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

RELÓGIO PROGRAMADOR COM RTC DS3234 E MICROCONTROLADOR ATMEGA328P

ELE1717 - Grupo 02 - Problema 04 - Implementação

ALYSSON FERREIRA DA SILVA
ANNY BEATRIZ PINHEIRO FERNANDES
ISAAC DE LYRA JUNIOR
STHEFANIA FERNANDES SILVA
WESLEY BRITO DA SILVA

Natal, 4 de agosto de 2021

Resumo

O relatório a seguir descreve a implementação de um programador horário usando o ATmega328P para o controle do sistema e o (*Real Time Clock* - Relógio de Tempo Real) RTC DS3234 para controle das horas, minutos e dias. Além disso, é usada uma IHM (Interface Homem-Máquina) com *display*, leds e botões, para interagir com o usuário. Para realizar tal implementação foi utilizada a máquina de estados de alto nível projetada para a construção do programa em *assembly*. Além disso, foi preciso estudar a comunicação SPI (*Serial Peripheral Interface*) do ATmega328P, o *datasheet* do DS3234 e do MAX7912. No que tange o projeto recebido, muito foi aproveitado e apenas algumas alterações foram realizadas.

Palavras-chave: Sistemas Digitais, ATMEGA328P, DS3234, MAX7219, Relógio Programador.

Lista de Imagens

Sumário

1 IMPLEMENTAÇÃO

Nesta seção serão apresentadas as soluções projetadas e implementadas para a realização do relógio programador. Algumas alterações foram necessárias e isso pode ser visto no tópico posterior. Uma decisão que ficou inalterada foi o diagrama de operação do relógio, mostrado no Anexo B.

1.1 Correções do projeto

1.1.1 Chave seletora

Para a chave seletora, no projeto passado, ela tinha a parte seletora do NA e NF ligadas com o pino digital do microcontrolador e o EX estava ligado com o *source* de um mosfet, onde seu dreno estava ligado com a entrada IN e um resistor de *pull-down* juntamente com a saída ligada com *source*. No atual projeto, devido a algumas tentivas que deram errado na implementação da utilização desse mosfet, foi utilizado em sua substituição uma porta *AND*.

1.1.2 *Clock* do RTC

Uma decisão do projeto passado é que o *clock* necessário para a utilização do RTC deveria ser de 4MHz, porém ao ser analisada essa proposta, foi visto que era uma frequência muito alta para o correto funcionamento do RTC em ambientes físicos, pois este estaria operando no seu limite. Com isso em mente, foi feito um ajuste em que se reduziu o valor de 4 MHz para 500KHz, alterando o seu prescaling ($f_{clk}/16$).

1.1.3 Acionamento dos *displays*

Foi proposto que para o acionamento dos displays fosse utilizado o protocolo interface serial de 2 fios (I2C) por meio do CI TM1637, pois utilizaria menos portas. Porém, foi julgado que como o protocolo SPI já seria implementado e, visto que, para utilizar mais um servo seria necessário adicionar apenas mais um porta do microcontrolador, então foi mudado o protocolo de I2C para SPI e a troca do CI TM1637 para o MAX7219.

1.1.4 Manipulação dos LEDS

Quanto a manipulação dos LEDs que representam o dia da semana e os modos de operação do programador horário, em projeto foi definida a utilização do método

charlieplexing para controle da ativação destes. Mas esta ativação se dá para mais de um LED por vez e, também, há a oscilação de um destes na frequência de 2Hz.

Com isso, durante a tentativa de implementação, se tornou clara a inviabilidade de aplicação deste método para todos os LEDS. Logo, foi decidido que o *charlieplexing* seria usado para a ativação dos LEDS do dia da semana (D1 à D7), os quais permanecem sólidos em sua ativação. No entanto, para os LEDS dos modos de operação, foram designados quatro pinos do microcontrolador que iriam controlar os quatro LEDS restantes (T, O, F, W) através das linhas de código em *Assembly*.

2 RESULTADOS

Neste capítulo são apresentandos os detalhes dos procedimentos realizados para implementar o relógio programador.

2.1 Implementação em *Assembly*

2.1.1 Implementação dos estados do relógio programável

Seguindo o fluxograma mostrado no Anexo B, foi feita a implementação de cada um dos estados definidos no projeto. Para isso, algumas definições precisaram ser feitas.

Começando com o SPI, este foi definido conforme mostra a Figura 1, a *label* "SPI_MasterInit" seta quais portas são utilizadas como saída do SPI e define os valores do registrador SPCR. Assim o SPI é definido como mestre, a comunicação é ativada, o prescale é setado como 16 e o modo de operação do SPI é definido. Esse modo de operação é setado de duas formas diferentes, para o servo MAX7912 o modo setado é o 0 (CPOL=0 e CPHA=0), já para o servo DS3234 o modo setado é 1 (CPOL=0 e CPHA=1). Já a *label* "SPI_MasterTransmit" é responsável por transmitir o dado e "Wait_Transmit" é a *label* responsável por garantir que todos os bits serão transmitidos.

