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

RELATÓRIO DE IMPLEMENTAÇÃO DO PROBLEMA 3

Iago Pereira Cassel

Lian Guedes Silva

Marco Antonio Saraiva do Nascimento

Rodrigo Rodrigues Bazerra

Sâmela Bruna Ferreira

Natal-RN
OUTUBRO/2020

Iago Pereira Cassel

Lian Guedes Silva

Marco Antonio Saraiva do Nascimento

Rodrigo Rodrigues Bazerra

Sâmela Bruna Ferreira

RELATÓRIO DE IMPLEMENTAÇÃO DO PROBLEMA 3

Relatório apresentado à disciplina de Sistemas Digitais, correspondente à avaliação da 6º semana do semestre 2020.6 do curso de Engenharia Mecatrônica da Universidade Federal do Rio Grande do Norte, sob orientação do **Prof. Samaherni Morais Dias.**

Professor: Samaherni Morais Dias.

Natal-RN
OUTUBRO/2020

Lista de Figuras

Sumário

1 INTRODUÇÃO

Este relatório é referente à implementação de um projeto criado pelo grupo 2 da disciplina de sistemas digitais no semestre de 2020.6 do curso de engenharia mecatrônica na UFRN para controlar a abertura de um cofre . O projeto deverá se implementado no microcontrolador uC AVR(ATMega 328P). O sistema digital deverá apresentar a seguinte interface.

Figura 1: Aparência da interface homem-máquina do sistema digital a ser projetado

Fonte: fonte do professor

Na imagem abaixo é mostrado o detalhamento individual dos elementos da interface do cofre, que mais a seguir no relatório, será definido que periféricos serão utilizados para representar cada elemento:

Figura 2: Elementos da interface do cofre

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: fonte do professor

2 COMPONENTES

2.1 Microcontrolador AVR

Os microcontroladores AVR da fabricante ATMEL são microcontroladores de 8 bits, desenvolvidos sob a tecnologia RISC - Reduced Instruction Set Computer (Computador com Set de Instruções Reduzido) e arquitetura HARVARD que separa a memória de dados da memória de programa. Desta forma um microcontrolador AVR tem um barramento para dados e outro para programa. Esta separação de barramentos permite uma maior velocidade no tratamento dos dados e do programa.

2.2 Potenciômetro

Potenciômetro é um componente eletrônico que cria uma limitação para o fluxo de corrente elétrica que passa por ele, e essa limitação pode ser ajustada manualmente, podendo ser aumentada ou diminuída. Os potenciômetros e os resistores têm essa finalidade de limitar o fluxo de corrente elétrica em um circuito, a diferença é que o potenciômetro pode ter sua resistência ajustada e o resistor comum não pode, pois ele possui um valor de resistência fixo.

2.3 Pushbutton

Um Pushbutton (botão de pressão) é uma chave que contém um botão que ao ser pressionado abre ou fecha os contatos do dispositivo, abrindo ou fechando o circuito onde ele está conectado. É comum que um pushbutton possua ação de contato momentânea, o que significa que a conexão é aberta ou fechada apenas momentaneamente, enquanto o botão estiver sendo pressionado.

2.4 LED RGB

Os LEDs RGB são a junção de três LEDs em um só, ele é formado por um vermelho (R de red), um verde (G de green) e um azul (B de blue). A definição das cores é baseada no princípio de mistura aditiva das cores. Por exemplo: Ao adicionar a cor vermelha ao verde, obteremos a cor amarela; adicionar a cor vermelha ao azul, obteremos a cor magenta (violeta-púrpura); ao adicionar o verde ao azul, obteremos a cor cyan (ciano); e ao adicionar as 3 cores primárias com a mesma intensidade, será obtida a cor branca.

2.5 Display de 7 segmentos

Um display de sete segmentos, é um tipo de display (mostrador) barato usado como alternativa a displays de matriz de pontos mais complexos e dispendiosos. Displays de sete segmentos

são comumente usados em eletrônica como forma de exibir uma informação alfanumérica (binário, octadecimal, decimal ou hexadecimal) que possa ser prontamente compreendida pelo usuário sobre as operações internas de um dispositivo. Seu uso é corriqueiro por se tratar de uma opção barata, funcional e de fácil configuração.

2.6 Conversor analógico-digital (ADC)

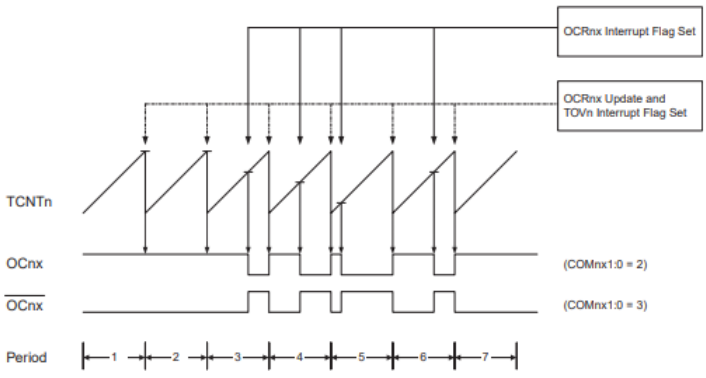
O microcontrolador ATMega 328P possui um conversor interno de sinal analógico para sinal digital, este converte uma tensão de entrada analógica em um valor digital de 10 bits por meio de aproximações sucessivas. O valor mínimo representa GND e o valor máximo representa a tensão de referência, que por padrão é a tensão do pino AREF menos 1 LSB,o ADC é conectado a um multiplexador analógico de 8 canais que permite oito entradas de tensão únicas.

