

MARCIO CARVALHO DE BRITO FILHO 20190153992
ALAX GABRIEL CAVALCANTE LIMA DE OLIVEIRA 20190001785
IAGO PEREIRA CASSEL 20170030770
JARDEL GREGORIO DO NASCIMENTO 20190001829

**PROJETO DE UMA CENTRAL DE ALARME
DIGITAL EM UM MICROCONTROLADOR
AVR(ATMega 328P)**

Natal
Dezembro de 2020

MARCIO CARVALHO DE BRITO FILHO 20190153992
ALAX GABRIEL CAVALCANTE LIMA DE OLIVEIRA 20190001785
IAGO PEREIRA CASSEL 20170030770
JARDEL GREGORIO DO NASCIMENTO 20190001829

PROJETO DE UMA CENTRAL DE ALARME DIGITAL EM UM MICROCONTROLADOR AVR(ATMega 328P)

Décimo Segundo Relatório apresentado à disciplina de Sistemas Digitais, correspondente ao semestre 2020.6 do curso de Engenharia Mecatrônica da UFRN referente à implementação de um projeto de circuito de um alarme digital baseado em microcontrolador AVR(ATMega 328P).

Universidade Federal do Rio Grande do Norte

Curso de Engenharia Mecatrônica

Disciplina de sistemas digitais

Natal

Dezembro de 2020

Resumo

O presente trabalho detalha o projeto em termos de periféricos, protocolos e máquina de estados para a implementação em linguagem C de uma central digital de alarme utilizando o microcontrolador AVR (ATMEGA328p).

Palavras-chaves: AVR, ATMEGA328p, Central de Alarme, C.

Lista de ilustrações

Figura 1 – Interface Homem Máquina da Central de Alarme	8
Figura 2 – Registrador de dados UDRn	10
Figura 3 – Frame Format	11
Figura 4 – Registrador de dados UCSRnA	11
Figura 5 – Registrador de dados UCSRnB	12
Figura 6 – Registrador de dados UCSRnC	13
Figura 7 – Registrador da taxa de <i>baud</i>	15
Figura 8 – Registrador de endereço da EEPROM	17
Figura 9 – Registrador de dados da EEPROM	17
Figura 10 – Registrador de controle e status da EEPROM	17
Figura 11 – Formato do arquivo de log	20
Figura 12 – Endereços e prioridade de interrupções	21
Figura 13 – Registrador TCCR1B	22
Figura 14 – Registrador TCCR1A	22
Figura 15 – Registrador TIMSK1	22
Figura 16 – Registrador OCR1A	22
Figura 17 – Registrador TCNT1	23
Figura 18 – Esquemático Teclado de 16 Teclas	26
Figura 19 – LCD 16x2 - LM016L	28
Figura 20 – Registradores do MAX6902	33
Figura 21 – Registradores do MAX6902	34
Figura 22 – Endereços dos registradores e descrição	34
Figura 23 – Conversão paralelo-serial em CI 74165	37
Figura 24 – Erro máximo recomendado da taxa de transmissão do receptor para o modo de velocidade normal	38
Figura 25 – Exemplos de configurações UBRn para frequências de oscilador comumente usadas	39
Figura 26 – Máquina de estados finitos de modos de operação	39
Figura 27 – Tela do modo recuperação	40
Figura 28 – Tela do modo desativado	40
Figura 29 – Tela do modo checar senha usuário	41
Figura 30 – Tela do modo ativado	42
Figura 31 – Tela do modo pânico	42
Figura 32 – Tela do modo programação	43
Figura 33 – Máquina de estados finitos do modo de programação	43
Figura 34 – Tela do estado de config select	44

Figura 35 – Tela do estado de habilitação	44
Figura 36 – Tela do estado de escolha do usuário	45
Figura 37 – Tela do estado da nova senha	45
Figura 38 – Tela do estado de habilitação do sensor	45
Figura 39 – Tela do estado de selecionar sensor para associar a uma zona	46
Figura 40 – Tela do estado de selecionar zona a qual sensor será associado	46
Figura 41 – Tela do estado de habilitar zona	47
Figura 42 – Tela do estado do temporizador de ativação	47
Figura 43 – Tela do estado do temporizador de timeout	48
Figura 44 – Tela do estado do temporizador da sirene	48
Figura 45 – Tela do estado de desabilitação	48
Figura 46 – Tela do estado de desabilitação do sensor	49
Figura 47 – Tela do estado de selecionar sensor para desassociar de zona	49
Figura 48 – Tela do estado de selecionar zona a qual sensor será desassociado	50
Figura 49 – Tela do estado de desabilitar zona	50
Figura 50 – Passo 1: Selecionando o sensor 5	51
Figura 51 – Passo 2: Selecionando a zona 2	51
Figura 52 – Passo 3: Modo ativado monitorando os sensores	51
Figura 53 – Passo 4:	52
Figura 54 – Circuito final da Central de Alarme	53

