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

DEPARTAMENTO DE COMPUTAÇÃO DE DIVINÓPOLIS — DECOM-DV

Microprocessadores e Microcontroladores

Primeira Atividade Avaliativa

Aluno: MATEUS HENRIQUE PEREIRA

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

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 verdadeiras as assertivas:

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

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

c) I, III, IV e V.

d) II e V.

e) II e VI.

Questão 2

Acerca de entradas e saídas digitais no Arduino e a conexão de periféricos ao microcontrolador, avalie as seguintes assertivas:

I. O comando `pinMode(13, OUTPUT)` configura o pino 13 como saída.

II. Para uma conexão em série do pino `+5V` do Arduino a um resistor de 330Ω passando pelo ânodo

de um LED, em sequência pelo cátodo do LED e finalizando no pino `13` do Arduino, o comando `digitalWrite(13, HIGH)` acenderá o LED se ele estiver apagado.

III. Para uma conexão em série do pino `+5V` do Arduino a um resistor de 330Ω passando pelo ânodo de um LED, em sequência pelo cátodo do LED e finalizando no pino `13` do Arduino, o comando `digitalWrite(13, HIGH)` apagará o LED se ele estiver aceso.

IV. O comando `digitalWrite(13, !digitalRead(13))` inverte o estado do dispositivo conectado ao pino 13 qualquer que seja seu modo de conexão (ligado ao `+5V` ou ao `GND` do Arduino).

São corretas as assertivas:

a) I e II, apenas.

b) I e III, apenas.

c) I e IV, apenas.

d) I, III e IV apenas.

e) I, II, III e IV.

Questão 3

Acerca do ATmega328P, microcontrolador presente no arduino UNO e NANO, avalie as assertivas:

I. A CPU do ATmega328P é baseada na arquitetura Harvard com um conjunto complexo de instruções (CISC);

II. O ATmega328P possui 32KB de memória flash;

III. A CPU do ATmega328P é baseada na arquitetura Von Neummann com um conjunto reduzido de instruções (RISC);

IV. É possível usar as 6 entradas analógicas do ATmega328P simultâneamente;

V. Para qualquer pino presente no microcontrolador, o ATmega328P não permite que um mesmo pino possa executar mais de uma função;

VI. Exceder os limites operacionais de tensão e corrente do ATmega328P não danifica o microcontrolador.

Quais estão corretas?

a) Nenhuma das assertivas.

b) II.

c) II, IV, V.

d) II, IV, V, VI.

e) Todas as assertivas.

Questão 4

(ENADE 2005 - 11) Apesar de todo o desenvolvimento, a construção de computadores e processadores continua, basicamente, seguindo a arquitetura clássica de von Neumann. As exceções a essa regra encontram-se em computadores de propósitos específicos e nos desenvolvidos em centros de pesquisa. Assinale a opção em que estão corretamente apresentadas características da operação básica de um processador clássico:

a) Instruções e dados estão em uma memória física única; um programa é constituído de uma seqüência de instruções de máquina; uma instrução é lida da memória de acordo com a ordem dessa seqüência e, quando é executada, passa-se, então, para a próxima instrução na seqüência.

b) Instruções e dados estão em memórias físicas distintas; um programa é constituído de um conjunto de instruções de máquina; uma instrução é lida da memória quando o seu operando-destino necessita ser recalculado; essa instrução é executada e o resultado é escrito no operando de destino, passando-se, então, para o próximo operando a ser recalculado.

c) Instruções e dados estão em uma memória física única; um programa é constituído de um conjunto de instruções de máquina; uma instrução é lida da memória quando todos os seus operandos-fonte estiverem prontos e disponíveis; essa instrução é executada e o resultado é escrito no operando de destino, passando-se, então, para a instrução seguinte que tiver todos seus operandos disponíveis.

d) Instruções e dados estão em memórias físicas distintas; um programa é constituído de um conjunto de instruções de máquina; uma instrução é lida da memória quando todos os seus operandos-fonte estiverem prontos e disponíveis; essa instrução é executada e o resultado é escrito no operando de destino, passando-se, então, para a instrução seguinte que estiver com todos os seus operandos disponíveis.

e) Instruções e dados estão em memórias físicas distintas; um programa é constituído de uma seqüência de instruções de máquina; uma instrução é lida da memória de acordo com a ordem dessa seqüência e, quando é executada, passa-se, então, para a próxima instrução na seqüência.

Questão 5

Considere o código abaixo:

```
const int ledPin = 13;
const int interruptPin = 2; // only pin 2 and 3 can be used for interrupts

volatile int state = LOW;

void blink() { // ISR function
    state = !state; // toggle the state
}

// missing setup function

void loop() {
    digitalWrite(ledPin, state);
}
```

É desejado que o LED conectado ao pino 13 comute de estado a cada vez que o botão conectado ao pino 2 for pressionado. O *pushbutton* conectado ao pino 2 do arduino também está conectado ao pino `GND` do microcontrolador.

Qual alternativa contém a implementação da função `setup()` que atende a esse requisito no Arduino UNO?

a)

```
void setup() {
    pinMode(2, INPUT_PULLUP);
    pinMode(ledPin, OUTPUT);
    attachInterrupt(
        digitalPinToInterrupt(interruptPin),
        blink,
        KEEPING
    );
}
```

b)

```
void setup() {
    pinMode(2, INPUT_PULLUP);
    pinMode(ledPin, OUTPUT);
    attachInterrupt(
        digitalPinToInterrupt(interruptPin),
        blink,
        CHANGE
    );
}
```

c)

```
void setup() {
    pinMode(2, INPUT_PULLUP);
    pinMode(ledPin, OUTPUT);
    attachInterrupt(
        digitalPinToInterrupt(interruptPin),
        blink,
        HIGH
    );
}
```

d)

```
void setup() {
    pinMode(2, INPUT_PULLUP);
    pinMode(ledPin, OUTPUT);
    attachInterrupt(
        digitalPinToInterrupt(interruptPin),
        blink,
        FALLING
    );
}
```

e)

```
void setup() {
    pinMode(2, INPUT_PULLUP);
    pinMode(ledPin, OUTPUT);
    attachInterrupt(
        digitalPinToInterrupt(interruptPin),
        blink,
        LOW
    );
}
```

Questão 6

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;`