UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
ENGENHARIA DE COMPUTAÇÃO

Wagner Spinato Chittó

TRABALHO 2:

Considere que um microcontrolador deve ler 3 valores de temperatura, ligados nos conversores ADC0, ADC1 e ADC2. Os valores lidos vão de XºC (lido como valor 0 no adc) até (YºC, lido como 1023 no adc). Faça um programa que fique lendo repetidamente o valor dos 3 conversores e mostre a média dos valores lidos como um número em 3 displays de 7 segmentos.

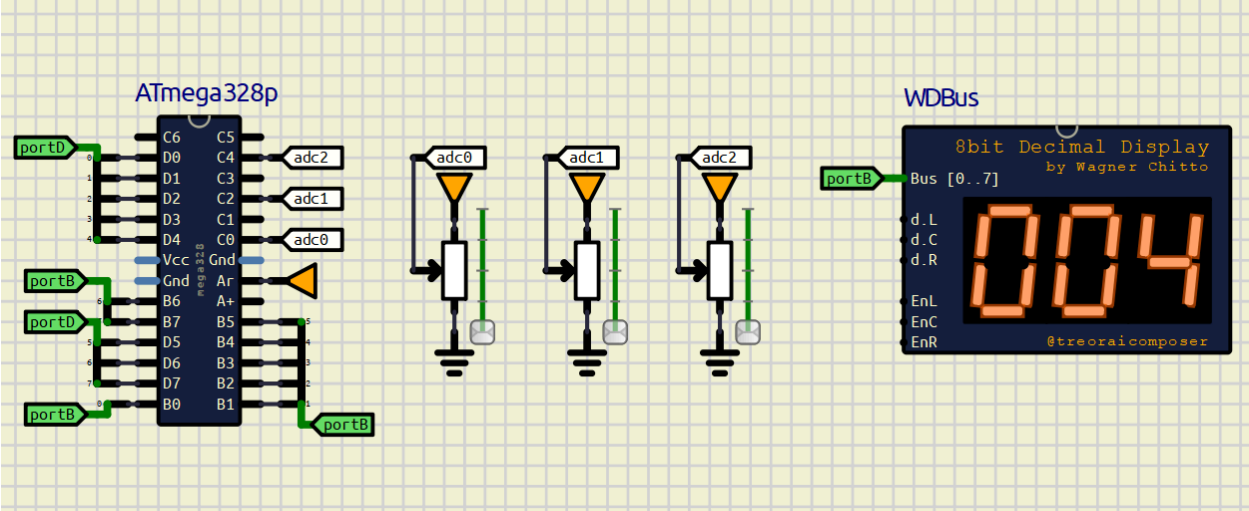
X = último dígito da sua matrícula +1.

Y = penúltimo dígito da sua matrícula +130.

Santa Maria, RS
19/04/2025

Componentes:

- ATmega328
- Grupo de 3 valores de temperatura (0V - 5V) ligados em PC0, PC2 e PC4.
- Um package customizado feito por mim, contendo
    - 3 Displays de 7 Segmentos
    - Entrada em bus de 8 bits



Sobre o package customizado:

Implementei este circuito para uso presente e futuro, uma vez que já foi avaliado os conhecimentos sobre conversão de binário para decimal e também mapeamento de saída para 7 segmentos. Para poupar código e tempo dos trabalhos futuros, peço permissão para usar este circuito:

- Recebe a entrada de 8 bits binários
- Usa também um ATmega rodando um programa semelhante ao final do Trabalho:
    - Lê os 8 bits de entrada binária.
    - Faz a conversão para decimal.
    - Mapeia as saídas para três displays de 7 segmentos.
- Arquivos do package estarão anexados no envio do trabalho.
    - É necessário colocar os arquivos na sua pasta “data” do SimulIDE para que ele abra com o package. Se não colocar na pasta, o display não carrega para o usuário.

