GUIA DO EX de micro 22/06/2020
As anotações em vermelho explicam o que fazer, o que não está em vermelho é o meu relatório em si.

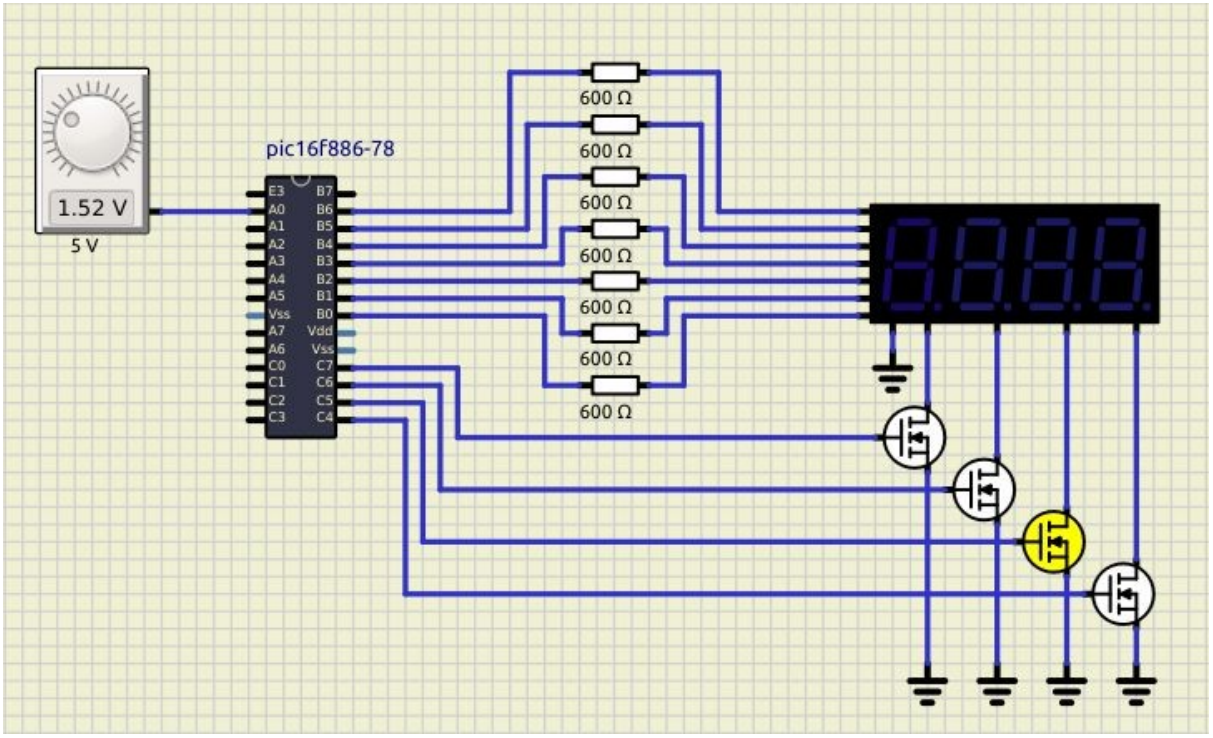
PMR3406 - Microprocessadores - Aula de 22/06/20
Gustavo Marangoni Rubo - 4584080

Considerações gerais:
Nesse exercício você vai precisar montar um circuito no simulIDE, codar o programa no MPLAB e depois escolher um MOSFET, de acordo com alguns requisitos. É de longe o exercício mais demorado até agora (quase um lab).

