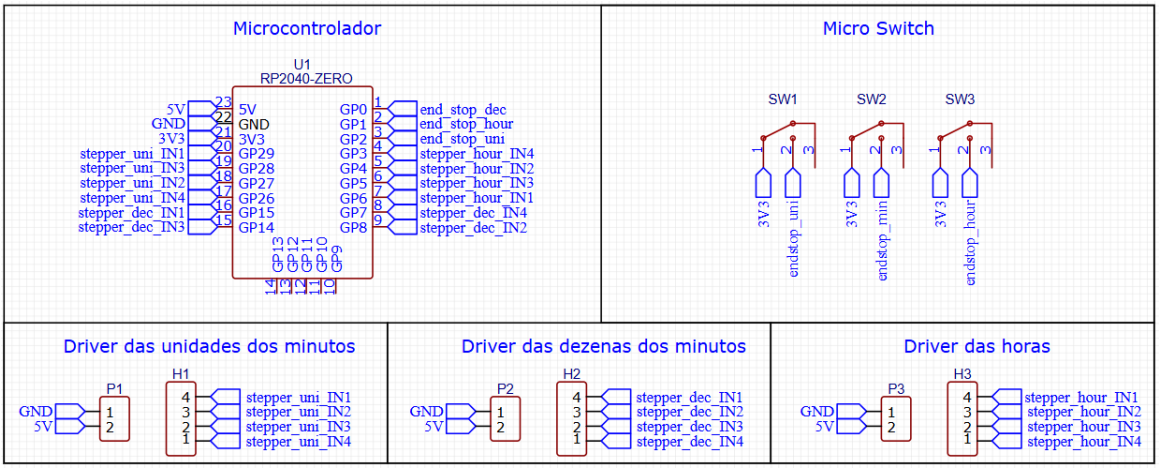
Figura 6 – Relógio Flip



3.2 Componentes eletrônicos

A figura 7 representa o esquemático dos eletrônicos utilizados.

Figura 7 – Pinagem dos componentes



3.3 Software

Para programar o RP2040 existem diversas opções, que variam na linguagem e na performance. O método escolhido foi utilizar o SDK disponibilizada pela própria Raspberry Pi, que suporta as linguagens C e C++, este é o método que melhor aproveita ao que o microcontrolador tem a oferecer.

O código está disponível de forma integral no repositório do projeto no *github.com*: https://github.com/ptBauke1/tic-tac

Para facilitar o desenvolvimento utilizou-se de uma biblioteca para controle dos motores de passo para o SDK, que foi disponibilizada por meio de um repositório no *github*. Na biblioteca, estão disponíveis funções para movimentar o motor em um passo de cada vez, um número determinado de passos e uma angulação definida pelo usuário. A Figura 8 apresenta a função utilizada para mover o motor em um único passo.

Figura 8 – Função stepper_step_once

```
void stepper_step_once(stepper_t *s) {
    s->position += s->direction;
    if (s->position == s->steps_per_revolution) {
        s->position = 0;
    } else if (s->position < 0) {
        s->position = s->steps_per_revolution - 1;
    }
    gpio_put_masked(s->gpio_mask, s->stepping_sequence[s->position % 4]);
}
```

A parte principal do código é a função main(), que é composta em 2 partes: o setup e o loop.

O setup roda apenas uma vez e é responsável em inicializar os objetos, pinos, periféricos e variáveis utilizados. Já o loop é uma parte que roda repetidamente um determinado conjunto de instruções, ele é o responsável por mudar o horário, tanto a variável que controla o tempo, quanto os motores para atualizar o display.

3.2.1 Inicialização

Quando o relógio é energizado, os motores giram até a posição “home”, que é definida no momento que o switch é acionado pelo relevo existente no tambor. Essa posição é importante pois serve como uma posição de “zero” para os motores de passo.