O ADC gera um resultado de 10 bits que é apresentado nos Registradores de Dados ADCH e ADCL, por padrão os oito bits menos significativos são armazenados no ADCL e os 2 bits mais significativos são guardados nos bits 0 e 1 do registrador ADCH.

2.7 Temporizador/contador de 8 bits com PWM

Temporizador/contador é um módulo temporizador de 8 bits de uso geral, com duas saídas independentes e com suporte PWM. Para o uso do led RGB é utilizado o suporte PWM (pulse width modulation) ou modulação da largura do pulso que consiste em emular um sinal analógico através das larguras dos pulsos digitais de um sinal. O modo fast PWM disponível no ATMega 328P utiliza um modo de operação single-slope, onde o contador começa a contar de baixo até o topo, e quando chega no topo reinicia novamente de baixo, quando o contador alcança o valor da interrupção o nível do sinal digital é alterado, o funcionamento do modo fast PWM está ilustrado na Figura 3.

Figura 3: FAST PWM



Fonte: Datasheet AVR 8-bit Microcontroller

O sinal digital resultante pode então ser utilizado para variar a tensão que entrará no led

RGB, alterando assim as cores desse led.

3 DESENVOLVIMENTO DO PROJETO

3.1 Circuito

O circuito é apresentado nos anexos.

Foi definido que o ATMega 328P trabalhará com um clock de 1MHz, esse clock se faz necessário para que a conversão AD seja instantânea, para o PWM funcionar corretamente e para o delay de 0.5s.

O potenciômetro possui resistência de 10k Ω , um de seus pinos é conectado em uma fonte de tensão de 5V e outro é conectado ao terra, o pino central é conectado a entrada analógica PC4 do ATMega 328P.

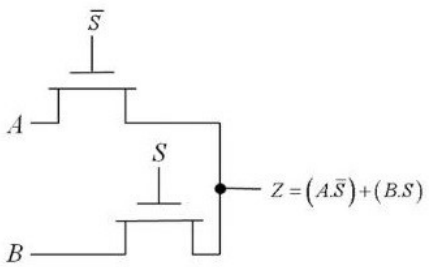
As portas PORTB[7:0] e PORTC[3:0] se comportam como output, e possuem o resultado da conversão analógico-digital em BCD, esses valores em BCD entram no CI 4511 que converte este número BCD para ser apresentado nos displays de 7 segmentos em decimal, as portas PORTB [3:0] possui a casa das dezenas, PORTB[7:4] a casa das centenas e PORTC[3:0] as unidades.

Os pinos PD3, PD5 e PD6 são utilizados pelos timers, essas portas transmitem o sinal analógico emulador pelo PWM, e são conectados ao led RGB.

Os pinos PORTD0 e PORTD1 são conectados aos PushButtons que foram especificados no projeto.

O pino PD1 é utilizado como uma flag para controlar um dos leds do 7 segmentos. Um circuito composto por 2 mosfets para cada display é criado de maneira a se comportar como um multiplexador, quando a flag tiver em 0 será passado o que estiver na saida QG do CI BCD obedecendo a conversão normal, mas caso a flag seja 1 será passado um sinal alto para o LED G do display e dessa forma será possível mostrar '- - -' no 7 segmentos. Abaixo segue a imagem da lógica seguida.

Figura 4: Mosfets



Fonte: Indian Intitute of Technology Guwahati

Sendo assim todos os pinos podem ser melhor visualizados na tabela a seguir:

Portas	**Finalidade**
PORTB0	Dezena D0
PORTB1	Dezena D1
PORTB2	Dezena D2
PORTB3	Dezena D3
PORTB4	Centena C0
PORTB5	Centena C1
PORTB6	Centena C2
PORTB7	Centena C3
PORTC0	Unidade U0
PORTC1	Unidade U1
PORTC2	Unidade U2
PORTC3	Unidade U3
PORTC4	Potenciômetro
PORTD0	Botão ON
PORTD1	Botão ADD
PORTD2	Flag
PORTD3	RGB Azul
PORTD5	RGB Verde
PORTD6	RGB Vermelho
TOTAL	19

Tabela 1: Pinagem

3.2 Conversão analógico-digital

O pino central do potenciômetro será conectado ao pino PC0 do ATMega 328P que é uma entrada analógica, utilizando o conversor analógico-digital de 10 bits presente no microcontrolador esta entrada será convertida para um sinal digital.

