Universidade Federal do Rio Grande do Norte
Centro de Tecnologia - CT
Departamento de Engenharia Elétrica

COFRE DIGITAL BASEADO NO ATMEGA328P COM INSTRUÇÕES EM ASSEMBLY

ELE1717 - Grupo 02 - Problema 03 - Implementação

Alysson Ferreira da Silva
Ana Beatriz Marinho Neves
Elias Gurgel de Oliveira
Isaac de Lyra Junior
Sthefania Fernandes Silva

Natal, 20 de julho de 2021

Resumo

O ATMEGA328P é um microcontrolador comumente utilizado nos mais diversos sistemas digitais, principalmente, por ser simples e barato. Diante disso, o seguinte relatório tem como objetivo desenvolver a implementação de um cofre digital usando como microprocessador o ATMEGA328P e como linguagem de programação *Assembly*. Para isso, foi necessário ler o *datasheet* do microcontrolador, entender seu funcionamento e definir as instruções e periféricos necessários na implementação. De posse disso, foi criado, no *software Microchip Studio*, um código em *Assembly* que contemplasse todas as particularidades do cofre. Para simular o funcionamento do sistema, com a interface homem-máquina, foi usado o *software Proteus*. A simulação executada mostra que a implementação do projeto foi bem sucedida, apesar da necessidade de algumas alterações do projeto original.

Palavras-chave: Sistemas Digitais, *Assembly*, ATMEGA328P, *Proteus*, *Microchip Studio*.

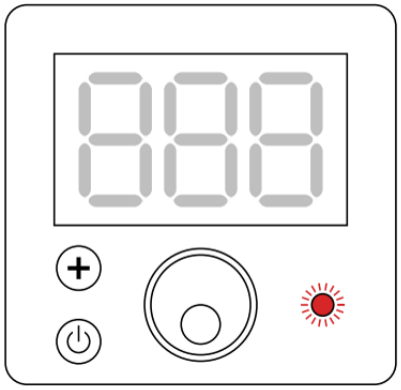
Lista de Imagens

Sumário

1 IMPLEMENTAÇÃO

A implementação do cofre digital - baseado na configuração do ATMEGA328P - foi feita de forma que a aparência do sistema seja conforme a Figura 1.

Figura 1 – Interface homem-máquina do cofre digital.



Fonte: Dados do problema.

Para criar tal interface, foram necessários os componentes mostrados na Figura 4. Note que o LED (*light-emitting diode*) usado será do tipo RGB (*red*, *green*, *blue* - em tradução livre: vermelho, verde e azul, respectivamente), logo a cor laranja precisará ser uma combinação de duas ou mais dessas três cores. Para isso foi sugerido o uso de uma saída modulada em PWM (*Pulse Width Modulation* - Modulação por Largura de Pulso) do microcontrolador.

Figura 2 – Descrição dos elementos da interface homem-máquina do cofre digital.

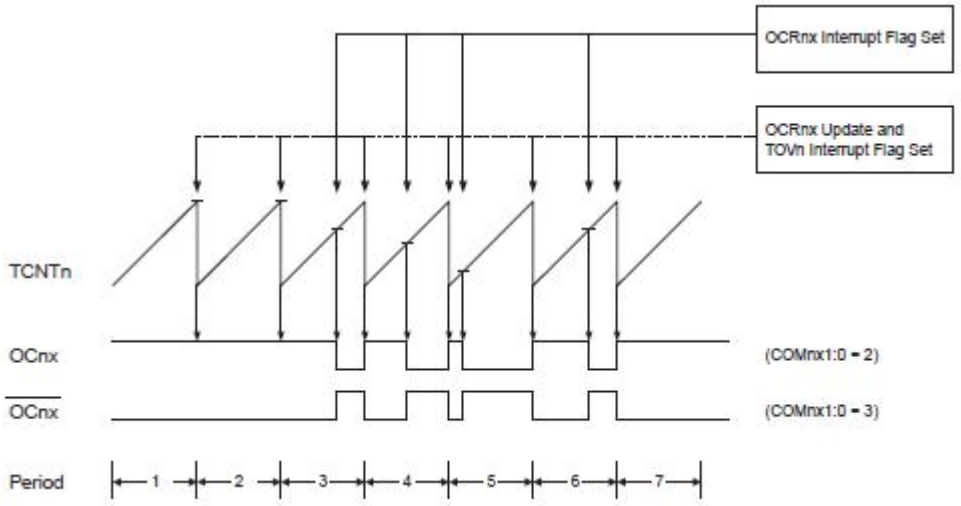
Elemento	Descrição
	Display para exibição de valor entre 0 e 999
	Potenciômetro para ajuste de valor (volta completa)
	Botão de inicializar/cancelar o processo (tipo *pushbutton*)
	Botão de adicionar valor (tipo *pushbutton*)
	Led RGB (Fechado; Senha; Processando; Aberto)

Fonte: Dados do problema.

O ATmega328P possui seis portas que podem ser configuradas como entrada ou saída PWM. Um sinal PWM possui dois níveis lógicos (alto e baixo) e o que caracteriza essa modulação é a quantidade de tempo, dentro de um período, que o sinal fica em nível

lógico alto. Esse período de tempo, em porcentagem do ciclo, é chamado de *Duty Cycle*. Na Figura 3 é evidenciado um gráfico com a técnica de modulação *fast PWM* utilizado no ATmega328P.

Figura 3 – Técnica de fast PWM usada no ATmega328P.



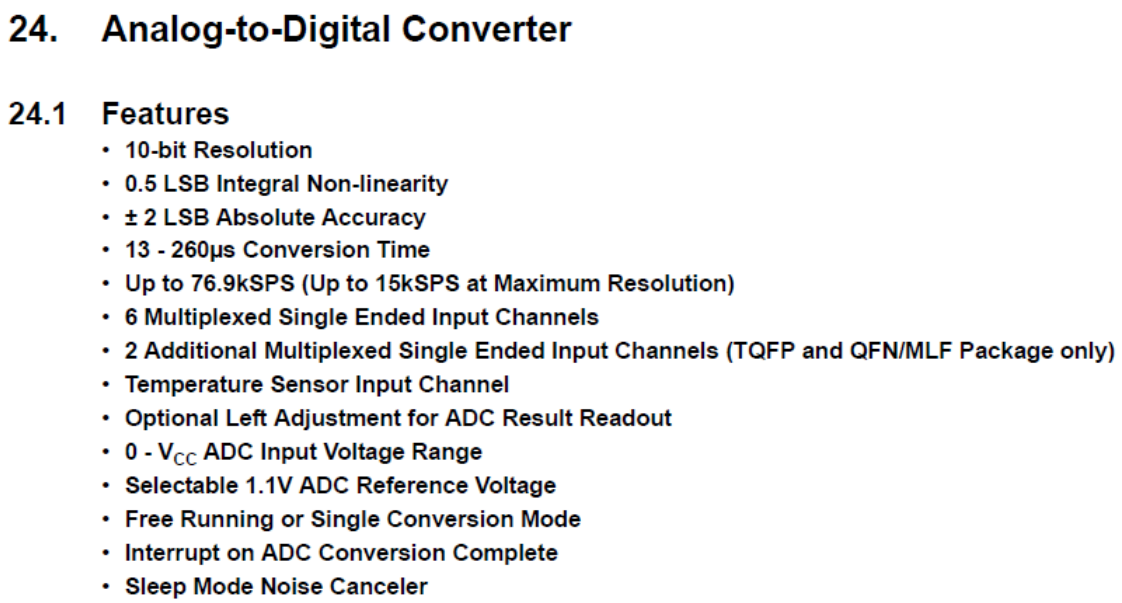
Fonte: Microchip (2015).

Outro ponto importante é que o *display* deve exibir valores dentro do intervalo de 000 a 999, no entanto, o valor inserido pelo potenciômetro é convertido para um valor de 10 bits - configuração intrínseca do microcontrolador - que permite escrever de 000 a 1023. Nesse sentido, faz-se necessário uma limitação dos valores do potenciômetro. A sugestão do projeto não foi acatada, pois foi encontrada uma forma mais simples de ser realizar tal alteração, esta é explica do tópico de correções.

