MICROPROCESSADORES

ANO LECTIVO: 2023/2024

CURSO: ENG. ELETROTÉCNICA E DE COMPUTADORES

REGIME: D TURMA: PL3

MINIPROJECTO PROPOSTA 9 - VERSÃO C

* + FOSC=8 MHz
  + Baud=9600 bps
  + MPX4250 pressure sensor
  + Bipolar stepper motor (MOTOR-BISTEPPER)
  + Display TFT gráfico c/ resolução 240x320 pixels c/ driver ILI9341
  + LM017L 32x2 alphanumeric LCD
  + 25LC256 256 kbit I2C serial EEPROM

AUTORES:

|  |  |
| --- | --- |
| NOME: Diogo Cravo | NOME: Paulo Sousa |
| Nº ALUNO: 2222030 | Nº ALUNO: 2222031 |
|  |  |
|  |  |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Resumo do Projeto**

|  |
| --- |
| O objetivo deste projeto é criar um sistema que monitoriza e controla a pressão na câmara de combustão de um veículo. Esse sistema será desenvolvido utilizando o microcontrolador PIC 18F46K22.    Figura - Diagrama de blocos do projeto  No programa existe um display que funciona através de SPI onde é possível observar o valor atual da pressão na câmara, o valor atual do grau de abertura da válvula de admissão de ar. Caso a pressão ultrapasse o limite máximo ou mínimo a informação de que o alarme foi ativado e a sua causa (baixa ou alta pressão) será exibido no display.  A alarme ativa automaticamente quando o valor baixa dos 30kPa (pressão baixa) ou quando o valor passa dos 120kPa (pressão alta), estes valores podem ser alterados através da interface série (Usart). O alarme é um LED com intermitência de 0.5 Hz e um buzzer que atua com uma intermitência de 2 Hz. O alarme pode ser desativado através de um botão de pressão que gera uma interrupção no programa.  Com a Interface série (Usart) será possível alterar entre controlo automático e manual da válvula (motor de passo). No modo automático o ângulo da válvula será diretamente proporcional à pressão lida. No modo manual será possível ajustar o ângulo da válvula através do terminal. No terminal será possível a alteração dos valores críticos limite de pressão, mínimo e máximo, para o alarme. |

**Estruturas e Funções**

|  |  |
| --- | --- |
| **Estruturas Utilizadas** | **Funcionalidade** |
| struct sensor mpx4250 | Tem a finalidade de organizar os dados do sensor de pressão. |
| struct valve stepper | Tem a finalidade de organizar os dados do motor de passo. |
| struct data adc | Tem a finalidade de organizar os valores do adc. |

Tabela 1 - Estruturas utilizadas

|  |  |
| --- | --- |
| **Funções Utilizadas** | **Funcionalidade** |
| void shutdownAlarm(); | Função que desliga o alarme |
| void triggerAlarm(); | Função que desliga o alarme |
| bool pressureOutsideThreshold(); | Função analisa se o valor da pressão está além dos limites estabelecidos, indicando assim se o alarme deve ser acionado. Se estiver fora dos limites devolve o valor *true* se não estiver fora devolve *false*. |
| Int setPressureThreshold(int original\_threshold) | Função que permite ajustar os limites superior e inferior da pressão nos quais o alarme é acionado. |
| void rotateSteps(int steps) | Função que roda o motor de passo. Esta função recebe steps como parâmetro de entrada. |
| int readDigitFromSerial(); | Função que aceita apenas números inseridos pelo utilizador no terminal. |
| void mainMenu(); | Função que exibe o menu principal do controle de pressão da câmara de combustão no terminal." |
| void valveMenu(); | Função que exibe o menu do controlo da válvula no terminal. |

