Relatório Projeto Final – Robô Palletizer

Tony Ferreira[1], Vítor Babo[2]

[1] FEUP/3MIEEC_A2, up201706545@fe.up.pt
[2] FEUP/3MIEEC_A2, up201704781@fe.up.pt

Introdução

No âmbito da unidade curricular Sistemas Baseados em Microprocessadores, o projeto final escolhido consistiu na elaboração de um robô palletizer.

O principal objetivo deste projeto passa por consolidar os conhecimentos abordados durante o semestre, entre eles, o projeto de hardware, a programação de um microcontrolador e o uso dos seus periféricos (neste caso, Timers, o conversor AD, EEPROM).

Este robô pretendia-se que fosse capaz de reconhecer a cor de uma peça (vermelha, verde, azul, preta) e depositá-la na caixa da cor respetiva. Para tal utilizamos 3 servos (MG996R), sendo um deles de rotação continua para a movimentação do "braço", um eletroíman para agarrar as peças e um sensor de cor (TCS3200).

Esquemático da interface

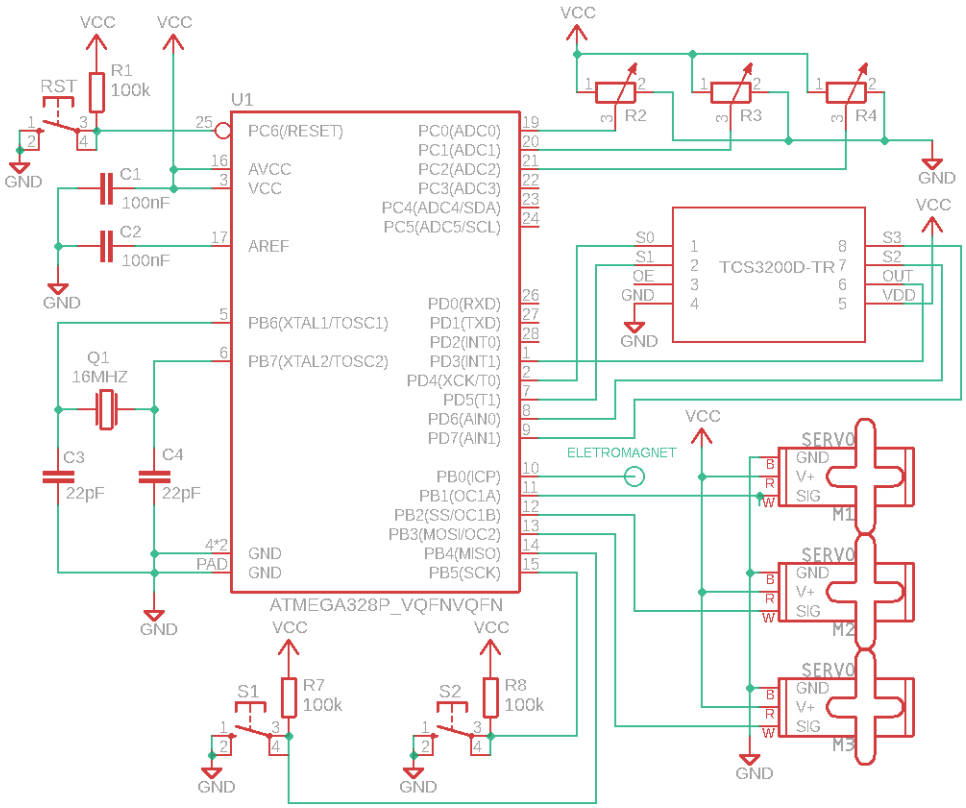


Figura 1 – Esquemático da interface.

Servos

Um servomotor é um mecanismo de circuito fechado que usa feedback de posição para controlar seu movimento e posição final. A entrada para seu controle é um sinal (analógico ou digital) representando a posição comandada para o eixo de saída.