Em sequência, os motores se movem até a posição que representa o horário fornecido pelo RTC. A lógica dessa movimentação é governada pela função step_to_digit. Primeiramente, é determinado o número de abas e de dígitos do motor selecionado, com essa informação, é calculada a posição alvo dado em passos que o atuador deve se rotacionar. Caso seja o motor das dezenas, é determinada uma segunda posição, uma vez que possui dois conjuntos de dígitos indo de 0 a 5.

O alvo é calculado a partir da equação abaixo:

target_pos = starting_offset + ((num_digits + digit - starting_digits) % num_digits) * STEPS_PER_REV/num_flaps;

Onde:

target_pos : posição desejada
starting offset: passos necessários para que a aba toque o limitador do mancal
num_digits : número de dígitos no tambor
digit : dígito dado pelo horário, que é calculado conforme a Figura 9
starting_digits: dígito da posição home
STEPS_PER_REV: passos por revolução do motor
num_flaps: número de abas no tambor

Figura 9- Cálculo do dígito a ser mostrado

```
uint8_t uni = t.min % 10;
uint8_t dec = t.min / 10;
uint8_t hour = t.hour % 12;
```

3.2.2 – RTC

O RTC embarcado é inicializado com um horário pré-determinado aleatório, representado na Figura 10, e utiliza-se desse conjunto para a incrementação conforme o tempo passa. Para que o relógio atualize utilizando o tempo real, foi necessário criar um script em Python (Figura 11) que envia por comunicação serial o horário presente do computador para o microcontrolador logo após que é ligado, através da porta serial.

Figura 10 – Struct do tempo

```
datetime_t rtc_time = {
  .year  = 2024,
  .month = 10,
  .day   = 23,
  .dotw  = 3, // 0 é Domingo, logo 5 é sexta
  .hour  = 17,
  .min   = 58,
  .sec   = 00
};
```

Figura 11 – Script em python

```python
import serial
import time
from datetime import datetime

# Configura a porta serial e a taxa de baud
ser = serial.Serial('COM7', 115200, timeout=1)
time.sleep(2)  # Aguarde a inicialização da porta serial

input("Iniciar...")

horario = datetime.now()
# Dados a serem enviados
data = f"{horario.year}-{horario.month}-{horario.day}"    # Data no formato "YYYY-MM-DD"
hora = f"{horario.hour}:{horario.minute}:{horario.second}"      # Hora no formato "HH:MM:SS"
dia_da_semana = f"3"    # Dia da semana (0 = domingo, 1 = segunda, ..., 6 = sábado)

# Enviar cada string seguida de uma nova linha para separação
ser.write((data + "\n").encode())         # Envia a data
time.sleep(0.1)                           # Pequeno intervalo entre envios

ser.write((hora + "\n").encode())         # Envia a hora
time.sleep(0.1)

ser.write((dia_da_semana + "\n").encode())  # Envia o dia da semana
time.sleep(0.1)

print("Informações enviadas para configurar o RTC:")
print("Data:", data)
print("Hora:", hora)
print("Dia da semana:", dia_da_semana)

# Fechar a porta serial
ser.close()
```

No momento que o RP2040 recebe os dados é feito uma separação e conversão de variável char para int, assim é possível distinguir os números de horas, minutos e segundos os quais são direcionados para os determinados lugares na memória para que ocorra o acréscimo do horário correto. A Figura 12 apresenta a função responsável em realizar essa função no *firmware* do projeto.