• Cálculo dos resistores:
Algumas pessoas tiveram resultados diferentes aqui, então não tenho completa certeza.
O Jun mostrou na aula o datasheet do display, e vamos usar o valor de Vf dele aqui.
Resistor vermelho: Vf = 2V (Typ)
If = 5mA
R = (Vss - Vf)/If = (5 - 2)/0,005 = 600Ω

• Cálculo da interrupção do timer 0 (cada 5ms):
Essa parte já fizemos diversas vezes, como no ex da aula 13/04.
Fosc = 20MHz, Tosc = 50ns, Tosc*4 = 0,2µs
Prescaler 1:128 ⇒ 0,2*128 = 25,6µs
5000/25,6 = 195,3125
195

• Circuito desligado:
PODEMOS usar MOSFETs do tipo P ou do tipo N. Mas resumindo, tipo N é melhor aqui, então use isso.
DICA: indo na aba “properties” do simulIDE sem ter algo selecionado, você vai ver a propriedade “animate”. Essa propriedade muda a cor dos fios de acordo com o nível de tensão deles. É muito útil pra visualizar o que está acontecendo, mude pra “true”.
O desenho do circuito é exatamente o que o Jun mostrou na aula, então pode fazer igual.



• Código do MPLAB (arquivo main.c):

O código base aqui é o mesmo de todos os labs, mas você vai ter que comentar (ou remover) algumas linhas que fazem referência a portas que não estamos usando mais. Comente todas as referências ao LCD e debug.

Lista de tarefas a fazer:

- configurar o timer e a interrupção dele (feito no ex de 13/04)
- configurar as saídas do display
- configurar as saídas dos drivers
- configurar a entrada do potenciômetro e o ADC (feito no ex de 27/04)
- ler o resultado do ADC (feito no ex de 27/04)
- codificar os caracteres de 0 a 9 como saídas pro display (ex: 0 -> 0b0111110)
- escrever os caracteres na EEPROM
- ler os caracteres na EEPROM
- fazer a lógica de imprimir o resultado do ADC no display

Essa lista parece intimidadora, e é por que eu dividi as coisas em muitos passos pequenos.

```
// PIC16F886 Configuration Bit Settings

// 'C' source line config statements

// CONFIG1
#pragma config FOSC = EC       // Oscillator Selection bits (EC: I/O function on RA6/OSC2/CLKOUT
pin, CLKIN on RA7/OSC1/CLKIN)
#pragma config WDTE = OFF      // Watchdog Timer Enable bit (WDT disabled and can be enabled by
SWDTEN bit of the WDTCON register)
#pragma config PWRTE = OFF     // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = ON      // RE3/MCLR pin function select bit (RE3/MCLR pin function is MCLR)
#pragma config CP = OFF        // Code Protection bit (Program memory code protection is disabled)
#pragma config CPD = OFF       // Data Code Protection bit (Data memory code protection is
```

```
disabled)
#pragma config BOREN = ON      // Brown Out Reset Selection bits (BOR enabled)
#pragma config IESO = ON       // Internal External Switchover bit (Internal/External Switchover
mode is enabled)
#pragma config FCMEN = ON      // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is
enabled)
#pragma config LVP = OFF       // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV
on MCLR must be used for programming)

// CONFIG2
#pragma config BOR4V = BOR40V   // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)
#pragma config WRT = OFF       // Flash Program Memory Self Write Enable bits (Write protection
off)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>
#include <stdio.h>
#include "always.h"
#include "delay.h"
#include "io.h"
//#include "lcd.h"
//#include "adc.h"
//#include "debug.h"

// Variáveis globais
int valorPot;
char caracteres[10];

// função exponente
int pow (int num, int exp) {
        int res =1;
        for (int i = 0; i < exp; i ++) res *= num;
        return res;
}

// Função para tratamento de interrupções
void interrupt isr(void) {

  // Tratamento da interrupção do Timer 0
  if (T0IE && T0IF) {
        // Executar a cada 5ms
        delay_ms(2);
        GO_DONE =1;
        while(GO_DONE);
        valorPot = (ADRESH <<8) + ADRESL;

        delay_ms(2);

        TMR0 =255 - 195; //interrupções a cada 5ms
        T0IF =0;       //reseta a interrupt flag
  }
}

// Inicialização do Timer 0
void t0_init(void) {
        T0CS =0;   //usar timer0
        PSA =0;//prescaler usa timer0
        //define prescaler em 256:
        OPTION_REGbits.PS = 0b110;
```

```
        TMR0 = 255 - 195; //interrupções a 5ms
        T0IE = 1;   //habilita interrupções do timer0
}

// Inicialização do PortA aqui
void portA_init(void) {
        TRISA0 = 1; // configura como entrada
        ANS0 = 1;   // configura como analógica
        ADCON0bits.ADCS = 0b10; // divisor: Fosc/32
        ADCON0bits.CHS = 0b0000; // seleciona o canal a ser convertido
        VCFG0 = 0; // Vdd = 5V
        VCFG1 = 0; // Vss = 0V
        ADFM = 1; // justificar à direita
        ADON = 1; //ligar o modo conversor
}

// Inicialização do PortB aqui
void portB_init(void) {
        //configurar todas as portas B como saídas
        TRISB = 0b00000000;
}

// Inicialização do PortC aqui
void portC_init(void) {
        //configurar como saídas as portas 4 a 7
        TRISC &= 0b00001111;
}

void vetor_caracteres_init(void){
  for (int i = 0; i < 10; i++)
        caracteres[i] = EEPROM_READ(i);
  caracteres[0] = 0b01111110;
}

// Programa Principal
void main(void) {

  /* Escrevendo dados na EEPROM
   * (só é necessário rodar uma vez)
  EEPROM_WRITE(0, 0b01111110);
  EEPROM_WRITE(1, 0b00110000);
  EEPROM_WRITE(2, 0b01101101);
  EEPROM_WRITE(3, 0b01111001);
  EEPROM_WRITE(4, 0b00110011);
  EEPROM_WRITE(5, 0b01011011);
  EEPROM_WRITE(6, 0b01011111);
  EEPROM_WRITE(7, 0b01110000);
  EEPROM_WRITE(8, 0b01111111);
  EEPROM_WRITE(9, 0b01110011);*/

  // Inicializações
  t0_init();          // inicializa Timer 0
  portA_init();
  portB_init();       // inicializa portB
  portC_init();       // inicializa portB
  vetor_caracteres_init();
  ei();               // macro do XC8, equivale a GIE = 1, habilita interrupções

  // Loop principal (infinito))
  while(1) {
        //escrever os quatro números, da casa menos significantes para mais
```