Para este trabalho como tínhamos de controlar 3 servos em simultâneo optamos por usar uma divisão do período do sinal pelo número de servos. Ao dividir os 20ms por 3 servos (20000µs/3 = 6666.66(6) µs/servo) como não era um valor inteiro optamos por dividir por 4 e deixar uma das divisões inutilizadas (time frame: 5ms para cada servo) obtendo uma gama de tamanho suficiente visto que para estes servos esta se situa entre 1.2ms (mínimo) e 2.1ms (máximo). Com recurso ao timer1 ativamos a saída de cada servo no início do período de tempo associado, desligamos quando passou o tempo definido e ativamos no início do time frame seguinte para o seguinte servo. O servo 3 na base do robô tinha um funcionamento diferente dos restantes na medida que rodava continuamente e foi necessária uma abordagem diferente.

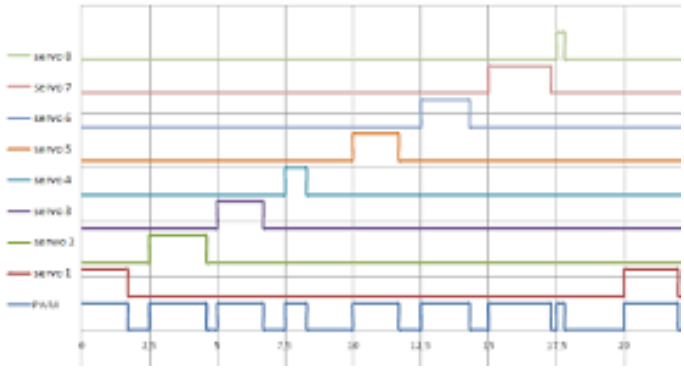


Figura - Divisão sinal PWM para cada servo

ADC

A conversão AD foi utilizada para a o posicionamento do servo 3 (de rotação continua e colocado na base do robô) através do potenciómetro (R4) colocado no eixo de rotação e ligado em PC2. Foi ainda ativada a conversão em PC0, PC1 para os potenciómetros (R2, R3) de posicionamento manual dos servos 1 e 2.

```
void init_adc (void){
  // Definir Vref=AVcc
  ADMUX = ADMUX | (1<<REFS0);
  // Desativar buffer digital em PC0
  DIDR0 = DIDR0 | (1<<PC0) | (1<<PC1) | (1<<PC2);
  // Pré-divisor em 128 e ativar ADC
  ADCSRA = ADCSRA | (7<<ADPS0)|(1<<ADEN);
}
```

Sensor de Cor

A deteção de cor é feita por análise da frequência dos sinais recebidos pelo sensor. Para a aquisição deste valor, registou-se, com um timer de 8bits, o intervalo de tempo com que o sinal fica *on* (entre um *rising-edge* e um *falling-edge*). No fim desta rotina, usa-se o total do *counter* para calcular a frequência. Posteriormente é repetido o ciclo para determinar a frequência do sinal para cada tipo de díodo presente no sensor (vermelho, azul e verde).

Através da análise da *datasheet* do sensor (TC3200) foi possível fazer a distinção das cores, que é feita por comparação das frequências medidas para cada item a detetar.

```
void colorsensor_init (){
  TCCR2A=0;
  TIMSK2=(1<<TOIE2);
}

ISR(TIMER2_OVF_vect) {
 TCNT2 = 0;                          // reload TC2
 incrementador++;
}

uint16_t FreqMeas(){
  if (PIND&(1 << 3)){
      while(PIND&(1 << 3));       // Espera o falling edge
  }
  TIFR2 = (7<<TOV2);
  while(!(PIND&(1 << 3)));        // Espera RE para inicio da contagem
      TCNT2=0;                    // Reset Counter
      incrementador=0;
      TCCR2B=(1<<CS20);           // Prescaller = F_CPU/1 Start Counting
  while(PIND&(1 << 3));           // Fim do pulso
      TCCR2B=0;                   // Stop Timer
  return (F_CPU/(2*(incrementador*255 + TCNT2)));
}

char get_color (){
  uint16_t red, blue, green;
  PORTD &= ~(1<<PD7); PORTD &= ~(1<<PD6);  //(!S3,!S2) freq red
  red = FreqMeas();
  PORTD |= (1<<PD7); PORTD &= ~(1<<PD6);   //(S3,!S2) freq blue
  blue = FreqMeas();
  PORTD |= (1<<PD7); PORTD |= (1<<PD6);    //(S3,S2) freq green
  green = FreqMeas();
  if(red>blue && red>green && green<blue && red>10000) return 'R';
  else if(blue>red && blue>green && red<green && blue>10000) return 'B';
  else if(green>red && green>blue && green>10000) return 'G';
  else return 'D';
}
```