Além de exibir os valores inseridos, o *display* existem duas configurações específicas para informar quando o cofre está aberto ou fechado. Isso pode ser observado através do, já comentado, *LED RGB*.

Ainda sobre o potenciômetro, para o valor inserido ser entendido pela máquina é preciso ser feita uma conversão analógica-digital (AD). Os conversores AD são utilizados para traduzir um sinal analógico em uma representação digital, quanto maior o número de bits para a representação do sinal melhor será a resolução do conversor. De acordo com o *datasheet* do microcontrolador ATMEGA328P, o conversor AD possui as características apresentadas na Figura 4.

Figura 4 – Características do Conversor Analógico Digital do microcontrolador ATMEGA328P



24. Analog-to-Digital Converter

24.1 Features

- 10-bit Resolution
- 0.5 LSB Integral Non-linearity
- ± 2 LSB Absolute Accuracy
- 13 - 260µs Conversion Time
- Up to 76.9kSPS (Up to 15kSPS at Maximum Resolution)
- 6 Multiplexed Single Ended Input Channels
- 2 Additional Multiplexed Single Ended Input Channels (TQFP and QFN/MLF Package only)
- Temperature Sensor Input Channel
- Optional Left Adjustment for ADC Result Readout
- 0 - V_{CC} ADC Input Voltage Range
- Selectable 1.1V ADC Reference Voltage
- Free Running or Single Conversion Mode
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler

Fonte: Microchip (2015).

No tópico Conversor AD, em 2.1.1, será detalhado como foram configurados os registradores no código fonte para realizar a conversão do valor analógico do potenciômetro para os valores que aparecerão nos *displays*.

Para que a senha inserida seja processada pelo cofre, o usuário necessita esperar 0,5 segundos. Logo, foi preciso usar um periférico de contagem do ATMEGA328P, o *Timer/Counter*. Neste ponto, o grupo anterior não havia mencionado qual seria o *Timer* utilizado dentre os três existentes, porém haviam indicado que seria utilizado o mesmo *Timer* para a contagem de 0.5s e para gerar o PWM do led laranja. Neste ponto, o grupo decidiu utilizar o *Timer* 1 pela quantidade de ciclos necessários (15625), pois necessitava de um registrador capaz de armazenar mais que 8 bits. Além disso, foi decidido que a implementação do PWM junto ao contador poderia ser árdua e complicada, principalmente por estar sendo usando o modo CTC (*Clear Timer on Compare Match*) e por causa disso foi utilizado 2 contadores ao invés de um.

Por último, o *clock* utilizado foi o interno do microcontrolador na frequência 8MHz.

1.1 Correções do projeto

Algumas alterações foram necessárias para mitigar erros e para facilitar a implementação do projeto. Assim, foi preciso alterar o limitador de entrada (000 a 999); o modo de armazenamento das senhas e o fluxograma de funcionamento do cofre.

1.1.1 Limitador de entrada

Considerando o *datasheet* do componente, observa-se que numa variação de tensão de 0 a 5v em um pino de entrada, conectado logicamente a um conversor AD, resulta em uma variação de 0 a 1023 na saída do conversor. Esse valor não é compatível com o *display* da IHM (Interface Homem-Máquina) por este ser limitado a apenas três dígitos. Esse problema pode ser contornado de duas formas: Tratamento interno pelo *software* ou limitação externa do sinal que sensibiliza o pino conectado ao conversor A/D. Para este projeto foi escolhida a segunda opção por ser mais prática do que efetuar tratamento numérico interno ao micro controlador, o que exigiria mais desse componente. O circuito desenvolvido e a explicação da sua obtenção serão apresentados mais adiante.

1.1.2 Fluxograma de funcionamento do sistema

Para definir a atuação do cofre digital, com todas as suas especificações, foi feito o fluxograma mostrado na Figura 5. É válido salientar que de maneira geral, o diagrama original foi preservado mas alguns detalhes foram alterados.

Primeiramente, o fluxograma começa pelo estado "Desligado"e se mantém nele até que o botão PW (*pushbutton* de ligar/cancelar) seja pressionado, nesse momento todos os contadores usados estão zerados. Quando o botão for pressionado, a máquina muda para o estado ESPERAR SENHA, aqui o led que estava na cor vermelha muda para a cor azul e o *display*, que outrora estava com todos os leds apagados, é habilitado para exibir os valores inseridos na máquina. A máquina permanecerá nesse estado a menos que algum dos botões (*pushbutton* de ligar/cancelar ou de adicionar) sejam pressionados. Nesse momento o usuário pode alterar o valor do potenciômetro livremente, assim caso o botão BOTADD seja pressionado será feito o processamento da senha nos 4 estados seguintes, que juntos formam INSERIR SENHA. Já se PW for pressionado é feito o cancelamento das inserções de senhas e a máquina volta para DESLIGADO.

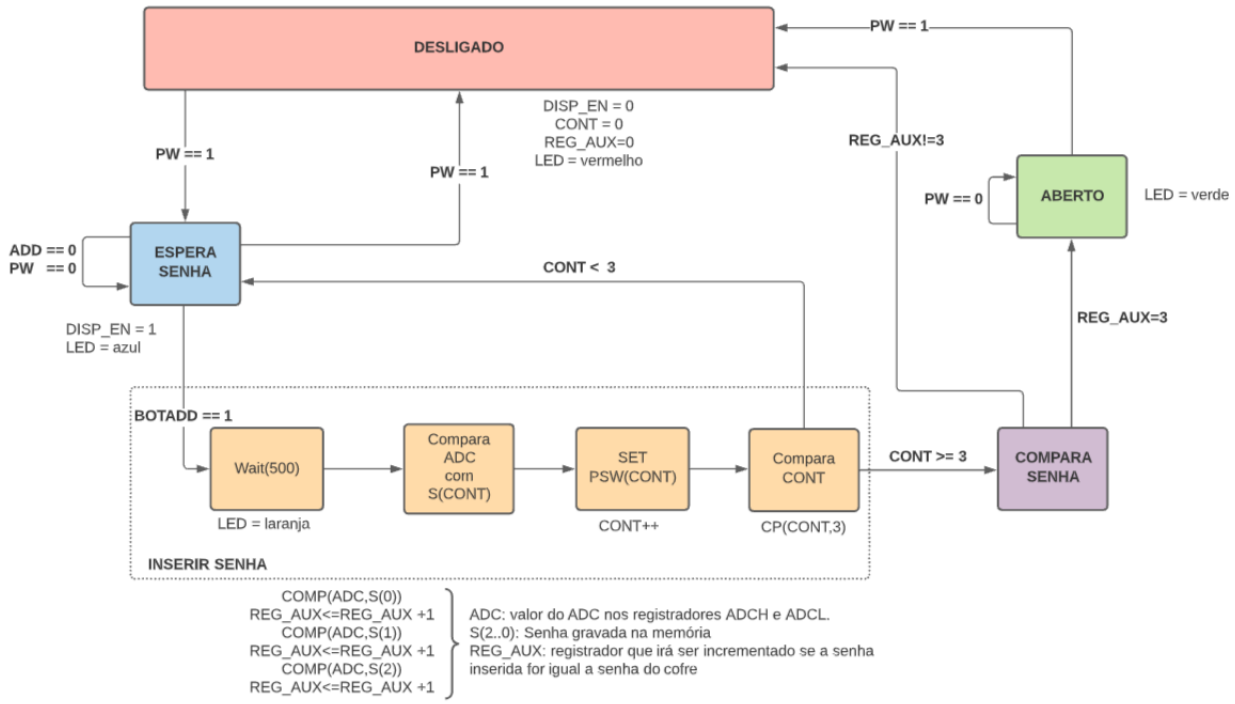
O estado "Inserir Senha"foi uma das alterações feitas para facilitar o entendimento do código *Assembly*. Ele contém o estado "Wait(500)", onde a máquina é forçada a esperar 500 milissegundos com o LED aceso na cor laranja, mostrando para o usuário que a senha está sendo processada. Finalizado o temporizador, uma comparação entre o valor armazenado nos registradores ADCH e ADCL (valores inseridos) com os registradores que estão armazenando a senha deve ser feita, isso ocorre em "Compara ADC com S(cont)". A maneira encontrada de guardar o resultado dessa comparação foi utilizando o registrador REGAUX, o qual será incrementado todas as vezes que a senha inserida for igual a senha do cofre, caso contrário não haverá incremento. Essa foi outra alteração realizada visando facilitar a implementação em *Assembly*.