Figura 1 – Configuração SPI para o RTC DS3234.

```
SPI_MasterInit:
    ldi r17,(1<<DDB3)|(1<<DDB5)|(1<<DDB2)|(1<<DDB1)
    out DDRB,r17
    ldi r17,(1<<SPE)|(1<<MSTR)|(1<<CPHA)|(1<<SPR0)
    out SPCR, r17
    ret

SPI_MasterTransmit:
    out SPDR, r16
    Wait_Transmit:
        in r16, SPSR
        sbrs r16, SPIF
        rjmp Wait_Transmit
        ret
```

Fonte: Autores.

Foram reservados 3 registradores (r17, r18 e r19) para armazenar os valores das horas (RH), minutos (RM) e dias (RW), respectivamente. Estes serão utilizados para exibir no *display* a hora e o minuto atual, para incrementação ou decrementação dos horários e também para enviar ao RTC o valor alterado.

Em posse disso foi criada a *label* Timer_H, nela solicita-se que o led T seja aceso através da chamada "rcall ACENDE_LEDT" e são feitas verificações dos pinos 4, 5, 6. Isso é feito com o intuito de verificar se um botão foi ou não pressionado, nesse sentido, o pino 4 tem como entrada o botão de incremento, o pino 5 o botão de decremento e o pino 6 o botão "R".

Figura 2 – Estado Timer_H.

```
timer_h:
    sei
    rcall ACENDE_LEDT
    cli
    sbic PIND, 4
    rcall increm_hours
    sbic PIND, 5
    rcall decrem_hours
    sbic PIND, 6
    rjmp RB_filter0
    rjmp timer_h
```

Fonte: Autores.

Quando a *label* responsável pelo incremento (increment_hours) ou decremento (decrement_hours) da hora é chamada ocorre a solicitação do timer de 0.5 segundos servindo como um *debouncing* do botão. Enquanto ocorrer o incremento, há a comparação do valor atual do registrador "RH" com 24, visando não extrapolar o valor limite das horas (nem dos minutos), se o valor foi maior ou igual, o registrador será travado no valor 23. Caso esteja ocorrendo um decremento, o valor também é comparado com 23 e se for maior ou igual, então o "RH" é travado em 0 no registrador. A Figura 3 mostra a lógica do incremento e decremento das horas.

Essa lógica também serve para os minutos, porém com os valores de comparação em 60 e setando em 0 e 59. Para os dias da semana se aplica a mesma ideia com os intervalos de 1 a 8. Nesse sentido, o Timer_M e Week possuem um código semelhante ao do Timer_H, mudando sempre os leds e o tipo de comparação que precisa ser realizada.

Note que na Figura 3, quando o pino 6 (pino conectado ao botão "R") é pressionado a próxima label ainda não é o Timer_M, isso foi feito para evitar que o relógio realize operações que o usuário não deseja. Nesse sentido, conforme mostra a Figura 4, a máquina somente irá para o estado Timer_M quando o botão for "despressionado".

Figura 3 – Configuração do incremento e decremento da hora.

```
153  decrem_hours:
154      dec RH
155      call timer05
156      cpi RH, 24
157      brsh lock_hours_dec
158      ret
159
160  lock_hours_dec:
161      ldi RH, 0
162      ret
163
164  increm_hours:
165      inc RH
166      call timer05
167      cpi RH, 24
168      brsh lock_hours_inc
169      ret
170
171  lock_hours_inc:
172      ldi RH, 23
173      ret
```

Fonte: Autores.

Figura 4 – Controle do botão "R".

```
RB_filter0:
    sbic PIND, 6
    rjmp RB_filter0
    rjmp timer_m
```

Fonte: Autores.

Para enviar os valores de hora, minutos e dias para o RTC é chamada a *label* "SPI_MasterTransmit" como mostra a Figura 5. Primeiro, define-se que o pino B2 (utilizado como chave seletora do servo RTC) como valor lógico baixo. Feito isso seleciona-se o endereço de escrita correspondente ao dado enviado, no exemplo mostrado é o endereço das horas do RTC, e após isso é enviado o dado que irá preencher tal endereço. Por fim, deve-se setar o valor 1 ao pino B2. Tal lógica é feita para todos os três dados utilizados pelo relógio programador.

De maneira análoga ao que foi explicado, é feito no estado ajuste de agendamento. A principal diferença é que os endereços setados do RTC, para horas, minutos e dias e são os do alarme 1 e alarme 2.

Figura 5 – Envio da hora para o RTC.

```
//seta as horas
cbi PINB, 2
ldi r16, 0x82 ; endereço de escrita das horas
call SPI_MasterTransmit
mov r16, RH
call SPI_MasterTransmit
sbi PINB, 2

cbi PINB,2
sbi PINB,2
```

Fonte: Autores.

2.2 Acendimentos dos LEDS

Nesta implementação foi elaborado duas formas para o acionamento dos leds: Uma para os LEDs da semana e outra para o acionamento dos LEDs dos processos. Para os LEDs indicadores dos processos, foi pensado na utilização das portas DDRC2, DDRC3, DDRC4 e DDRC5, para, respectivamente, o LED T, LED W, LED O e o LED F. Como é mostrado na Figura 6, para o acendimento, se o valor da porta já for 1, então sai da label, caso contrário, aciona 1 no pino. Para desligar, caso o LED esteja ligado, então insere o valor 0 na porta, caso já seja 0, então o processo volta a rotina anterior.