Lista de tabelas

Tabela 1 – Tabela de seleção do modo de operação do USART	14
Tabela 2 – Tabela de seleção da paridade dos dados	14
Tabela 3 – Tabela de seleção do modo de operação do USART	14
Tabela 4 – Tabela de seleção do modo de operação da EEPROM	18
Tabela 5 – ADMUX	24
Tabela 6 – ADCSRA	24
Tabela 7 – Seleção da tensão de referência do ADC	25
Tabela 8 – Seleção do canal de entrada	25
Tabela 9 – Divisor de clock	26
Tabela 10 – Tabela de pinos do LCD	29
Tabela 11 – Códigos de comandos do LCD	30
Tabela 12 – Endereço das posições do LCD	30
Tabela 13 – Portas do protocolo SPI.	31
Tabela 14 – SPSR (SPI Status Register)	31
Tabela 15 – SPCR (SPI Control Register)	32
Tabela 16 – SPDR (SPI Data Register)	32

Lista de abreviaturas e siglas

CI	Circuito Integrado
RTC	Real Time Clock
SPI	Serial Peripheral Interface
SPSR	SPI Status Register
SPCR	SPI Control Register
SPDR	SPI Data Register
SPIF	Sinalizador de interrupção SPI
WCOL	Write Colision Flag
SPI2X	Double SPI Speed
SPIE	SPI Interrupt Enable
SPE	SPI Enable
DORD	Data Order
USART	Universal Synchronous Asynchronous Receiver Transmitter
UART	Universal Asynchrounous Receiver/Transmitter
LCD	Liquid Crystal Display
EEPROM	Electrically Erasable Programmable Read-Only Memory

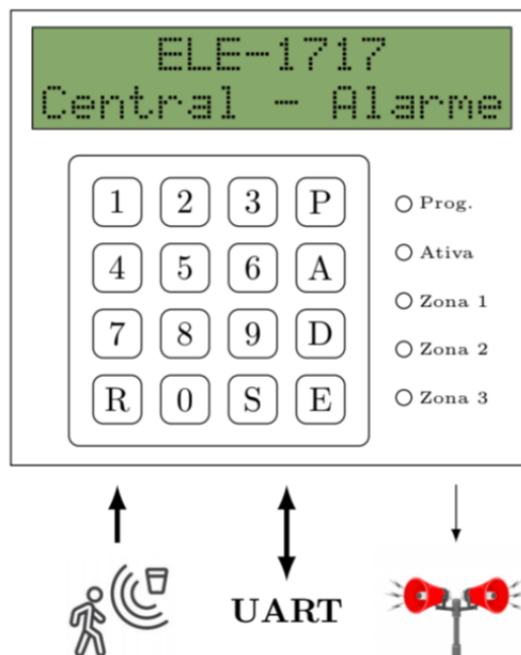
Sumário

	Sumário	7
1	Introdução	8
2	Desenvolvimento	10
2.1	USART	10
2.1.1	Implementação do USART	15
2.2	EEPROM	16
2.2.1	Implementação do EEPROM	18
2.3	Arquivo LOG	19
2.4	Interrupções	20
2.5	Timer	21
2.5.1	Implementação Timer	23
2.6	Conversor Analógico - Digital	24
2.7	Teclado 16 Teclas	26
2.7.1	Implementação Teclado	27
2.8	Display LCD	28
2.8.1	Pinagem do LCD	29
2.8.2	Códigos de comando do LCD	30
2.9	SPI	31
2.9.1	RTC MAX6902	33
2.9.2	Implementação SPI	35
2.10	Expansão de entradas e saídas	37
2.10.1	Aumento do número de entradas - Conversão Paralelo-Serial	37
2.10.1.1	Implementação dos sensores	37
2.11	Clock	38
2.12	Máquina de estados finitos	39
2.13	Exemplo de operação na central de alarme	50
3	Resultados	53
4	Conclusão	54
	REFERÊNCIAS	55