Logo após isso, no estado "SET PSW"o contador CONT é incrementado. Esse

contador é importante devido a recursividade na inserção de senhas que está sendo feita, assim é preciso ser verificado se as três senhas foram inseridas na máquina para as próximas etapas serem executadas. Portanto, em "Compara CONT"é verificado se todas as senhas foram inseridas comparando o valor guardado no contador com 3, caso CONT seja inferior a 3 a máquina volta para o estado "Espera Senha", caso CONT seja igual a 3 a máquina vai para o estado "Compara Senha". Durante todas essas etapas o led laranja estava acionado

Em "Compara Senha" é feita uma comparação do REGAUX com 3, se o resultado for verdadeiro significa que a senha inserida está correta e o cofre pode ir para o estado "Aberto", onde o LED ficará verde e o *display* vai ficar com o formato '- - -'. Caso contrário, a senha inserida está errada e a máquina vai para o estado "Desligado", desativando o *display* e colocando o LED na cor vermelha.

Figura 5 – Fluxograma do cofre digital.



Fonte: Autores.

2 RESULTADOS

De posse das correções necessárias, o projeto foi implementado utilizando o *software Microchip Studio*, o qual é uma plataforma voltada para desenvolver projetos com os microcontroladores da ATMEL. Esta possui um ambiente que permite escrever projetos em C/C++ ou *Assembly*. Nesse caso, foi utilizada a linguagem *Assembly*.

Para simular o sistema com a interface homem-máquina foi utilizado o *software Proteus*, a explanação de ambas as simulações estarão nos tópicos seguintes.

2.1 Código em Assembly

O primeiro passo a ser feito foi definir quais pinos das portas B, C ou D do ATMEGA328P seriam utilizadas como entrada ou saída. Isso é feito utilizando os registradores DDRX (onde X pode ser B, C ou D), os quais são responsáveis por definir se o pino é de saída, caso o valor lógico seja 1 ou de entrada, caso o valor lógico seja 0. Dessa forma, as portas foram setadas conforme mostra a Figura 6.

Figura 6 – Definição das portas.

```
;DEFINIÇÃO DE ENTRADAS E SAÍDAS
ldi r16, 0b11111111
out DDRB, r16 ; define todas as portas PB como saída
ldi r16, 0b11111000
out DDRD, r16 ; define inputs/outputs das portas PD
cbi DDRC, 1
cbi DDRC, 2
sbi DDRC, 3
sbi DDRC, 4
```

Fonte: Autores.

2.1.1 Conversor AD

Para o código do conversor AD primeiro utilizamos um registrador de uso local para carregar o valor que colocaremos no registrado ADMUX, onde definimos que a tensão do pino de referência AREF será a tensão do AVCC com os capacitores externos, ou seja, REFS1 é '0' e REFS0 é '1'.

Depois de definir o ADMUX, iremos para o registrador DIDR0 que será onde indicaremos qual é o pino que está conectado o nosso potenciômetro, no nosso projeto é o ADC0. Em seguida, nosso sistema pode começar o processamento das instruções começando no estado DESLIGADO.

Quando o usuário começar a ajustar o valor da senha no potenciômetro, a máquina estará no estado ESPERA SENHA onde o *LED RGB* estará sinalizando a cor azul e sua quarta porta do PORTC estará em nível lógico baixo avisando que a tranca da porta do cofre está fechada. Logo depois, será definido quais serão os *bits* carregados no registrador ADCRSA. O *bits* mais significativo refere-se ao ADEN que é o habilitador do conversor AD, ou seja, ele estará em '1'. O segundo *bit* mais significativo é o ADSC que também estará em '1' pois ele inicia o processo da conversão. Em seguida, temos o *bit* do ADATE que estará em '1'já que a conversão AD iniciará na borda positiva do sinal. Já os *bits* ADIF e ADIE estarão em '0', pois o primeiro sinaliza que a interrupção está concluída e a segunda também está relacionado com a finalização da interrupção. Os três últimos *bits*, ADPS[2:0], selecionará o Prescale do conversor AD, no nosso projeto selecionamos 128, ou seja, ADPS2, ADPS1 e ADPS0 estão todos em nível lógico alto.

Após configurar o registrador ADCRSA, armazenamos em dois registradores locais os valores que estão nos registradores ADCL e ADCH. Realizamos uma comparação entre o registrador que armazenamos o valor de ADCH para descobrirmos se os dois *bits* mais significativos que está saindo do conversor AD possuem os valores '00' ou '01' ou '10' ou '11'.

Na saída, o PORTD armazenará os dois *bits* mais significativos do conversor AD e o PORTB armazenará os demais *bits*. Em seguida, o próximo comando é para retornar para a sub rotina, ou seja, a máquina retornará ao estado ESPERA SENHA e verificará se a senha adicionada estava correta.

Na Figura 7 é mostrada a configuração do conversor AD utilizada.

Figura 7 – Configuração dos conversor AD.

```
;DEFINIÇÃO DO CONVERSOR AD
ldi r16, 0b01000000
sts ADMUX, r16 ; define ADMUX como sendo com REFS = 01, ou seja pega a tensão
               ; de referência do AVCC em conjunto com capacitor
ldi r16, 0b00000001
sts DIDR0, r16

rjmp desligado
```

Fonte: Autores.

2.1.2 Led RGB e PWM

A cor azul do led é acionada por uma saída digital (níveis lógicos alto e baixo), a cor vermelha é acionada por uma saída PWM com os valores 0 e 255 (que pode ser traduzida como níveis lógicos alto e baixo), por conta da indicação do projeto recebido. Já a cor verde também é acionada por uma saída em PWM, contudo é usado o valor 127 em conjunto com o acendimento da cor vermelha, para formar a cor laranja. Em outros casos,

são usados os valores 0 e 255 (para sinalizar o "0" lógico e o "1" lógico, respectivamente) para o acendimento da cor verde.

Conforme falado anteriormente, um sinal modulado em PWM consiste na manutenção do nível lógico alto durante um certo tempo dentro de um período do sinal (*duty cycle*). Isso pode ser traduzido como um valor RMS (*root mean square*) do sinal, implicando na alteração da intensidade da cor do led.

No caso do microcontrolador em questão, o período do sinal PWM possui uma resolução de 8 bits, dividindo o período em 256 valores, com isso, um valor contido entre 0 e 5 v será mapeado entre 0 e 255.

Figura 8 – Configuração dos leds.

```
setledred:
    ldi r16, 255
    sts 0x48, r16 ; define OCR0B
    ldi r16, 0
    sts 0x47, r16 ; define OCR0A
    cbi PORTD, 7
    ret

setledblue:
    ldi r16, 0
    sts 0x48, r16 ; define OCR0B
    ldi r16, 0
    sts 0x47, r16 ; define OCR0A
    ret

setledorange:
    ldi r16, 255
    sts 0x48, r16 ; define OCR0B
    ldi r16, 127
    sts 0x47, r16 ; define OCR0A
    cbi PORTD, 7
    ret

setledgreen:
    ldi r16, 0
    sts 0x48, r16 ; define OCR0B
    ldi r16, 255
    sts 0x47, r16 ; define OCR0A
    cbi PORTD, 7
    ret
```

Fonte: Autores.