Tabela - Funções desenvolvidas

**Explicação dos Registos**

|  |
| --- |
| 1. **Portos IO**   **Definir Porto Digital/Analógico (1/0)**  ● ANSELA=0b00000001;  ● ANSELB=0b00000000;  ● ANSELC=0b00000000;  ● ANSELD=0b00000000;  ● ANSELE=0b00000000;  **Definir Porto Saída/Entrada (1/0)**  ● TRISA = 0b00001111;  ● TRISB = 0b11111111;  ● TRISC = 0b11111111;  ● TRISD = 0b11111111;  ● TRISE = 0b00000111; |

|  |
| --- |
| 1. **Interrupção** **INT0 (desativar alarme)**   Ativa prioridades nas interrupções  ● RCONbits.IPEN = 1;  Ativa as funções de alta e baixa prioridade, respetivamente.  ● INTCONbits.GIEH = 1;  ● INTCONbits.GIEL = 1;  (INT0 é sempre alta prioridade)  Apaga a flag da interrupção  ● INTCON INTCONbits.INT0IF = 0;  Ativa a interrupção  ● INTCONbits.INT0IE = 1;  Flanco descendente  ● INTCON2bits.INTEDG0 = 0; |

|  |
| --- |
| 1. **Timer 0**   **Cálculos:**  O sinal luminoso deverá ser ligado com a intermitência de 2 Hz  ● F=2Hz, com DC=50%  ● TH=TL=500/2 = 250ms  ● Para CONTInicial = 0  ● Logo, **Prescaler = 8**  ● Depois, calculamos o CONTInicial  **Registos:**  ● TMR0=3036;  ● T0CONbits.T08BIT = 0; //Contador de 16 bits  ● T0CONbits.T0CS = 0; //Seleção de Relógio Interno (Fosc/4)  ● T0CONbits.T0SE = 0; //Seleção do bordo para o relógio externo  ● T0CONbits.PSA = 0; //Prescaler ativo  Prescaler = 8, logo bits = 010  ● T0CONbits.T0PS2 = 0; //Prescaler bit 2  ● T0CONbits.T0PS1 = 1; //Prescaler bit 1  ● T0CONbits.T0PS0 = 0; //Prescaler bit 0  ● T0CONbits.TMR0ON = 1; //Inicia contagem  Configuração da interrupção do Timer 0  INTCONbits.TMR0IE = 1; //Autoriza a Interrupção  INTCONbits.TMR0IF = 0; //Apaga a flag de Interrupção  INTCON2bits.TMR0IP = 0; //Interrupção de baixa prioridade |

|  |
| --- |
| 1. **Timer 1 (ADC)**   **Cálculos:**  ● T = 50ms  ● Para CONTInicial = 0  ● Logo, **Prescaler = 2**  ● Depois, calculamos o CONTInicial  **Registos:**  ● TMR1 = 15536; //Inicialização do temporizador a 15536  ● T1CONbits.TMR1CS= 0; //Seleção de Relógio Interno (Fosc/4)  ● T1CONbits.T1CKPS= 0b01; //Prescaler a 2  ● T1CONbits.T1OSCEN = 0; //Oscilador secundário desligado  ● T1CONbits.T1SYNC = 0; // Relógio sincronizado com oscilador externo  ● T1CONbits.RD16 = 1; //Lê 16 bits  ● T1CONbits.TMR1ON = 1; //Inicia a contagem  //Configuração da interrupção do Timer 1  ● PIE1bits.TMR1IE = 1; //Autoriza a Interrupção  ● PIR1bits.TMR1IF = 0; //Apaga a flag de Interrupção  ● IPR1bits.TMR1IP = 0; // Interrupção de Baixa prioridade |