1 Introdução

Este relatório focou na implementação da central de alarme projetada pelo grupo anterior. Uma central de alarme é amplamente usada para monitorar um determinado local com o intuito de identificar falhas, o exemplo mais comum de uso desse sistema seria uma central para identificação de incêndios. Para este projeto a central acionará uma sirene sempre que algum sensor de movimento detectar um movimento em um ambiente. Abaixo na figura 19 se encontra a interface homem máquina para este projeto.

Figura 1 – Interface Homem Máquina da Central de Alarme



Fonte: Datasheet Atmega328p

Um display LCD para para exibição dos parâmetros e dados para a central é usado, leds para sinalizações do modo configuração e ativação para zonas, também um canal de entrada para os sensores, um canal de saída para a sirene e um canal de comunicação UART para acesso ao arquivo de log.

A central de alarme possui 5 modos de operação. O primeiro é o desativado, no qual a sirene não será acionada mesmo com a indicação de movimento por algum dos sensores de entrada. O segundo é o modo ativado, no qual, de acordo com a configuração da central de alarme, a sirene pode ser acionada a partir da indicação de movimento por um dos sensores de entrada. O terceiro modo, no qual a sirene não é acionada e é permitido a reconfiguração do sistema. O quarto modo é chamado de pânico, no qual o usuário através do pressionamento da tecla S pode acionar ou desacionar a sirene de modo manual. Por fim, o quinto e o último modo é o de recuperação, o qual se trata de um

modo de transição necessário e utilizado apenas para retornar as configurações da central aos parâmetros de fábrica.

2 Desenvolvimento

2.1 USART

Um *Universal Synchronous Asynchronous Receiver Transmitter* abreviado como USART é um Transmissor/Receptor do tipo Universal Síncrono e Assíncrono. É um periférico do microcontrolador para comunicação de dados de forma serial, ou seja, ele converte bytes de entrada e saída de dados em um fluxo de bits serial e como o próprio nome já remete essa comunicação pode ser do tipo assíncrona ou síncrona.

O USART permite o uso de comunicação do tipo full duplex no qual é capaz de transmitir e receber ao mesmo tempo, com dados de 5 a 9 bits, bits de parada, modos de paridade e variedades nas taxas de transmissão.

Dado que, a central de alarme deve criar um arquivo de log, no qual irá registrar dados como usuário, dia da semana e a hora que a central foi configurada, ativada ou desativada, o USART terá como propósito a consulta desse arquivo. Sendo assim é necessário entender como funciona seus registradores internos no microcontrolador AVR Atmega328p.

O USART possui um total de cinco registradores, um para os dados a serem transmitidos e recebidos, três registradores de controle e status e um para a taxa de *baud*, a seguir será detalhado esses registradores.

O registrador de buffer de transmissão do USART e os registrador de buffer de recepção compartilham do mesmo endereço I/O. O registrador de transmissão (TXB) será o destino para os dados gravados no local do registrador UDRn na figura 2 e ler esse registrador irá retornar o conteúdo do RXB.

Figura 2 – Registrador de dados UDRn

UDRn – USART I/O Data Register n

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDRn (Read)
	TXB[7:0]								UDRn (Write)
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	

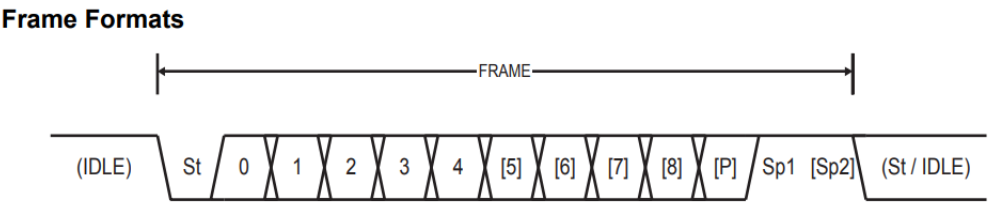
Fonte: Datasheet Atmega328p

Para caracteres de 5,6 ou 7 bits, os bits superiores não utilizados são ignorados pelo transmissor e setados para zero pelo receptor.

