Escola de Engenharia da UFMG

Projeto de Sistemas Embutidos

Laboratório 3 - Pisca LED com Temporizador e Interrupção

Nome: Giovanni Martins de Sá Júnior

Matrícula: 2017001850

Semestre: 2023/2

Conteúdo

1 Configuração de GPIO

- Neste terceiro laboratório, para configurar um pino de GPIO com a entrada sensível a interrupções no Arduino Uno, são feitas as seguintes configurações:

1. Configuração de GPIO:

    - **pinMode():** usado para configurar os pinos como entrada (botão) e saída (LED).

    - **attachInterrupt():** usado para configurar a interrupção no pino do botão.

    - **digitalWrite():** usado para controlar o estado do LED.

2. Configurações do Temporizador

    - TCCR1A e TCCR1B são registradores de controle do Timer/Counter 1.

    - TCNT1 é o registrador do contador, que é zerado para começar a contar a partir de zero.

    - OCR1A é o valor de comparação que determina quando ocorre a interrupção.

    - WGM12 é o bit responsável por configurar o modo CTC (Clear Timer on Compare Match).

    - CS12 e CS10 são bits de seleção do prescaler, configurando o fator de divisão do clock.

    - TIMSK1 é o registrador de máscara de interrupção do Timer/Counter 1, e OCIE1A habilita a interrupção de comparação A.

3. Seleção do Clock

    - O valor intervaloTemporizador * 16 é calculado para determinar o valor a ser comparado no temporizador.

- No caso do Arduino Uno com um clock de 16 MHz, isso resulta em um intervalo de 250 ms (0,25 segundos).

- O prescaler é configurado como 1024 para dividir o clock por esse fator.

4. Funcionamento do Temporizador

   - A rotina de interrupção (ISR(TIMER1-COMPA-vect)) é chamada quando o temporizador atinge o valor de comparação.

   - temporizadorExpirado é marcado como verdadeiro, indicando que o temporizador atingiu o intervalo desejado.

   - No loop principal, você verifica se temporizadorExpirado é verdadeiro e realiza ações específicas com base nisso.

2 Implementação

- A seguir, é listado abaixo a implementação feita para o terceiro laboratório:

```
const int pinoLed = 2;
const int pinoBotao = 13;
int estadoBotao = 0;

enum Estados { AGUARDANDO, LED_ACESO, LED_APAGADO, PISCANDO,
    FINALIZANDO };
Estados estado = AGUARDANDO;

volatile bool botaoPressionado = false;
volatile bool temporizadorExpirado = false;

unsigned long millisAnterior = 0;
int contagemPiscadas = 0;  // Contador para o n mero de piscadas

const unsigned long intervaloTemporizador = 250;  // Intervalo do
     temporizador em milissegundos

void setup() {
  pinMode(pinoLed, OUTPUT);
  pinMode(pinoBotao, INPUT);

  // Configurar a interrup  o para o pino do bot o
  attachInterrupt(digitalPinToInterrupt(pinoBotao),
    botaoInterrupcao, RISING);

  // Configurar o temporizador
  setupTemporizador();
}

void loop() {
  unsigned long millisAtual = millis();

  // Verificar se o temporizador expirou
  if (temporizadorExpirado) {
    temporizadorExpirado = false;

    switch (estado) {
```

```
35      case PISCANDO:
36        digitalWrite(pinoLed, !digitalRead(pinoLed)); // Inverter
    o estado do LED
37        break;
38    }
39  }
40
41  // Restante do seu c digo permanece inalterado
42  switch (estado) {
43    case AGUARDANDO:
44      estadoBotao = digitalRead(pinoBotao);
45      if (estadoBotao == HIGH) {
46        delay(50); // Debounce
47        while(digitalRead(pinoBotao) == HIGH); // Espera liberar
48        estado = LED_ACESO;
49        millisAnterior = millisAtual;
50      }
51      break;
52
53    case LED_ACESO:
54      digitalWrite(pinoLed, HIGH);
55      if (millisAtual - millisAnterior >= 1000) {
56        estado = LED_APAGADO;
57        millisAnterior = millisAtual;
58      }
59      break;
60
61    case LED_APAGADO:
62      digitalWrite(pinoLed, LOW);
63      if (millisAtual - millisAnterior >= 2000) {
64        estado = PISCANDO;
65        millisAnterior = millisAtual;
66      }
67      break;
68
69    case PISCANDO:
70      if ((millisAtual - millisAnterior >= 250) && (
   contagemPiscadas < 6)) {
71        digitalWrite(pinoLed, !digitalRead(pinoLed)); // Inverte
   o estado do LED
```

```
72          millisAnterior = millisAtual;
73          contagemPiscadas++;
74        }
75        if (contagemPiscadas >= 6) {
76          contagemPiscadas = 0; // Reinicia o contador de piscadas
77          estado = FINALIZANDO;
78        }
79        break;
80
81      case FINALIZANDO:
82        digitalWrite(pinoLed, LOW);
83        estado = AGUARDANDO;
84        break;
85    }
86  }
87
88  // Fun  o de interrup  o para o bot o
89  void botaoInterrupcao() {
90    botaoPressionado = true;
91  }
92
93  // Configurar o temporizador
94  void setupTemporizador() {
95    noInterrupts();  // Desabilitar interrup  es durante a
       configura  o
96
97    // Configurar o temporizador 1 (para Arduino Uno)
98    TCCR1A = 0;      // Modo normal
99    TCCR1B = 0;      // Limpar registrador de controle
100   TCNT1 = 0;       // Zerar contador
101
102   // Calcular o valor para OCR1A (com base no clock do Arduino
       Uno de 16MHz)
103   OCR1A = (intervaloTemporizador * 16) - 1;
104
105   // Configurar o temporizador para operar no modo CTC (Clear
       Timer on Compare Match)
106   TCCR1B |= (1 << WGM12);
107
108   // Selecionar o prescaler (1024 para intervalo de 250ms)
```

```
109   TCCR1B |= (1 << CS12) | (1 << CS10);
110
111   // Habilitar a interrup  o de compara  o A
112   TIMSK1 |= (1 << OCIE1A);
113
114   interrupts();  // Habilitar interrup  es ap s a
        configura  o
115 }
116
117 // Rotina de interrup  o para o temporizador
118 ISR(TIMER1_COMPA_vect) {
119   temporizadorExpirado = true;
120 }
```

Listing 1: Implementação do Código do terceiro laboratório

3 Resultados

A realização do terceiro laboratório trouxe laboratório trouxe uma dificuldade um pouco maior que a esperada, pelo fato da necessidade de se corrigir os erros do segundo laboratório. Contudo, os testes para o terceiro laboratório funcionaram conforme o esperado.