2.1.3 *Timer* de 0,5 segundos

O timer proposto necessita de 15625 ciclos para completar 0.5 segundos, por causa disso foi necessário utilizar o timer 1 que possui 16 bits e está disponível no atmega328p para configuração e uso. Dito isso, a configuração utilizada para operacionalizar esta contagem iniciou pela definição do valor 15625 nos registradores OCR1AH (registrador de 8 bits com os bits mais significativos de OCR1A) e OCR1AL (registrador de 8 bits com os

bits menos significativos de OCR1A). Nesse sentido, foi passado para OCR1AH o valor em hexadecimal 0x3d e para OCR1AL o valor 0x09.

Além disso, no registrador referente ao registro de controle "A", do timer 1 (TCCR1A), o bit mais significativo foi habilitado para escolher a limpeza de OC1A e OC1B quando o resultado da comparação do contador com OCR1A for verdadeiro. Ressalta-se que a escolha desse bit foi realizada para caso o grupo necessitasse utilizar esta função no futuro. Os outros bits deste registrador foram setados em zero. No registrador de controle "B"(TCCR1B) foi passado o valor 0x4c. Neste caso, o sexto bit deste registrador (ICES1) foi assinalado em nível lógico alto para que na subida do clock fosse realizada a captura de valor e consequentemente fosse habilitada a interrupção por captura de valor, para caso fosse necessário fazer uso desta função. Ademais, assumiu-se que WGM11 e WGM10 do registrador TCCR1A assumiriam 0, assim como, a variável WGM13 do registrador TCCR1B que somado a WGM12 = 1 indicamos o uso do modo Clear Timer on Compare match (CTC) para resetar o contador 1.

Para selecionar o prescale de 256 para compactuar com a formula indicada foi passado nível lógico alto para o bit CS12 e nível lógico baixo para CS11 e CS10 no registrador TCCR1B. E como não foi utilizado os geradores de onda no timer 1, foi passado um vetor nulo para o registrador de controle C (TCCR1C). Para as operações acima foram utilizados os mnemônicos ldi, para carregar uma constante em um registrador de uso geral, e sts para guardar em uma constante na sRAM o valor salvo em um registrador. A configuração descrita acima pode ser vista na Figura 9.

Figura 9 – Configuração do Timer.

```
;DEFINIÇÃO DO TIMER 1
ldi r16, 0x3d
sts 0x89, r16 ; OCR1AH
ldi r16, 0x09
sts 0x88, r16 ; OCR1AL
ldi r16, 0x00
sts 0x80, r16 ; TCCR1A
ldi r16, 0x00
sts 0x82, r16 ; TCCR1C
ldi r16, 0x23
sts 0x6f, r16 ; TIMSK1
ldi r17, 0x02
ldi r16, 0x4c
sts 0x81, r16 ; TCCR1B
```

Fonte: Autores.

Quando necessário chamar o timer em código é realizado um jmp para label timer05. Esta label tem duas instâncias: uma com uso do mnemônico sbi para setar nível lógico alto no bit 1 do registrador TIFR1 e realizar o reset do contador, e outra com a label comeca que utiliza o mnemônico sbis no bit 1 do mesmo registrador anterior para verificar

quando o contador irá atingir o limite imposto de ciclos naturalmente. Assim que o número de ciclos é atingido, entende-se que passou 0.5 e ele retorna através do mnemônico ret para onde foi realizada a chamada do timer. Caso contrário, será dado rjmp para comeca verificando quantas vezes forem necessárias o bit 1 do registrador TIFR1 até este assumir nível lógico alto. Este trecho de código pode ser observado na Figura 10.

Figura 10 – Label timer05.

```
timer05:
    sbi TIFR1, 1
    comeca:
        sbis TIFR1, 1
        rjmp comeca
        ret
```

Fonte: Autores.

2.1.4 Fluxograma de funcionamento do código

Em concordância com o que foi explicado no tópico 1.1.2 foi feita a implementação em *Assembly* do cofre, no entanto, alguns detalhes foram adicionados. Um deles foi um estado para verificar se o botão foi "despressionado"ou não. Isso se fez necessário devido a, principalmente, frequência do *clock* de 8MHz, assim, um botão dessincronizado com ele faria com que a máquina executasse muitas etapas - não esperadas pelo usuário - em pouco tempo. Foi feito mais de um *label* para garantir isso, na Figura 11 é mostrado o *label* usado para garantir que o sistema somente irá do estado "Desligado"para o estado "Espera Senha"se o *push button* PW não estiver mais sendo pressionado.

Figura 11 – Label botão síncrono.

```
desligado:
    rcall setledred
    cbi PORTC, 3
    sbi PORTC, 4
    sbis PINC, 1
    rjmp desligado
    rjmp botao_sinc

botao_sinc:
    sbic PINC, 1
    rjmp botao_sinc
    rjmp esperasenha
```

Fonte: Autores.

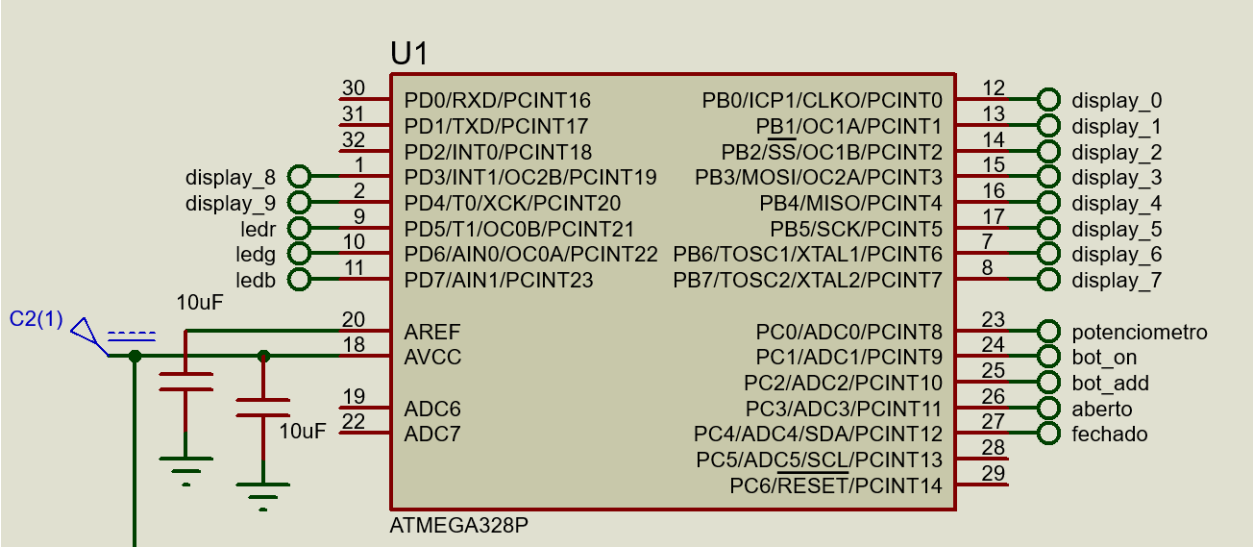
Outro detalhe é que em cada estado serão chamados os *labels* dos leds e do conversor. No que tange o *label* do conversor, isso é feito porque a conversão será desativada no estado "Insere Senha"com o intuito de evitar que o usuário mude o valor inserido durante o processamento da senha. Já a chamada dos leds é para setar a cor correta no rgb.

O código completo em *Assembly* de todos tópicos citados está disponível no Anexo B.

2.2 Simulação no *Proteus*

No código *Assembly* foram definidos quais pinos seriam de entradas ou de saída, no Proteus esses pinos foram selecionados e nomeados conforme mostra a Figura 12.

Figura 12 – Pinos selecionados do microcontrolador.



Fonte: Autores.