O USART possui um padrão de formato de dados(*Frame Format*) constituído de bits de sincronização, no caso, bits de inicio e parada do caractere e bits de paridade para

checagem de erros. Abaixo na figura 3 se encontra as possíveis combinações para compor esse *Frame Format*. Os bits em colchetes são bits não obrigatórios.

Figura 3 – Frame Format



Fonte: Datasheet Atmega328p

- **St** Bit de início, sempre baixo.
- **n** Bits de dados (0 to 8).
- **P** Bit de Paridade. Pode ser par ou ímpar.
- **Sp** Bit de Parada, sempre alto.
- **IDLE** Sem transferência na linha de comunicação (RxDn ou TxDn). Uma linha IDLE deve estar em nível lógico alto.

O primeiro registrador de controle e status do USART é o UCSRnA mostrado na figura 4 abaixo.

Figura 4 – Registrador de dados UCSRnA

UCSRnA – USART Control and Status Register n A

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREN	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Fonte: Datasheet Atmega328p

- **Bit 7 - RXCn (Recepção Completa):** Esta flag indica se existem dados que ainda não foram lidos no buffer de recepção, sendo assim quando este bit possui nível lógico alto existem dados no buffer de recepção que precisam ser lidos, quando esses dados são lidos ou jogados fora, o bit volta possuir nível lógico baixo.
- **Bit 6 - TXCn (Transmissão Completa):** A flag TXCn possui nível lógico alto quando o registrador de transmissão foi enviado por completo, para que esse bit volte a ser zero se faz necessário a escrita "manual"do valor '1' nessa flag, ou o uso de uma interrupção.

- **Bit 5 - UDREn (Registrador de dados vazio):** O bit UDREn indica se o buffer de transmissão UDRn está vazio, caso UDREn='1', significa que o buffer já está pronto para receber novos dados.
- **Bit 4 - FEn (Erro de Frame):** Esse bit é setado caso exista um erro nos dados recebidos. Possui nível lógico '1' quando o bit de parada dos dados recebidos é um. Este bit é válido até que o buffer de recepção (UDRn) seja lido. É importante definir esse bit como zero sempre ao gravar em UCSRnA.
- **Bit 3 - DORn (Data OverRun):** Indica com nível lógico '1' que o buffer de recepção está cheio (2 caracteres) e um novo caractere está esperando no registrador *Receive Shift* e um novo bit de início é detectado. Este bit é válido até que o buffer de recepção (UDRn) seja lido. É importante definir esse bit como zero sempre ao gravar em UCSRnA.
- **Bit 2 - UPEn (Erro de Paridade):** Esse bit indica que o próximo caractere no buffer de recepção teve um erro de paridade ao ser recebido e verificação de paridade estava ativa. Este bit UPEn é válido até que o buffer de recepção (UDRn) seja lido. É importante definir esse bit como zero sempre ao gravar em UCSRnA.
- **Bit 1 - U2Xn (Dobra a velocidade de transmissão):** Esse bit apenas tem efeito para o modo de operação assíncrona. Utilizar em '0' caso deseje usar o modo síncrono mas Quando '1' irá reduzir o divisor da taxa de *baud* dividindo-o por 8 e consequentemente dobrando a taxa de transferência para a comunicação assíncrona. Para este projeto vale ressaltar que o bit acima, ou seja, o modo de operação deverá ser setado para trabalhar de maneira assíncrona, para funcionar como o UART.
- **Bit 0 - MPCMn (Modo de Comunicação com Multi-Processadores):** Quando '0' desabilita esse modo e quando '1' todos os dados recebidos que não tiverem informações de endereço serão ignorados.

O segundo registrador de controle do USART é o UCSRnB, figura 5, nesse registrador pode ser realizada a ativação de: interrupções e da recepção ou transmissão de dados pelo USART.