Para que essa conversão seja realizada é preciso preencher os registradores ADCSRA e ADMUX de maneira adequada às condições desejadas para a conversão, o registrador ADMUX está representado na Figura 5, os bits 7 e 6 selecionam a tensão referencial que será utilizada na conversão, essa seleção segue o padrão apresentado na Figura 6, no nosso caso escolheremos AREF como a tensão de referência logo os bits devem ser REFS[1:0]="00", o bit 5, caso seja 1, ajusta o resultado da conversão para a esquerda nos registradores ADCH e ADCL, porém manteremos a apresentação padrão logo o bit ADLAR='0' , o bit 4 é reservado, e os bits 3 até 0 selecionam qual entrada analógica será conectada ao ADC, no nosso caso utilizaremos a entrada ADC0 logo os bits são MUX[3:0]="0100", sendo assim as seguintes instruções são utilizadas para armazenar o valor desejado no ADMUX.

```
LDI R16, 0X04
STS ADMUX, R16
```

Figura 5: Registrador ADMUX

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Fonte: Datasheet AVR 8-bit Microcontroller

Figura 6: Seleção para a tensão de referência

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

Fonte: Datasheet AVR 8-bit Microcontroller

Tabela 2: Definição do ADMUX. Fonte: Própria (2020).

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
ADMUX	0	0	0	0	0	1	0	0

Para saber a tensão de referência é preciso seguir a seguinte formula:

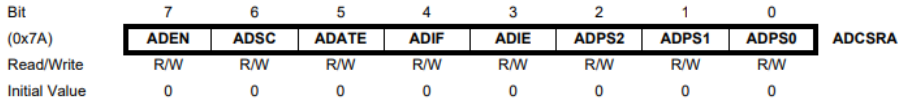
$$ADC = \frac{Vin \times 1024}{Vref}$$

O potenciômetro tem um dos seu pinos laterais conectados a uma fonte de tensão de 5V e o outro pino lateral conectado com o terra, sendo assim a tensão máxima no pino central do potenciômetro é aproximadamente 5V, logo Vin(max)=5V, já que o cofre suporta um número de 0 até 999, quando o Vin for máximo o ADC deve ser igual a 999. Utilizando a fórmula citada obtemos que Vref=5.1246, e essa tensão deve ser conectada ao pino Aref do ATMega 328P.

O registrador ADCSRA é representado na Figura 7, o bit 7, caso 1, ativa o ADC logo é preciso fazer ADEN =’1’, o bit 6 inicia a conversão, logo deve ser ativado apenas em um momento posterior no código, sendo assim ADSC =’0’, o bit 5 é um gatilho automático que não será utilizado, assim ADATE=’0’,o bit 4 é uma flag que indica quando a conversão for finalizada, o bit 3 interrompe a conversão logo ADIE=’0’, e os bits 2 até 0 determinam o fator de divisão entre o clock da máquina e o clock de entrada do ADC, para escolher um fator de divisão de 2 os bits foram definidos como ADPS[2:0]="001", sendo assim as instruções que inicializam o registrador são:

```
LDI R16, 0X81
STS ADCSRA, R16
```

Figura 7: Registrador ADCSRA



Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Fonte: Datasheet AVR 8-bit Microcontroller

Tabela 3: Definição do ADRSC. Fonte: Própria (2020).

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
ADCSRA	1	0	0	0	0	0	0	1

Para iniciar a conversão é necessário fazer ADSC='1', e esse código deve ficar em looping até que a conversão seja feita por completo, isso é indicado quando o bit de interrupção ADIF for '1', essa implementação é feita com as instruções a baixo:

```
StartADC:    LDS R16, ADCSRA
     ORI R16, 1<<ADSC  ;indica que irei carregar o valor '1' no pino adsc do
     STS ADCSRA, R16   ;registrador ADCSRA dando start na conversão


KeepPolling: lds R16, ADCSRA
     sbrs R16,ADIF ;indica que devo dar skip na proxima instrução caso o bit
     rjmp KeepPolling;de interrupção esteja ativo
```

3.3 Conversão binário-BCD

Como já foi citado anteriormente a conversão analógico-digital resulta em um número binário de 10 bits, porém esse número deve ser apresentado nos displays de 7 segmentos, então deve-se converter este número binário de 10 bits para a representação BCD, onde o número é dividido em unidade, dezena e centena, não é necessário a casa dos milhares pois os valores do cofre vão apenas de 0 até 999.

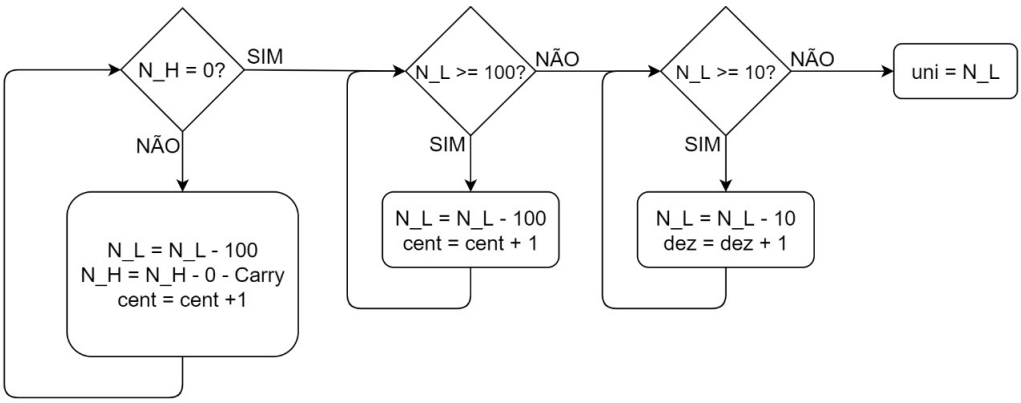
O número em BCD é então enviado para os registradores I/O PORTD [7:0] e PORTB [3:0] que devem estar configurados como output, isso é feito pelas linhas de código

```
     LDI R16, 0X0F;indica que os pinos 3,2,1,0 do portb devem se comportar
     OUT DDRB, R16 ; como output e os pinos 4,5,6,7 como input
     LDI R16, 0XFF
     OUT DDRD, R16 ;todos os pinos do portd devem se comportar como output
```