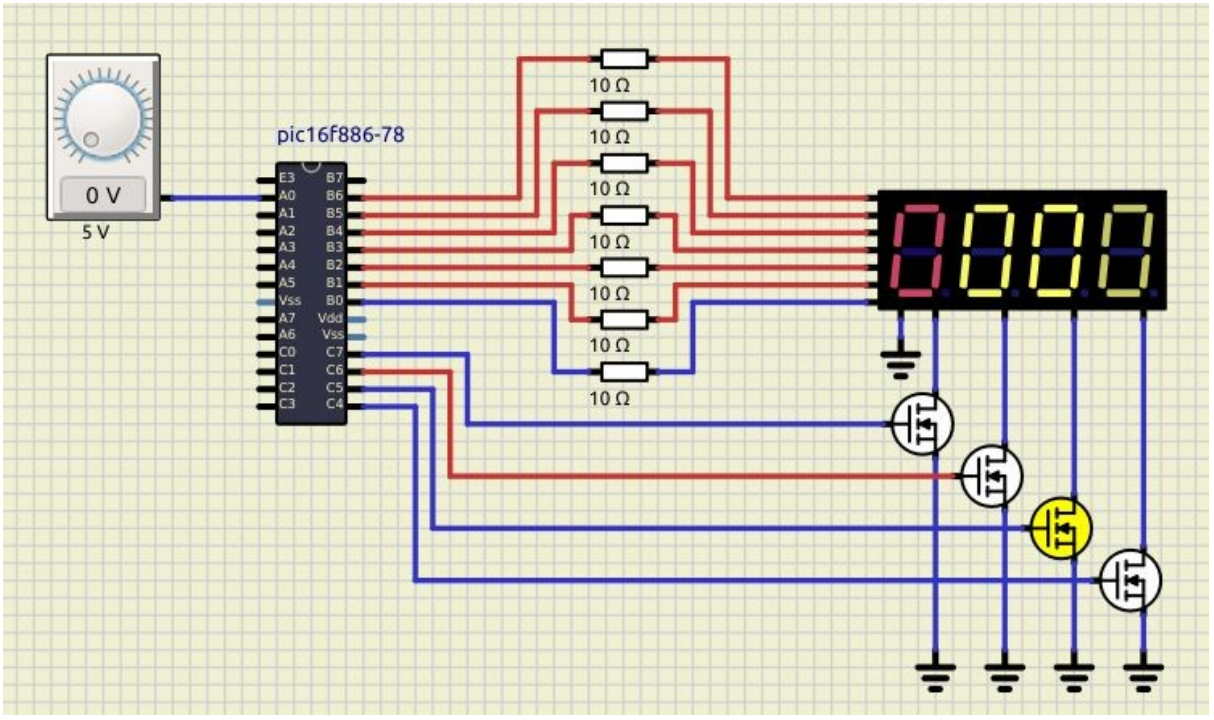
```
        for (int i = 0; i < 4; i++) {
        int num = (int)((valorPot % pow(10, i+1)) / pow(10, i));
        PORTB = caracteres[num];
        PORTC = (1 << (i + 4)) |0b00000000;
        delay_ms(10); // espera de 10 ms -> refresh de 25Hz pro display inteiro
        }
  }
}
```

