UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
DEPARTAMENTO DE COMPUTAÇÃO E AUTOMAÇÃO
COMPONENTE: SISTEMAS DIGITAIS

Marco Antonio Moreira Carujo
Edmilson Dias Tavares Filho

Natal / RN

Abril de 2017

Sumário **2**

INTRODUÇÃO

O projeto é a realização de um secador industrial de grãos, onde nele tivemos que utilizar um Arduíno com o micro controlador ATMEGA328p, uma FAN simples de computador, e sensores de luz e temperatura.

O funcionamento do secador industrial será dado pela seguinte forma: ele deve ser ligado por uma chave e funcionará por três minutos, durante esse tempo a FAN (secador) ligará e se comportará de acordo com o tempo e com as informações dos sensores de temperatura e luz.

O micro controlador terá a missão de fazer o controle do nosso secador, ele irá ler os dois sinais analógicos dos dois sensores e um sinal digital, que será nossa chave para ligar o sistema; irá mandar quatro sinais PWM: o primeiro para controlar a velocidade do secador, o segundo para controlar a intensidade do LED de acordo com a velocidade do secador, o terceiro para controlar a intensidade de mais um LED proporcionalmente a informação lida no sensor de luz , e o quarto para controlar outro LED através da informação lida no sensor de temperatura; por fim um sinal digital direcionado a um LED que deve indicar a ativação do nosso sistema. Além disso, o sistema foi programado para ser desligado ao fim do processo de secagem, determinado em 3 minutos.

CONFIGURAÇÃO

Na imagem abaixo vamos ver configurações com comentários sobre descrições das linhas de código em C para o micro controlador.

```
DDRB = 0b00001110; // Configurando B1 e B2 e B3 como saida(PWM)
DDRD = 0b10001000; // Configurando D3(PWM) e D7(Digital)
TCCR2A = 0b10100011; // Configurando registrador do PWM B1 e B2
TCCR2B = 0b00000001; // Configurando registrador do PWM B1 e B2
TCCR1A = 0b10100011; // Configurando registrador do PWM B3 e D3
TCCR1B = 0b00000001; // Configurando registrador do PWM B3 e D3
TCCR0A = 0b00000000; // configurando o contador para overflow
TCCR0B = 0b00000101; // configurando o contador para overflow
ADMUX = 0b01000000; // valor de referência
ADCSRA = 0b10000111; // habilita o leitor do analogico e o escalonamento
TIMSK0 = 0b00000001; // Habilita a configuração de Overflow
```

Figura 1 - Configurações de Portas e Contadores

Nas duas primeiras linhas fazemos a porta B3 como a saída para o LED que é proporcional ao sensor de temperatura, B2 como saída para o acoplador óptico, posteriormente transistor fazendo o chaveamento de corrente na FAN, B1 como a saída para o LED que será proporcional a tensão na FAN, D7 como saída para um LED que indicará se o sistema está ligado e D3 como saída para um LED que será proporcional ao sensor de luminosidade.

Para os registradores TCCR2 e TCCR1 fizemos as mesmas configurações, as saídas PWM como não inversores (COM0A1 e COM0B1 com valor 1) e selecionando o modo de operação para "Fast PWM" (WGM02, WGM01 e WGM00 com valor 1).

Já para o registrador TCCR0 estaremos utilizando ele para controlar o tempo de ação do nosso secador , que é de três minutos, sendo assim vamos utilizar as interrupções para fazer a nossa lógica, dividimos o 'Clock' de 16MHz por 1024 o que é equivalente a 15.625 KHz, esse é o 'Clock' do contador de 8 bits que contará até 255 para podermos ter uma interrupção, o que torna nosso clock de interrupção equivalente a 61,27Hz. Com esse valor fizemos nossa lógica para que seja cumprido os três minutos da curva.

Seguindo no código, o registrador ADMUX tem uma missão simples de colocar o valor de referência para a conversão analógica para digital como reservada, equivalente a 5V do micro controlador. O TIMSK0 apenas como descrito habilita ocorrência da interrupção quando houver um overflow no contador do TCCR0.

LEITURA ANALOGICA

A leitura do analógico será feita usando os registradores ADMUX e ADCSRA, e após o termino da leitura, o registrador ADC de 8 bits terá a informação desejada.

```
ADMUX = 0b10000000; //escolher porta a ser usada no caso A0
ADCSRA |= 0b01000000; // inicializar a leitura analogico
while (!(ADCSRA & 0b00010000)); //espera por interupção que indica final de leitura
SL = ADC;

ADMUX = 0b10000001; //escolher porta a ser usada no caso A1
ADCSRA |= 0b01000000; // inicializar a leitura analogico
while (!(ADCSRA & 0b00010000)); //espera por interupção que indica final de leitura
ST = ADC;
```

Figura 2 - Leitura Analógica

Com o ADMUX configurado com o valor de referência 5V, mas o diferencial no momento está nas configurações de entrada do ADMUX, onde MUX3, MUX2, MUX1 e MUX0 estão com o valor 0, fazendo que seja leito a porta ADCS0 e armazena na variável SL (sensor de luz).

