Universidade de São Paulo-USP
Escola de Engenharia de São Carlos-EESC
SEL0629 - Aplicações de Microprocessadores I
Relatório Prática III

Vinicius William da Silva - Número USP: 11233842
Prof. Marcelo Andrade da Costa Vieira

São Carlos, 2023

1 Introdução

Nessa prática será estudado o funcionamento do PWN para controlar a velocidade da ventoinha da placa Kit Pic Genios. Após isso, é feito o cálculo da velocidade em RPM e exibido no *display* LCD.

O objetivo dessa prática é primeiramente configurar o *duty cycle* do PWN, utilizar 5 botões para alterar a velocidade da ventoinha e exibir o valor no *display* LCD. Para isso, foi utilizado a própria biblioteca do PWN do Easy Pic V7. A segunda parte da prática consiste em conectar o Kit Pic Genios e utilizar o sensor de infravermelho para calcular a velocidade de giro da ventoinha da placa e enviar esse dado para o Easy Pic.

2 Cálculos

Para o cálculo do RPM da ventoinha utilizamos tanto o TIMER1 como contador, quanto o TIMER0 para o gerar interrupções em intervalos de 1 s. O cálculo realizado para o último está disposto a seguir.

$$T = (2^n - TMR) \cdot CM \cdot PS)$$

Para o valor de tempo de interrupção do TIMER ser de aproximadamente 1 segundo optou-se pelos valores:

$$T = (2^{16} - 3000) \cdot (4/(8 \cdot 10^6)) \cdot 32$$

$$T = 1,000576s$$

Convertendo 3000 para hexadecimal temos o valor OX0BB8, assim atribuímos ao *timer* TMR0H o valor 0X0B e ao TMR0L o valor 0XB8.

Como é recebido apenas a frequência no TIMER1 é necessário realizar sua conversão para RPM. A ventoinha utilizada possui 7 hélices, isso significa que uma frequência de 7 Hz na verdade indica apenas uma rotação por segundo.

Além disso, é preciso multiplicar o valor por 60 para obter o resultado por minutos.

$$RPM = \frac{TMR1 \cdot 60}{7}$$

3 Código de Simulação

```
#define TH_0 0X0B // 1 segundo - prescale 32
#define TL_0 0XB8
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// End LCD module connections

int one_seg_flag = 0; //flag para atualizar dados
int TH = 0;
int TL = 0;

// Interrupcao do TIMER 0
void TIMER0() iv 0x0008 ics ICS_AUTO {
```

```
    TMR0H = TH_0; // Valor recarregado no TIMER 0
    TMR0L = TL_0;

    TH = TMR1H; //Atualiza variaveis globais
    TL = TMR1L;

    TMR1H = 0; //Reseta contador
    TMR1L = 0;

    one_seg_flag = 1; //Atualiza flag de 1 seg

    INTCON.f2 = 0; // Limpa a flag de interrupcaoo
}

void main() {
    char duty[20] = "Duty␣Cycle:␣";
    char value_string[8] = "␣0␣%";
    char rotation_string[10];
    float duty_value;
    int total;

    anselb = 0; //Portas digitais
    anselc = 0;
    anseld = 0;

    trisb = 0; //Output
    trisc = 0b00000001; //Apenas RC0 (Timer1) como input
    trisd = 255; //Input

    PWM1_Init(5000); // Inicializa PWM1 em 5KHz
    PWM1_Start(); // Inicia PWM1
```

```
    PWM1_Set_Duty(0); // Default 0

    Lcd_Init(); // Inicializa LCD
    Lcd_Cmd(_LCD_CURSOR_OFF); // Cursor off
    Lcd_Cmd(_LCD_CLEAR); // Limpa Display

    T0CON = 0b10000100; // Inicia o TIMER 0, com prescale 32

    INTCON = 0b10100000; // Interrupcao

    TMR0H = TH_0; // Valor carregado no TIMER 0
    TMR0L = TL_0;

    T1CON = 0b10000101; //Contador no timer 1
    T1GCON = 0;
    TMR1H = 0;
    TMR1L = 0;

    while(1){

        if(portd.rd0 == 0){
            PWM1_Set_Duty(0); // velocidade em 0%
            sprintf(value_string,"␣0␣%c",'%') ;
        }
        if(portd.rd1 == 0){
            PWM1_Set_Duty(63); // velocidade em 25%
            sprintf(value_string,"25␣%c",'%') ;
        }
        if(portd.rd2 == 0){
            PWM1_Set_Duty(127); // velocidade em 50%
            sprintf(value_string,"50␣%c",'%') ;
```

```
        }
        if(portd.rd3 == 0){
            PWM1_Set_Duty(192); // velocidade em 75%
            sprintf(value_string,"75␣%c",'%') ;
        }
        if(portd.rd4 == 0){
            PWM1_Set_Duty(255); // velocidade em 100%
            sprintf(value_string,"100%c",'%') ;
        }
        if(one_seg_flag){ //Atualiza RPM
            //Le valores
            total = TH;
            total = total << 8;
            total += TL;
            total = total * 60 / 7; //RPM
            one_seg_flag = 0;
            Lcd_Cmd(_LCD_CLEAR); //Limpa display
        }
        //Exibe no display
        sprintf(rotation_string,"RPM:␣%d",total);
        Lcd_Out(2,1,rotation_string);
        Lcd_Out(1,1,duty);
        Lcd_Out(1,13,value_string);
    }
}
```

