



**UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA MECATRÔNICA
SISTEMAS DIGITAIS**

IMPLEMENTAÇÃO DE CIRCUITO BASEADO EM MICROCONTROLADOR AVR(ATMega 328P)

Humberto Gama de Carvalho Neto - 20190001810
Oswaldo de Oliveira Vicente - 20200001097
Atyson Jaime de Sousa Martins - 20190153956
Alax Gabriel Cavalcante Lima de Oliveira - 20190001785

Natal, RN, Outubro de 2020

HUMBERTO GAMA DE CARVALHO NETO
OSWALDO DE OLIVEIRA VICENTE
ATYSON JAIME DE SOUSA MARTINS
ALAX GABRIEL CAVALCANTE LIMA DE OLIVEIRA

IMPLEMENTAÇÃO DE CIRCUITO BASEADO EM MICROCONTROLADOR AVR(ATMega 328P)

Sexto Relatório apresentado à disciplina de Sistemas Digitais, correspondente a sexta semana da disciplina do semestre 2020.6 do curso de Engenharia Mecatrônica da UFRN Referente à implementação de um circuito de um cofre baseado em microcontrolador AVR(ATMega 328P).

Professor: Samaherni Moraes Dias

Natal, RN, outubro de 2020

Sumário

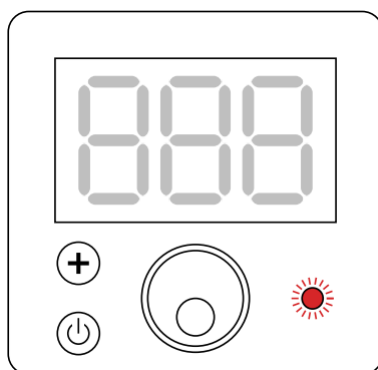
1	Introdução	1
1.1	Apresentação do Sistema	1
1.2	Etapas do Projeto.....	1
2	Desenvolvimento	2
2.1	Componentes do Circuito.....	2
2.1.1	Microcontrolador AVR ATmega328P	2
2.2	Periféricos	2
2.2.1	Displays de 7 segmentos	3
2.2.2	Decodificador 7 segmentos.....	3
2.2.3	Potenciômetro.....	4
2.2.4	LED RGB	4
2.2.5	Push Button	5
2.3	Montagem dos circuitos no Proteus	5
2.4	Fluxograma	6
2.5	Máquina de estados	8
2.6	Clock	10
2.7	Delay	10
2.8	Conversor ADC.....	11
2.9	Amostragem no display.....	14
3	Conclusão	13

1 Introdução

1.1 Apresentação do Sistema

O presente relatório tem o objetivo de descrever um sistema digital, assim como sua implementação, que controla o funcionamento de um cofre baseado no microcontrolador AVR (ATMega 328P). Para a abertura dele é necessário inserir uma senha que é composta por três valores variando de 0 a 999. Esses valores são inseridos através de um botão rotativo e um botão de confirmação. O sistema ainda possui um botão para iniciar/cancelar o processo de inserção de senha e um LED auxiliando o usuário. A interface do sistema é demonstrada na figura 1.

Figura 1: Interface do sistema



Fonte: Disponibilizado pelo professor

O processo de abertura do cofre segue os seguintes passos: primeiro, deve-se pressionar o botão *Power* para iniciar, ajustar o valor desejado no botão rotativo e então pressionar o botão *Add*. O processo deve ser repetido três vezes. Antes de iniciar este processo o LED deve estar na cor vermelha, na fase de ajuste do valor passado a cor deve estar azul, na fase de confirmação do valor a cor deve permanecer laranja por 0,5 segundos. Ao fim deste processo, caso a senha esteja correta a cor será verde sinalizando a abertura do cofre, caso contrário sua cor voltará para o vermelho do início, podendo repetir o processo.

Neste projeto, a senha correta será definida previamente via software e não poderá ser alterada pelo usuário.

1.2 Etapas do Projeto

Os componentes eletrônicos necessários para a implementação do circuito são: microcontrolador AVR ATMega 328P, 2 pushbuttons, 1 potenciômetro, 3 displays de 7 segmentos, 1 Led RGB, 2 capacitores, 1 transistor, 1 CI 74HC14, 3 CI BCD 7446, 22 portas NOT e 28 resistores. Esses componentes são detalhados ao longo do relatório.

Outra fase importante no desenvolvimento do projeto foi a máquina de estados de alto nível que é responsável por toda a lógica do comportamento do sistema. Por fim temos a etapa de implementação por meio de instruções passadas ao microcontrolador via linguagem Assembly. Utilizamos o Proteus para fazer os circuitos e simular o código.

2 Desenvolvimento

2.1 Componentes do Circuito

2.1.1 Microcontrolador AVR ATMega328P

O principal componente do sistema é o AVR ATMega 328P, esse microcontrolador de 8 bits combina alta performance com baixo consumo de energia, baseado em arquitetura RISC avançada, ele possui 131 instruções usando 32 registradores de propósito geral. Esse microcontrolador possui 28 pinos entre eles estão I/Os digitais e analógicos com funcionalidades de PWM (Modulação por Largura de Pulso, do inglês Pulse Width Modulation) e ADC (Conversor Analógico-Digital, também do inglês Analog-to-Digital Converter).