Código do Trabalho 2:

```
; Author : Wagner Spinato Chitto
; Matricula: 201421493
;
; Ler 3 valores de temperatura ligados nos conversores ADC0, ADC1 e ADC2
; Os valores lidos vao de 4 (lido como 0 no adc) ate 139 (lido como 1023 no ADC)
; Mostrar a media dos valores lidos como um número em 3 displays de 7 segmentos.

start:
        ldi r17, 0b10000111             ; setup do conversor
        sts ADCSRA, r17

        ser r17
        out DDRD, r17                   ; PORTD is output
        out DDRB, r17                   ; PORTB is output
        clr r17
        out DDRC, r17                   ; PORTC is input

loop:
        clr r17
        clr r26
muxadc0:
        cpi r17, 0x00                   ; r17 = registrador do incremento do mux
        brne muxadc1
        ldi r16, 0b01000000
        sts ADMUX, r16                  ; use adc0
        jmp muxisset
muxadc1:
        cpi r17, 0x01
        brne muxadc2
        ldi r16, 0b01000010
        sts ADMUX, r16                  ; use adc1
        jmp muxisset
muxadc2:
        ldi r16, 0b01000100
        sts ADMUX, r16                  ; use adc2
        cpi r17, 0x02
        breq muxisset
        jmp loop

muxisset:                               ; mudanca do mux
        lds r18, ADCSRA
        ori r18, 0b01000000             ; liga o bit 6 e mantem os demais
```

```
        sts ADCSRA, r18                 ; inicia a conversao

testa:
        lds r18, ADCSRA
        sbrc r18, 6                     ; se ADCSRA[6] == 0, espera terminar a conversao
        rjmp testa

        lds r19, ADCH                   ; le parte alta da conversao
        lds r18, ADCL                   ; le parte baixa da conversao

trio:
        cpi r17, 0x00
        breq salva0
        cpi r17, 0x01
        breq salva1
        cpi r17, 0x02
        breq salva2

salva0:
        movw r20, r18                   ;save[21:20]
        inc r26
        inc r17
        sbrs r26, 2                     ; skip se r16 = 4
        jmp muxadc0
        jmp soma
salva1:
        movw r22, r18                   ;save[23:22]
        inc r26
        inc r17
        sbrs r26, 2                     ; skip se r16 = 4
        jmp muxadc0
        jmp soma
salva2:
        movw r24, r18                   ;save[25:24]
        inc r26
        inc r17

soma:
        ldi r26, 0x01                   ; limpa r26
        clr r28
        clr r29
        adc r28, r20
        adc r29, r21
        adc r28, r22
```

```
adc r29, r23
adc r28, r24
adc r29, r25                    ; [r29:r28] total da soma

; DIVISAO POR APROXIMACAO
clr r24
clr r25                         ; resultado em [25:24]

; somar as partes 1/4, 1/16, 1/64 e 1/128
; shifts              2, 4, 6 e 7
; divisao por 3 = 0.333 aproximadamente 0.336

;.25
movw r30, r28
lsr r31
ror r30
lsr r31
ror r30
movw r24, r30

;.3125
movw r30, r28
lsr r31
ror r30
lsr r31
ror r30
lsr r31
ror r30
lsr r31
ror r30
add r24, r30
adc r25, r31

;.328
movw r30, r28
lsr r31
ror r30
lsr r31
ror r30
lsr r31
ror r30
lsr r31
ror r30
lsr r31
```

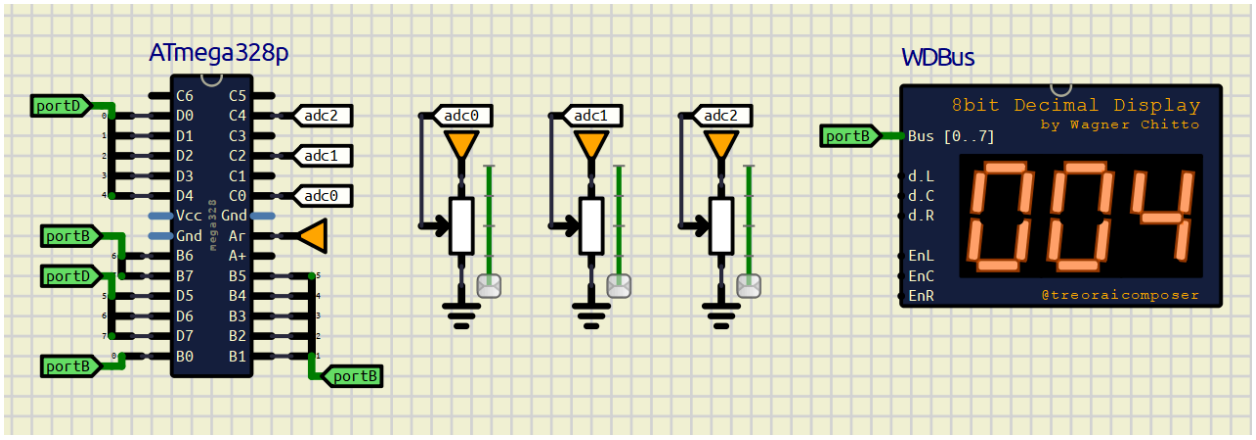
```
ror r30
lsr r31
ror r30
add r24, r30
adc r25, r31

;.336
movw r30, r28
lsr r31
ror r30
lsr r31
ror r30
lsr r31
ror r30
lsr r31
ror r30
lsr r31
ror r30
lsr r31
ror r30
lsr r31
ror r30
add r24, r30
adc r25, r31

; Mapeamento b[0 .. 1000 0000 0100]
;         para b[0100 .. 1000 1011] d[4 .. 139]

; Funcao de reducao
; y = ((x * 135) / 2052) + 4
; y = ((x * 5 ) / 38 ) + 4

; multiplica r25:r24 por 5
; (r25:r24 << 2) + r25:r24
; Livres:  [r31:r28]
; Destino: [r25:r24]

movw  r28, r24                          ; copia r25:r24 para r29:r28

; shift para fazer x4
lsl  r28
rol  r29
lsl  r28
rol  r29
```