Figura 6 – Acionamento e desligando dos LEDs dos processos.

```
49      ;ledT = C2, ledW = C3, ledO = C4, ledF = C5
50      ACENDE_LEDT:
51      sbis PORTC,2
52      sbi PINC, 2
53      ret
54
55      ACENDE_LEDW:
56      sbis PORTC,3
57      sbi PINC, 3
58      ret
59
60      ACENDE_LEDO:
61      sbis PORTC,4
62      sbi PINC, 4
63      ret
64
65      ACENDE_LEDF:
66      sbis PORTC, 5
67      sbi PINC, 5
68      ret
```

```
70      APAGA_LEDT:
71      sbis PORTC,2
72      ret
73      cbi PINC, 2
74      ret
75      APAGA_LEDW:
76      sbis PORTC,3
77      ret
78      cbi PINC, 3
79      ret
80
81      APAGA_LEDO:
82      sbis PORTC,4
83      ret
84      cbi PINC, 4
85      ret
86
87      APAGA_LEDF:
88      sbis PORTC,5
89      ret
90      cbi PORTC, 5
91      ret
```

Fonte: Autores.

Para o piscar dos LEDs, como é vista na Figura 7, há uma comparação entre dois registradores, r22 e r24. O registrador r22 é um registrador que guarda o valor 0x01 enquanto o registrador r24 alterna entre 0 e 1 indicando quando os LEDs que piscam ficaram acesos ou apagados. Para isso, sempre que ocorrer uma interrupção por comparação

do OCR1A é invertido o sinal do r24 no tratamento da interrupção na label Muda_Pisca. Assim feito, a label Pisca_LedX verifica se r24 é 0 ou 1 (se os leds vão apagar ou acender) e salta para acender o led ou apagar o led conforme o valor de r24.

Figura 7 – Pisca LEDs.

```
102     PISCA_LEDT:
103     CPSE r22, r24
104     jmp APAGA_LEDT
105     jmp ACENDE_LEDT
106
107     PISCA_LEDW:
108     CPSE r22, r24
109     jmp APAGA_LEDW
110     jmp ACENDE_LEDW
111
112     PISCA_LEDO:
113     CPSE r22, r24
114     jmp APAGA_LEDO
115     jmp ACENDE_LEDO
116
117     PISCA_LEDF:
118     CPSE r22, r24
119     jmp APAGA_LEDF
120     jmp ACENDE_LEDF
```

```
42      Muda_Pisca:
43      cli
44      CPSE r24, r22
45      jmp coloca
46      jmp tira
47
48      coloca:
49      INC r24
50      sei
51      reti
52      tira:
53      DEC r24
54      sei
55      reti
```

Fonte: Autores.

Para os LEDs da semana, como dito anteriormente, foi elaborado a lógica do *Charlieplexing*. Seguindo o conceito da estrutura proposta foram elaboradas equações de resultados de cada acionamento que o LED requeria. Nisto, para o LEDX é aplicado a fórmula de cada um. No caso, é enviado 1 para o terminal positivo do led e 0 para o terminal negativo do LED (definidos como entrada), e para não acionar os outros LEDs acidentalmente, é aplicado o conceito de *tris-tate* (alta impedância) que consiste em definir as portas restantes como entrada para o registrador r16. Na Figura 8, temos como um exemplo a implementação do D1.

Figura 8 – Acionamento de uma porta para a técnica Charlieplexing.

```
60  LED1:
61      ldi r20, 0x01
62      cp r19, r20
63      breq acende_led1
64      ldi r20, 0x03
65      cp r19, r20
66      breq acende_led1
67      ret
68
69      acende_led1:
70      sbis PORTD, 3
71      jmp seta1
72      jmp retorna1
73
74      seta1:
75      ldi r16, 0b00001001
76      out DDRD, r16
77
78      sbi PIND, 3
79      cbi PORTD, 0
80      cbi PORTD, 1
81      cbi PORTD, 2
82
83      retorna1:
84      ret
```

Fonte: Autores.

2.3 Implementação no Proteus

Nesta seção vão ser abordados os esquemáticos utilizados para simular o código desenvolvido, tanto quanto à comunicação com os CIs, como quanto as conexões dos LEDs, do *display*, da chave e dos botões. Estes esquemáticos foram desenvolvidos no software *Proteus*.

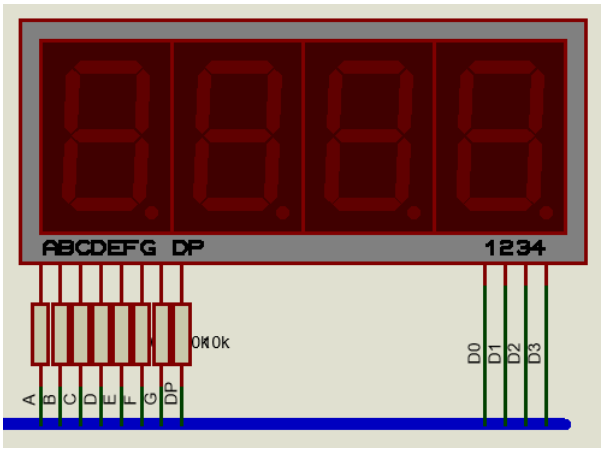
2.3.1 Implementação do *display*

O projeto requer um *display* com quatro dígitos, cada com 7 segmentos para realizar a apresentação dos valores de hora e minuto atual no programador horário. Dito isto, foi necessária a escolha de um CI que realizasse a comunicação do *display* com o microcontrolador.