Figura 5 – Registrador de dados UCSRnB

UCSRnB – USART Control and Status Register n B

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - RXCIEn (interrupção da flag RX):** Este bit tem como função ativar a interrupção gerada quando o USART completa por total uma recepção de dados, isto é feito definindo RXCIEn=1.
- **Bit 6 - TXCIEn (interrupção da flag TX):** Este bit tem como função ativar a interrupção gerada quando o USART completa por total uma transmissão de dados, isto é feito definindo TXCIEn=1.
- **Bit 5 - UDRIEn (interrupção caso o registrador de dados UDRn esteja vazio):** Este bit tem como função ativar a interrupção gerada quando o registrador UDRn do USART está vazio, indicado pela flag UDREn.
- **Bit 4 - RXENn (ativação do receptor do USART):** Ao ser setado, esse bit ativa a recepção de dados pelo buffer de recepção do USART.
- **Bit 3 - TXENn (ativação do transmissor do USART):** Ao ser setado, esse bit ativa a transmissão de dados pelo USART, quando se é definido TXENn=0, o transmissor só será de fato desativado quando todos os dados no registrador forem de fato enviados.
- **Bit 2 - UCSZn2 (ativação do transmissor do USART)**
- **Bit 1 - RXB8n (bit 8 do dado recebido):** O bit RXB8n é o oitavo bit recebido quando o USART está operando com frames de 9 bits.
- **Bit 0 - TXB8n (bit 8 do dado à ser transmitido):** O bit TXB8n é o oitavo bit do dado que será enviado caso o USART esteja operando com frames de 9 bits.

O terceiro registrador de controle do USART é o UCSRnC, demonstrado na figura 6, onde é possível realizar o controle do modo operacional do USART, e de detalhes sobre o envio e recepção de dados.

Figura 6 – Registrador de dados UCSRnC

USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
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	1	1	0	

Fonte: Datasheet Atmega328p

- **Bit (7:6) - UMSELn1:0 (seleção do modo de operação do USART):** Os modos de operação do USART podem ser selecionados seguindo a tabela:

Tabela 1 – Tabela de seleção do modo de operação do USART

UMSELn1	UMSELn0	Modo
0	0	Modo assíncrono
0	1	Modo síncrono
1	0	Reservado
1	1	Modo SPI

Tabela 2 – Tabela de seleção da paridade dos dados

UPMn1	UPMn0	Modo de paridade
0	0	Desativado
0	1	Reservado
1	0	Paridade par
1	1	Paridade ímpar

- **Bit (5:4) - UPMn1:0 (seleção do modo de paridade):** Estes bits selecionam o tipo de paridade que será gerado e enviado em cada frame de acordo com a tabela 2.
- **Bit 3 - USBSn (seleção dos Stop bits):** Este bit seleciona a quantidade de Stop bits que serão utilizados durante a transmissão dos dados, caso USBSn='1' são utilizados 2 Stop bits, caso contrário é utilizado 1 Stop bit.
- **Bit (2:1) - UCSZn1:0 (seleção do tamanho do frame):** Juntamente com o bit UCSZn2, estes bits selecionam o tamanho dos dados que serão recebidos ou enviados pelo USART, de acordo com a tabela 3

Tabela 3 – Tabela de seleção do modo de operação do USART

UCSZn2	UCSZn1	UCSZn0	Tamanho do frame
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reservado
1	0	1	Reservado
1	1	0	Reservado
1	1	1	9-bit

- **Bit 0 - UCPOLn (seleção da polaridade do clock):** Usado apenas no modo síncrono

Por fim os registradores da taxa de transmissão, o UBRRnL e UBRRnH, demonstrado na figura 7, no qual é guardado a taxa de *baud*.

Figura 7 – Registrador da taxa de *baud*

UBRRnL and UBRRnH – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	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	
	0	0	0	0	0	0	0	0	

Fonte: Datasheet Atmega328p

- **Bit 15:12 - Reservado:** Esses bits são reservados para uso futuro. Para compatibilidade com futuros dispositivos, esse bit deve ter sinal lógico '0' quando UBRRnH é escrito.
- **Bit 11:0 - Registrador da taxa de *Baud*:** O UBRRnH contém os quatro bits mais significativos e o UBRRnL os 8 bits menos significativos da taxa de *baud*. Transmissões em andamento serão corrompidas caso haja mudança na taxa de transmissão. Escrever em UBRRnL irá desencadear uma atualização imediata do prescaler da taxa de transmissão.