Para a implementação do circuito serão utilizados 18 pinos I/O. Sendo 17 digitais e 1 analógico. Sendo os digitais: 12 pinos de saída para os BCDs, 3 pinos de saída para o LED RGB, 1 pino de entrada para o botão ligar, 1 pino de entrada para o botão add. Quanto ao analógico temos 1 pino de entrada para o potenciômetro.

O clock a ser trabalhado no projeto será o de 128Khz, mais a frente explicaremos melhor.

2.2 Periféricos

Para o circuito do cofre foi proposto que usássemos os seguintes componentes para a inserção e verificação da senha, assim como operação do sistema como um todo.






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 <i>pushbutton</i>)
	Botão de adicionar valor (tipo <i>pushbutton</i>)
	Led RGB (Fechado; Senha; Processando; Aberto)

Figura 2: Componentes necessários para o funcionamento do circuito
Fonte: Disponibilizado pelo professor

2.2.1 Displays de 7 segmentos

Inicialmente foi pensado em usar um método de multiplexação temporal baseada na persistência da visão. Esse método foi pensado porque o olho humano é incapaz de perceber alterações de um sinal luminoso, sem perceber nenhuma cintilação, com frequências acima de 48Hz, o que significa que se um LED piscar 48 vezes durante 1 segundo, ele aparentará aos olhos humanos como ligado constantemente.

Porém, devido a dificuldade em fazer esse método em Assembly no tempo disponibilizado, tendo em vista que tinham outros parâmetros a serem definidos e implementados, foi decidido utilizar três displays separados sendo um para cada: unidade, dezena e centena.

2.2.2 Decodificador 7 segmentos

Para transformar os bits que saem do microcontrolador em uma exibição decimal nos displays é necessário um decodificador. O utilizado no projeto é o BCD 7446 para converter para 7 segmentos.

Como foram utilizados três displays para representar a unidade, dezena e centena foram necessários 12 bits de saídas do microcontrolador, sendo 4 para o BCD referente à centena, 4 para a dezena e 4 para a unidade.

2.2.3 Potenciômetro

Para o ajuste dos valores da senha foi utilizado um potenciômetro linear rotativo com resistência de 10 k Ω . O potenciômetro possui três pinos VCC, GND, conectados respectivamente no microcontrolador e o terceiro pino onde é realizada a leitura da resistência conectado a uma porta ADC (PC0) do ATmega328P. Para converter o valor analógico para digital foi utilizado o conversor ADC do próprio microcontrolador.

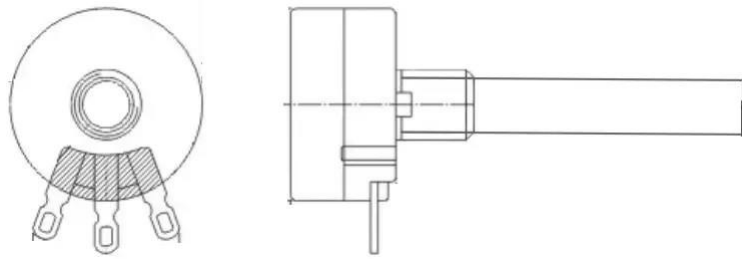


Figura 3: Potenciômetro linear rotativo
Fonte: autoria própria (Antigo grupo 1)

2.2.4 LED RGB

Foi escolhido em projeto um led RGB de catodo comum que é conectado a três portas digitais do microcontrolador (PB5...PB7) para a composição de cores de acordo com o estado do cofre. As portas referentes ao RGB serão conectadas com a adição de resistores de 100 e 150ohms.

Dessa forma é possível atingir as 4 cores necessárias para o projeto, com uma ressalva do laranja, não é possível obtê-la com essa configuração, com isso, decidimos usar a cor amarela para representar o laranja, uma vez que é possível atingi-la combinando o R (Red/Vermelho) com o G (Green/Verde). Decidimos fazer dessa forma para facilitar o entendimento e a montagem do circuito não sendo necessário usar portas PWM.

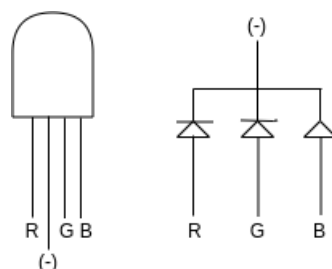


Figura 4: LED rgb de catodo comum
Fonte: autoriaprópria (Antigogrupol)

2.2.5 Push Button

O circuito dos pushbuttons usa uma técnica de debouncing por hardware, onde são utilizados dois resistores, um capacitor e uma porta inversora do CI 74HC14 (Schmitt Trigger).

Escolhemos essa técnica para evitar o efeito bouncing que seria um problema de oscilação do sinal no momento em que a tecla é pressionada podendo acontecer do processador acreditar que a chave foi pressionada mais de uma vez em intervalos curtos.

De forma breve, nesse circuito o capacitor faz um papel de amortecedor do sinal, criando um tempo de atraso para o sinal necessário para carregar o capacitor. Com isso as alterações rápidas do sinal são filtradas evitando o problema do bouncing.

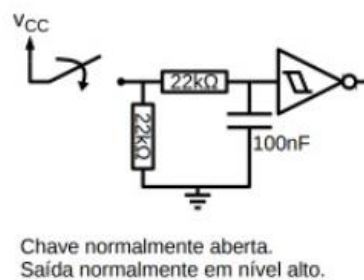


Figura 5: montagem de um pushButton normalmente aberto
Fonte: PDF: Aula 10B – Circuitos de Debounce por Nikolas Libert.