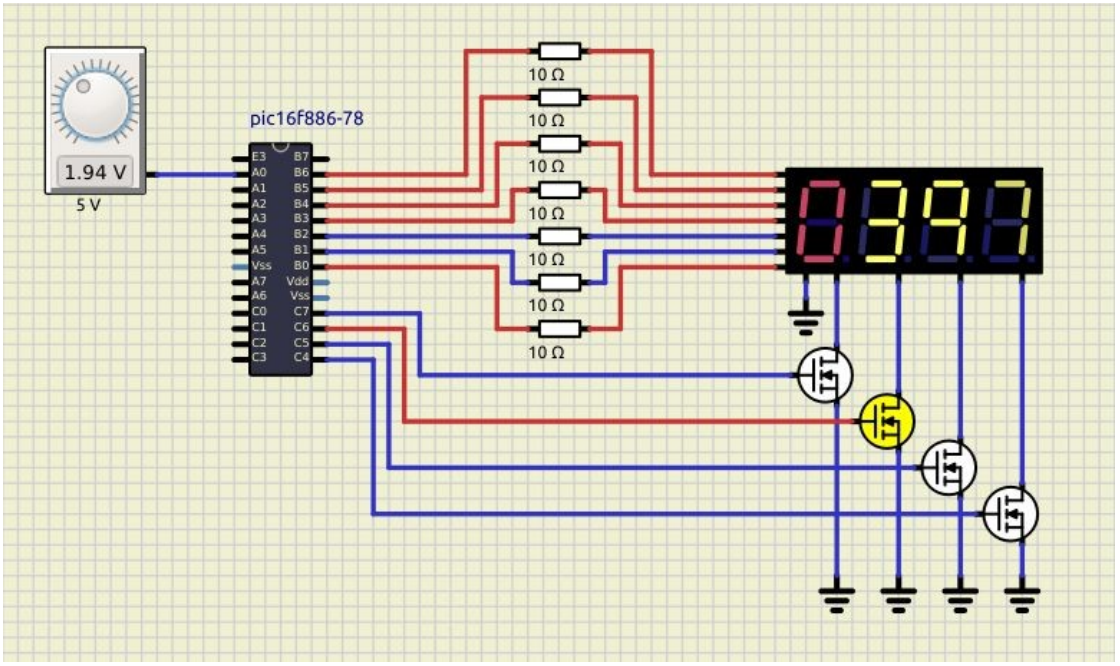
• Circuito em funcionamento:

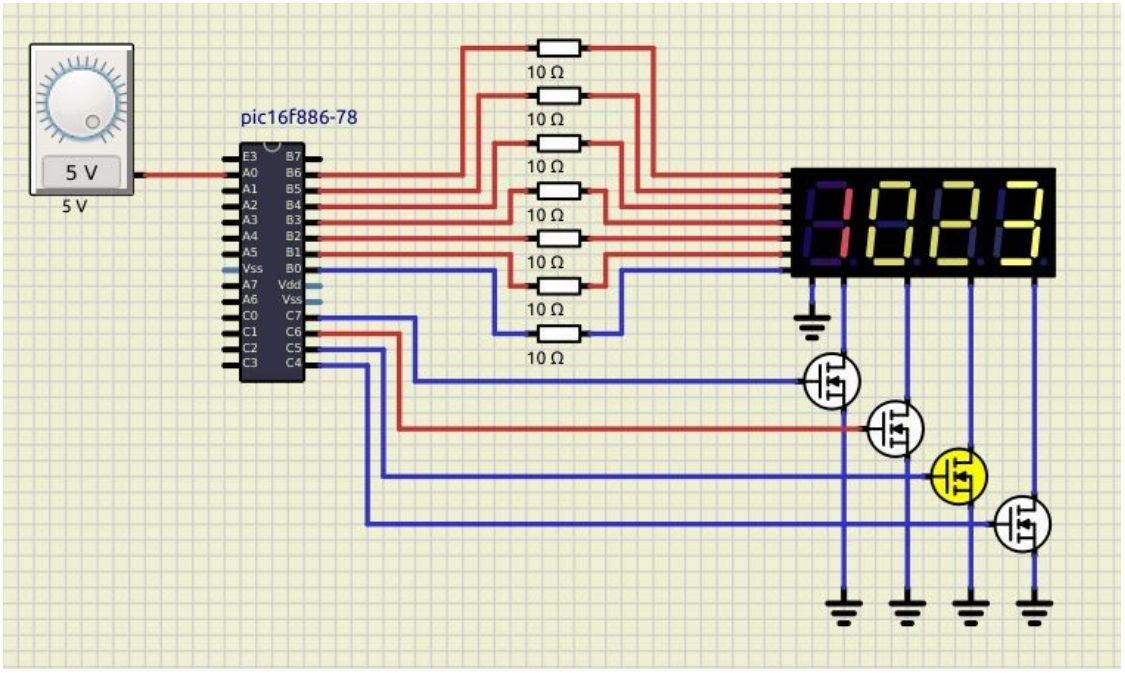
Nada a dizer aqui além do que já escrevi no relatório.

Obs1: o valor dos resistores foi calculado em 600Ω, mas o com esses valores, os LEDs ficam pouco visíveis no simulIDE. Por isso os valores foram mudados para 10, o que está errado e possivelmente danificaria os LEDs na realidade.

Obs2: por algum bug do simulIDE, o display tem três números amarelos. Porém, nas propriedades dele, a cor é vermelha.







• Escolha de MOSFET:

Similar a como foi da outra vez, mas corrigi de acordo com o que o Jun comentou na aula. Dessa vez não nos importamos com Rds(on), por que a dissipação de potência é muito baixa. De novo, use o lcsc.com para procurar MOSFETs.

Escolhemos um display de catodo comum, portanto devemos usar MOSFETs tipo N.