Nos pinos PORTD [7:4] é armazenada a casa das centenas do valor em BCD, nos pinos PORTD [3:0] as dezenas e nos pinos PORTB [3:0] é armazenada a unidade.

Para realizar essa conversão um valor é sequencialmente subtraído por 100 até que este valor esteja na casa das dezenas, ou seja, o número é menor que 100, então este valor é sequencialmente subtraído por 10 até que ele seja menor que 10, obtendo assim um valor em BCD, esse procedimento é ilustrado pelo fluxograma abaixo:

Figura 8: fluxograma da conversão BIN-BCD



Fonte: Autoria própria

Como o valor de 10 bits é armazenado em dois registradores é necessário realizar uma manipulação dos dados para realizar as subtrações, se o resultado da conversão for maior ou igual a 256 este valor será armazenado em dois registradores, sendo necessário duas operações de subtração, a primeira são os bits menos significativos do valor a ser convertido menos 100, e a segunda operação é uma subtração com carry entre os bits mais significativos e zero, essas operações são realizadas dessa maneira no código.

```
SUBI rl0, 100
SBCI rl1, 0
```

3.4 Atraso de 0.5s

Observando a máquina de estado desenvolvida no projeto, Figura 12, quando o usuário inserir um valor no cofre a máquina deve passar para o estado await_confirm, esse estado é responsável por gerar um atraso de 0.5s no cofre, mantendo o led RGB laranja durante esse atraso, isto é feito em uma sub-rotina no código do microcontrolador.

```
.def oLoopR  = R23 ; registrador externo de loop
.def iLoopRl = R24 ; registrador interno loop low
.def iLoopRh = R25 ; registrador interno loop high
.equ iVal  = 39998 ; valor interno do loop
```

```
    ldi  oLoopR,3;PARA 1MHZ
delay10ms:
    ldi iLoopRl,LOW(iVal) ; inicia o contador interno do loop
    ldi iLoopRh,HIGH(iVal) ; registradores loop high e low

iLoop: sbiw iLoopRl,1 ; decremento do registrador interno loop low
    brne iLoop ; pula para iLoop se registrador iLoopRl for != 0

    dec oLoopR ; decremento do registrador de saida loop
    brne delay10ms ; pula para delay10ms se registrador oLoopR for != 0

    nop ; apenas para perder ciclo de clock

    ret ; retorno da subroutine
```

São criados dois loopings, o delay10ms e um outro looping interno a esse, o iLoop, dentro do delay10ms o registrador iLoopRl é carregado com os bits menos significativos da constante iVal, esse registrador iLoopRl será decrementado dentro do iLoop até o seu valor ser igual à zero, então quando o iLoop é quebrado o registrador oLoopR é decrementado e o código retorna para o delay10ms, onde o registrador iLoopRl será novamente carregado com com os bits menos significativos da constante iVal e o iLoop se repetirá, quando o valor do registrador oLoopR for igual à zero o looping delay10ms é quebrado e o código retornará da sub-rotina, essas instruções utilizam pulsos de clock gerando assim o atraso no cofre.

3.5 PWM e led RGB

Para controlar a voltagem enviada para o led RGB, será utilizado o modo de operação 'Fast PWM Mode', que servirá para regular a frequência de ciclo PWM de maneira rápida o suficiente para o olho humano, para fazer isso, será utilizado os timers do microcontrolador. O Atmega328P possui 3 timers, um de 16 bits(timer1) e dois de 8 bits(timer0 e timer2), que serão utilizados para realizar a geração de sinal PWM. O cálculo para a frequência do ciclo PWM pode ser visto abaixo, onde o TOPvalue representa a saída do timer.

$$PWMfequency = \frac{clockspeed}{Prescallervalue * (1 + TOPValue)}$$

Cada timer possui duas portas de saída, no caso do timer0, as portas usadas são as D5(OCR0B) e D6(OCR0A), e no timer2, será utilizado a porta D3(OCR2B). Ambos os timers são controlados através de dois registradores, TCCR0A e TCCR0B para o timer0 e TCCR2A e TCCR2B para o timer2, funcionando da mesma forma para os dois temporizadores. O conteúdo

de cada registrador é mostrado logo abaixo, junto com suas funcionalidades:

Figura 9: Registrador A de controle do timer0

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

Timer/Counter Control Register 0 A

COM0A1	COM0A0	DESCRIPTION
0	0	OC0A disabled
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected WGM02 = 1: Toggle OC0A on Compare Match
1	0	None-inverted mode (HIGH at bottom, LOW on Match)
1	1	Inverted mode (LOW at bottom, HIGH on Match)

Applies only to PWM modes

COM0B1	COM0B0	DESCRIPTION
0	0	OC0B disabled
0	1	Reserved
1	0	None-inverted mode (HIGH at bottom, LOW on Match)
1	1	Inverted mode (LOW at bottom, HIGH on Match)

Applies only to PWM modes

Fonte: QEEwiki

Figura 10: Registrador B de controle do timer0

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TCCR0B	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00

Timer/Counter Control Register 0 A

CS02	CS01	CS00	DESCRIPTION
0	0	0	Timer/Counter2 Disabled
0	0	1	No Prescaling
0	1	0	Clock / 8
0	1	1	Clock / 64
1	0	0	Clock / 256
1	0	1	Clock / 1024

CS bits

MODE	WGM02	WGM01	WGM00	TOP	DESCRIPTION
0	0	0	0		Normal
1	0	0	1	0xFF	PWM Phase Corrected
2	0	1	0	OCRA	CTC
3	0	1	1	0xFF	Fast PWM
4	1	0	0	-	Reserved
5	1	0	1	OCR0A	PWM Phase Corrected
6	1	1	0	-	Reserved
7	1	1	1	OCR0A	Fast PWM

Waveform Generator Mode bits

Fonte: QEEwiki

As portas de saída do timer0 foram utilizados para as cores vermelha, verde e laranja, e a porta de saída do timer2 para a cor azul. A definição de cada bit de controle do temporizador para cada cor do led RGB será apresentado a seguir:

Tabela 4: Definição para a cor vermelha. Fonte: Própria (2020).

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TCCR0A	1	0	0	0	0	0	1	1
TCCR0B	0	0	0	0	0	0	0	1

Tabela 5: Definição para a cor verde. Fonte: Própria (2020).

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TCCR0A	0	0	1	0	0	0	1	1
TCCR0B	0	0	0	0	0	0	0	1

Tabela 6: Definição para a cor laranja. Fonte: Própria (2020).

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TCCR0A	1	0	1	0	0	0	1	1
TCCR0B	0	0	0	0	0	0	0	1

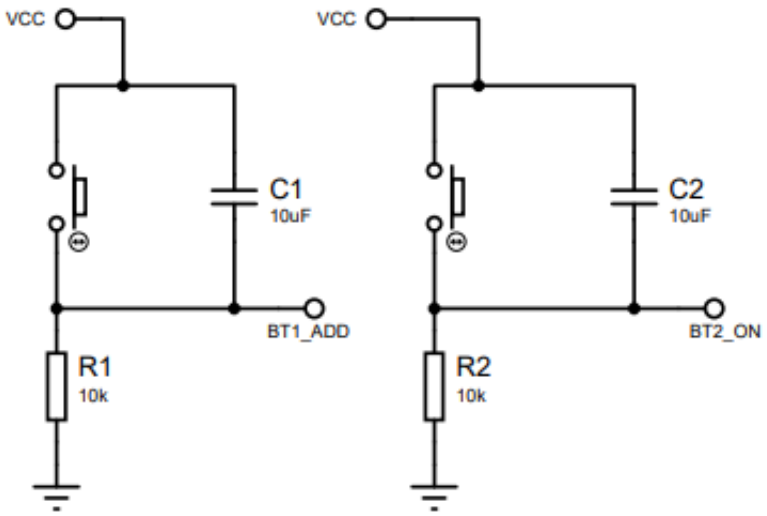
Tabela 7: Definição para a cor azul. Fonte: Própria (2020).

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TCCR2A	0	0	1	0	0	0	1	1
TCCR2B	0	0	0	0	0	0	0	1

3.6 PushButtons

O circuito demonstrado na Figura 11 foi criado para evitar o problema do debounce, que pode causar um falso positivo na saída do botão

Figura 11: debounce



Fonte: Autoria própria

3.7 Máquina de estados

A maquina de estados é implementada de acordo com o fluxograma demonstrado na Figura 12, onde a tag em vermelho foi alterada para que a máquina de estado funcione corretamente, para implementar essa máquina de estados foram utilizadas instruções de "branch"que fazem com que um computador comece a executar uma sequência de instruções diferente se uma determinada condição for verdadeira e, portanto, desvie sua ordem padrão de execução de instruções, analisando a implementação do estado wait_on pode-se compreender melhor como foram implementados os estados.

```
STATE_WAIT_ON:
in REG_BUTTON, PIND ;Pega o conjunto das entradas PIND, "Botão ON" é 0000 0001
andi REG_BUTTON, 1 ;Usa a máscara 0000 0001
cpi REG_BUTTON, 1 ;Compara com a máscara para ver se o "Botão ON" foi pressionado
breq STATE_WAIT_VALUE ;Pula para STATE_WAIT_VALUE caso tenha sido pressionado.
rjmp STATE_WAIT_ON ;Volta para STATE_WAIT_ON caso contrário.
```

A instrução in armazena o conteúdo do registrador PINB no registrador REG_BUTTON, a instrução andi aplica uma máscara no registrador REG_BUTTON, realizando um "clear"em todos os bits do registrador menos o bit 8, que é o pino conectado ao botão ON, após isso a instrução cpi compara o conteúdo da registrador REG_BUTTON com o valor "00000001", caso a comparação seja verdadeira o botão ON foi ativado e a instrução breq conduz a máquina de estados para o estado WAIT_VALUE, caso a comparação não seja verdadeira a instrução rjmp mantém a máquina de estados no estado atual.