4 Resultados e Discussão

Após carregar o código do programa no PIC18F45k22, o *display* LCD exibe na primeira linha o valor atual do *duty cycle*, e na segunda, a velocidade de giro da ventoinha em RPM. Por meio dos botões RD0 a RD4

controla-se o valor do *duty cycle* e dessa maneira é alterada a velocidade de giro da ventoinha de acordo com o botão pressionado.

Para a validação dos resultados foi utilizado um osciloscópio para medir a frequência média da onda emitida pelo infravermelho. A medida foi realizada usando uma ponta do osciloscópio no terra e a outra no pino RC0.

As figuras a seguir mostram a leitura realizada mostrada no LCD para PWM em 25%, 50%, 75% e 100%.

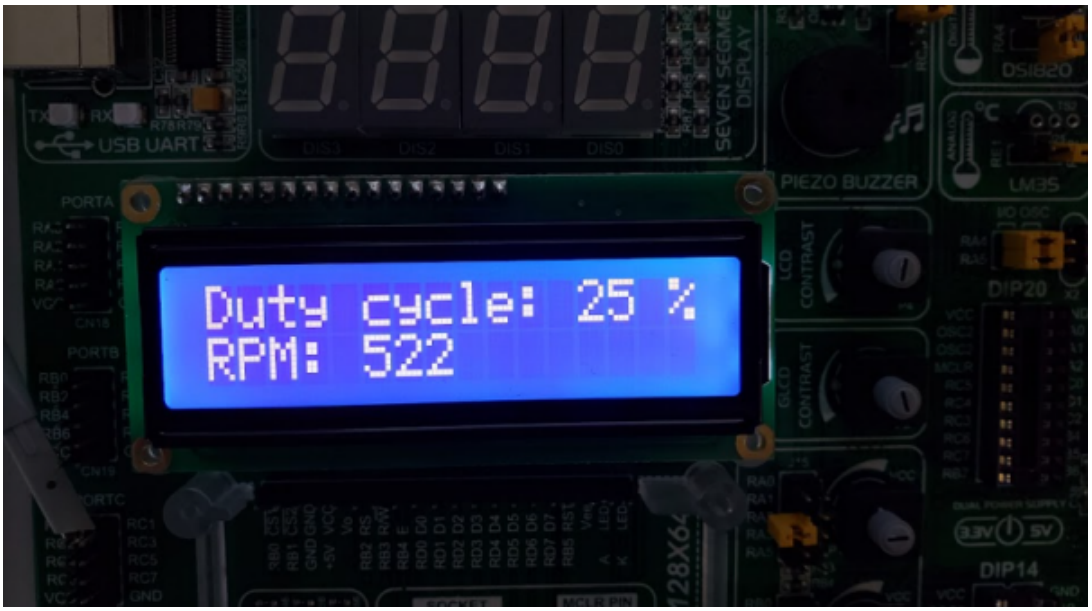

Figura 1: RPM apresentado para PWM 25%.