Foi decidido dentro das correções de projeto que essa comunicação seria realizada, também, através do protocolo SPI. As conexões do *display* com o CI MAX7219 está disposta na Figura 9. Já a comunicação do CI com o ATMega328p através do protocolo, é vísivel na Figura 10.

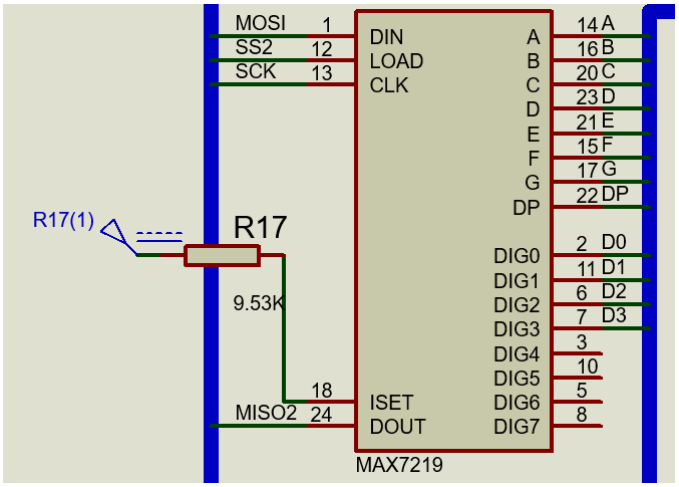
Repare que na Figura 10 é possível ver as 12 ligações do CI com o *display* à direita e, à esquerda, estão as 4 ligações necessárias com o protocolo para existir a transferência de dados de hora e minuto do display ao microcontrolador. A entrada ISET do MAX7219 é usada para ajustar o brilho dos dígitos.

Figura 9 – Esquemático - Display - CI MAX7219



Fonte: Autores.

Figura 10 – Esquemático - CI MAX7219



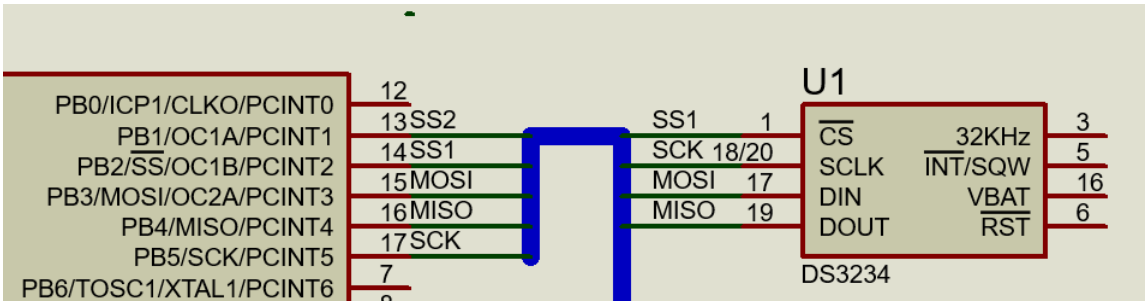
Fonte: Autores.

2.3.2 Implementação do RTC

O *Real-Time-Clock* (RTC) é um CI utilizado para fornece a data e hora atual do sistema e também para armazenar alarmes requeridos. O CI RTC escolhido pelo projeto do grupo anterior foi o DS3234. Este, por sua vez, utiliza do protocolo de comunicação SPI para realização da transferência de dados. A conexão deste CI com o microcontrolador se dá na Figura 11. À esquerda vemos as portas utilizadas do ATMega328p e à direita sua respectiva comunicação com o CI DS3234.

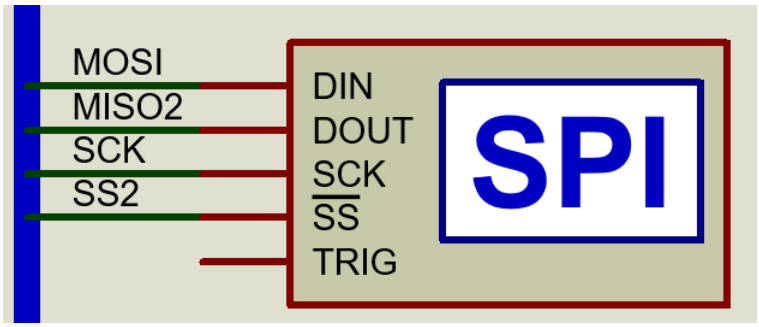
Para ajudar no entendimento da transferência dos dados através do protocolo, também foi utilizado um debugador de SPI próprio do software. Este pode ser vizualisado na Figura 12. Repare que a comunicação padrão SPI, um servo e um mestre, utiliza de 4 portas (SS, MOSI, MISO e SCK). No entanto, dado que utilizamos um segundo servo (o CI MAX7219) nesta comunicação, foi adicionada uma quinta porta com label de SS2 como descrita da Figura 11.

Figura 11 – Esquemático - Conexão RTC - ATMega328p



Fonte: Autores.

Figura 12 – Esquemático -Debugador SPI



Fonte: Autores.

2.3.3 Implementação dos LEDS

Como já comentado, a implementação da ativação dos 11 LEDS foi realizada a partir da aplicação de dois métodos diferentes: o *Charlieplexing* e o controle direto a partir do microcontrolador. Com isso, para garantir o funcionamento do código desenvolvido, foi implementado um esquemático no software *Protheus* para cada método.