Podemos afirmar que o estado primordial de nossa máquina é estado STATE_WAIT_VALUE, neste estado a máquina verifica o pressionamento dos botões, realiza a conversão analógica e

acende o LED azul da máquina através de sub-rotinas chamadas por um rcall, que realiza um pulo de lable para as funções definidas, utilizando boa parte das funções definidas pelos blocos de entrada do sistema. Ademais, ele verifica o pressionamento de ambos botões para transitar para diferentes estados.

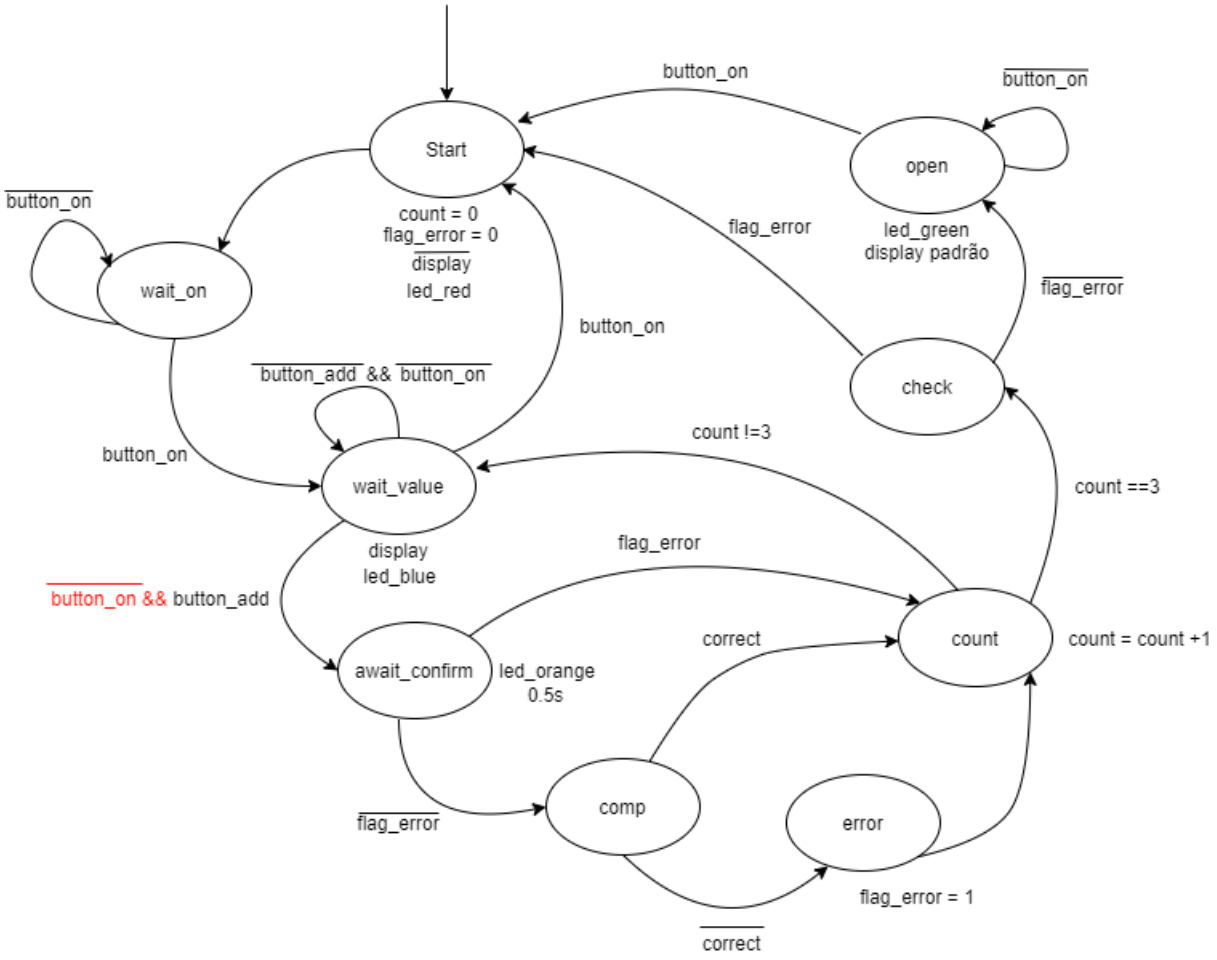
```
STATE_WAIT_VALUE:
rcall BUTTON_SYNC ;Espera que ambos botões estejam despressionados

STATE_WAIT_VALUE_POS_BUTTON_SYNC:
rcall azul ;Acende o LED Azul
rcall StartADC ;Realiza a conversão analógica e expressa ela na saída
in REG_BUTTON, PIND ;Pega o conjunto das entradas PIND, "Botão ON" é 0000 0001
andi REG_BUTTON, 1 ;Usa a máscara 0000 0001
cpi REG_BUTTON, 1 ;Compara com a máscara para ver se o "Botão ON" foi pressionado
breq STATE_START ;Pula para STATE_STATE_ON caso tenha sido pressionado
in REG_BUTTON, PIND ;Pega o conjunto das entradas PIND, "Botão ADD" é 0000 0010
andi REG_BUTTON, 2 ;Usa a máscara 0000 0010
cpi REG_BUTTON, 2 ;Compara com a máscara para ver se o "Botão ADD" foi pressionado
breq STATE_AWAIT_CONFIRM ;Pula para STATE_AWAIT_CONFIRM caso tenha sido pressionado
rjmp STATE_WAIT_VALUE_POS_BUTTON_SYNC ;Volta para STATE_WAIT_VALUE_POS_BUTTON_SYNC caso o contrário
```


```
STATE_WAIT_VALUE:
rcall BUTTON_SYNC ;Espera que ambos botões estejam despressionados

STATE_WAIT_VALUE_POS_BUTTON_SYNC:
rcall azul ;Acende o LED Azul
rcall StartADC ;Realiza a conversão analógica e expressa ela na saída
in REG_BUTTON, PIND ;Pega o conjunto das entradas PIND, "Botão ON" é 0000 0001
andi REG_BUTTON, 1 ;Usa a máscara 0000 0001
cpi REG_BUTTON, 1 ;Compara com a máscara para ver se o "Botão ON" foi pressionado
breq STATE_START ;Pula para STATE_STATE_ON caso tenha sido pressionado
in REG_BUTTON, PIND ;Pega o conjunto das entradas PIND, "Botão ADD" é 0000 0010
andi REG_BUTTON, 2 ;Usa a máscara 0000 0010
cpi REG_BUTTON, 2 ;Compara com a máscara para ver se o "Botão ADD" foi pressionado
breq STATE_AWAIT_CONFIRM ;Pula para STATE_AWAIT_CONFIRM caso tenha sido pressionado
rjmp STATE_WAIT_VALUE_POS_BUTTON_SYNC ;Volta para STATE_WAIT_VALUE_POS_BUTTON_SYNC caso o contrário
```