2.3 Montagem dos circuitos no Proteus

Abaixo é mostrado detalhadamente os circuitos de todos os periféricos falados acima, relacionando com os resistores necessários, e suas respectivas entradas e saídas para com o ATmega.

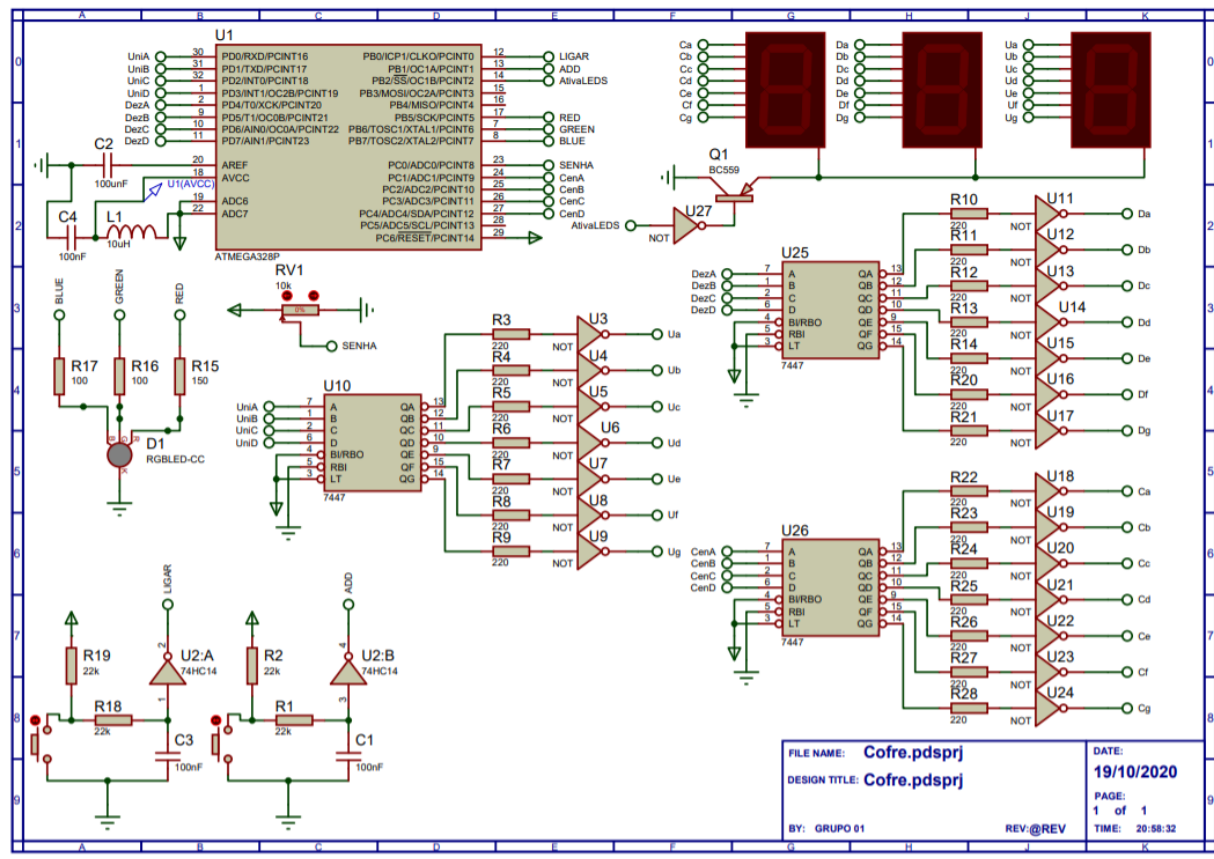


Figura 6: Montagem dos circuitos
 Fonte: Autoria Própria

2.4 Fluxograma

O fluxograma utilizado é o mesmo do grupo anterior, decidimos não modifica-lo pois aparenta está correto, a máquina de estados segue o seu funcionamento.