2.3.3.1 LEDS da Semana - *Charlieplexing*

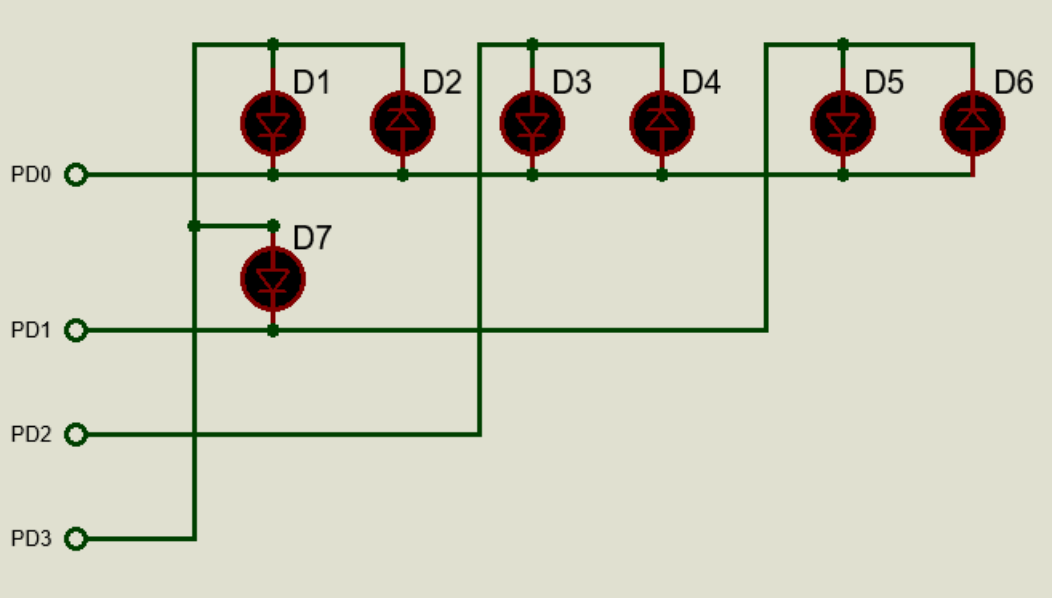
Para os sete LEDS que representam os dias da semana foi implementado o método *Charlieplexing*, que consiste basicamente numa combinação de entradas utilizando de alta impedância e fluxo dos diodos. A combinação de entradas para a quatidade exigida de LEDS se dá por meio da equação: n^2 - n, onde n representa o número de portas (linhas de controle) a serem utilizadas.

Dado que são 7 LEDS a serem ativados, o número mínimo de portas a serem utilizadas é igual à 4 ($4^2 - 4 = 12$), pois o número de 3 portas permite o controle apenas de 6 LEDS ($3^2 - 3 = 6$). Com isso, a Figura 13 mostra o esquemático referente à implementação do método *Charlieplexing* para os sete dias da semana. Nesta, o domingo é representado por D1, a segunda por D2, a terça por D3 e assim por diante até o D7 que representa o sábado.

Na Figura 14 estão dipostas as combinações das portas que controlam a ativação

de cada LED em questão. As portas D0, D1, D2 e D3, do ATMega328p, foram as escolhidas para este controle. Estas possuem 3 estados possíveis: Alto (apresentando 5V), baixo (apresentando 0V) e Alta impedância. A combinação destes estados permitem a aplicabilidade do método. Com isso, ao aplicar o código ao microcontrolador, vemos a ativação individual de cada LED conforme desejado.

Figura 13 – Esquemático - LEDS da Semana



Fonte: Autores.

Figura 14 – Fórmula de aplicação - LEDS da Semana

Legenda		
1	0	Tri-state
D+	D-	D*

Portas			
D0	D1	D2	D3

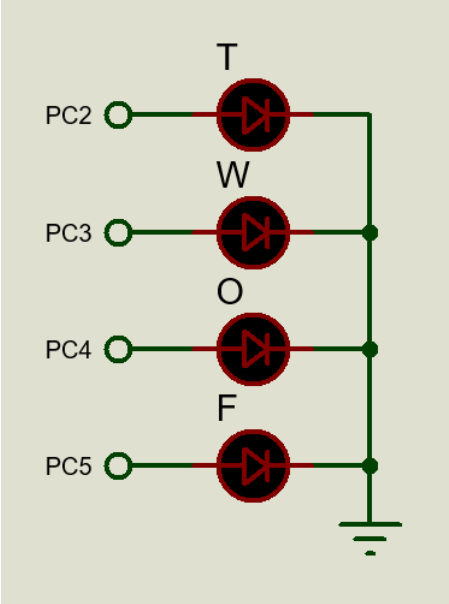
LED	"Formula"
1	D3+ D0- D1* D2*
2	D3- D0+ D1* D2*
3	D2+ D0- D3* D1*
4	D2- D0+ D1* D3*
5	D1+ D0- D2* D3*
6	D1- D0+ D2* D3*
7	D3+ D1- D2* D0*

Fonte: Autores.

