Saída Analógica

Experimento 2

1st Luis Henrique Ferracciu Pagotto Mendes
Engenharia de Computação
Universidade Tecnológica Federal do Paraná
Apucarana, Brasil
luis.mendes.2020@alunos.utfpr.edu.br

***Resumo*—Este relatório explora a aplicação de técnicas de temporização em sistemas microcontrolados. O objetivo principal é utilizar a função *millis()* para programação não bloqueante e configurar interrupções de temporizador (timers) para controle preciso de tempo e gerenciamento de tarefas concorrentes. O experimento aborda desde o controle de LEDs e a transmissão de dados seriais utilizando *millis()* até a implementação de efeitos de *fade* em LEDs, leitura de entradas analógicas e a execução simultânea de múltiplas tarefas através de interrupções de timer. Os métodos envolvem a montagem de circuitos simples com LEDs, potenciômetros e a programação do Arduino para executar as funcionalidades propostas, demonstrando a transição de abordagens de temporização simples para técnicas avançadas e eficientes.**

***Palavras-Chave*—Arduino, Temporização, millis(), Interrupção de Timer, PWM.**

I. Introdução

Para o desenvolvimento de sistemas embarcados, a temporização não bloqueante e precisa é importante para a execução paralela de diversas tarefas, e sem perda de desempenho. No Arduino, a função *millis()* informa o número de milissegundos desde a execução do programa, permitindo gerar temporizações, o que pode ser feito sem o uso de *delay()*, que bloqueia o *loop* principal do Arduino. Como também existem os timers que geram interrupções periódicas e são ajustados via registradores, o processador pode ser utilizado para outras tarefas, enquanto o hardware aguarda o evento de tempo.

O Experimento 6 consiste em explorar duas maneiras de gerar uma base de tempo: (i) com a função *millis()* e (ii) com as interrupções por timer configuradas via registradores. Com o Arduino Mega 2560, foram implementadas rotinas de modulação do brilho de LEDs via PWM e de transmissão de informações de horário e de leitura analógica via interface serial, ambas sem bloqueio.

II. Materiais e Métodos

Para a realização deste experimento e verificação dos resultados, foram utilizados os materiais a seguir:

A. Materiais Utilizados

- 1 Arduino Mega 2560;
- 1 Protoboard;
- 1 Fonte de alimentação USB 5V.
- 3 Diodos Emissores de Luz (LEDs);
- 1 Potenciômetro linear de 100kΩ;
- 3 Resistores de 330Ω;
- Jumpers macho-macho;

B. Montagem do Sistema

O circuito foi montado conforme descrito:

1) Conectar o ânodo do LED ao pino 13 do Arduino, em série com um resistor de 330 Ω, retornando ao GND da placa.
2) Ligar o potenciômetro entre +5V e GND, com o terminal central ao pino A0 do Arduino.
3) Inserir o Arduino e a protoboard na mesa de trabalho e estabelecer comunicação serial via cabo USB.

C. Métodos e Implementação

Exercício 6.1 – Controle de LED Analógico via Interrupção de Timer

- Configuração do Timer1 (16 bits) com prescaler 1:1, carregando o registrador TCNT1 para gerar estouro a cada 1 ms.
- Habilitação da interrupção de overflow (TOIE1) para Timer1.
- Na rotina ISR(TIMER1_OVF_vect), reinicialização de TCNT1 e incremento de contador de passos de PWM.
- No *loop* principal, a saída analógica PWM (analogWrite) varia de 0 a 255 ao longo de 300 ms de subida e 300 ms de descida, calculados com base no número de interrupções.

Exercício 6.2 – Leitura Analógica de Potenciômetro com Timer

- Utilização da mesma interrupção de Timer1.
- Dentro da ISR, leitura da entrada analógica A0 (analogRead) e armazenamento do valor lido.
- No *loop* principal, transmissão serial do valor do potenciômetro a cada ciclo completo de LED.

Exercício 6.3 – Transmissão de Minutos e Segundos Serialmente

- Também realizada dentro da rotina de interrupção, acumulando ciclos para formar segundos e minutos.
- A cada segundo completo (1000 interrupções), atualizar contadores de segundos e minutos.
- No *loop* principal, imprimir na interface serial, no formato **"Min: MM | Seg: SS"**, sincronizado com o controle de LED e leitura analógica.

D. Código Implementado

Em comparação ao método baseado em interrupções, elaborou-se um código que apenas usa a função *millis()* para fornecer um tempo não bloqueante e controlar o brilho do LED e a leitura do potenciômetro. Abaixo se descreve como funciona o código:

- Definições iniciais:
  - ledPin se refere ao pino de PWM (pino 11) que será utilizado para controlar a saída do LED.
  - potPin refere-se à entrada analógica (A0) onde o potenciômetro será lido.
  - cycle é 600ms, que equivale a um ciclo completo de subida (300ms) e de descida (300ms) do nível de brilho.
  - startTime armazena o instante em que o programa iniciou, em *setup()* através do *millis()*.
- *Loop* principal (loop()):
  - Leitura do instante atual now = millis() e calculo do tempo decorrido no ciclo atual: t = (now - startTime) % cycle.
  - Cálculo do valor de PWM:
  - Se t < cycle / 2 (subida), pwm = map(t, 0, cycle/2, 0, 255); ou seja, o brilho vai de 0 a 255 em 300 ms.
  - Caso contrário, (descida), pwm = map (t, cycle/2, cycle, 255, 0); ou seja, depois, ele se reduz de 255 a 0 nos próximos 300 ms.
  - Valor calculado para o LED é enviado com analogWrite (ledPin, pwm).
  - Leitura do valor analógico do potenciômetro com analogRead(potPin) e envio deste valor pela porta serial Serial.println(potValue).
  - E se for desejado, inserir um pequeno delay(100) para evitar inundar o buffer serial.

```
const byte ledPin   = 11;       // pino PWM para o LED
const byte potPin   = A0;       // entrada analógica do potenciômetro
const unsigned long cycle = 600; // 600 ms para um ciclo completo (300ms subir + 300ms descer)
unsigned long startTime;

void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
  startTime = millis();
}

void loop() {
  unsigned long now = millis();
  unsigned long t = (now - startTime) % cycle;

  // Cálculo do PWM do LED usando map():
  int pwm;
  if (t < cycle/2) {
    // 0 → 255 em 300ms
    pwm = map(t, 0, cycle/2, 0, 255);
  } else {
    // 255 → 0 em 300ms
    pwm = map(t, cycle/2, cycle, 255, 0);
  }
  analogWrite(ledPin, pwm);

  // Leitura do potenciômetro e envio pela Serial
  int potValue = analogRead(potPin);
  Serial.println(potValue);

  // Pequena pausa para não inundar a Serial (opcional)
  delay(100);
}
```

Figura 1. Imagem do Código

III. Conclusão

O Experimento 6 mostrou como implementar uma eficiente base de tempo no Arduino Mega 2560 sem utilização da função *delay()*, utilizando tanto a função *millis()* quanto interrupções de *timer* configuradas por registradores. O brilho de um LED pôde ser modulado suavemente via PWM, leituras feitas de um potenciômetro e a transmissão de valores de tempo no formato de minutos e segundos ocorreram simultaneamente, mantendo o *loop* principal disponível para outras atividades. O método com interrupções de *timer* foi particularmente robusto, assegurando uma precisão de, aproximadamente, 1ms e um ajuste entre várias tarefas.