Figura 12: Máquina de estados



Fonte: Projeto do Grupo 2

4 CONCLUSÃO

A partir da máquina de estados, que sofreu uma pequena alteração, e das especificações definidas durante a semana de projeto, foram necessárias a realização das definições faltantes antes de se iniciar o processo de implementação. Em seguida, o projeto que foi produzido objetivando uma aplicação sobre um microcontrolador genérico, foi executado sobre o microcontrolador ATmega328p.

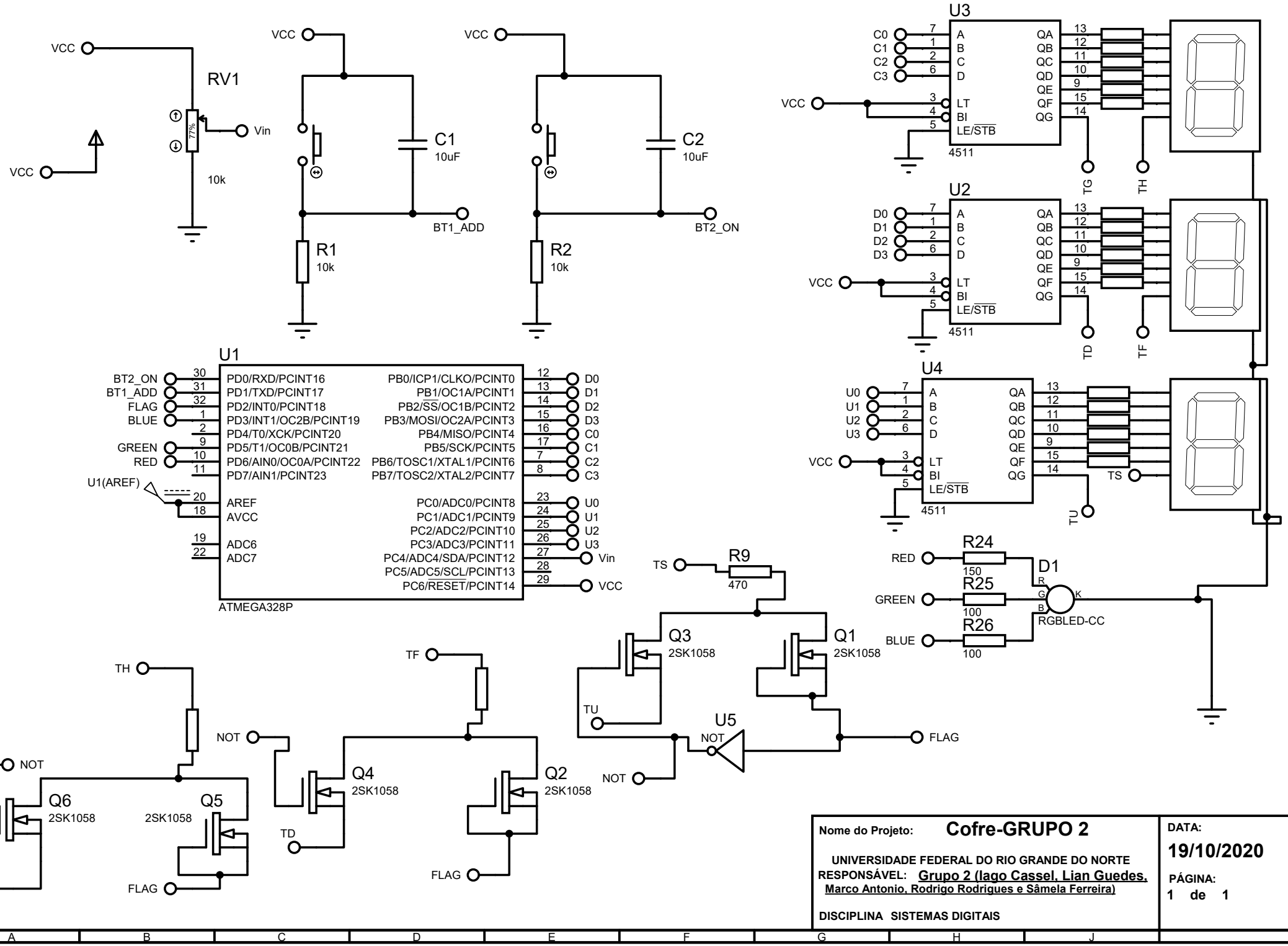
Dentre as principais dificuldades encontradas pelo grupo, foram: a impossibilidade de utilizar os dezesseis primeiros registradores, ocasionando em uma reorganização sintáxica no projeto, visto que não haviam informações acerca disso no datasheet; e a necessidade do aprendizado da linguagem assembly, pois a dificuldade não era resolver a lógica do projeto mas sim em passar para a linguagem escolhida.