Funcionamento Servo 3

```
uint8_t setServo3 (uint16_t pos){
//Retorna 1 enquanto servo 3 estiver ativo

  uint16_t actual_pos = read_adc(2);

  if ((abs(pos - actual_pos)) <= 1) {
    servoSet(3, 1500);   // Para um pulso de 1.5ms o servo para
    return 0;
  }
  else if( (abs(pos - actual_pos)) < 20 ){
    if(pos>actual_pos) servoSet(3, 1700);
    else if(pos<actual_pos) servoSet(3, 1300);
  }
  else
    servoSet(3, 1500 + (pos - actual_pos) * 7 );

  return 1;
}
```

A função obtém o valor da posição do servo 3 através da leitura ADC da tensão no potenciómetro (R4) retornando um valor entre 0 e 1023, sendo que 500 é a posição ajustada para a posição central (alinhado perpendicularmente com a caixa das peças). Com um pulso de 1.5ms o servo fica parado, superior a este valor roda para a direita e inferior para a esquerda. Tentamos ajustar a velocidade através de um "controlador P" para melhorar o desempenho, contudo durante as movimentações os valores da conversão AD alteravam-se por razões desconhecidas o que nos não permitiu ter muita precisão horizontal.

Modos de funcionamento

Introduzimos 2 modos de funcionamento, automático e manual os quais alternamos através da ativação do botão 1. O modo automático (modo 1) executa as movimentações do robô para todas peças até ao fim (Com início após pressionar o botão 2). No modo manual (modo 2) utilizamos dois potenciómetros (R2 e R3) e a leitura ADC para fazer a alteração do pulso nos servos 1 e 2. Ainda neste modo pressionando o botão 2 é possível ativar o eletroíman e fazer a leitura de cor com o sensor.

EEPROM

Foi utilizada para armazenar em memória não volátil o número de peças de cada cor processadas. Recorremos às seguintes funções para ler e escrever sendo que a adicionamos a possibilidade de guardar de variáveis de 16bits através da utilização de dois endereços por variável e concatenação de bytes embora não tenha sido utilizado.

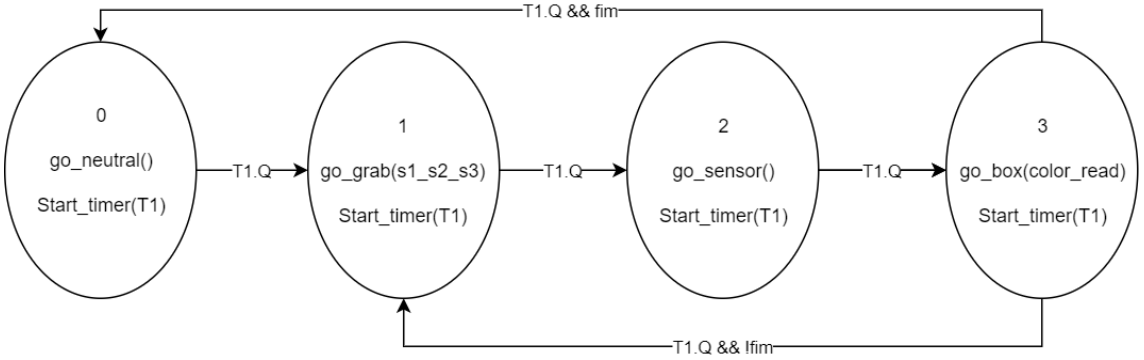
```
uint8_t EEPROM_read(uint8_t adrs){
    while(EECR & (1<<EEPE));   // Espera a conclusão da leitura anterior
    EEAR = adrs;               // Define o endereço do registo
    EECR |= (1<<EERE);         // Inicializa a leitura da EEPROM
    return EEDR;               // Retorna o valor em registo
}

void EEPROM_write(uint8_t adrs, uint8_t Data){
    while(EECR & (1<<EEPE));    // Espera a conclusão da escrita anterior
    EEAR = adrs;               // Define o endereço do registo
    EEDR = Data;               // Dados a serem guardados
    EECR |= (1<<EEMPE);
    EECR |= (1<<EEPE);         // Inicializa a escrita
}
void print_report(){

  RED_store   = EEPROM_read_low(RED_ADRS);
  GREEN_store = EEPROM_read_low(GREEN_ADRS);
  BLUE_store  = EEPROM_read_low(BLUE_ADRS);
  UNK_store   = EEPROM_read_low(UNK_ADRS);

  printf("Peças Operadas\n RED: %d, GREEN: %d, BLUE: %d, UNK: %d \n", RED
_store, GREEN_store, BLUE_store, UNK_store);
}
```