2.3.3.2 LEDS dos Modos de Operação - Controle Direto

Para o controle dos LEDS dos modos de operação, o método aplicado foi o de controle direto pois este permite a oscilação de cada LED sem grandes problemas. Para isso, foram destinadas as portas C2, C3, C4 e C5 para controlar os LEDS T, W, O e F. A oscilação destes LEDS na frequência de 2Hz é realizada através da utilização de uma interrupção global do timer1 do ATMega328p. Dado que o controle é feito majoritariamente através do código, o esquemático mostra apenas as conexões com o microcontrolador como na Figura 15.

Figura 15 – Esquemático - LEDS dos Modos de Operação



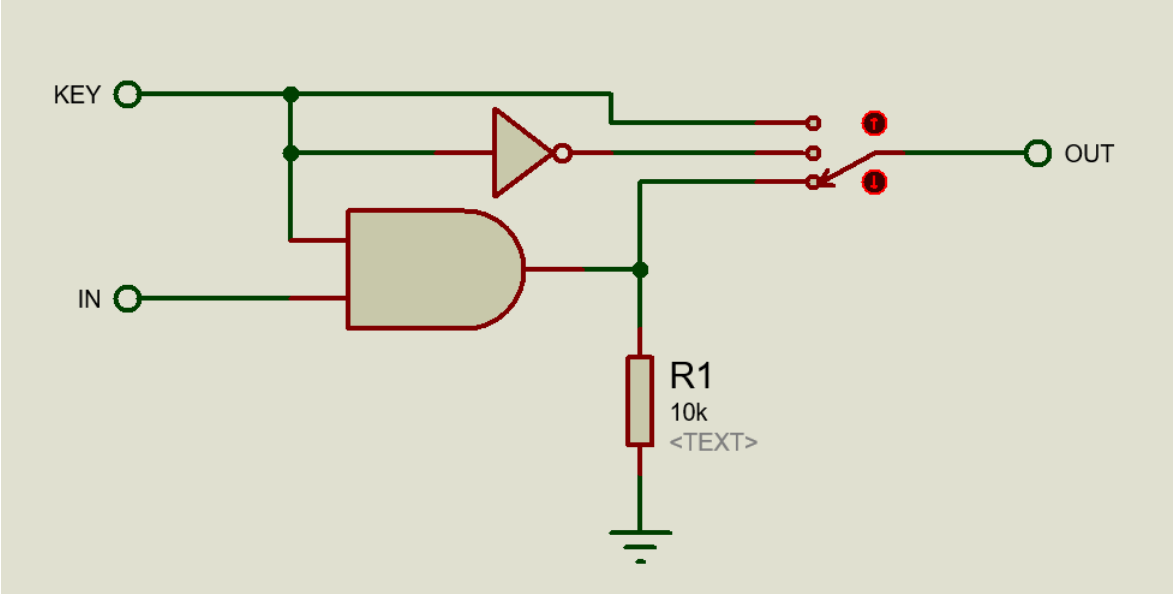
Fonte: Autores.

2.3.4 Implementação da chave de 3 vias e os botões

A implementação em esquemático da chave seletora de 3 vias usada no projeto sofreu uma alteração já comentada anteriormente. A alteração principal consiste na alteração da utilização de um mosfet por uma porta lógica AND como representado na Figura 16. A label de nome KEY representa a saída controlada pelo microcontrolador. Esta será 1 quando a data e hora atual estiverem dentro do intervalo dos alarmes agendados. A label de nome IN representada a entrada direta do programador horário e por fim, a label OUT representa a saída que tem o valor controlado pela chave.

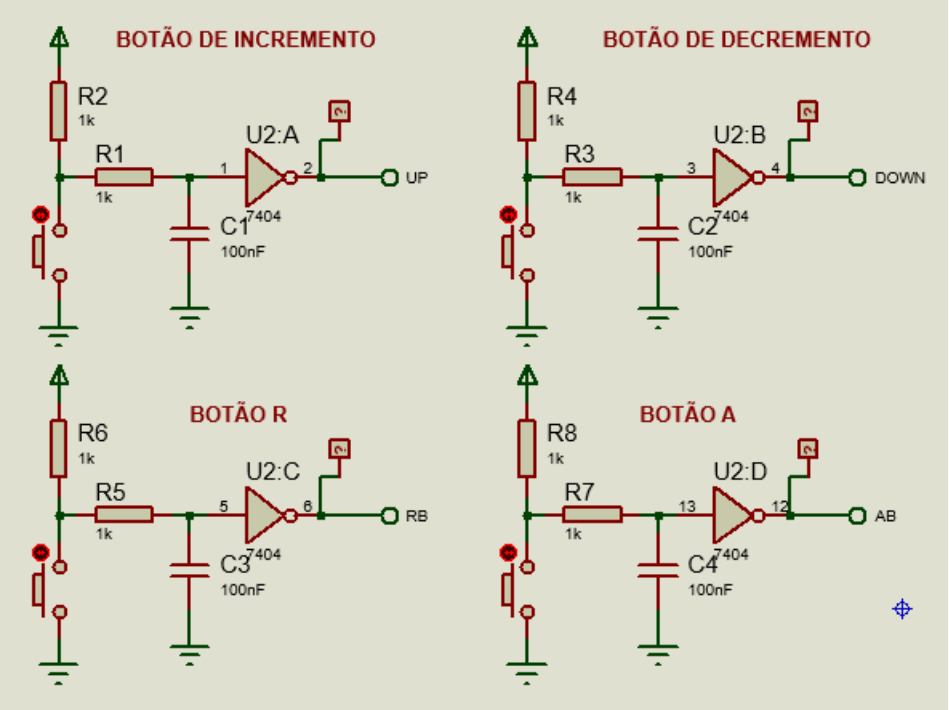
Ao que se refere à implementação dos botões existentes na interface homem-máquina, foi realizado o circuito de *debounce* para cada. As portas designadas para cada botão foram as PD4 para o INCREMENTO, PD5 para o DECREMENTO, PD6 para R e PD7 para A. O circuito aplicado pode ser visto na Figura 17.