|  |
| --- |
| 1. **Timer 2**   **Cálculos:**  O sinal sonoro deverá ser atuado com uma intermitência de 0.5 Hz  ● F=0.5 Hz, com DC=50%  ● TH=TL=500/2 = 250ms  ● X =(PRExPOST)  ● Para encontrar X: Definir PR2 como 255  Como X é um valor muito elevado é necessário reduzir o valor de T e acrescentar uma variável count ao programa.  Se por exemplo, T = 0.05  Ou seja, quando a variável count for igual a 40 ativa o buzzer.  ● Encontrar X com T = 0.05s  Logo, X => 112  ●Definir por exemplo:  PrescalerxPoscaler = 112 = (16x7)  Prescaler = 16  Poscaler = 7  **Registos**:  ●PR2 = 225;  ●TMR2 = 0;  //Postscaler  ●T2CONbits.T2OUTPS3 = 0;  ●T2CONbits.T2OUTPS2 = 1;  ●T2CONbits.T2OUTPS1 = 1;  ●T2CONbits.T2OUTPS0 = 0;  //Prescaler  ●T2CONbits.T2CKPS1 = 1;  ●T2CONbits.T2CKPS0 = 1;  ●PIE1bits.TMR2IE =1; //Autoriza a Interrupção  ●PIR1bits.TMR2IF =0; //Apaga a flag de Interrupção  ●IPR1bits.TMR2IP = 0; //Interrupção de Baixa Prioridade  //Inicia a contagem  ●T2CONbits.TMR2ON = 1; |

|  |
| --- |
| 1. **ADC**   **Converter o valor obtido do ADC em tensão do sensor de pressão**  Para obter a tensão do sensor de pressão utilizamos a seguinte formula:  Sendo que VREF = 5 e MAX\_ADC\_VALUE = 1023.  **Converter tensão proveniente do sensor de pressão em um valor correspondente de pressão:**  Depois de ler o datasheet do sensor de pressão MPX4250. Aprendemos que a relação da tensão e do seu valor de pressão tem uma relação linear.  ●Colocamos o sensor de pressão a 0kPa e obtivemos uma tensão de 0.263658V  ●Colocamos o sensor de pressão a 250kPa e obtivemos uma tensão de 4.87084V  ●Logo utilizando a fórmula da reta:  ●Para descobrir o declive da reta utilizamos a seguinte formula:  ●Substituindo os valores obtidos anteriormente:  ●Para calcular o b, utilizamos a seguinte formula:  ●Substituindo os valores obtidos anteriormente: |

|  |
| --- |
| 1. **Eusart**   **Cálculos**:  **Fosc** = 8Mhz, **Baud Rate** = 9600;  **BRG16 = 0 (8 BITS) Velocidade Baixa**    ● Resolvendo para SPBRGH:SPBRG      ● Calcular o Baud Rate    ● Calcular o erro em percentagem  **BRG16 = 0 (8 BITS) Velocidade Alta**    ● Resolvendo para SPBRGH:SPBRG      ● Calcular o Baud Rate    ● Calcular o erro em percentagem  **BRG16 = 1 (16 BITS) Velocidade Baixa**  Os cálculos são iguais ao BRG16 = 0 (8 BITS) Velocidade Alta.  **BRG16 = 1 (16 BITS) Velocidade Alta**    ● Resolvendo para SPBRGH:SPBRG      ● Calcular o Baud Rate    ● Calcular o erro em percentagem  **Registos**:  ● SPBRG = ;  **Configuração do registo TXSTA:**  CSRC=0 -> Não interessa para modo assíncrono  TX9=0 -> Seleção de transmissão a 8 bits de dados  TXEN=1 -> Habilitação do circuito de transmissão  SYNC=0 -> Seleção do modo de comunicação assíncrona  SENDB=0 -> Não emitir condição de Break  BRGH=1 -> Habilitação do modo High-Speed  TRMT =X (don't care) -> Bit só de leitura  TX9D=0 -> Nono bit de dados a transmitir (não usado)  ● TXSTA = 0b00100100;  **Configuração do registo RCSTA:**  SPEN=1 -> Habilitação da porta de comunicação série  RX9=0 -> Seleção de receção a 8 bits de dados  SREN=0 -> Não interessa para modo assíncrono  CREN =1 -> Habilitação do circuito de receção  ADDEN=0 -> Não interessa para modo de receção a 8 bits  FERR=X (don't care) -> Bit só de leitura  OERR=X (don't care) -> Bit só de leitura  RX9D=0 -> Nono bit de dados recebido (não usado)  ● RCSTA = 0b10010000; |