Referências

1 ATMEL. 8-bit AVR Microcontroller with 32KBytes In-System Programmable Flash. Orchard Parkway San Jose: 2011, 346.

2 To control the intensity of a LED via PWM. avr-asm-tutorial, 2017. Disponível em:http://www.avr-asm-tutorial.net/avr$_{e}n/micro_{b}eginner/5_{L}ed_{p}wm/5_{L}ed_{P}wm.html$

3 Delay Subroutine. rjhcoding, 2018. Disponível em: http://www.rjhcoding.com/avr-asm-delay-subroutine.php


VCC
RV1
77%
Vin
10k
VCC
C1
10uF
R1
10k
BT1_ADD
VCC
C2
10uF
R2
10k
BT2_ON
U3
C0
C1
C2
C3
A
B
C
D
LT
BI
LE/STB
QA
QB
QC
QD
QE
QF
QG
4511
VCC
TG
TH
U2
D0
D1
D2
D3
4511
TD
TF
U4
U0
U1
U2
U3
4511
TS
TU
U1
BT2_ON
BT1_ADD
FLAG
BLUE
GREEN
RED
U1(AREF)
PD0/RXD/PCINT16
PD1/TXD/PCINT17
PD2/INT0/PCINT18
PD3/INT1/OC2B/PCINT19
PD4/T0/XCK/PCINT20
PD5/T1/OC0B/PCINT21
PD6/AIN0/OC0A/PCINT22
PD7/AIN1/PCINT23
AREF
AVCC
ADC6
ADC7
PB0/ICP1/CLKO/PCINT0
PB1/OC1A/PCINT1
PB2/SS/OC1B/PCINT2
PB3/MOSI/OC2A/PCINT3
PB4/MISO/PCINT4
PB5/SCK/PCINT5
PB6/TOSC1/XTAL1/PCINT6
PB7/TOSC2/XTAL2/PCINT7
PC0/ADC0/PCINT8
PC1/ADC1/PCINT9
PC2/ADC2/PCINT10
PC3/ADC3/PCINT11
PC4/ADC4/SDA/PCINT12
PC5/ADC5/SCL/PCINT13
PC6/RESET/PCINT14
D0
D1
D2
D3
C0
C1
C2
C3
U0
U1
U2
U3
Vin
VCC
ATMEGA328P
TS
R9
470
Q3
2SK1058
Q1
2SK1058
TU
U5
NOT
FLAG
NOT
R24
150
R25
100
R26
100
RED
GREEN
BLUE
D1
R
G
B
K
RGBLED-CC
TH
NOT
Q6
2SK1058
Q5
2SK1058
TG
FLAG
TF
NOT
Q4
2SK1058
TD
Q2
2SK1058
FLAG
NOT
Nome do Projeto: Cofre-GRUPO 2
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
RESPONSÁVEL: Grupo 2 (Iago Cassel, Lian Guedes, Marco Antonio, Rodrigo Rodrigues e Sâmela Ferreira)
DISCIPLINA SISTEMAS DIGITAIS
DATA:
19/10/2020
PÁGINA:
1 de 1