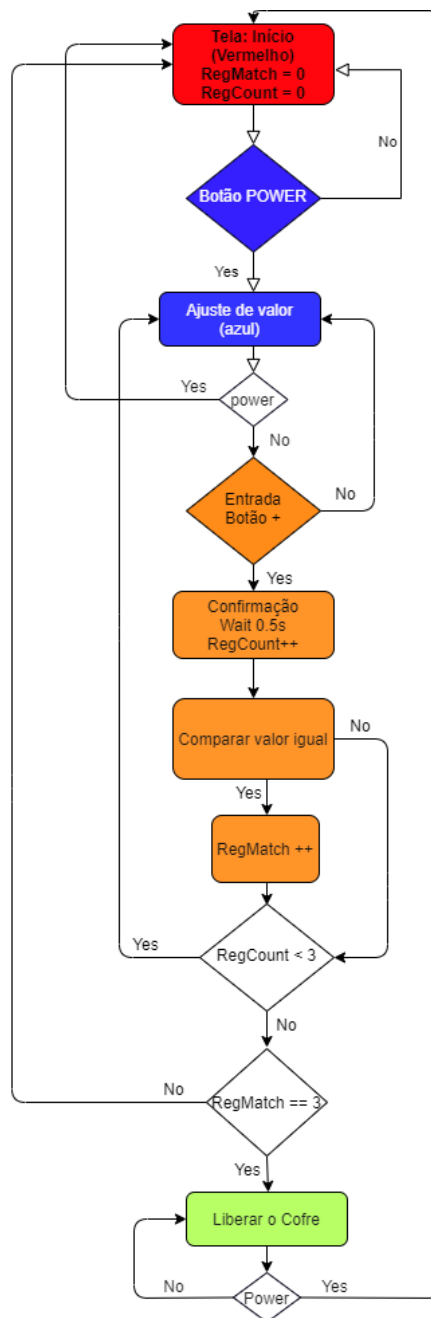


Figura 7: Fluxograma do funcionamento do projeto
 Fonte: Autoria própria (Antigo grupo 1)

2.5 Máquina de estados

Anteriormente foram feitas três máquinas de estados para demonstrar o processo de construção da máquina final, porém se tornou um pouco confuso, por isso, decidimos implementar apenas a última máquina, pois demonstra estar mais correta.

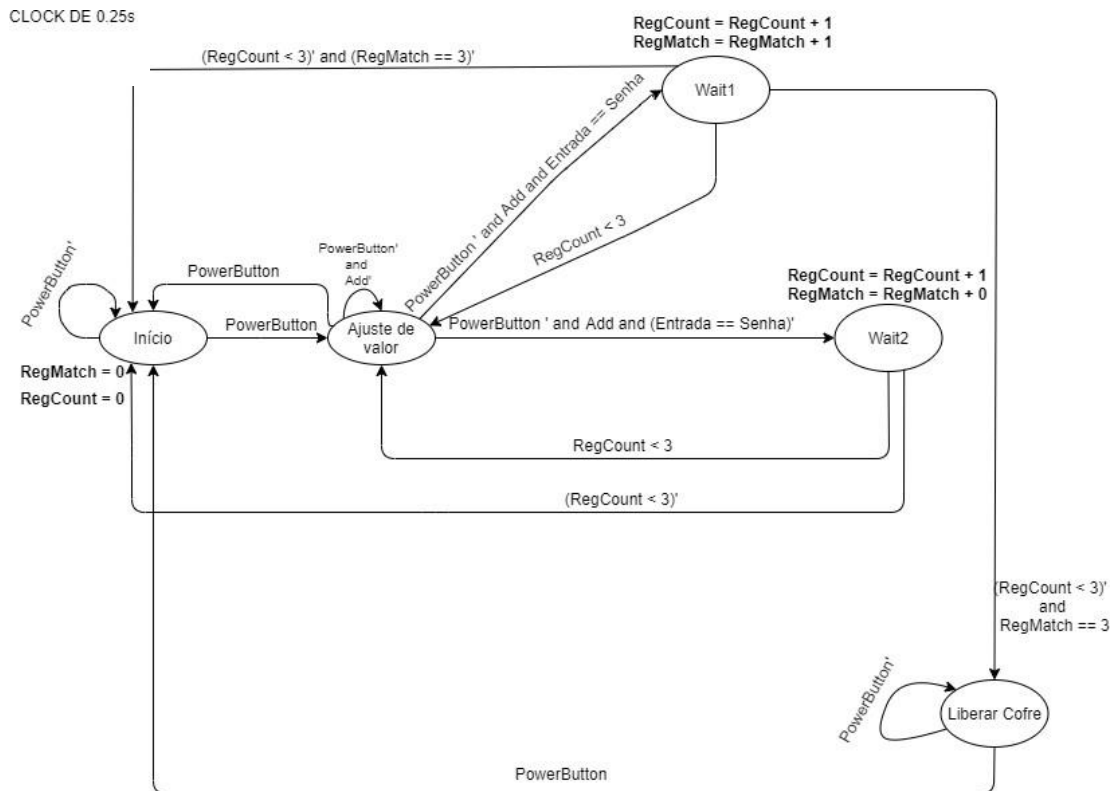


Figura 8: Máquina de estados
Fonte: Autoria própria (Antgo grupo 1)

Na máquina de estados os estados de Wait vão guardar o valor da quantidade erros ou acertos de acordo com a senha inserida. Para isso, foi adicionado um outro estado de Wait, (Wait2), no caso em que não haja o acerto da senha. Em ambos os casos, a distribuição de estados é a ideal, ou seja, depois do "ADD" da senha, o próximo estado será o de wait, e, por conseguinte, já voltará imediatamente para o estado de Ajuste de Valor, como deveria ser. Neste caso, a conexão entre o estado de Wait e Liberar Cofre será feita apenas para o Wait1, visto que essa possibilidade (liberar o cofre) nunca irá acontecer caso passe pelo estado de Wait2, que significa que pelo menos 1 das 3 entradas

foram erradas.

Estado "Início-> caso o PushButton de inicializar/cancelar o processo, o qual chamamos de PowerButton, esteja Desativado, irá permanecer no estado. Caso contrário, se for ativado, a máquina será direcionada para o estado de "Ajuste de valor". Além disso, nesse estado, serão zerados os registradores responsáveis por guardar a quantidade de valores colocados, bem como a quantidade de comparações de equidade entre as entradas e senhas.