2.2.1 O circuito limitador

Conforme exposto anteriormente, o circuito limitador que alimenta o pino do conversor A/D objetiva manter os valores convertidos dentro de uma faixa apropriada ao *display*. Para isto, foi desenvolvido o circuito da Figura 13. O valor do potenciômetro foi arbitrado em 1 kΩ para que fosse calculado o outro resistor apropriado ao propósito. Em seguida pode-se observar a sequência de passos:

a) Foi usado uma regra de três para definir a queda de tensão necessária no resistor R_1;

b) De posse da queda de tensão em R_1 e usando a equação para um divisor de tensão,

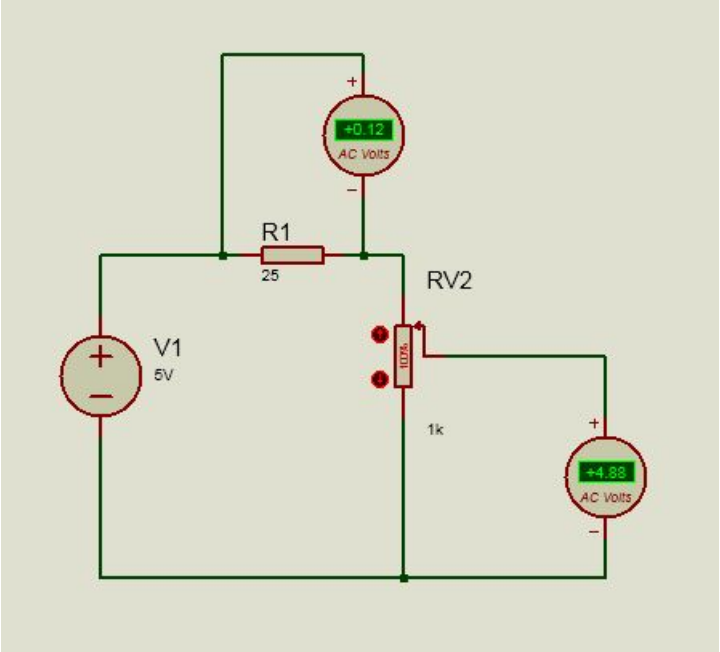
obtivemos um valor aproximado para R_1 em 25 Ω.

$$\begin{aligned} 5 &\rightarrow 1024 \\ U &\rightarrow 999 \\ U &= 4,88v \end{aligned}$$

Assim, foi calculada a queda de tensão em R_1: $U_{R1} = 5 - 4,88 \therefore U_{R1} = 0,12v$. Com essa informação, foi possível calcular R_1 conforme desenvolvimento abaixo, considerando que $U = 5v$ e $R_2 = 1k\Omega$:

$$\begin{aligned} U &= \frac{(R_1 + R_2) * U_{R1}}{R_1} \\ R_1 &= 25\Omega \end{aligned}$$

Figura 13 – Circuito limitador.



Fonte: Autores.

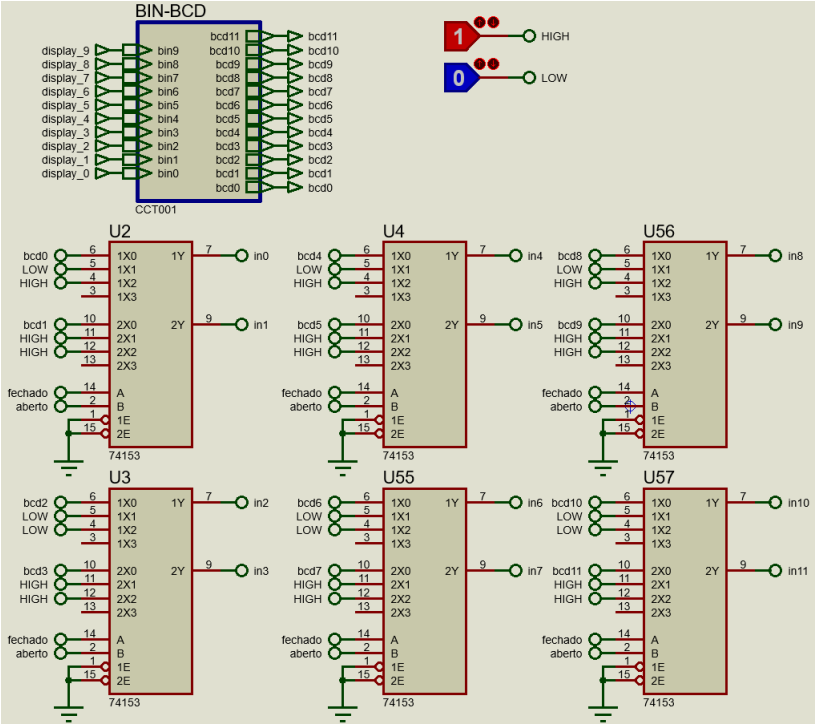
2.2.1.1 *Display*

O *display* é responsável por exibir a senha que está sendo manipulada pelo usuário, para que isso seja possível foi utilizado uma lógica externa ao microcontrolador, porém utilizando os 10 bits fornecidos por ele.

Tudo se inicia inserindo os 10 bits em um conversor responsável por converter o código binário para um padrão *Binary Coded Decimal* (BCD), esses bits foram configurados para serem fornecidos através das portas PB_n sendo $n = 0 \sim 7$, PD_3 e PD_4 do ATMEGA328P. O resultado da conversão é colocada em um multiplexador cujo o CI comercial utilizado foi o 74LS153, que irá receber também os valores ”1010” e ”1011” e

a chave seletora que também é disponibilizada pelo microcontrolador através das portas PC_3 e PC_4. A Figura 14 mostra o bloco BIN-BCD e os multiplexadores utilizados na implementação.

Figura 14 – BIN-BCD e Multiplexadores

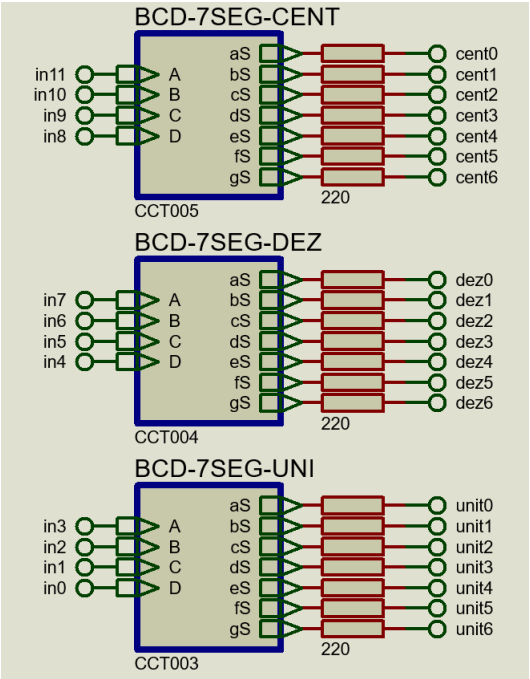


Fonte: Autores

A lógica é simples, quando o cofre está no estado inicial o *display* permanece apagado e a porta PC_3 fica em nível lógico alto, isso faz com que seja selecionado na saída do multiplexador o valor ”1010”, ao mudar para o estado de inserir a senha a porta PC_3 volta a nível lógico baixo e a saída do multiplexador será o valor binário da conversão do BIN-BCD, por fim, caso o usuário acerte as 3 senhas do cofre, a porta PC_4 adquire nível lógico alto e é feito a seleção na saída do multiplexador para o valor ”1011”.

A saída do multiplexador entra em um conversor responsável por decodificar o código BCD para o padrão que os *displays* de 7 segmentos conseguem decodificar. Esse decodificador transforma as entradas no range de ”0000” $\sim$ ”1011”, sendo ”0000” até ”1001” os valores decimais de $0 \sim 9$ e os outros dois valores são responsáveis por formar os padrões especiais do cofre, o desligado com nenhum segmento aceso e o padrão de aberto, onde só o segmento do meio é acendido. Na Figura 15 pode ser visto os decodificadores responsáveis por ligar os segmentos do display.