Figura 16 – Chave Seletora de 3 Vias



Fonte: Autores.

Figura 17 – Botões



Fonte: Autores.

3 CONCLUSÃO

A implementação do projeto disposto passou por algumas mudanças com o intuito de economizar alguns recursos do microcontrolador, como é o caso da não utilização do protocolo I2C. Também, durante o desenvolvimento do assembly, foram encontradas algumas dificuldades na hora de configurar os LEDs utilizando o *charlieplexing*, principalmente os LEDs dos processos. Por fim ficou decidido que somente os leds dos dias das semanas iriam utilizar tal técnica.

Outro desafio foi a utilização do *display* e a comunicação SPI do ATmega328p com o MAX7912. A priori exibir valores do microcontrolador para o *display* por meio do MAX7912 foi uma etapa bem sucedida, mas realizar a leitura do RTC e exibir no *display* foi uma tarefa que não conseguimos realizar por múltiplos problemas.

Além disso, não foi possível realizar o ajuste de agenda com os leds acesos. Apesar de ser possível acender os leds nos modos de operação necessários, o problema encontrado consistia em um "salto" indevido nos modos de operação. Assim, por exemplo, quando estávamos colocando o horário inicial do agendamento, ao chegar no momento de escolher o dia da semana, o programa pulava para o estado de definir a hora final do agendamento. Não conseguimos resolver esse problema, que aliado aos outros nos impediu de entregar o projeto completamente implementado.

Assim, o que foi possível implementar foram os estados ajuste de relógio (com leds funcionanado) e agendamento (sem leds funcionando), devido a não implementação do *display* o estado "Run" não foi implementado.

Referências

ATMEL CORPORATION. *ATMega328p*: Atmel 8-bit microcontroller with 4/8/16/32kbytes in-system programmable flash. Califórnia, USA, 2015. 660 p.

DIAS, S. M. *Problema 04: Implementação*. [S.l.]: Departamento de Engenharia Elétrica, 2021.

INTEGRATED, M. *Datasheet DS3234*. 2015. 28 jul. 2021. Disponível em: <https://datasheets.maximintegrated.com/en/ds/DS3234.pdf>.

MAXIM. *Serially Interfaced, 8-Digit LED Display Drivers*. 1997. Acessado em: 25 jul. 2021. Disponível em: <https://pdf1.alldatasheet.com/datasheet-pdf/view/73745/MAXIM/MAX7219.html>.

ANEXO A – Relato semanal

Líder: STHEFANIA FERNANDES SILVA

A.1 Equipe

Tabela 1 – Identificação da equipe

Função no grupo	**Nome completo do aluno**
Redator	WESLEY BRITO DA SILVA
Debatedor	ANNY BEATRIZ PINHEIRO FERNANDES
Videomaker	ALYSSON FERREIRA DA SILVA
Auxiliar	ISAAC DE LYRA JUNIOR

Fonte: Autores.

A.2 Defina o problema

O problema consiste em projetar o circuito de um relógio programável que permite a alteração e agendamento de horários (horas e minutos) e dias.

Para alterar o horário do relógio há três modos de operação: Timer_H, Timer_M e Week, os quais são sinalizados para usuários através do led T aceso; led T piscando e leds T e D1 acesos e led W piscando, respectivamente. Já para agendar um evento há 6 modos de operação: On_H (led O aceso), On_M (led O piscando) e Week_On (leds O e D1 acesos e led W piscando), responsáveis por definir o horário e dia inicial e Off_H (led F aceso), Off_M (led F piscando) e Week_Off (leds F e D1 acesos e led W piscando) responsáveis por definir o horário e dia final do evento. Na interface homem-máquina também há um display que exibe a hora atual e os valores que estão sendo alterados e sete leds que representam os sete dias da semana.

Além disso, há uma chave de 3 vias que define a saída "Out" do relógio. Quando a chave está em "NA", temos que "Out" possuí nível lógico alto, quando a hora atual estiver no interior do intervalo de horário agendado pelo usuário e 0 para as demais situações. Quando em "NF" a saída Out se comporta de forma oposta ao caso da "NA". Por último, em "EX", a saída "Out" terá o mesmo valor da entrada In quando a hora atual estiver no interior do intervalo agendado pelo usuário e 0 para as demais situações.

A.3 Registro de *brainstorming*

As reuniões iniciaram na quarta-feira e perduraram até a segunda-feira. Começando pela quarta-feira, nela foi conversado sobre os detalhes do projeto, visando discutir o que seria implementável ou não, nela foi decidido que tentaríamos tudo o que foi proposto pelo grupo projetista. Na quinta-feira foram definidos os cargos do grupo e começou a se realizar a comunicação SPI, para isso lemos e relemos o *datasheet* do DS3234, até então todo o grupo estava realizando a implementação junto. No fim da reunião ficou decidido que três integrantes (Alysson, Anny e Wesley) iriam ficar responsáveis pelo *charlieplexing* dos leds, enquanto eu e Isaac iríamos fazer a comunicação do ATMEGA com RTC definindo o ajuste de relógio e ajuste de agendamento.