O valor guardado nesse registrador UBRRn e a taxa de *baud* depende das seguintes fórmulas para o modo de operação assíncrono:

$$baud = \frac{f_{osc}}{16 \cdot (UBRRn + 1)} - 1 \quad (1)$$

$$UBRRn = \frac{f_{osc}}{16 \cdot BAUD} - 1 \quad (2)$$

De acordo com as escolhas padrões de Baud Rate, a frequência escolhida foi de 9600 bps.

2.1.1 Implementação do USART

Para implementar o USART primeiro foi criada uma função para inicializar com a configuração dos registradores que setam o formato do frame transmitido/recebido, como também o enable dos modos de transmissão e recepção.

```
#define BAUD 9600    // Baud Rate
#define UBRR_PROJETO CLK/16/BAUD-1
//(((F_CPU / (BAUDRATE * 16UL))) - 1)
```



```

void USART_inicializacao(unsigned int ubrr){

UBRR0H = (unsigned char)(ubrr>>8);           // Configura o Baud Rate High
UBRR0L = (unsigned char)ubrr;                 // Configura o Baud Rate Low
UCSR0B = (1<<RXEN0)|(1<<TXEN0);              // Dá enable no transmissor de receptor
UCSR0C = (2<<UPM00)|(0<<USBS0)|(3<<UCSZ00);   // Configura 8 bits de palavra e 1 bit de parada
}

```

Foi configurado um frame com palavra de 8 bits, um bit de parada e um bit de paridade par. Dessa forma, foi inserido a chave "011" no conjunto de bits que representam os UCSZ0n, para configurar a palavra de oito bits. Além disso uma chave "10" nos bits UPM0n para configurar dar enable na paridade par e USBS0 igual a "0" para identificar um bit de parada.

Após isso, foram feitas as funções de transmissão e recebimento de dados através da USART. Para função de transmissão temos:

```

void USART_transmissao(unsigned char mensagem_usart){

while (!(UCSR0A & (1<<UDRE0)));              // Espera o esvaziamento do buffer de transmissão
UDR0 = mensagem_usart;                       // Coloca a mensagem no buffer
}

```

A mensagem contida no UDR0 é carregada por uma variável do sistema, que será transmitida pelo protocolo. Entretanto, isso só é feito após o sistema verificar que o buffer da transmissão esvaziou, ou seja, que toda transmissão anterior foi feita. Por fim, para função de recepção temos:

```

unsigned char USART_recepcao(void){

while (!(UCSR0A & (1<<RXC0)));              // Espera uma mensagem ser recebida
return UDRn0;                               // Retorna a mensagem do buffer
}

```

De forma similar a função de transmissão, a mensagem contida no UDR0, recebida pelo protocolo, é retornada pela função do sistema. Entretanto, isso só é feito após o sistema verificar que o buffer da recepção esvaziou, ou seja, que toda recepção anterior foi feita.

2.2 EEPROM

A EEPROM é um tipo de memória não-volátil integrada em diversos microcontroladores e computadores, o ATmega possui uma EEPROM DE 1K bytes de memória, ela é

um espaço de dados separados, na qual bytes podem ser escrito ou lidos.

O fato de que a EEPROM é uma memória não-volátil, ou seja, a partir do momento em que os dados são gravados, eles serão conservados mesmo se o microcontrolador seja desconectado de sua fonte de energia, torna este tipo de hardware ideal para o armazenamento de arquivos de registro, cujos dados não podem ser perdidos, como o arquivo de log necessário para a central de alarme, que registra informações sobre os acessos ao sistema realizados pelos usuários.

Para realizar a leitura e escrita de dados na EEPROM é necessário o uso de três registradores, o EEAR, demonstrado na figura 8, que recebe o endereço no qual deve ser realizada a leitura ou escrita de dados, o EEDR, figura 9, é o registrador que contém o conteúdo que deve ser escrito na memória, no caso de uma operação de escrita, ou o conteúdo que foi lido da memória, no caso de uma operação de leitura.

Figura 8 – Registrador de endereço da EEPROM

Bit	15	14	13	12	11	10	9	8	
0x22 (0x42)	–	–	–	–	–	–	–	EEAR8	EEARH
0x21 (0x41)	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

Fonte: Datasheet Atmega328p

Figura 9 – Registrador de dados da EEPROM

Bit	7	6	5	4	3	2	1	0	
0x20 (0x40)	MSB							LSB	EEDR
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 Atmega328p

Por fim é necessário realizar a configuração do registrador EECR, figura 10.