Estado "Ajuste de valor" -> Caso seja pressionado o PowerButton, haverá o direcionamento para o estado de Início. Caso o Powerbutton não seja pressionado e o botão de adicionar valor, o qual chamamos de ADD, também não seja pressionado, irá permanecer no mesmo estado. Na hipótese do PowerButton estar desativado e ADD ativado, dependerá da comparação, como mostrada no fluxograma, entre a Entrada e a Senha. Caso igual, a direção será o Estado Wait1, caso diferente, Wait2.

Estado "Wait2-> O estado Wait2 corresponde a hipótese em que a entrada do usuário não foi igual à senha registrada. Portanto, irá haver o incremento na contagem de vezes que o usuário adicionou um valor e, posteriormente, se essa contagem não tiver chegado em 3, retornará ao estado de "Ajuste de Valor". Caso tenha chegado em 3, retornará ao início.

Estado "Wait1-> O estado Wait1 corresponde a hipótese em que a entrada do usuário foi igual à senha registrada. Portanto, haverá um incremento na quantidade de vezes em que a entrada foi igual à senha e, também, irá haver o incremento na contagem de vezes que o usuário adicionou um valor. Posteriormente, se a contagem de quantidade de inserções de valor não tiver chegado em 3, retornará ao estado de "Ajuste de Valor". Caso essa tenha chegado em 3 e o registrador que guarda a quantidade de acertos de senha não estiver em 3, retornará ao início. Por fim, se a quantidade de inserções de valor tenha chegado em 3 e também constar 3 equidades entre as entradas e senhas, o destino será o estado de "Liberar Cofre".

Estado "Liberar Cofre-> Finalmente, após a tranca ser liberada, caso não haja o pressionamento do PowerButton, a tranca continuará no estado de "Liberar Cofre", ou seja, liberada. Caso o usuário deseje trancar, novamente, basta apertar o PowerButton e haverá o direcionamento para o estado de Início.

2.6 Clock

Como dito anteriormente o clock utilizado foi o de 128KHz. Ele foi escolhido para ser utilizado por ser uma das opções de clock disponíveis no ATmega 328P, sendo necessário apenas configurar os bits do vetor de seleção interno do microcontrolador: CKSEL[3..0] para “0011”. Seu período é menor que o tempo de espera necessário de 0,5s.

8.2 Clock Sources

The device has the following clock source options, selectable by flash fuse bits as shown below. The clock from the selected source is input to the AVR[®] clock generator, and routed to the appropriate modules.

Table 8-1. Device Clocking Options Select⁽¹⁾

Device Clocking Option	CKSEL3..0
Low power crystal oscillator	1111 - 1000
Full swing crystal oscillator	0111 - 0110
Low frequency crystal oscillator	0101 - 0100
Internal 128kHz RC oscillator	0011
Calibrated internal RC oscillator	0010
External clock	0000
Reserved	0001

Note: 1. For all fuses “1” means unprogrammed while “0” means programmed.

Figura 9: Fontes de clock
Fonte: Datasheet do AVR ATmega 328P

2.7 Delay

Uma das imposições feitas pelo projeto se referiu ao tempo de espera de meio segundo entre a adição de uma senha e a mudança de cor do led para que outro número seja inserido. Para tal tarefa, pensamos em implementar da seguinte forma.

Foi definido via código uma sub-rotina com uma série de comandos "inúteis", cada qual gastando ciclos de clock. Assim como no modo de utilização dos timers, se faz necessário calcular quantos ciclos de clock geram uma parada de 0,5 segundos, e assim calcula-se quantas vezes o laço deve ser ativado.

No livro “AVR e Arduino Técnicas de Projeto” é mostrado um exemplo de código em assembly, para o microcontrolador estudado, de chamada de sub-rotina de atraso para manter um led aceso por determinado tempo.

```

DEC R3      //Decrementa o registrador R3, começa com o valor 0x00.
BRNE Atraso //Enquanto R3>0 fica decrementando R3, desvio para atraso.
DEC R2      //Decrementa R2, começa com o valor 0x00.
BRNE Atraso //Enquanto R2 > 0 volta a decrementar R3.
RET         //Retorno da sub-rotina.

```

Dessa forma, a instrução DEC consome um ciclo, a instrução BRNE consome dois ciclos e na última vez, quando não há mais desvios, consome um ciclo. Como os valores de R3 e R2 começam em zero, o primeiro decremento leva ao valor 255 (considerando que ambos são números de 8 bits) e o decremento de R3 é repetido dentro do laço de R2, espera-se que haja 256 decrementos de R3 vezes 256 decrementos de R2, considerando 3 ciclos para DEC e BRNE temos aproximadamente $(3 \times 256 \times 256) + (3 \times 256)$ o que resulta em 197.376 ciclos. A parcela 3×256 é o tempo gasto no decremento de R2.

Nesse exmplo foi utilizado um clock de 16Mhz (período de 62,5 ns), com isso temos que da chamada da sub-rotina ate o seu retorno foi gasto cerca de 12,23 ms.

Como o clock utilizado no nosso projeto é de 128Khz (período de 7,81us), a quantidade de ciclos necessária para obter-se 0,5 segundos é de aproximadamente:

$$X = 0,5 / 7,81\mu s \rightarrow X = 64.020 \text{ ciclos}$$

O que resulta em um laço de decretação de R3 em R2 com ambos os valores iniciando em 146, dando assim um total de $(3 \times 146 \times 146) + (3 \times 146) = 64.386$ ciclos.