Requisitos:

Id > 45mA

Vgs > 5V

Vds > 5V

Vgs(th) < 2,4V

MOSFET escolhido: **DMN601VK**

DUAL N-CHANNEL ENHANCEMENT MODE FIELD EFFECT TRANSISTOR

Features

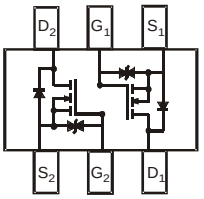
- Dual N-Channel MOSFET
- Low On-Resistance
- Low Gate Threshold Voltage
- Low Input Capacitance
- Fast Switching Speed
- Low Input/Output Leakage
- Ultra-Small Surface Mount Package
- **Totally Lead-Free & Fully RoHS Compliant (Notes 1 & 2)**
- **Halogen and Antimony Free. "Green" Device (Note 3)**
- **An Automotive-Compliant Part is Available Under Separate Datasheet (DMN601VKQ)**

Mechanical Data

- Case: SOT563
- Case Material: Molded Plastic, "Green" Molding Compound; UL Flammability Classification Rating 94V-0
- Moisture Sensitivity: Level 1 per J-STD-020
- Terminal Connections: See Diagram
- Terminals: Finish - Matte Tin Annealed over Copper Leadframe. Solderable per MIL-STD-202, Method 208 e3
- Weight: 0.006 grams (Approximate)

SOT563

TOP VIEW



TOP VIEW
Internal Schematic

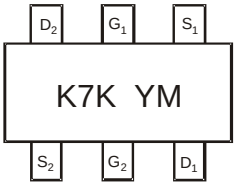
Ordering Information (Note 4)

Part Number	Case	Packaging
DMN601VK-7	SOT563	3,000/Tape & Reel

Notes:
1. No purposely added lead. Fully EU Directive 2002/95/EC (RoHS) & 2011/65/EU (RoHS 2) compliant.
2. See http://www.diodes.com/quality/lead_free.html for more information about Diodes Incorporated's definitions of Halogen- and Antimony-free, "Green" and Lead-free.
3. Halogen- and Antimony-free "Green" products are defined as those which contain <900ppm bromine, <900ppm chlorine (<1500ppm total Br + Cl) and <1000ppm antimony compounds.
4. For packaging details, go to our website at http://www.diodes.com/products/packages.html.

Marking Information

SOT563



K7K = Marking Code
YM = Date Code Marking
Y = Year (ex: D = 2016)
M = Month (ex: 9 = September)

Date Code Key

Month	2005	…	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024
Code	S	…	C	D	E	F	G	H	I	J	K	L

Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Code	1	2	3	4	5	6	7	8	9	O	N	D

Maximum Ratings (@T_A = +25°C, unless otherwise specified.)

Characteristic		Symbol	Value	Unit
Drain-Source Voltage		V_{DSS}	60	V
Gate-Source Voltage		V_{GSS}	±20	V
Drain Current (Note 5)	Continuous Pulsed (Note 6)	I_D	305 800	mA

Thermal Characteristics (@T_A = +25°C, unless otherwise specified.)