Figura 12 – Função para configurar o horário

```c
void serial_communication() {
  char strings[STRING_COUNT][STRING_LENGTH] = {{0}};
        int str_index = 0;
        int char_index = 0;

        // Ler três strings da serial
        while (str_index < STRING_COUNT) {
            int c = getchar_timeout_us(100000);  // Timeout de 100 ms
            if (c != PICO_ERROR_TIMEOUT) {
                if (c == '\n') {
                    strings[str_index][char_index] = '\0';  // Termina a string
                    str_index++;
                    char_index = 0;
                } else if (char_index < STRING_LENGTH - 1) {
                    strings[str_index][char_index++] = (char)c;
                }
            }
        }
        // Exibe as strings recebidas
        printf("Strings recebidas:\n");
        for (int i = 0; i < STRING_COUNT; i++) {
            printf("String %d: %s\n", i + 1, strings[i]);
        }
        // Conversão das strings para configurar o RTC
        // String 1: Data "YYYY-MM-DD"
        rtc_time.year = atoi(strtok(strings[0], "-"));
        rtc_time.month = atoi(strtok(NULL, "-"));
        rtc_time.day = atoi(strtok(NULL, "-"));
        // String 2: Hora "HH:MM:SS"
        rtc_time.hour = atoi(strtok(strings[1], ":"));
        rtc_time.min = atoi(strtok(NULL, ":"));
        rtc_time.sec = atoi(strtok(NULL, ":"));
        // String 3: Dia da semana (0 a 6)
        rtc_time.dotw = atoi(strings[2]);
        // Configura o RTC com a data e hora especificadas
        rtc_init();
        rtc_set_datetime(&rtc_time);
}
```

3.2.3 – Loop

O loop consiste em checar o horário e atualizar as abas conforme o necessário. A estrutura do código está representada na Figura 13.

Figura 13 – loop principal

```
while (true) {
    // loop principal
    rtc_get_datetime(&t);
    uint8_t uni = t.min % 10;
    uint8_t dec = t.min / 10;
    uint8_t hour = t.hour % 12;

    step_to_digit(&stepper_uni, uni, 20);
    step_to_digit(&stepper_dec, dec, 20);
    step_to_digit(&stepper_hour, hour, 20);
    sleep_ms(20);
}
```

4. Conclusão

O desenvolvimento deste projeto de flip-clock possibilitou uma experiência prática valiosa na integração de componentes eletrônicos com design mecânico e programação. A escolha de motores de passo, controlados pelo microcontrolador RP2040 e drivers ULN2003, demonstrou-se acertada, garantindo a precisão necessária para o funcionamento correto dos dígitos. Como dito anteriormente, todo o projeto e vídeo do flip-clock funcionando está documentado no *github*: https://github.com/ptBauke1/tic-tac

O projeto proporcionou aos alunos um aprofundamento na aplicação de conceitos de eletrônica analógica e digital, além de promover habilidades em programação e prototipagem 3D. A realização deste relógio flip-clock contribuiu para consolidar o aprendizado adquirido em sala de aula e preparou os alunos para desafios futuros em projetos de automação e controle.

5. Referências

1. YU, Alex. **Flip-Clock**. Disponível em: https://github.com/alexyu132/flipclock. Acesso em: 10 nov. 2024.

2. RASPBERRY PI LTD. **RP2040 Datasheet**. Disponível em: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf. Acesso em: 10 nov. 2024.

3. GONZÁLEZ, Antonio. **Pico-stepper**. Disponível em: https://github.com/antgon/pico-stepper/tree/main. Acesso em: 10 nov. 2024.

4. RASPBERRY PI LTD. **Pico examples**. Disponível em: https://github.com/raspberrypi/pico-examples/tree/master. Acesso em: 10 nov. 2024.

5. WAVESHARE. **Rp2040-Zero**. Disponível em: https://www.waveshare.com/wiki/RP2040-Zero. Acesso em: 10 nov. 2024.

6. SPEBANMOH. **Micro Switch KW10**. Disponível em: https://base.spebanmoh.com/thtimages/Switch_Micro_KW10/KW10%20Miniature%20Micro%20Switch-SPEC.pdf. Acesso em: 10 nov. 2024.

7. SUNROM. **ULN2003**. Disponível em: https://www.sunrom.com/m/4245. Acesso em: 10 nov. 2024.

8. MURTA, Gustavo. **Guia do Motor de Passo 28BYJ-48 + Driver ULN2003**. Disponível em: https://blog.eletrogate.com/guia-completo-do-motor-de-passo-28byj-48-driver-uln2003/. Acesso em: 10 nov. 2024.

7. TOSHINA. **Datasheet ULN2003,04APG/AFWG**. Disponível em: https://blog.eletrogate.com/wp-content/uploads/2018/07/ULN2003APG.pdf. Acesso em: 10 nov. 2024.