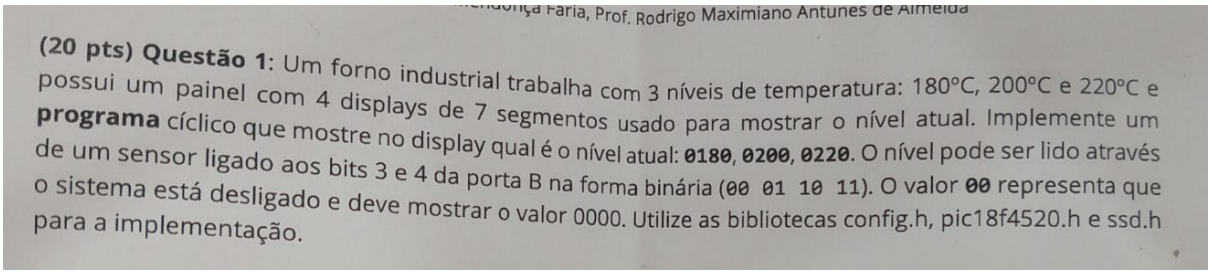
Resolução primeira prova de ECOP04 - 2023.2



...nça Faria, Prof. Rodrigo Maximiano Antunes de Almeida

(20 pts) Questão 1: Um forno industrial trabalha com 3 níveis de temperatura: 180°C, 200°C e 220°C e possui um painel com 4 displays de 7 segmentos usado para mostrar o nível atual. Implemente um **programa** cíclico que mostre no display qual é o nível atual: **0180**, **0200**, **0220**. O nível pode ser lido através de um sensor ligado aos bits 3 e 4 da porta B na forma binária (`00 01 10 11`). O valor **00** representa que o sistema está desligado e deve mostrar o valor 0000. Utilize as bibliotecas config.h, pic18f4520.h e ssd.h para a implementação.

Lógica:

Os bits 3 e 4 estão conectados ao sensor de temperatura, e operam em conjunto para realizar a leitura do sinal, portanto, a saída depende de ambos.

PORTB[3]	PORTB[4]	Saída
0	0	0000
0	1	0180
1	0	0200
1	1	0220

A abordagem mais simples é utilizar uma cascata de 4 IFs, para testar todos os valores possíveis de combinação, da seguinte forma.

```
#include "config.h"
#include "ssd.h"
#include <pic18f4520.h>

void main(void) {

    ssdInit();

    TRISB = 0x00; //Como nenhuma porta está sendo utilizada como
entrada, essa configuração serve

    BitClr(TRISB, 3); //Outra configuração possível
    BitClr(TRISB, 4);

    for(;;){
      ssdUpdate();
        if(BitTst(PORTB,3) && BitTst(PORTB,4)){
```

```
        //11, Saída = 0220
            ssdDigit(0, 0);
            ssdDigit(2, 1);
            ssdDigit(2, 2);
            ssdDigit(0, 3);
        } else if(BitTst(PORTB,3) && !BitTst(PORTB,4)){
        //10, Saída = 0200
            ssdDigit(0, 0);
            ssdDigit(2, 1);
            ssdDigit(0, 2);
            ssdDigit(0, 3);
        } else if(!BitTst(PORTB,3) && BitTst(PORTB,4)){
        //01, Saída = 0180
            ssdDigit(0, 0);
            ssdDigit(1, 1);
            ssdDigit(8, 2);
            ssdDigit(0, 3);
        }else{
        //00, saída = 0000;
            ssdDigit(0, 0);
            ssdDigit(0, 1);
            ssdDigit(0, 2);
            ssdDigit(0, 3);
        }
    }
}
```

(25 pts) Questão 2: Um sistema de radar monitora a velocidade dos veículos e acende uma luz verde (Porta B, bit 0) quando o veículo não ultrapassou 60km/h, entre 60km/h e 70km/h uma luz vermelha (Porta B, bit 1) é acionada e acima dessa velocidade, além da luz vermelha, uma câmera (Porta C, bit 0) é acionada para registrar a placa do veículo. Depois de um tempo as lâmpadas se apagam. Implemente um **programa** para esse sistema sabendo que a velocidade é obtida em quilômetros por hora através da função adcRead() da biblioteca adc.h, esta função retorna zero quando nenhum veículo foi identificado.