Máquina de Estados



- **go_neutral():** Coloca o robô numa posição "neutra", alinhado a meio.
- **go_grab(s1, s2, s3):** Vai buscar a peça designada. S1 e S2 são os tempos em microssegundos dos pulsos a enviar para os servos 1 e 2, respetivamente. S3 é o valor ADC lido do potenciómetro associado ao servo 3 (de rotação continua). O eletroíman é ativado quando este estiver em cima da peça na posição (s1,s2,s3).
- **go_sensor():** Posiciona o robô depois de agarrar a peça em cima do sensor de cor para ser feita a leitura da cor. Guarda a cor da peça numa variável (*color_read*) que será utilizada para escolher a caixa.
- **go_box(color_read):** Larga a peça na caixa da cor lida pelo sensor.

Estas rotinas são repetidas para todas as peças através da função **piece(s1, s2, s3).**

(0,0) S1: 1550 S2: 2080 S3: 470	**(0,1)** S1:1552 S2: 2080 S3: 500	**(0,2)** S1: 1550 S2: 2080 S3: 530
(1,0) S1: 1480 S2: 1990 S3: 470	**(1,1)** S1: 1482 S2: 1990 S3: 500	**(1,2)** S1: 1480 S2: 1990 S3: 530
(2,0) S1: 1360 S2: 1885 S3: 470	**(2,1)** S1: 1360 S2: 1885 S3: 500	**(2,2)** S1: 1360 S2: 1885 S3: 530

Tabela 1: Esquema da posição das peças na caixa e valores associados.

Conclusões

Consideramos que este projeto foi uma mais-valia no que diz respeito à familiarização com o ATMega328 e diversos periféricos no desenvolvimento dos conhecimentos adquiridos nesta unidade curricular, ao nível do projeto de hardware e software.

Foi um trabalho desafiante e no final não conseguimos atingir completamente os objetivos desejados mas ficamos satisfeitos com o resultado e podemos entender as pequenas complicações que podem surgir no desenvolvimento de um projeto.

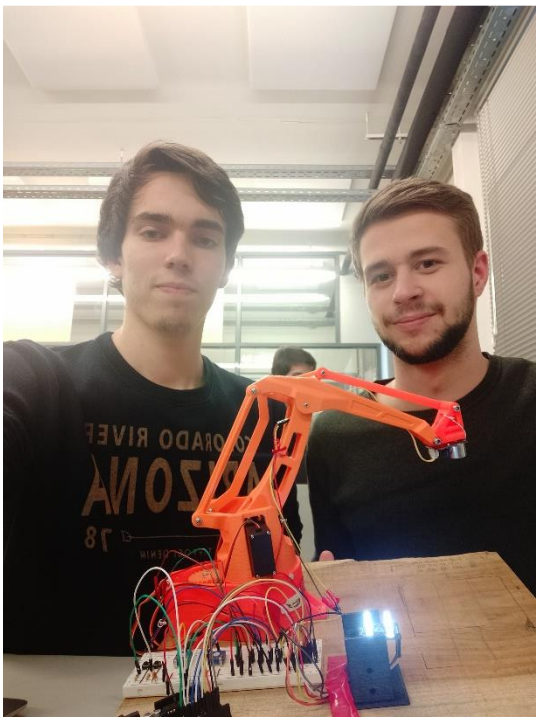
Anexos

A – Esquemático Interface

B – Fotos Montagem

C – Código

D – Datasheets



Fim