2.8 Conversor ADC

Um dos pontos principais para serem estudados foi como seria feita a leitura do potenciômetro em Assembly.

Para essa primeira questão foi visto que para habilitar a conversão ADC do microcontrolador é necessário inicializar dois registradores: ADCSRA e ADMUX. Ambos são de 8 bits.

Cada bit desses registradores tem uma determinada função, no caso do ADCSRA o

bit 7 (ADEN) e 6 (ADSC) dão enable e iniciam a conversão respectivamente, ambos devem estar em 1 para que isso ocorra. O bit 4 (ADIF) é setado em 1 quando a conversão ADC tiver sido completada e os registros de dados atualizados. Os bits de 0 a 2 (ADPS 2:0) determinam o fator de divisão entre a frequência do sistema e o clock de entrada para o conversor ADC. Por padrão a frequência do conversor ADC precisa ser entre 50Khz e 200Khz, no projeto estamos utilizando um clock de 128Khz, ou seja, já esta dentro da margem, porém, foi utilizado o fator de conversão 2 [001], resultando em 64Khz.

Table 21-5. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Figura 10: Tabela de redução de frequência

Fonte: Datasheet do AVR ATmega 328P

Quanto ao ADMUX os bits 7 e 6 (REFS 1:0) escolhem a tensão de referência para o conversor ADC que pode ser escolhida entre essas opções. A opção escolhida foi a de [01], Avcc.

Table 21-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V Voltage Reference with external capacitor at AREF pin

Figura 11: Tabela de seleção da tensão de referência

Fonte: Datasheet do AVR ATmega 328P

A conversão ADC segue a seguinte fórmula.

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

Figura 12: Equação de conversão ADC
Fonte: Datasheet do AVR ATmega 328P

Onde o ADC é o valor máximo que o potenciômetro pode atingir que no caso é de 999 e o VIN é a tensão máxima do potenciômetro que é de aproximadamente 4,88V. Dessa forma podemos encontrar que a tensão de referência é de aproximadamente 5V. Foi necessário utilizar um filtro passa-baixa no pino do AVCC.

O bit 5 (ADLAR) é o bit que vai definir como será salvo os resultado da conversão.

21.9.3.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
(0x79)	–	–	–	–	–	–	ADC9	ADC8	ADCH
(0x78)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

21.9.3.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	ADC1	ADC0	–	–	–	–	–	–	ADCL
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Figura 13: Tabela do ADLAR
Fonte: Datasheet do AVR ATmega 328P

O resultado da conversão é salvo em dois registradores de 8 bits: ADCH e ADCL que é feito de acordo com o bit ADLAR. No projeto foi utilizado o ADLAR = 0 resultando nos 2 MSB salvos em ADCH e outros em ADCL.

Por fim, os bits 3..0 são responsáveis por selecionar a entrada do microcontrolador a ser conectada ao conversor, no projeto nos utilizamos a primeira opção: ADC0 [0000].

Table 21-4. Input Channel Selections

MUX3..0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8 ⁽¹⁾
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

Figura 14: Tabela do ADLAR
 Fonte: Datasheet do AVR ATmega 328P

2.9 Amostragem no display

A segmentação dos valores para os display é baseada numa divisão de grupos seguida de sucessivas subtrações. Com a divisão do valor lido em ADCH e ADCL, há 4 combinações possíveis:

- $ADCH = "00000000" + ADCL = "xxxxxxx"$, o que representa, na prática, 000 + um valor até 255;
- $ADCH = "00000001" + ADCL = "xxxxxxx"$, o que representa, na prática, 256 + um valor até 255;
- $ADCH = "00000010" + ADCL = "xxxxxxx"$, o que representa, na prática, 512 + um valor até 255;
- $ADCH = "00000011" + ADCL = "xxxxxxx"$, o que representa, na prática, 768 + um valor até 255;

De acordo com a combinação dos dois bits menos significativos do ADCH, os valores práticos de centena, dezena e unidade serão somados aos valores segmentados do ADCL.

Por exemplo: se o valor é 527, ADCH contém "00000010" e ADCL contém "00001111". O valor prático de ADCH representa 512, enquanto o valor prático de ADCL representa 15. ADCL, então, será sujeito a repetidas subtrações de modo que o registrador da centena

contenha 0, o registrador da dezena contenha 1 e o registrador da unidade contenha 5. Em seguida, o registrador da centena é somado com 5, o registrador da dezena é somado com 1 e o registrador da unidade é somado com 2, resultando assim nos dígitos 5, 2 e 7 nos registradores.

3 Conclusão

O código se encontra no ANEXO 1 e sua simulação no Proteus, no vídeo. Ele funcionou corretamente, passando por todos os estados e realizando a verificação das senhas, esperando o intervalo de 0,5 a cada inserção de senha e liberando o cofre quando houvesse o acerto das três senhas ou trancando o cofre caso houvesse alguma senha errada, o único problema foi na parte de amostragem do valor do potenciômetro do display, apenas a unidade está funcionando corretamente, a dezena e centena ficam alternando valores repetidamente. Esse problema se deu porque o valor do potenciômetro está sendo atualizado a todo instante nos displays, para resolver isso seria necessário criar um estado ou algum label de verificação de conversão, onde apenas atualizaria o valor caso houvesse alguma variação do valor do potenciômetro, porém, devido ao tempo não conseguimos consertar.