Lógica - A velocidade do veículo é obtida pela função adcRead() já em km/h, então só é necessário realizar a comparação com os limites de velocidade estabelecidos no enunciado.

```
#include "config.h"
#include "adc.h"
#include <pic18f4520.h>

void main(void) {
```

```
    adcInit();

    TRISB = 0x00; //Como nenhuma porta B está sendo utilizada como
entrada, essa configuração serve
    TRISC = 0x00; //Como nenhuma porta C está sendo utilizada como
entrada, essa configuração serve

    //Outra configuração possível
    BitClr(TRISB, 0); //Luz Verde
    BitClr(TRISB, 1); //Luz Vermelha
    BitClr(TRISC, 0); //Camera

    int velocidade = 0;
    //Necessário uma variável auxiliar pro valor de teste ser sempre o
mesmo
    //Pois a leitura de ADC é dinâmica e pode haver variações
    for(;;){
        //Necessário testar se é diferente de zero, pois as luzes não
podem ficar acesas se não houverem carros
        velocidade = adcRead();
        if(Velocidade != 0){
            if(velocidade < 60){
                BitSet(PORTB, 0); //acende a luz verde
                delay(1000); //delay para apagar a luz
            }else if(velocidade >=60 && velocidade <= 70){
                BitSet(PORTB, 1); //acende a luz vermelha
                delay(1000); //delay para apagar a luz
            }else if(velocidade > 70){
                BitSet(PORTB, 1); //acende a luz verde
                BitSet(PORTC, 0); //liga a câmera
                delay(1000); //delay para apagar a luz
            }
        }

        //Após passarem os delays desligam-se os periféricos;
        //Ou se não houverem carros, os periféricos se mantém
desligados;

        BitClr(PORTB, 0); //apaga a luz verde
        BitClr(PORTB, 1); //apaga a luz vermelha
        BitClr(PORTC, 0); //apaga a luz camera
    }
}
```

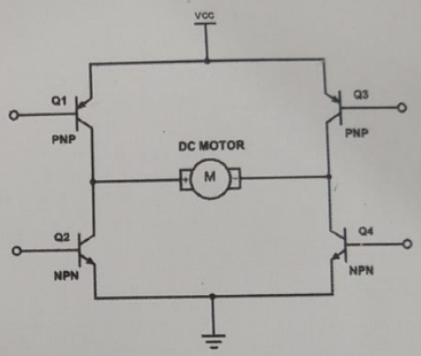
(25 pts) Questão 3: Um sistema de defesa militar pode fazer disparos (Porta A, bit 0) quando detecta a presença de uma aeronave inimiga (Porta D, bit 3). O acionamento, por questão de segurança, só deve começar quando dois militares ativarem dois botões simultaneamente (bits 1 e 2 da porta D). Caso somente um botão seja pressionado, o sistema deve ignorar o comando e esperar que o operador solte o botão antes de checar novamente. Faça um **programa** para esse sistema. Podem ser utilizadas as bibliotecas pic18f4520.h e config.h.

Lógica - Caso uma aeronave inimiga for detectada pelo sensor na Porta D bit 3, o sistema fica preparado para atirar, mas o disparo só é liberado ao pressionar os dois botões ao mesmo tempo. Se os botões forem pressionados em tempos diferentes, o sistema deverá entrar em loop até que as teclas sejam soltas.

```
#include "config.h"
#include <pic18f4520.h>

void main(void) {

    TRISA = 0x00; //Como nenhuma porta A está sendo utilizada como
entrada, essa configuração serve
    TRISD = 0x0E; //0000 1110 bits -- 1 2 3 como entrada

    //Outra configuração possível
    BitClr(TRISA, 0); //Arma
    BitSet(TRISD, 1); //Botão 1
    BitSet(TRISD, 2); //Botão 2
    BitSet(TRISD, 3); //Sensor


    for(;;){
        //Se o sensor captar uma aeronave, entra em modo de disparo
        if(BitTst(PORTD, 3)){
            //apenas o botão 1 foi pressionado, espera os dois serem
soltos
            if(BitTst(PORTD,1) && !bitTst(PORTD,2)){
                while (BitTst(PORTD,1) || bitTst(PORTD,2));
            }
            //apenas o botão 2 foi pressionado, espera os dois serem
soltos
            if(!BitTst(PORTD,1) && bitTst(PORTD,2)){
                while (BitTst(PORTD,1) || bitTst(PORTD,2));
            }
```

```
            //Ambos os botões foram pressionados, disparo autorizado
            if(BitTst(PORTD,1) && bitTst(PORTD,2)){
                BitSet(PORTA,0);
                BitClr(PORTA,0);
            }
        }
        //Sem alvos, arma desativada
        BitClr(PORTA,0);
    }
}
```