Nos dias seguintes (sexta, sábado e domingo) os dois "subgrupos" ficaram dedicando seu tempo a realizar sua tarefa, encontrando e corrigindo erros ao longo da implementação. No domingo, o subgrupo responsável pelo *charlieplexing* desistiu de implementar os 11 leds com tal método, pois não foi encontrada uma forma de piscar e deixar um led estático com o método implementado. Além disso, foi desistida da ideia de implementar o *display* usando o I2C, diante da complexidade do protocolo e também diante da possibilidade de usar o SPI adicionando apenas um servo a mais. Diante disso, foi decidido que seria usado o MAX7912 para exibir no *display* o horário do relógio programador.

Na segunda-feira, foi iniciado o relatório e os últimos erros dos leds, *display* e dos modos de operação foram corrigidos, assim como, foi feito o slide e vídeo da implementação.

A.4 Pontos-chaves

O ponto chave do projeto foi a compreensão do protocolo SPI, da técnica *charlieplexing* e dos CI's DS3234 e MAX7912. Além da necessidade do bom entendimento do ATMEGA328P e das suas instruções.

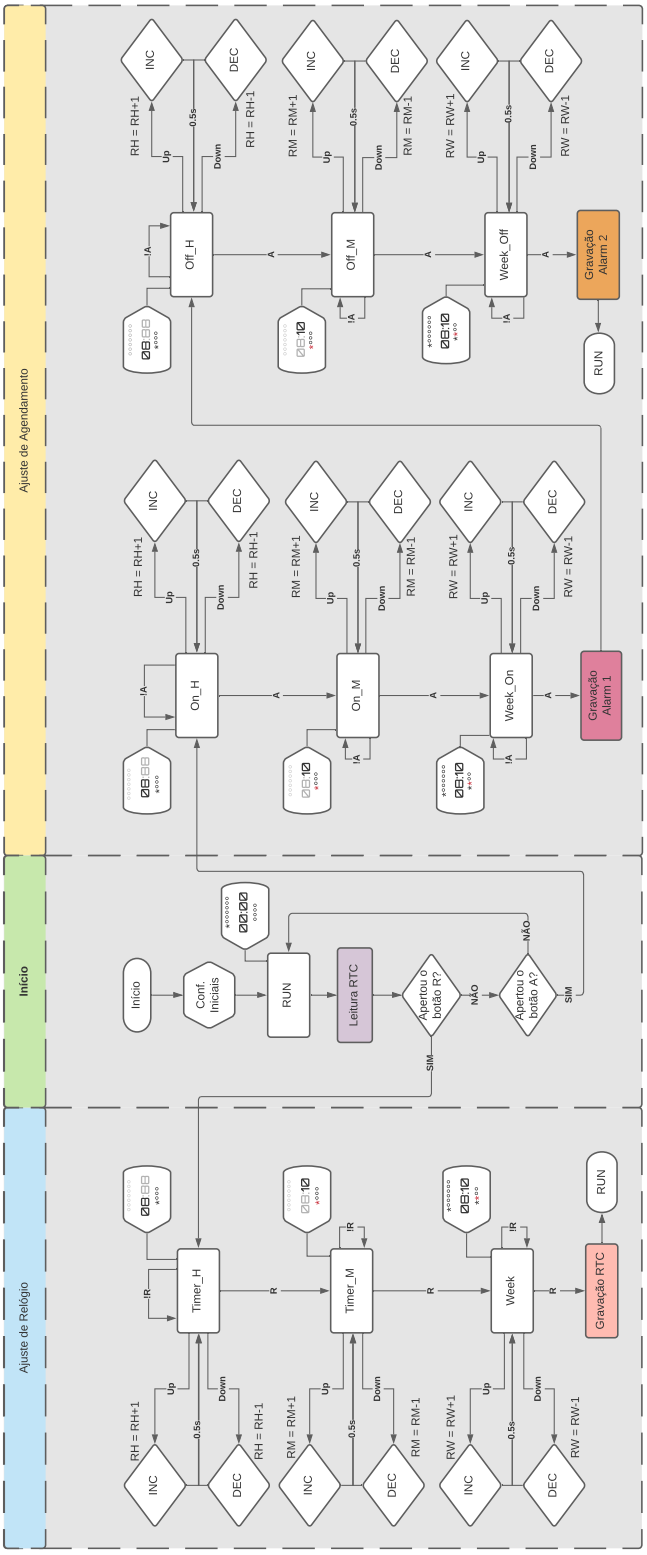
A.5 Questões de pesquisa

Os tópicos que foram pesquisados foram os CI's DS3234 e MAX7912, o protocolo SPI do ATMEGA328P, como também as instruções do microcontrolador.

A.6 Planejamento da pesquisa

Como fonte de pesquisa foram utilizados os *datasheets* dos CI's mencionados e também vídeo aulas no Youtube.

ANEXO B – Diagrama de blocos do relógio



ANEXO C – Esquemático no Proteus