Com o ADMUX sendo novamente configurado com o valor de referência 5V, mas o diferencial no momento está nas configurações de entrada do ADMUX, onde MUX3, MUX2, MUX1 recebem 0 e MUX0 está com o valor 1, fazendo que seja leito a porta ADCS1e armazena na variável ST (sensor de temperatura).

CURVA DE SECAGEM

Como falado na introdução a nossa curva de secagem é de 3 minutos, se olharmos que a frequência de overflow é aproximadamente de 61 Hz, os nossos cálculos apontam que 10980 overflows será o suficiente para que tenhamos os 3 minutos de secagem.

```
ISR(TIMER0_OVF_vect) {
  overflow_count = overflow_count + 1; // 1 overflow é equivalente a 16ms
  if (overflow_count < 1830 ) {
    pwm = floor(overflow_count * (307.0 / 1830.0)) + 150.0;
  }
  else if (overflow_count >= 1830  && overflow_count < 3660 ) {
    pwm = 307.0 + 150;
  }
  else if (overflow_count >= 3660 && overflow_count < 5490 ) {
    pwm = floor(((460.0 / 1830.0) * overflow_count) - 613.0) + 150.0;
  }
  else if (overflow_count >= 5490 && overflow_count < 7320 ) {
    pwm = 767.0 + 150.0;
  }
  else if (overflow_count >= 7320 && overflow_count < (10980) ) {
    pwm = floor(((-767.0 / 3570.0) * overflow_count) + 2340.0) + 150.0;
  }
  else {
    chave = false;
    overflow_count = 0;
  }
}
```

Figura 3 - Função de Overflow

Como a interrupção foi ativada com o registrador TIMSK0, então a cada nova interrupção a nossa função ISR é chamada e dentro dela está a lógica da nossa curva. O 'overflow_count' é nossa variável que estará contando as interrupções baseando-se no valor dela, e desta forma, vamos colocando o valor do PWM da FAN conforme os trechos da curva de secagem.

SAÍDAS

As saídas do micro controlador são quatro: OCR1A (PWM da FAN), OCR1B (PWM do LED de temperatura), OCR2A (PWM do LED de luz), OCR2B (PWM do LED equivalente a FAN).

```
OCR1A = pwm - (SL / 10.0) - (ST / 10.0); // FAN
OCR1B = ST; // LED do Sensor de Temperatura
OCR2A = SL; // LED do Sensor de Luz
OCR2B = pwm - (SL / 10.0) - (ST / 10.0); // Led da FAN
```

Figura 4 - Saídas PWM

Dentro da lógica do secador industrial, a velocidade da FAN será inversamente proporcional a luz e a temperatura, então com valores maiores de ST (sensor de temperatura) e SL (sensor de luz), o valor da variável ‘pwm’ é reduzido, controlando a velocidade da FAN.

```
PORTD = (1 << PD7);
```

Figura 5 - Saída digital

A saída digital ela é utilizada para ligar o LED que informa quando o sistema está ligado e funcionando, dentro da lógica do micro controlador a porta recebe o valor 1 quando a ‘chave’ que faz o sistema funcionar for ‘true’, e recebe o valor 0 quando a ‘chave’ for ‘false’.

COMUNICAÇÃO

A comunicação do nosso micro controlador com software foi feita através do USART aplicado no USB.

```
unsigned char USART_receive(void) {

  while (!(UCSR0A & (1 << RXC0)));
  return UDR0;

}

void USART_send( unsigned char data) {

  while (!(UCSR0A & (1 << UDRE0)));
  UDR0 = data;

}

void USART_putstring(char* StringPtr) {

  while (*StringPtr != 0x00) {
    USART_send(*StringPtr);
    StringPtr++;
  }
```

Figura 6 - Funções do USART

Utilizando o registrador UDR0, que é utilizado tanto para envio quanto para recebimento, o que caracteriza uma comunicação do tipo 'half-duplex', ou seja, a informação percorre um único sentido por vez. Temos ainda uma verificação de disponibilidade do registrador, através dos registradores UDRE0 e RXC0 respectivamente.

Já na parte do software teve que ser feito um “protocolo” em Java, pois a comunicação envia uma *String* e cabe ao protocolo feito no software tratar essa *String.*

```
char aux[] = "";
itoa(pwm, aux, 10);
USART_putstring(aux);
USART_putstring(",");

char aux2[] = "";
itoa(SL, aux2, 10);
USART_putstring(aux2);
USART_putstring(",");

char aux3[] = "";
itoa(ST, aux3, 10);
USART_putstring(aux3);
USART_putstring(",");

USART_putstring("1");
USART_putstring("\n");
```

Figura 7 - Formação da String para USART

Como vemos na formação a *String* a ser enviada tem o padrão de “A, B, C, D\n” onde A é o informação do PWM, B é a informação do sensor de luz, C é a informação do sensor de temperatura e D é a informação do sistema, sendo 1 para ligado e 0 para desligado. O ‘\n’ é o caractere final de *String*, causando um quebra linha, quando no software será determinante para informar o fim do dado.

CONCLUSÃO

Com o projeto terminado e funcionando, podemos aprender as funcionalidades do micro controlador e suas ferramentas para utilizar o ATMEGA328p como controlador de um processo de secagem industrial. Conseguimos ainda realizar o monitoramento via software dos parâmetros do sistema pela comunicação USART.