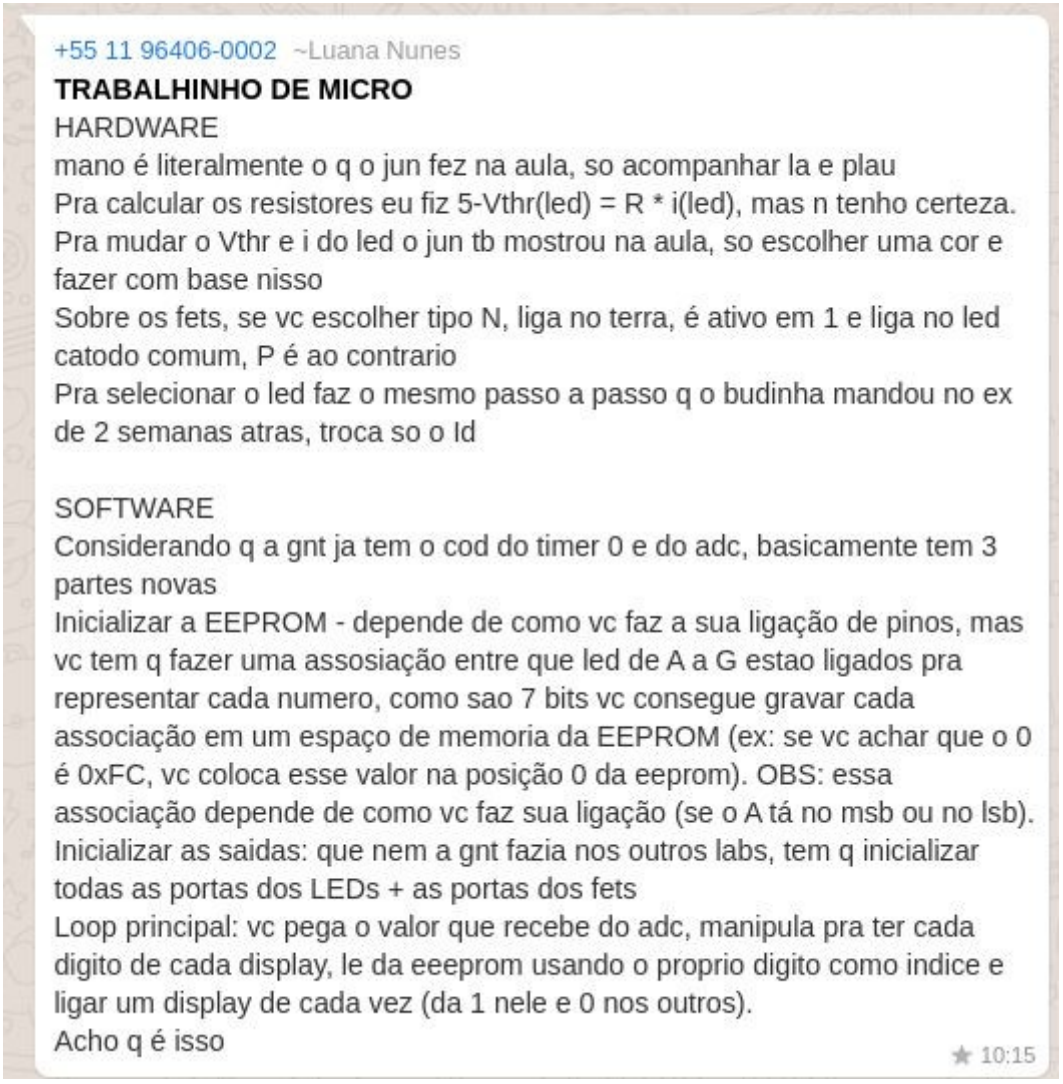
Characteristic	Symbol	Value	Unit
Total Power Dissipation (Note 5)	P_D	250	mW
Thermal Resistance, Junction to Ambient	$R_{\theta JA}$	500	°C/W
Operating and Storage Temperature Range	T_J, T_{STG}	-65 to +150	°C

Electrical Characteristics (@T_A = +25°C unless otherwise specified.)

Characteristic	Symbol	Min	Typ	Max	Unit	Test Condition
OFF CHARACTERISTICS (Note 7)						
Drain-Source Breakdown Voltage	BV_{DSS}	60			V	V_{GS} = 0V, I_D = 10μA
Zero Gate Voltage Drain Current	I_{DSS}			250	nA	V_{DS} = 50V, V_{GS} = 0V
Gate-Source Leakage	I_{GSS}			±500	nA	V_{GS} = ±10V, V_{DS} = 0V
				±100		V_{GS} = ±5V, V_{DS} = 0V
ON CHARACTERISTICS (Note 7)						
Gate Threshold Voltage	$V_{GS(th)}$	1.0	1.6	2.5	V	V_{DS} = V_{GS}, I_D = 250μA
Static Drain-Source On-Resistance	$R_{DS(ON)}$			2.0 3.0	Ω	V_{GS} = 10V, I_D = 0.5A V_{GS} = 4.5V, I_D = 200mA
Forward Transfer Admittance	$\|Y_{fs}\|$		284		ms	V_{DS} =10V, I_D = 0.2A
Diode Forward Voltage (Note 7)	V_{SD}	0.5		1.4	V	V_{GS} = 0V, I_S = 115mA
DYNAMIC CHARACTERISTICS						
Input Capacitance	C_{iss}			50	pF	V_{DS} = 25V, V_{GS} = 0V f = 1.0MHz
Output Capacitance	C_{oss}			25	pF	
Reverse Transfer Capacitance	C_{rss}			5.0	pF	

Notes: 5. Device mounted on FR-4 PCB.
6. Pulse width ≤ 10μs, Duty Cycle ≤ 1%.
7. Short duration pulse test used to minimize self-heating effect.

