CEFET-MG — Centro Federal de Educação Tecnológica de Minas Gerais

DEPARTAMENTO DE COMPUTAÇÃO DE DIVINÓPOLIS — DECOM-DV

Microprocessadores e Microcontroladores

Primeira Atividade Avaliativa

Aluno: ANIELLY GONÇALVES

Valor: 30 pontos (cada questão vale 5 pontos)

Turma: 2024/1

Prof. M. Sc. Diego Ascânio Santos

Respostas:

1	2	3	4	5	6

Questão 1

Quanto as interrupções, avalie as assertivas:

I. Interrupção é um mecanismo que permite a uma entidade externa interromper a execução de um programa sendo executado.

II. Chegada de dados em uma porta de entrada/saída pode ser um exemplo de interrupção.

III. Jammais podem ser associadas a eventos assíncronos.

IV. O pressionamento de um botão pode ser um exemplo de interrupção.

Quais são falsas?

a) I, II, III e IV.

b) I, II e IV.

c) I e II apenas.

d) I e IV apenas.

e) III apenas.

Questão 2

Júlio procurou seu professor de microcontroladores durante a aula alegando que o código que ele havia escrito para piscar o led interno (Led 13) do Arduino de 500 em $500ms$ não estava funcionando. O professor pediu para que Júlio mostrasse o código que havia escrito e assim, Júlio o fez:

```
void setup() {
    pinMode(13, INPUT);
}

void loop() {
    digitalWrite(13, HIGH);
    delay(500);
    digitalWrite(13, LOW);
}
```

Qual das seguintes alternativas resolve o problema de Júlio?

a) Trocar a função `pinMode(13, INPUT);` por `pinMode(13, OUTPUT);` — na linha 2 — da função

`setup()`.

b) Adicionar uma instrução `delay(500);` logo abaixo da instrução `digitalWrite(13, LOW);` — que está na linha 8 — da função `loop()`.

c) Retirar a instrução `delay(500)` — que está na linha 7 — da função `loop()`.

d) Trocar a função `pinMode(13, INPUT);` por `pinMode(13, OUTPUT);` — na linha 2 — da função `setup()` e reescrever a função `loop()` da seguinte forma:

```
void loop() {
    digitalWrite(
        13, !digitalRead(13)
    );
    delay(500);
}
```

e) Nenhuma das alternativas anteriores resolve o problema de Júlio

---

Questão 3

A respeito de entradas e saídas digitais do Arduino, resistores pull-up e pull-down, contatos normalmente abertos e normalmente fechados, avalie as assertivas:

I. Por padrão as entradas digitais do Arduino estão preparadas para receber sinais digitais em nível lógico TTL (0V a 5V).

II. Não é necessário realizar quaisquer tipos de adaptações para conectar circuitos digitais não-TTL (por exemplo, CMOS) ao Arduino.

III. Um contato normalmente aberto é um contato que, em repouso, não permite a passagem de corrente elétrica.

IV. O nível lógico de uma entrada digital do Arduino conectada a um contato normalmente aberto em seu estado de repouso é sempre 0V.

V. Resistores pull-up são utilizados para garantir que uma entrada digital do Arduino esteja sempre em nível lógico alto em seu estado padrão.

VI. O Arduino não dispõe de resistores pull-up internos, sendo necessário adicionar resistores externos para este fim.

São falsas as assertivas:

a) I, II, III, IV, V e VI.

b) II, III, IV, V e VI.

c) II, IV, V e VI.

d) II e V.

e) II e VI.

Questão 4

Interrupções no arduino podem ser habilitadas e associadas a funções de*callback* — que são executadas quando interrupções ocorrem — através do **ISR** (Interrupt Service Routine). Em relações aos recursos e limitações que as funções de *callback* apresentam, é correto afirmar que:

a) Funções de *callback* de interrupções podem receber argumentos e retornar valores.

b) É possível executar múltiplos *callbacks* ao mesmo tempo.

c) Funções de *callback* podem receber argumentos, mas, não podem retornar valores.

d) Podem ser interrompidas por outras interrupções.

e) O modificador de variável `volatile` precisa ser utilizado para permitir a modificação de variáveis globais nas funções de *callback* e na função principal.

Questão 5

Cláudia perguntou ao professor Ascânio se sua rotina de interrupções de overflow do `Timer2` estava correta para contar intervalos de tempo de 10 em 10 segundos, pois, estava com duvidas se seus cálculos de quantidade de *overflows* e o modo de *prescaling* que havia definido para o `Timer2` estavam corretos. O professor Ascânio verificou o código apresentado por Cláudia e disse que tanto os cálculos quanto o *prescaling* estavam corretos, mas, que a rotina de interrupção de *overflow* do `Timer2` — *Flag* `TOIE2` — estava desabilitada.

Qual deve ser a instrução que Cláudia deve adicionar à função `setup()` de seu código para habilitar a interrupção de *overflow* do `Timer2`?

a) `TIMSK2 = TIMSK2 | 0b00000001;`

b) `TIMSK2 = 0b00000000;`

c) `TIMSK2 = TIMSK2 | 0b00000000;`

d) `TIMSK2 = TIMSK2 | 0b00000010;`

e) `TIMSK2 = TIMSK2 | 0b00000100;`

Questão 6

Raul deseja fazer um LED comutar de estado a cada 5 segundos. Sem muito critério do entendimento viu nos códigos dos slides do prof. Ascânio o seguinte ISR que fazia o LED comutar de estado a cada 5 segundos:

Rotina de interrupção de Overflow associada ao `Timer2`

```
ISR(TIMER2_OVF_vect) {
    overflows ++;
    if (overflows == 306) {
        overflows = 0;
        // Toggle the LED state
        digitalWrite(13, !digitalRead(13));
    }
}
```

Entretanto, ao copiar os códigos, não se atentou ao modo correto de funcionamento e configurou — sem querer — o prescaler do `Timer2` para o fator de 256, através do registrador de controle `TCCR2B` pela seguinte instrução na função `setup()`:

```
    TCCR2B = 0b00000110;
```

Com isso, em vez do LED comutar de estado a cada 5 segundos, ele comutava a cada 1.25 segundos (aproximadamente). Qual dos seguintes modos de *prescaling* do `Timer2` faz com que o LED comute de estado a cada 5 segundos, como deseja Raul?

a) `TCCR2B = 0b00000000;`

b) `TCCR2B = 0b00000001;`

c) `TCCR2B = 0b00000011;`

d) `TCCR2B = 0b00000111;`

e) `TCCR2B = 0b00000110;`