Referências

- [1] VAHID, Frank. , *Sistemas digitais: projeto, otimização e HDLs.*, Artmed, Porto Alegre, 2008.
- [2] *ATMega 328P Datasheet*. Disponível em: <https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf> Acesso em: 10 de out. de 2020.
- [3] *Display Datasheet*. Disponível em: <<https://pdf1.alldatasheet.com/datasheet-pdf/view/703367/BETLUX/BL-T56A-31.html>> Acesso em: 10 de out. de 2020.
- [4] *Decoder Datasheet*. Disponível em: <https://www.ti.com/lit/ds/sdls111/sdls111.pdf?ts=1602356882572&ref_url=https%253A%252F%252Fwww.google.com%252F> Acesso em: 10 de out. de 2012
- [5] Leituras de chaves mecânicas e o processo de debouncing. **Embarcados**. Disponível em: < <https://www.embarcados.com.br/leitura-de-chaves-debounce/> > Acesso em: 17/10/2020
- [6] E-book Avr e Arduino: Técnicas de projeto. Disponível em: <<https://pt.scribd.com/read/421356943/Avr-E-Arduino-Tecnicas-De-Projeto#>> Acesso em: 15/10/2020

ANEXO 1

CÓDIGO EM ASSEMBLY

```
;=====
; Main.asm file generated by New Project wizard
; GRUPO 01
; Created: dom out 18 2020
; Processor: ATmega328P
; Compiler: AVRASM (Proteus)
;=====

;=====
; DEFINITIONS
;=====
;=====
; VARIABLES
;=====

; Reset Vector

    ldi r16, 0b1111_1100 ; Configura PB0, PB1 como entradas e as demais como saída
    out ddrb, r16

    ldi r16, 0x00ff ; Configura PORTD como saída;
    out ddrd, r16

    ldi r16, 0b0111_1110 ; Configura PC0, PC7 como entrada;
    out ddrc, r16

    ldi r16, 0b01000000 ; Seta a tensão de referência como AVCC e ADLAR em 0 (alinhamento a
direita), port em ADC0
    sts ADMUX, r16

    ldi r16, 0b01010001 ; MSMB em 0 = ADC DESLIGADO! (LIGA EM AJUSTE 1). Interrupt FLAG
0, Interrupt disabled, fator de divisão 2
```

```
sts ADCSRA, r16
```

```
rjmp Start
```

```
;=====
```

```
; CODE SEGMENT
```

```
;=====
```

Start:

```
; Write your code here
```

Inicio:

```
sbi portB,5 ; LIGAR LED VERMELHO
```

```
cbi portB,6 ; DESLIGA LED VERDE
```

```
cbi portB,7 ; DESLIGA LED AZUL
```

```
cbi portB,2 ; DESLIGA OS DISPLAYS
```

```
clr r21 ; Limpa o registrador de acertos
```

```
clr r20 ; Limpa o registrador de contagem
```

```
sbic pinB,0 ; Verifica se o botão LIGAR está em zero, se sim, pula uma instrução, se não, ler  
proxima instrução
```

```
rjmp AJUSTE1
```

```
rjmp Inicio
```

AJUSTE1:

```
cbi portB,5 ; DESLIGA LED VERMELHO
```

```
cbi portB,6 ; DESLIGA LED VERDE
```

```
sbi portB,7 ; LIGAR LED AZUL
```

```
sbi portB,2 ; LIGAR OS DISPLAYS
```

```
ldi r16, 0b11010001 ;MSB em 1 = ADC LIGADO! Desligado em ajuste 2. Interrupt FLAG 0,  
Interrupt disabled, fator de divisão 2
```

```
sts ADCSRA, r16
```

```
TESTEBIT: LDS R24, ADCSRA
```

```
SBRs R24, 4
```

```
RJMP TESTEBIT
```

lds r23, ADCL ; Joga os oitos LSBS da conversão em R23

lds r22, ADCH ; Joga os dois MSBS da conversão em R22

rcall DISPLAY

sbic pinB,0 ; Verifica se o botão LIGAR está em zero, se sim, pula uma instrução, se não, ler próxima instrução

rjmp Inicio

sbic pinB,1 ; Verifica se o botão ADD está em zero, se sim, pula uma instrução, se não, ler próxima instrução

rjmp AJUSTE2

rjmp AJUSTE1

AJUSTE2: ;aqui já entra com powerbutton = 0 e add = 1

ldi R16,0b01010001

sts ADCSRA,R16

RCALL AJUSTESENHA

INC R20

CPSE R22, R18

RJMP WAIT2

CPSE R23, R19

RJMP WAIT2

RJMP WAIT1

WAIT1:

SBI PORTB, PB6 ;ligar led verde

SBI PORTB, PB5 ;ligar led vermelho

cbi portB,7 ; LIGAR LED AZUL

LDI R16, 0b10010010 ; Carrega R16 com 146

LDI R17, 0b10010010

RCALL ATRASO ;Chama a sub-rotina de atraso

CBI PORTB, PB6 ;Desliga o led verde

CBI PORTB, PB5 ;Desliga o led vermelho

INC R21 ; Incrementa o reg_match

```
CPI R20, 3 ;Compara R20 com 3  
BRLO AJUSTE1 ;Desvia para ajuste 1 se R20<3  
CPI R21,3  
BRLO INICIO ;Desvia para o INICIO se R21<3  
RJMP LIBERAR
```