(30 pts) Questão 4: Um sistema de direcionamento de antenas possui um motor que ajusta a direção das antenas de 0° a 359° com passos de 1°. Uma ponte H é utilizada para controlar o motor. O sentido da corrente determina a direção do motor. Considere que os transistores são ativados com nível lógico 1. Faça **uma biblioteca** com uma função **init_motor()** que não recebe nem retorna nada que deve inicializar o sistema e uma função **motor()** que recebe dois parâmetros: o valor da posição_atual e a posição_desejada. Esta função deve comparar os dois valores e ligar o motor para o sentido correto. Valores maiores de 359 devem ser ignorados. Se a posição atual for igual à nova posição, todos os transistores devem ser desligados. Os transistores Q1, Q2, Q3 e Q4 utilizam os bits 4, 5, 6 e 7 da porta A.




```
//adc.h
void adcInit(void);
int adcRead(void);

//config.h
Necessário para configurar o hardware
Possui comandos: #pragma …

//Observação: TRIS = 0 → Saída
```

```
//pic18f4520.h
Define TRISA, PORTA,... bem como:
BitSet(arg,bit)   BitClr(arg,bit)
BitFlp(arg,bit)   BitTst(arg,bit)

//ssd.h
void ssdInit(void);
void ssdUpdate(void);
void ssdDigit(int val, int pos);
```

Lógica - É necessário declarar a biblioteca motor de acordo com o enunciado

```
#ifndef MOTOR_H
    #define MOTOR_H
    void motor(unsigned int posicao_atual, unsigned int
posicao_desejada);
    void init_motor(void);

#endif
```

Lógica - Após definida a biblioteca, entra a lógica das funções

- init_motor(void): Como função de inicialização, é responsável por configurar as portas a serem utilizadas, PORTA 4 | 5 | 6 | 7 como saídas

```
#include "motor.h"
#include <pic18f4520.h>

void init_motor(void){
    TRISA &= 0x0F;
    //Dessa forma, os bits 3 | 2 | 1 | 0 que forem 1, se manterão como 1
    //enquanto os bits 7 | 6 | 5 | 4 que forem 1, se tornarão 0

    //Outra implementação
    BitClr(TRISA, 7);
    BitClr(TRISA, 6);
    BitClr(TRISA, 5);
    BitClr(TRISA, 4);
}
```

- motor(posicao_atual, posicao_desejada): Primeiramente, define-se a combinação de transistores para cada sentido, sendo Q1 e Q4 para sentido horário | Q2 e Q3 para sentido anti-horário.

```
void motor(unsigned int posicao_atual, unsigned int posicao_desejada){
    //Caso a posição desejada seja maior que 359, a função deve ignorar
    if(posicao_desejada <= 359){
        if(posicao_atual < posicao_desejada){
            //Caso a posição desejada seja maior que a posição atual, sentido horário ativado;
            //Liga Q1 e Q4
            BitSet(PORTA, 4);
            BitSet(PORTA, 7);

            //Desliga Q2 e Q3
            BitClr(PORTA, 5);
            BitClr(PORTA, 6);
        } else if(posicao_atual > posicao_desejada){
```

```
            //Caso a posição desejada seja menor que a posição
  atual, sentido anti-horário ativado;;
            //Liga Q2 e Q3
            BitSet(PORTA, 5);
            BitSet(PORTA, 6);

            //Desliga Q1 e Q4
            BitClr(PORTA, 4);
            BitClr(PORTA, 7);
        } else  if(posicao_atual == posicao_desejada){
            //Caso forem iguais, motores desligados;
            //Desliga todas os transistores
            BitClr(PORTA, 4);
            BitClr(PORTA, 5);
            BitClr(PORTA, 6);
            BitClr(PORTA, 7);
        }
    }
  }

```