|  |
| --- |
| 1. **SPI**   **Registos:**  **Configuração do registo SSP2STAT:**  bit 7 SMP = 0: SPI Data Input Sample bit  bit 6 CKE = 1: SPI Clock Edge Select bit (SPI mode only)  bit 5 D/A = 0: Data/Address bit (I2C mode only)  bit 4 P = 0: Stop bit  bit 3 S = 0: Start bit  bit 2 R/W = 0: Read/Write bit information (I2C mode only)  bit 1 UA = 0: Update Address bit (10-bit I2C mode only)  bit 0 BF = 0: Buffer Full Status bit  ● SSP2STAT = 0b01000000;  **Configuração do registo SSP2CON1:**  bit 7 WCOL=0: Write Collision Detect bit  bit 6 SSP2OV=0: Receive Overflow Indicator bit(1)  bit 5 SSP2EN=0: Synchronous Serial Port Enable bit  bit 4 CKP=0: Clock Polarity Select bit  bit 3-0 SSP2M=0;  ● SSP2CON1 = 0x00;  **Configuração do registo SSP2ADD:**  ● SSP2ADD = 0x01; |

**Anexo 1.**

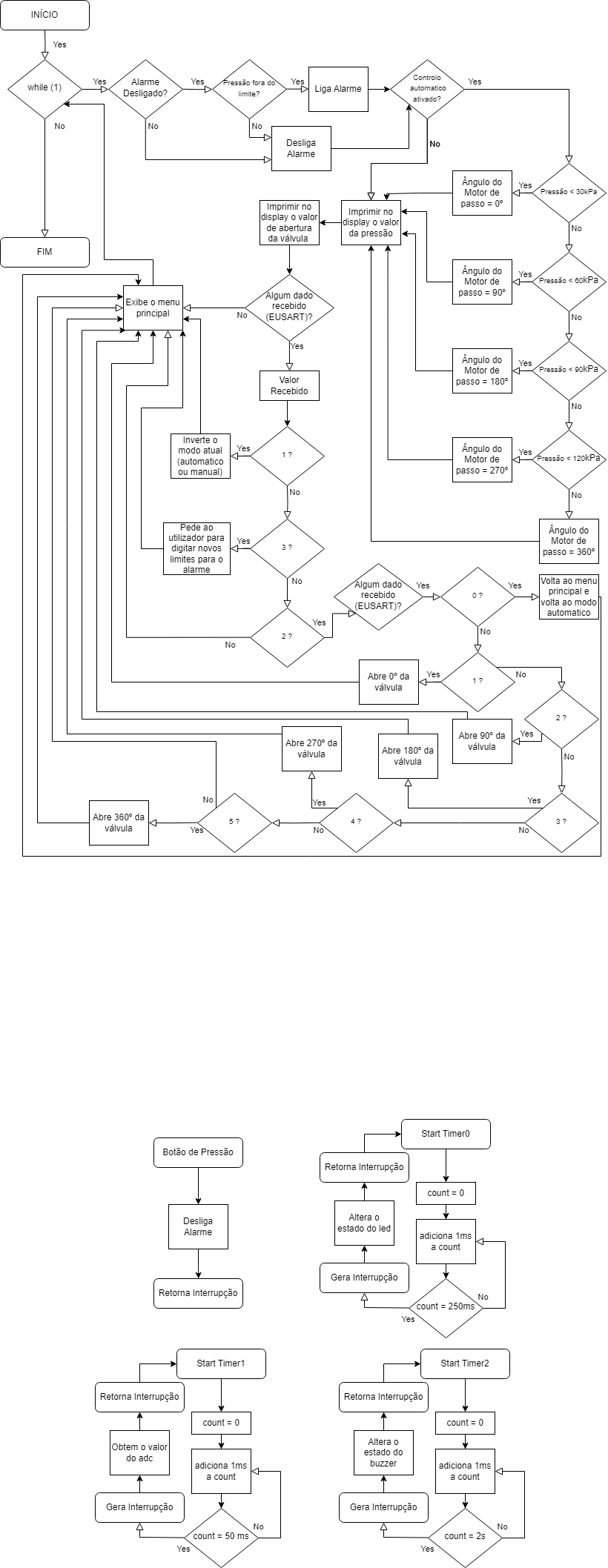


Figura - Fluxograma do projeto

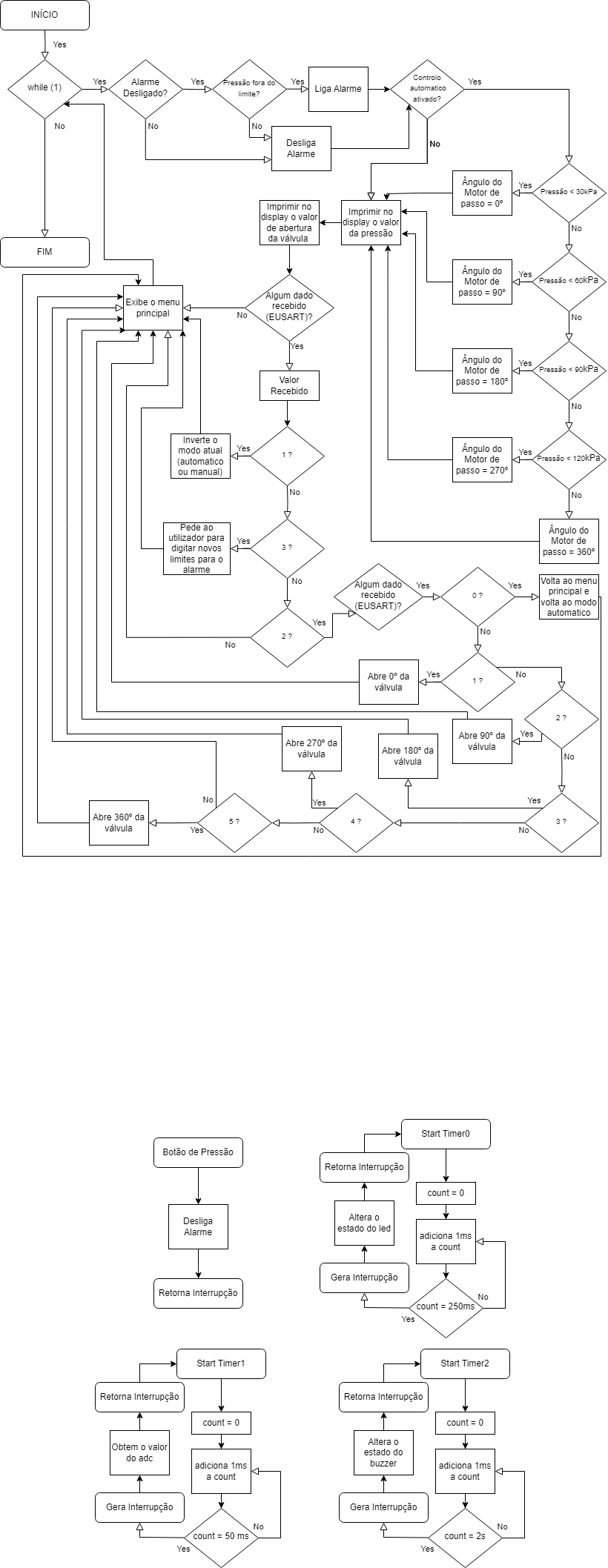
  
**Anexo 2.**

Figura - Fluxograma da interrupção e dos timers