WAIT2:

```
SBI PORTB, PB6 ;ligar led verde  
SBI PORTB, PB5 ;ligar led vermelho  
cbi portB,7 ; LIGAR LED AZUL  
LDI R16, 0b10010010 ; Carrega R16 com 146  
LDI R17, 0b10010010  
RCALL ATRASO ;Chama a sub-rotina de atraso  
CBI PORTB, PB6 ;Desliga o led verde  
CBI PORTB, PB5 ;Desliga o led vermelho  
CPI R20, 3 ;Compara R20 com 3  
BRLO AJUSTE1 ;Desvia para ajuste 1 se R20<3  
RJMP INICIO
```

LIBERAR:

```
cbi portB,5 ; DESLIGA LED VERMELHO  
cbi portB,7 ; DESLIGA LED AZUL  
sbi portB,6 ; LIGAR LED VERDE  
ldi r16, 0x0000  
out portB, r16 ;ACIONA PADRAO DOS DISPLAYS  
cbi portC, 1  
cbi portC, 2  
cbi portC, 3  
cbi portC, 4  
  
sbic pinB,0 ; Verifica se o botão LIGAR está em zero, se sim, pula uma instrução, se não, ler  
proxima instrução
```


rjmp Inicio
rjmp LIBERAR

ATRASO:

DEC R16 ; Decrementa R16 começando em 146
BRNE ATRASO ; Enquanto R16>0 fica decrementando R16
LDI R16, 0b10010010
DEC R17 ; Enquanto R17>0 volta a decrementar R17
BRNE ATRASO
RET

AJUSTESENHA:

cmp R20, 0 ;comparar regcount com 0
BREQ SENHA0
cmp R20, 1 ;comparar regcount com 1
BREQ SENHA1
cmp R20, 2 ;comparar regcount com 2
BREQ SENHA2
RET

SENHA0: ;015

LDI R18,0b00000000
LDI R19,0b00000000
ret

SENHA1: ;396

LDI R18,0b00000000
LDI R19,0b10000000
ret

SENHA2: ;170

LDI R18,0b00000000
LDI R19,0b00000000

ret

DISPLAY:

CPI r22, 0b00000010

BREQ qedoze

CPI r22, 0b00000001

BREQ dcinqs

CPI r22, 0b00000011

BREQ smoito

JMP CENTENA

qedoze:

LDI r22, 0b00000010

LDI r27, 0b00000001

LDI r28, 0b00000101

jmp CENTENA

dcinqs:

LDI r22, 0b00000110

LDI r27, 0b00000101

LDI r28, 0b00000010

jmp CENTENA

smoito:

LDI r22, 0b00001000

LDI r27, 0b00000110

LDI r28, 0b00000111

jmp CENTENA

zero:

LDI r22, 0b00000000

LDI r27, 0b00000000

LDI r28, 0b00000000

jmp CENTENA

CENTENA:

CPI r23, 0b01100100

BRLO CENTENAAUX

SUBI r23, 0b01100100

INC r31

JMP CENTENA

CENTENAAUX:

ADD r31, R28

DEZENA:

CPI r23, 0b00001010

BRLO DEZENAAUX

SUBI r23, 0b00001010

INC r30

JMP DEZENA

DEZENAAUX:

ADD r30, R27

CPI r30, 0b00001010

BRSB carry

JMP UNIDADEAUX

carry:

INC r31

SUBI r30, 0b00001010

UNIDADEAUX:

ADD r23, r22

CPI r23, 0b00001010

BRSB carryuni

JMP ATRIBUICAO

carryuni:

INC r30

SUBI r23, 0b00001010

CPI r30, 0b00001010

BRSB carrycent

JMP ATRIBUICAO

carrycent:

INC r31

SUBI r30, 0b00001010

ATRIBUICAO:

; unidade

SBRC r23, 0

SBI PORTD, 0

SBRB r23, 0

CBI PORTD, 0

SBRC r23, 1

SBI PORTD, 1

SBRB r23, 1

CBI PORTD, 1

SBRC r23, 2

SBI PORTD, 2

SBRB r23, 2

CBI PORTD, 2

SBRC r23, 3

SBI PORTD, 3

SBRS r23, 3

CBI PORTD, 3

; dezena

SBRC r30, 0

SBI PORTD, 4

SBRS r30, 0

CBI PORTD, 4

SBRC r30, 1

SBI PORTD, 5

SBRS r30, 1

CBI PORTD, 5

SBRC r30, 2

SBI PORTD, 6

SBRS r30, 2

CBI PORTD, 6

SBRC r30, 3

SBI PORTD, 7

SBRS r30, 3

CBI PORTD, 7

; centena

SBRC r31, 0

SBI PORTC, 1

SBRS r31, 0

CBI PORTC, 1

SBRC r31, 1

SBI PORTC, 2

```
SBRS r31, 1
CBI PORTC, 2
SBRC r31, 2
SBI PORTC, 3
SBRS r31, 2
CBI PORTC, 3
SBRC r31, 3
SBI PORTC, 4
SBRS r31, 3
CBI PORTC, 4
```

```
ret
```

```
;=====
```