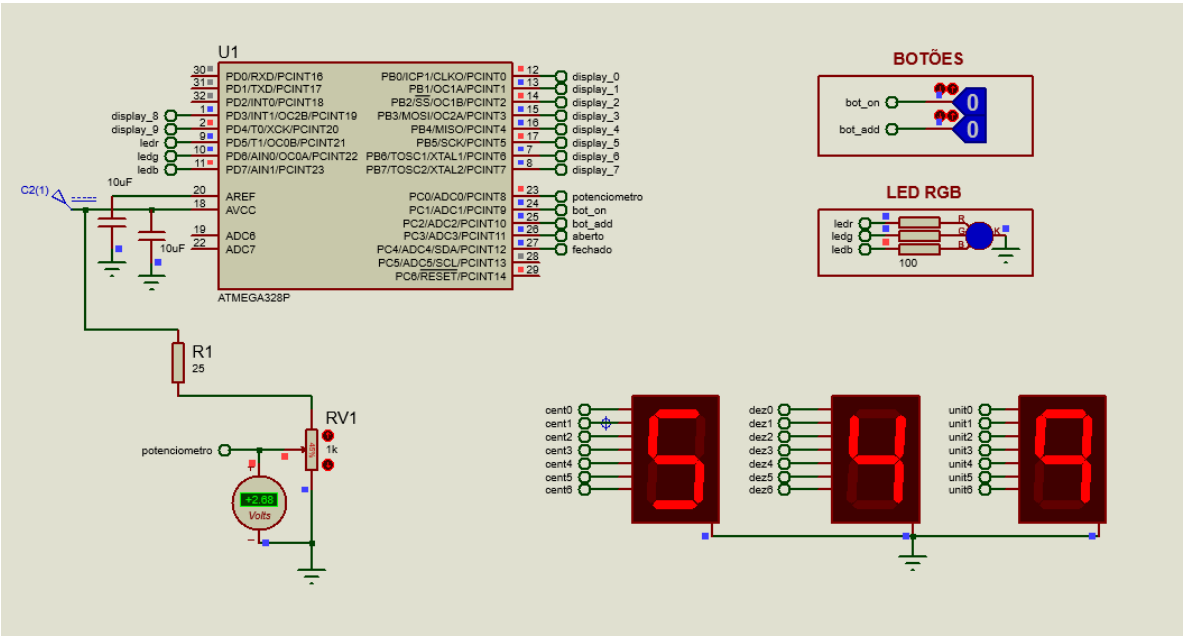
Figura 15 – Decodificadores BCD para 7 segmentos



Fonte: Autores

Por fim, a junção das lógicas explicadas nos tópicos 2.1 e 2.2 possibilitou a simulação do cofre digital no *Proteus*. Na Figura 16 pode-se observar o resultado final da implementação, nela é mostrado o estado em que o sistema aguarda a inserção da senha pelo usuário, o qual é nomeado "Espera Senha".

Figura 16 – Simulação no *Proteus*.



Fonte: Autores.

3 CONCLUSÃO

O ATMEGA328P é um microcontrolador comumente utilizado nos mais diversos sistemas digitais, principalmente, por ser simples e barato. Nesse relatório foi implementado um cofre digital utilizando o ATMEGA328P como microprocessador e *Assembly* como linguagem de programação. Além disso, para simular a interface homem-máquina foi utilizado o *software Proteus*.

A priori, o projeto de cofre digital projetado possuía algumas inconsistências, que foram corrigidas, e também foram feitas algumas alterações visando diminuir as variáveis utilizadas e facilitar o entendimento das operações.

No que tange a simulação, o código *Assembly* foi testado no *Microchip Studio* e no *Proteus* e ambos mostraram que o cofre se comporta conforme esperado, mostrando resultados coerentes com a ação executada.

Referências

MICROCHIP, C. *ATmega328P Datasheet*. 2015. Acessado em: 18 jul. 2021. Disponível em: <https://datasheetspdf.com/pdf-file/1469778/ATMEL/ATmega328P/1>.

MICROCHIP, C. *AVR® Instruction Set Manual*. 2021. Acessado em: 18 jul. 2021. Disponível em: <https://ww1.microchip.com/downloads/en/DeviceDoc/AVR-InstructionSet-Manual-DS40002198.pdf>.

ANEXO A – Relato semanal

Líder: Alysson Ferreira da Silva

A.1 Equipe

Tabela 1 – Identificação da equipe

Função no grupo	**Nome completo do aluno**
Redator	Isaac de Lyra Junior
Debatedor	Elias Gurgel de Oliveira
Videomaker	Ana Beatriz Marinho Neves
Auxiliar	Sthefania Fernandes Silva

Fonte: Autores.

A.2 Defina o problema

O problema consiste em implementar o projeto em um uC AVR (ATMega328p) que atue como um sistema digital capaz de controlar a abertura de um cofre. O código foi desenvolvido em assembly e o sistema possui 1 display que vai de 0 até 999, um potênciometro de volta completa para regulagem do valor, 2 pushbutton um para o power e outro para adicionar valores e um led RGB configurado para acender vermelho, azul, verde e laranja.

O sistema possui 4 configurações que são: Abertura de cofre, Falha na senha, Cancelamento e Trancando cofre. O procedimento de abertura de cofre inicia com o usuário apertando o botão power, isto acende o display e o led rgb setado para azul que logo passa para a etapa de ajuste de valor ondem devem ser feitas 3 ajustes de valor e 3 confirmação de valor. Para o usuário colocar a senha desejada na máquina deverá ser pressionado o botão de adicionar valor sempre que um valor for ajustado. Após ter inserido as 3 senhas e feito as 3 comparações o sistema deverá ser capaz de interpretar se a senha está correta ou não. Caso esteja correta o cofre deverá ser aberto e o led verde acendido.

O procedimento de falha na senha é similar ao de abertura de cofre. A parte destoante entre os 2 processos está no fato de que se a senha estiver incorreta após as 3 inserções e verificações a máquina deverá exibir a configuração inicial, cuja qual está configurada para ser led RGB setado para vermelho e display apagado. O procedimento de cancelamento pode ser nos estágios de inserção de senha ao apertar o botão power novamente, assim a máquina deverá voltar para seu estado inicial. Por fim, O processo de

trancar o cofre dar-se ao clicar o botão power após a sua abertura, voltando para o estado inicial onde o cofre está trancado.

A.3 Registro de *brainstorming*

O grupo não se reuniu após a aula do dia 13/07 e no dia 14/07, esses dias ficaram designados para estudar as funções do ATMega328p que seriam utilizadas durante a semana de implementação. Dito isso, a primeira reunião do grupo foi dia 15/07 onde foram decididos os cargos de cada componente do grupo que resultou em: Isaac (Redator), Elias (Debatedor), Ana (Videomaker) e Sthefania (Auxiliar).

Ademais, demos início a etapa de correção de projeto. Nesse dia foi realizda uma discussão conceitual sobre o PWM e todos os componentes do grupo concordaram que seria árdua a implementação do uso de um único Timer para implementação do PWM para o led laranja e o contador de 0.5s, pois o counter precisaria resetar em 15625 para reiniciar 0.5s (utilizando modo CTC) e o PWM precisaria ir até metade do valor do contador que é +- 32500 o que faria o contador resetar antes de bater o duty cicle em 50%. Por causa disso, decidimos usar o timer 0 e o timer 1 para realizar tais ações. Verificou-se a proposta para limitar os valores de entrada advindos do potênciometro e a função sugerida era da biblioteca dos arduinos o que dificultava a implementação do projeto somente em Assembly, por causa disso decidiu-se utilizar um circuito analógico para limitando o sinal de maneira externa ao ATMega328p.

No dia 16/07 o grupo deu início a implementação do projeto iniciando pelo conversor ADC. Neste momento o grupo identificou um bug no software proteus que deixava o teste do código incapaz de ser executado. A partir disso, o grupo dedicou a reunião deste dia para tentar resolver o bug encontrado mas não obtivemos sucesso. No dia 17/07, o grupo novamente se empenhou em resolver o bug no software ou a encontrar erros no código produzido no microship studio. Porém, após várias tentativas o grupo resolveu conversar com outros grupos e após algumas tentativas o código passou a funcionar sem alterações.