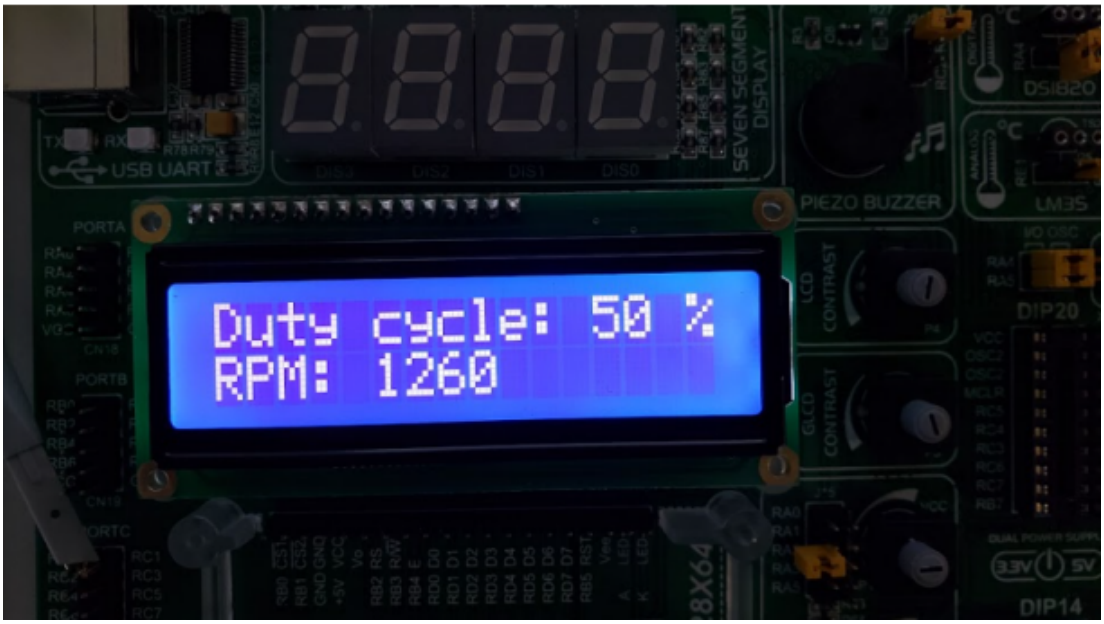

Figura 2: RPM apresentado para PWM 50%.