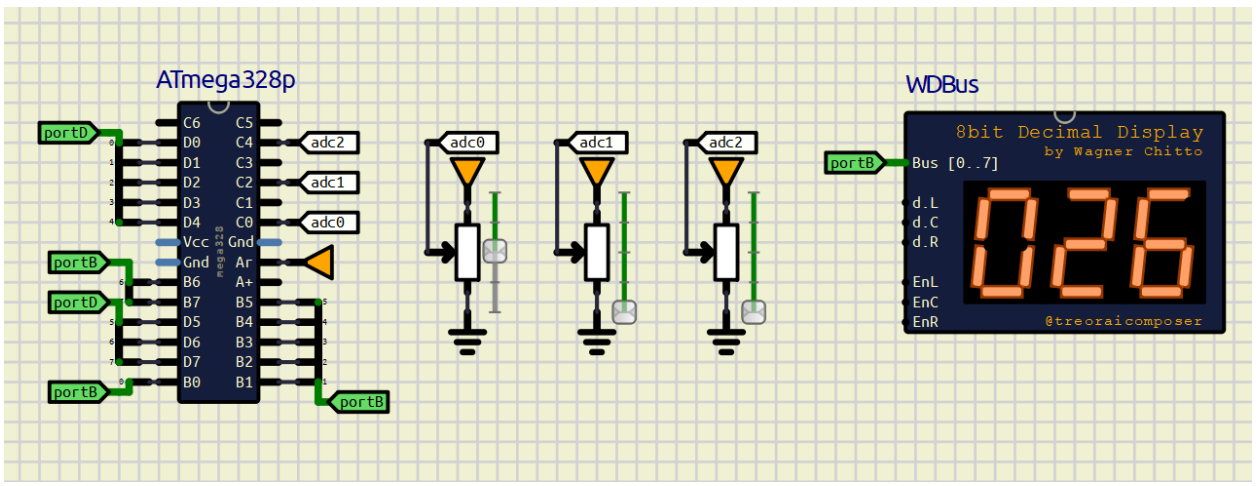
```
        ; soma com o original x5
        add  r28, r24
        adc  r29, r25

        ; max   0x 140f
        ;                   0b 0001 0100 0000 1111

        clr r24
        clr r25
        clr r30
        ldi r16, 16                     ; contador de bits
        ldi r31, 38                     ; divisor

div_loop:
        ;desloca [r29:r28] para a esquerda, resto entra
        lsl r28                         ; <<1
        rol r29
        rol r30                         ; resto recebe bit deslocado
        cp r30, r31                     ; compara resto com divisor
        brlo no_sub
        sub r30, r31                    ; subtrai divisor do resto

        ; ajusta quociente (desloca 1 para a esquerda + 1)
        lsl r24
        rol r25
        ori r24, 0x01                   ; coloca 1 no bit menos significativo
        rjmp next

no_sub:
        lsl r24                         ; desloca quociente sem somar 1
        rol r25

next:
        dec r16
        brne div_loop

        ; add 4
        ldi r16, 4
        add r24, r16

print:
        out PORTB, r24                  ; escreve o valor em portB
        rjmp loop
```

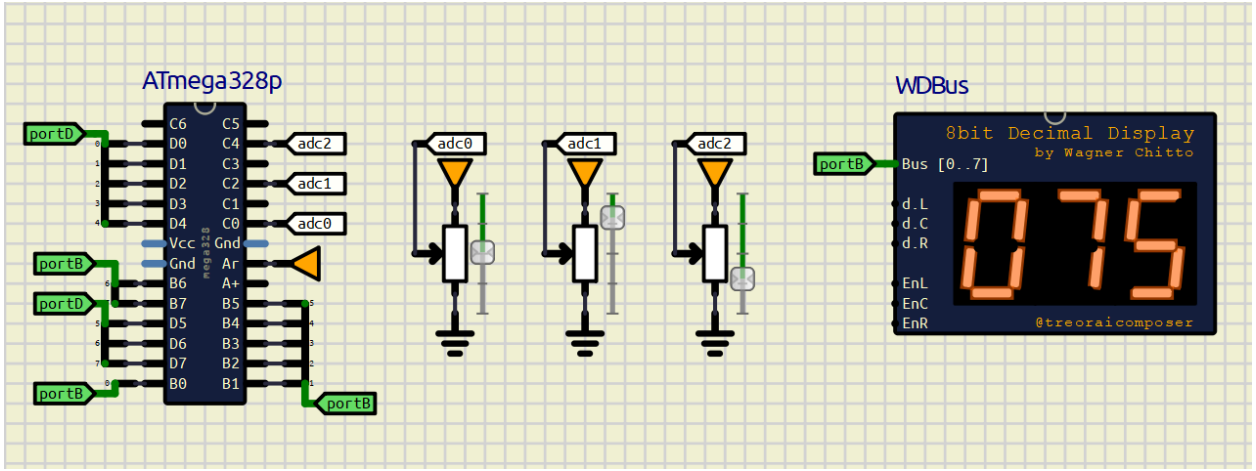
Vídeo do programa em execução:

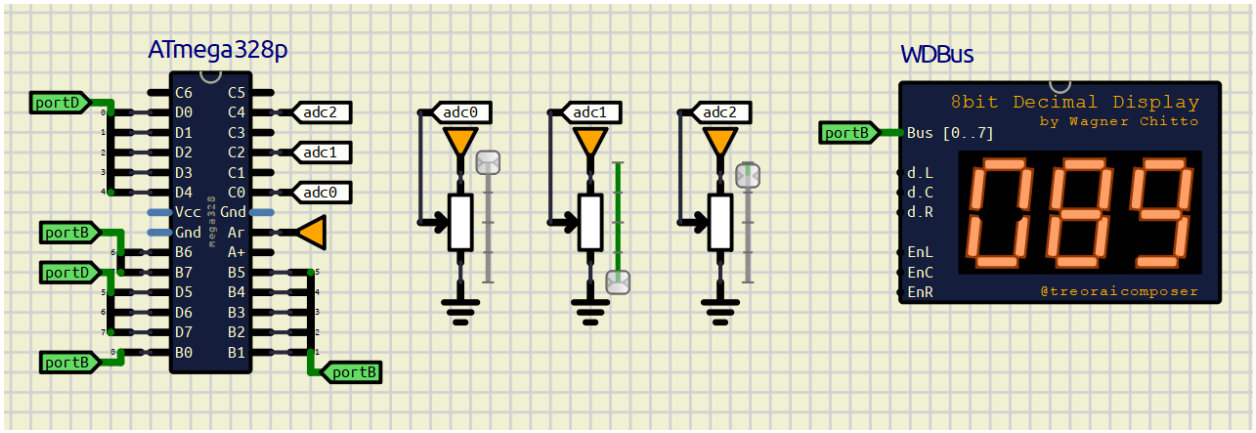
https://youtu.be/E-3cQat22LQ

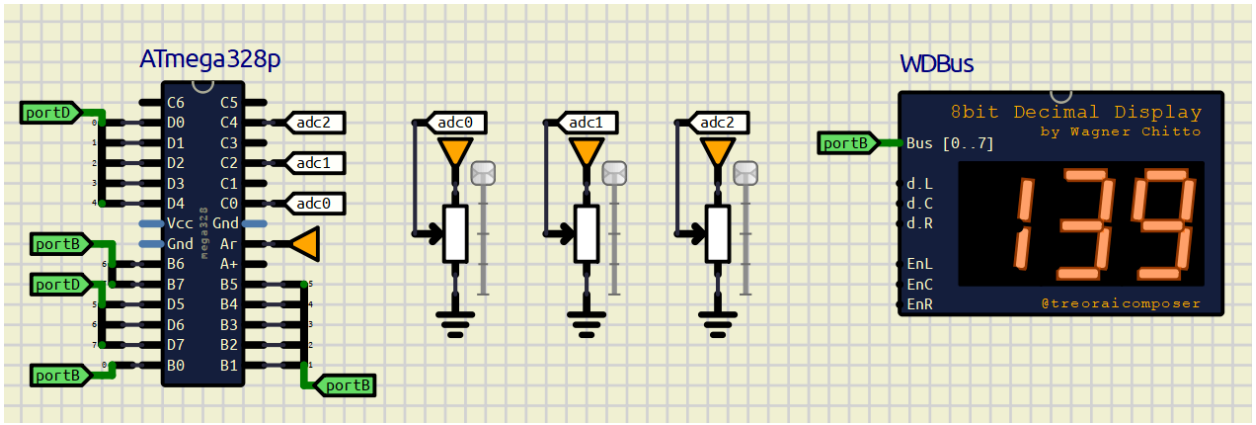
Fotos:











Vídeo do programa em execução:

https://youtu.be/E-3cQat22LQ