REFERÊNCIAS:



+55 11 96406-0002 ~Luana Nunes

TRABALHINHO DE MICRO

HARDWARE

mano é literalmente o q o jun fez na aula, so acompanhar la e plau

Pra calcular os resistores eu fiz 5-Vthr(led) = R * i(led), mas n tenho certeza.

Pra mudar o Vthr e i do led o jun tb mostrou na aula, so escolher uma cor e fazer com base nisso

Sobre os fets, se vc escolher tipo N, liga no terra, é ativo em 1 e liga no led catodo comum, P é ao contrario

Pra selecionar o led faz o mesmo passo a passo q o budinha mandou no ex de 2 semanas atras, troca so o Id

SOFTWARE

Considerando q a gnt ja tem o cod do timer 0 e do adc, basicamente tem 3 partes novas

Inicializar a EEPROM - depende de como vc faz a sua ligação de pinos, mas vc tem q fazer uma assosiação entre que led de A a G estao ligados pra representar cada numero, como sao 7 bits vc consegue gravar cada associação em um espaço de memoria da EEPROM (ex: se vc achar que o 0 é 0xFC, vc coloca esse valor na posição 0 da eeprom). OBS: essa associação depende de como vc faz sua ligação (se o A tá no msb ou no lsb).

Inicializar as saidas: que nem a gnt fazia nos outros labs, tem q inicializar todas as portas dos LEDs + as portas dos fets

Loop principal: vc pega o valor que recebe do adc, manipula pra ter cada digito de cada display, le da eeeprom usando o proprio digito como indice e ligar um display de cada vez (da 1 nele e 0 nos outros).

Acho q é isso

★ 10:15

Xu

> +55 11 96406-0002 ~Luana Nunes
> **TRABALHINHO DE MICRO**
> HARDWARE
> mano é literalmente o q o jun fez na aula, so acompanhar la e plau…

se alguém fizer e estiver tendo problemas na eeprom, e resovler, me avisa pfv kkkkkkk o meu n ta lendo

10:25

EXERCÍCIO DE MICRO
Se vc estiver com problemas pra ler valores da EEPROM, ou teu display funciona quando vc seta um valor "na mão" mas não funciona quando pega carregando um valor da EEPROM, me chama. Eu e o @Pedro estávamos com esse problema mas descobrimos uma famosa gambiarra que da bom. Esse problema é só por conta do SimulIDE, de verdade não ocorreria, e parece q n ocorre no RealPIC tbm

12:18

Xu

> Xu
> **EXERCÍCIO DE MICRO**
> Se vc estiver com problemas pra ler valores da EEPROM, ou teu display funciona quando vc seta um valor "na mão" mas não funciona quando pega carregando u…

É q acho q é muito trampo pra explicar aqui no grupo se ngm ta tendo problema kkk mas primeiro de tudo
-Nao salve valores na PRIMEIRA posição da EEPROM, seta como 255 e usa sempre pra sempre, entao leia assim:
EEPROM_READ(++valor);
-Tendo esse problema, clique com a direita no PIC e clica em Save EEPROM data, modifique na mão num editor de texto os valores q vc quer q fique salvo depois clique em LOAD EEPROM data desses valores
-Deve funcionar, se n funcionar fecha e abre e tenta de novo
Se **mesmo assim** não funcionar (meu caso), apaga teu PIC, carrega outro e já da LOAD EEPROM data

12:21