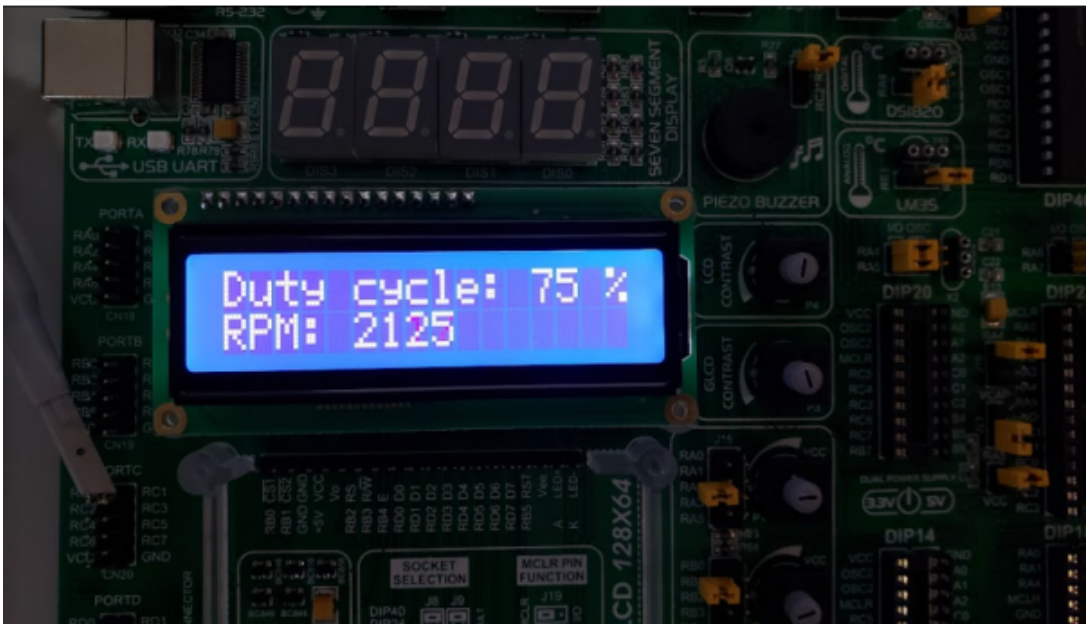

Figura 3: RPM apresentado para PWM 75%

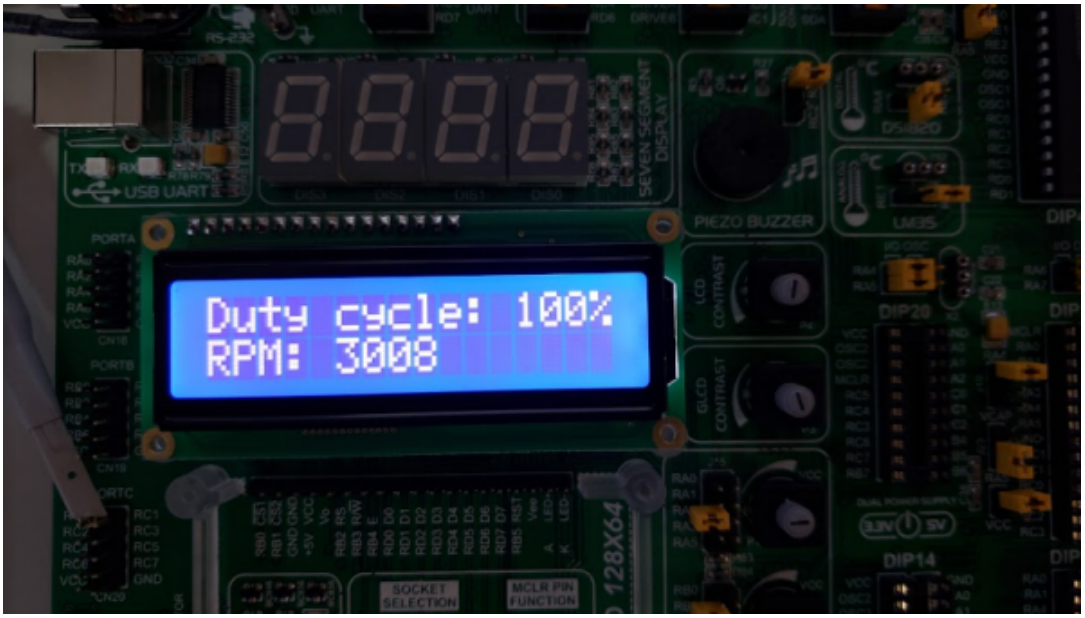

Figura 4: RPM apresentado para PWM 100%

A **tabela 1** mostra a comparação entre os valores esperados e obtidos, além do erro e desvio percentual.

Tabela 1: Comparação do valor ideal e obtido para rotação da ventoinha.

PWM	Osciloscópio (Hz)	RPM ideal Osciloscópio (rpm)	RPM LCD (rpm)	Desvio Percentual (%)
25	61,52	527	522	1,01
50	147,57	1265	1259	0,40
75	248,79	1265	1259	0,33
100	350,88	3008	3010	0,06

Após a análise da tabela 1, pode-se observar que o resultado foi de acordo com o esperado, tendo uma variação dos valores esperados e obtidos próximo de zero. Sendo assim, a leitura da frequência de giro da ventoinha e conversão para RPM foi bem sucedida.

5 Conclusão

Pode-se concluir que a prática foi um sucesso, dada a precisão que foi possível de se obter. Foi possível compreender o uso do PWN por meio do uso de sua biblioteca e alterar o *duty cycle* para o valor desejado. Além disso, houve o conhecimento de como conectar o Kit PIC Genios e por meio do sensor infravermelho realizar a leitura de frequência da ventoinha.

Os pequenos erros observados provavelmente deram-se por conta da própria limitação dos instrumentos de medida.