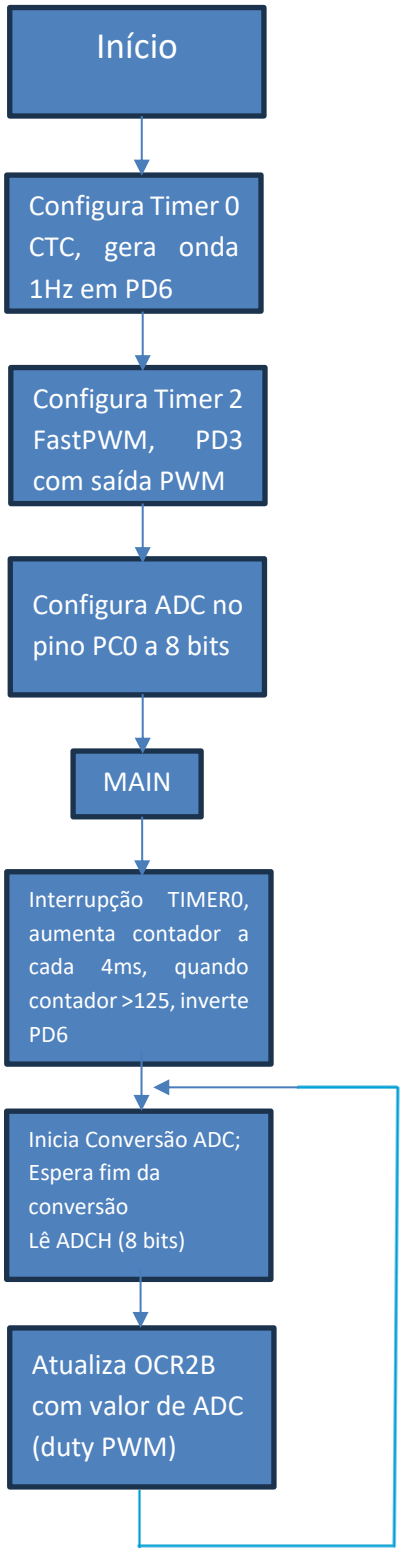
Sistema Mínimo

Rafael Afonso, 1230531@isep.ipp.pt

Índice

FLUXOGRAMA


Início
Configura Timer 0 CTC, gera onda 1Hz em PD6
Configura Timer 2 FastPWM, PD3 com saída PWM
Configura ADC no pino PC0 a 8 bits
MAIN
Interrupção TIMER0, aumenta contador a cada 4ms, quando contador >125, inverte PD6
Inicia Conversão ADC;
Espera fim da conversão
Lê ADCH (8 bits)
Atualiza OCR2B com valor de ADC (duty PWM)

CÓDIGO

```
#include <avr/io.h>
#include <avr/interrupt.h>

volatile uint8_t contador = 0;
uint8_t valor_adc = 0;

ISR(TIMER0_COMPA_vect) {
    contador++;
    if (contador >= 125) { // 125 * 4ms = 0.5s
        PORTD ^= (1 << PD6);
        contador = 0;
    }
}

uint8_t adc_init(){
    ADMUX = (1 << REFS0) | (1 << ADLAR); // AVcc referência, 8 bits, seleciona ADC0
    ADCSRA |= (1 << ADEN) | (1 << ADSC);// habilita e começa a conversão
    while (ADCSRA & (1 << ADSC));// espera que ADSC fique a 0
    return ADCH; //lê os 8 bits mais significativos
}

void onda1Hz_init(void) {
    DDRD |= (1 << PD6);
    PORTD &= ~(1 << PD6);

    // Timer0 CTC mode
    TCCR0A = (1 << WGM01);   // CTC
    TCCR0B = (1 << CS02) | (1 << CS00); // prescaler 1024

    OCR0A = 62;
    TIMSK0 = (1 << OCIE0A); // liga interrupção
    sei(); // enable global
}
// Inicializa PWM
void pwm_init(void) {
    DDRD |= (1 << PD3);                    // PD3 como saída (Arduino pin 3)
    TCCR2A |= (1 << COM2B1) | (1 << WGM21) | (1 << WGM20); //COM2B1 para OC2B no PD3 (Fast
PWM)
    TCCR2B |= (1 << CS21);                 // prescaler 8
    OCR2B = 128;                           //duty cycle inicial 50% on OCR2B

void inic(void){
    DDRC &= ~(1 << PC0); // ADC entrada
    onda1Hz_init();
    pwm_init();
}
```

```
int main(void) {

inic();
    while (1) {
        valor_adc = adc_init();
        OCR2B = valor_adc;  //PWM OCR2B (PD3)
    }
}
```

Timers e ADC

.Timer0

Este Timer de 8 bits foi usado para gerar uma onda quadrada com 1Hz de frequência.
O Timer foi configurado para operar no modo CTC, o que faz com que o contador (TCNT0) volte a zero quando atinge o valor guardado no OCR0A.

$$f_{OCnx} = \frac{f_{clk\_I/O}}{2 \times N \times (1 + OCRnx)}$$

Sabendo que a frequência do clock é 16Mhz e procuramos 1Hz (1 ciclo por segundo), uma hipótese é configurar o período do OC0A para 25ms e registar 20 comparações.
$25 \times 10^{-3} = \frac{Prescaler}{16 \times 10^6} \times (OCR0 + 1)$, com o prescaler a 1024, o OCRO dá 389. Este valor não pode ser usado pois não cabe em 8 bits.
Assim, em vez de calcular o período para 25ms e registar 20 comparações, calcula-se para 4ms e registam-se 125 comparações. Isto dá um valor de OCR0 de 62.
Juntamente com o bit OCIE0A, do registo TIMSK0, e com o bit-I do registo de estados, que quando estão a 1 possibilitam uma interrupção a cada comparação, entre o OCR0A e o TCNT0, é possível agora ter um temporizador de 1Hz.

.Timer2

Este Timer também de 8 bits tem como objetivo controlar uma saída através de PWM.
O Timer foi configurado para Fast PWM, onde o contador incrementa até 255 e volta a 0.
Como estamos a controlar o brilho de um led, o ideal é usar frequências elevadas (>5Khz).
Foi usado o PD3 com o OCR2B para libertar o pino PB3, onde liga o MOSI do AVR-ISP-6.

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \times 256}$$

Atribuí-se o valor de 8 ao prescaler, ficando assim uma frequência de 7.8Khz associada ao Timer. Como o valor máximo do OCR2B é 255 o duty cycle é 100% neste ponto.

Assim, quando lemos o ADCH (8 bits) proveniente do potenciómetro e atribuímos este valor ao OCR2B, controlamos por PWM o duty cycle do led.

.ADC0

Escolheu -se o canal 0 do ADC, que fica no pino 14 do porto D, que lê a tensão de 0-5V e a converte para um valor digital de 10 bits (0-1024). Como só queremos 8 bits, põe-se a 1 o bit ADLAR no registo ADMUX, que apenas nos deixa com os 8 bits mais significativos.
Ao começar a conversão o bit ADSC do registo ADCSRA fica a 1 e retorna a 0 quando esta acaba, assim a condição para lermos o resultado de uma conversão é esperar o 0 neste bit e retornar o valor de ADCH (são pretendidos apenas 8 bits).

ESQUEMA ELÉTRICO

Neste projeto foi adotado o Arduino Uno para facilitar as ligações físicas do hardware. O esquema elétrico a seguir representa apenas o AtMega328P sem o resto da placa de desenvolvimento com todas as ligações necessárias para que este funcionasse sem a mesma, juntamente com a montagem do sistema mínimo deste projeto.