O dia 18 foi dedicado a completar o resto do projeto. Nesse sentido, foram produzidos os códigos referente ao PWM e os leds, ao timer de 05s, aos pushbuttons, a "máquina de estados finitos"e o circuito para limitar o sinal enviado pelo potênciometro. Todas as implementações citadas foram terminadas e o grupo conseguiu ao final do dia realizar a simulação do projeto com êxito. Com o projeto completado o dia 19/07 foi dedicado para elaboração do relatório.

A.4 Pontos-chaves

Podem ser listados como ponto chaves para a implementação do projeto o estudo dos mnemônicos dispostos no datasheet do ATMega328p, ou seja, a familiarização com o assembly e o estudo das funções disponíveis no ATMega e como fazer a configuração dos pinos, portas e registradores sejam eles de uso geral, sejam eles reservados para guardar flags de operação e constantes do sistema.

A.5 Questões de pesquisa

As pesquisas circurdaram o funcionamento do ATMega328p, ou seja, suas configurações e funcionalidades. Dito isso foram estudados: como fazer PWM utilizando os Timers, como configurar o Timer para realizar a contagem de 0.5s, quais são os mnêumonicos que serão utilizados, como configurar as portas e os pinos, como realizar a configuração de clock do sistema, como utilizar o prescale dos contados, etc. Eventualmente foram necessários pesquisas alguns erros e bugs que ocorriam durante a implementação do projeto.

A.6 Planejamento da pesquisa

As pesquisas surgiram conforme a necessidade do grupo. De início ficou acordado todos estudarem as funcionalidades do ATMega328p e como configura-las. Mas, as pesquisas direcionadas foram sendo destinadas ao longo da implementação. Nos dias 13 e 14 o grupo dedicou esforços no datasheet do ATMega328p, no dia 16 pesquisou-se como configurar o conversor ADC disponível no ATMega e tentou-se resolver os bugs relativos a implementação no proteus. No dia 17, a busca por soluções do bug supracitado contiuaram e no dia 18 pesquisou-se sobre as demais implementações do projeto como: PWM, Timers, Assembly, Portas/Pinos, etc.

ANEXO B – Código *Assembly*

```
;
; cofre.asm
;
; Created: 17/07/2021 19:27:45
; Author : Isaac, Alysson, Ana, Sthefania, Elias
;


; Replace with your application code
.include "m328pdef.inc"
.org 0x0000

config:
  ;DEFINICAO DAS SENHAS
  ldi r26, 0 ;XL
  ldi r27, 0 ;XH
  ldi r28, 0 ;YL
  ldi r29, 0 ;YH
  ldi r30, 0 ;ZL
  ldi r31, 0 ;ZH

  ;DEFINICAO DO PWM
  ldi r16, 0b10100011
  sts 0x44, r16 ; define TCCR0A
  ldi r16, 0b00000011
  sts 0x45, r16 ; define TCCR0B

  ;DEFINICAO DO TIMER 1
  ldi r16, 0x3d
  sts 0x89, r16 ; OCR1AH
  ldi r16, 0x09
  sts 0x88, r16 ; OCR1AL
  ldi r16, 0x00
  sts 0x80, r16 ; TCCR1A
  ldi r16, 0x00
  sts 0x82, r16 ; TCCR1C
  ldi r16, 0x23
  sts 0x6f, r16 ; TIMSK1
  ldi r17, 0x02
  ldi r16, 0x4c
  sts 0x81, r16 ; TCCR1B
```

```
  ;DEFINICAO DE ENTRADAS E SA DAS
  ldi r16, 0b11111111
    out DDRB, r16 ; define todas as portas PB como sa da
  ldi r16, 0b11111000
  out DDRD, r16 ; define inputs/outputs das portas PD
  cbi DDRC, 1
  cbi DDRC, 2
  sbi DDRC, 3
  sbi DDRC, 4

  ;DEFINICAO DO CONVERSOR AD
    ldi r16, 0b01000000
    sts ADMUX, r16 ; define ADMUX como sendo com REFS = 01, ou seja pega a
        tens o
            ; de refer ncia do AVCC em conjunto com capacitor
  ldi r16, 0b00000001
  sts DIDR0, r16

  rjmp desligado

setledred:
  ldi r16, 255
  sts 0x48, r16 ; define OCR0B
  ldi r16, 0
  sts 0x47, r16 ; define OCR0A
  cbi PORTD, 7
  ret

setledblue:
  ldi r16, 0
  sts 0x48, r16 ; define OCR0B
  ldi r16, 0
  sts 0x47, r16 ; define OCR0A
  ret

setledorange:
  ldi r16, 255
  sts 0x48, r16 ; define OCR0B
  ldi r16, 127
  sts 0x47, r16 ; define OCR0A
  cbi PORTD, 7
  ret

setledgreen:
  ldi r16, 0
  sts 0x48, r16 ; define OCR0B
```

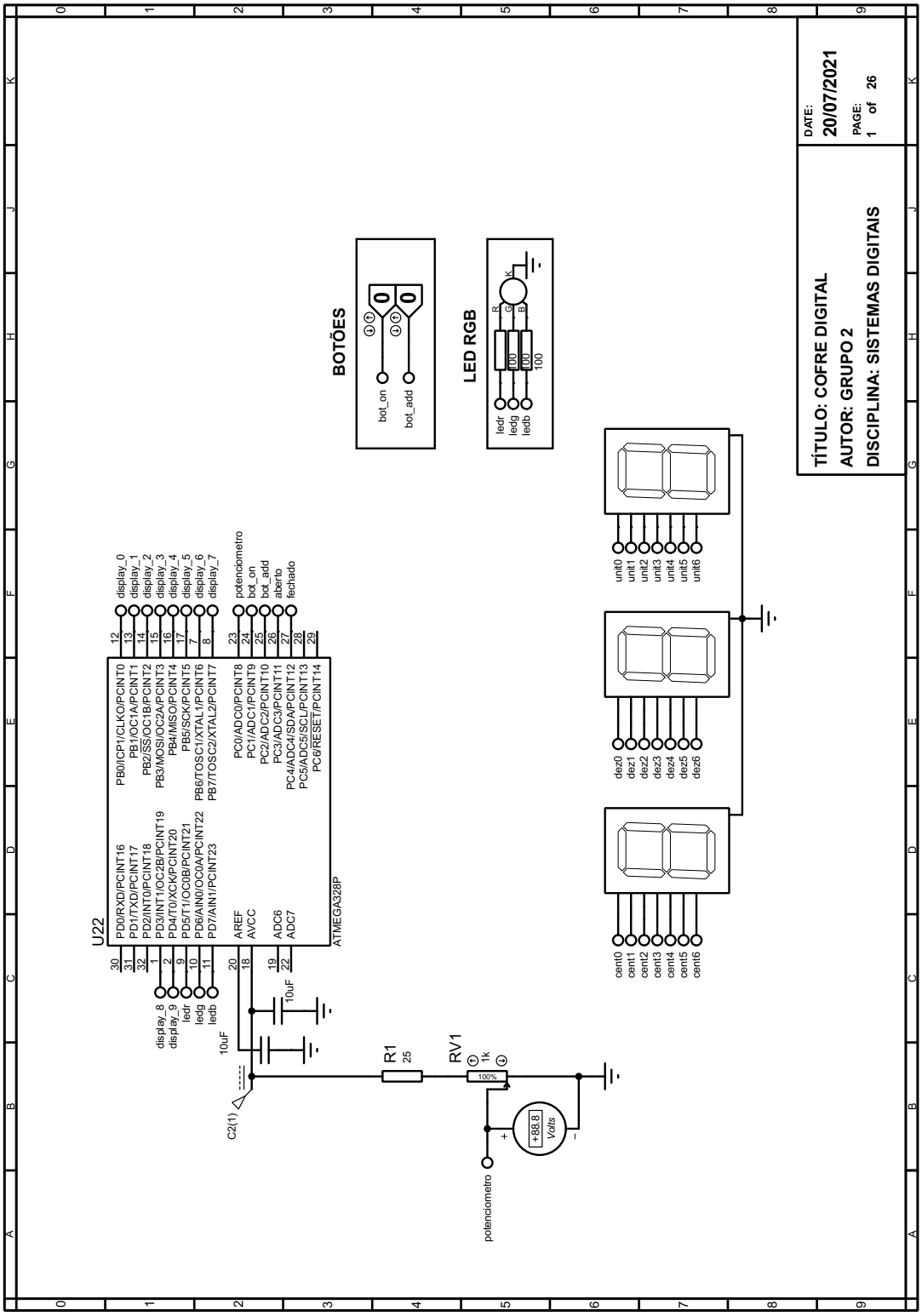
```
  ldi r16, 255
  sts 0x47, r16 ; define OCR0A
  cbi PORTD, 7
  ret

delay:
  clr r17
  ldi r18, 10
  delay_loop:
    dec r17
    brne delay_loop
    dec r18
    brne delay_loop
    ret

timer05:
  sbi TIFR1, 1
  comeca:
    sbis TIFR1, 1
    rjmp comeca
    ret

conv:
  ldi r16, 0b11100111
    sts ADCSRA, r16 ; define ADCSRA
  rcall delay
  lds r18, ADCL
  lds r19, ADCH
    lds r16, ADCL
  lds r17, ADCH
  cpi r17, 3
    breq atribuir3
    cpi r17, 2
    breq atribuir2
    cpi r17, 1
    breq atribuir1
  rjmp atribuir0

    atribuir3:
    ldi r17, 0b10011000
    rjmp saidapd
    atribuir2:
    ldi r17, 0b10010000
    rjmp saidapd
    atribuir1:
    ldi r17, 0b10001000
    rjmp saidapd
```

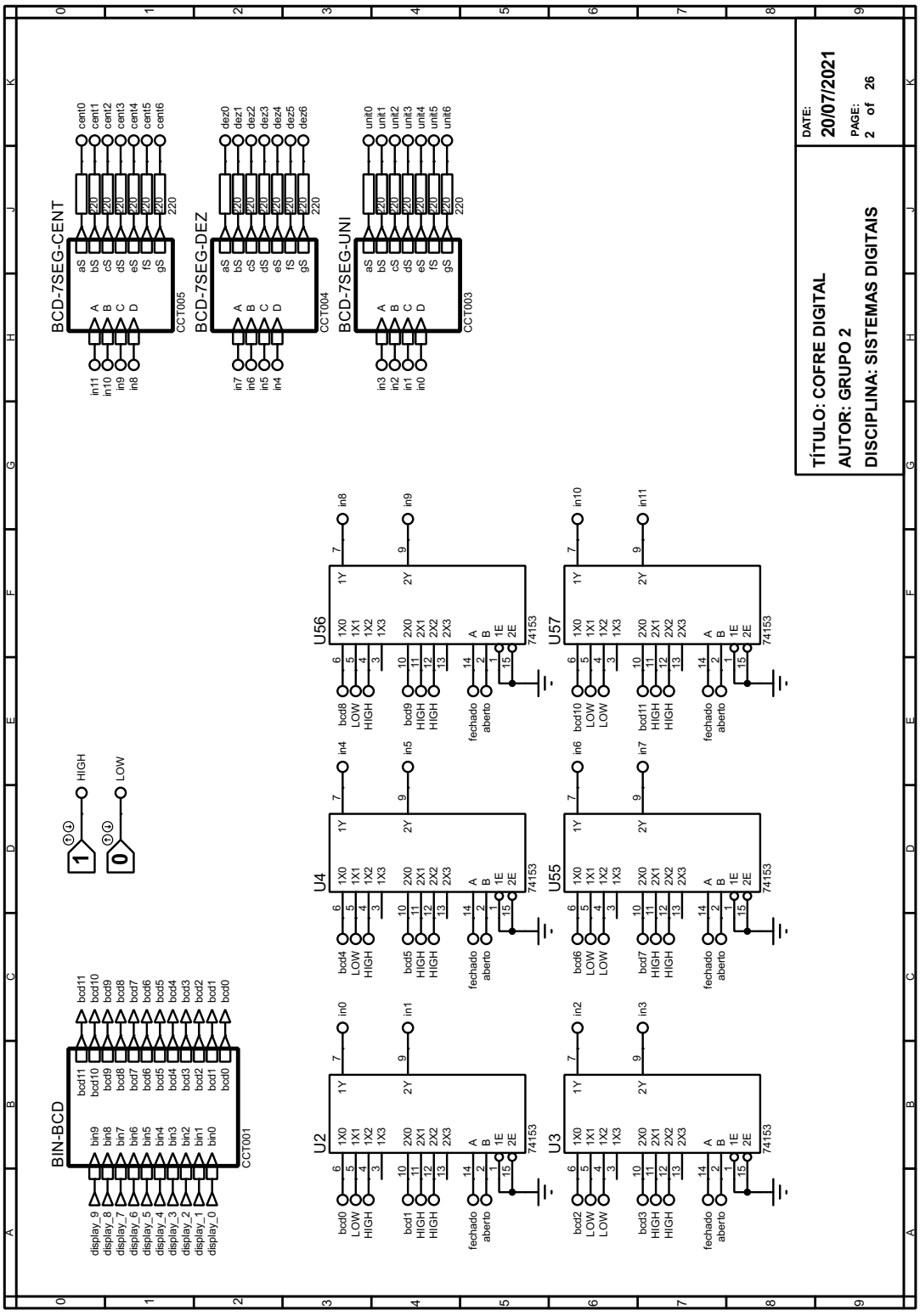
```
  atribuir0:
    ldi r17, 0b10000000
    rjmp saidapd

  saidapd:
    out PORTD, r17
    out PORTB, r16
    ret

desligado:
  rcall setledred
  cbi PORTC, 3
  sbi PORTC, 4
  sbis PINC, 1
  rjmp desligado
  rjmp botao_sinc

botao_sinc:
  sbic PINC, 1
  rjmp botao_sinc
  rjmp esperasenha

esperasenha:
  rcall setledblue
  cbi PORTC, 4
  rcall conv
  rjmp verificaon

verificaon:
  sbis PINC, 1
  rjmp verificaadd
  rjmp botao_sinc3

botao_sinc3:
  sbic PINC, 1
  rjmp botao_sinc3
  rjmp desligado

verificaadd:
  sbis PINC, 2
  rjmp esperasenha
  rjmp botao_sinc2

botao_sinc2:
  sbic PINC, 2
  rjmp botao_sinc2
  rjmp inseresenha
```

```
inseresenha:
  rcall setledorange
  ldi r16, 0b00000111
  sts ADCSRA, r16 ; desliga convers o
  rcall timer05
  cpi r20, 0
  breq comp_senha1
  cpi r20, 1
  breq comp_senha2
  cpi r20, 2
  breq comp_senha3

  comp_senha1:
    inc r20
    cpse r18, r26
    rjmp esperasenha
    cpse r19, r27
    rjmp esperasenha
    inc r21
    rjmp esperasenha

  comp_senha2:
    inc r20
    cpse r18, r28
    rjmp esperasenha
    cpse r19, r29
    rjmp esperasenha
    inc r21
    rjmp esperasenha

  comp_senha3:
    cpse r18, r30
    rjmp comparasenha
    cpse r19, r31
    rjmp esperasenha
    inc r21
    rjmp comparasenha

comparasenha:
  rcall setledorange
  cpi r21, 3
  breq aberto
  rjmp desligado

aberto:
  rcall setledgreen
```

```
  sbi PORTC, 3
  clr r21
  clr r20
  sbis PINC, 1
  rjmp aberto
  rjmp botao_sinc4

botao_sinc4:
  sbic PINC, 1
  rjmp botao_sinc4
  rjmp desligado
```

ANEXO C – Esquemático do projeto no Proteus




BCD-7SEG-CENT
BCD-7SEG-DEZ
BCD-7SEG-UNI
BIN-BCD
CCT001
CCT003
CCT004
CCT005
U2
U3
U4
U55
U56
U57
74153
HIGH
LOW
TÍTULO: COFRE DIGITAL
AUTOR: GRUPO 2
DISCIPLINA: SISTEMAS DIGITAIS
DATE: 20/07/2021
PAGE: 2 of 26