Figura 10 – Registrador de controle e status da EEPROM

Bit	7	6	5	4	3	2	1	0	
0x1F (0x3F)	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	EECR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	X	X	0	0	X	0	

Fonte: Datasheet Atmega328p

Os bits EEPM1 e EEPM0 (5 e 4) selecionam qual operação será realizada pelo microcontrolador quando o bit EEPE, que inicia a operação de escrita, for definido como um. Essa seleção ocorre de acordo com a tabela 4.

Tabela 4 – Tabela de seleção do modo de operação da EEPROM

EEPM1	EEPM0	operação
0	0	apagar e escrever
0	1	somente apagar
1	0	somente escrever
1	1	reservado

O bit EERIE tem como função ativar uma interrupção para o momento em que a escrita dos dados tenha sido finalizada, já os bits EEMPE e EEPE são ambos responsáveis pela realização da operação de escrita na memória, o bit EEPE é o indicador que uma operação de escrita deve ser realizada na EEPROM, ou seja, quando este bit é definido em nível lógico alto a operação de escrita na memória se iniciaria, porém caso o bit EEMPE não for setado previamente ao bit EEPE, mesmo que o EEPE seja definido como '1' a operação de escrita não ocorrerá, ao fim de uma operação de escrita o bit EEPE é definido com '0' automaticamente pelo microcontrolador.

Sendo assim para a escrita de dados na EEPROM é necessário seguir os passos:

1. Esperar o bit EEPE='0';
2. Definir no registrador EEAR o endereço da memória no qual os dados devem ser escritos;
3. Definir no registrador EEDR os dados que devem ser escritos na memória;
4. Definir o bit EEMPE='1', enquanto EEPE='0';
5. Definir o bit EEPE='1' após 4 ciclos de clock.

Por fim o bit EERE inicia uma leitura do endereço da memória presente no registrador EEAR, novamente se faz necessário a checagem do bit EEPE para que se seja realizada esta operação, pois caso EEPE ainda seja igual a 1, não só a leitura não ocorrerá como também a alteração do EEAR para o endereço de leitura não será realizada.

2.2.1 Implementação do EEPROM

Diferente do USART, a EEPROM não precisa de uma função de inicialização. Só foi preciso configurar as funções escrita e leitura de dados. Para função de escrita temos:

```
void EEPROM_write(unsigned int endereco_escrita_eeprom,
                  unsigned char mensagem_escrita_eeprom){

while(EECR & (1<<EEPE));           // Espera a finalização da ultima escrita
```

```

EEAR = endereco_escrita_eeprom;           // Configura o endereço de escrita
EEDR = mensagem_escrita_eeprom;           // Passa a mensagem
EECR |= (1<<EEMPE);                       // Da enable na escrita
EECR |= (1<<EEPE);                         // Começa a escrita
}

```

A EEPROM espera a ultima escrita finalizar para iniciar uma nova. Com isso, o endereço da escrita e a mensagem a ser escrita são adicionadas nos registradores de controle reservados da ATmega. Após isso o bit EEMPE é habilitado para dar enable na escrita. Em sequência do enable na escrita, o bit EEPE é setado em nível alto para iniciar o processo de escrita na EEPROM.

Para função de leitura temos:

```

unsigned char EEPROM_read(unsigned int endereco_leitura_eeprom){

while(EECR & (1<<EEPE));                   // Espera a finalização da ultima escrita
EEAR = endereco_leitura_eeprom;           // Configura o endereço de leitura
EECR |= (1<<EERE);                         // Da enable na leitura
return EEDR;                             // Retorna a mensagem da leitura
}

```

Como na escrita, a EEPROM espera a ultima escrita finalizar para iniciar uma leitura. O registrador EEAR é carregado com o endereço do dado a ser lido na memória. Após o carregamento do endereço, o bit EERE é setado em 1 para dar enable na leitura dos dados. Por fim, a função retorna o que foi lido na memória e guardado no registrador EEDR.

2.3 Arquivo LOG

O arquivo log é um registro de todos os acessos realizados ao sistema da central de alarme, cada registro deve possuir as informações sobre qual dos usuários acessou o sistema, além do horário e dia da semana que o acesso ocorreu. Estas informações devem ser armazenadas na EEPROM integrada ao ATmega 328p, e a única maneira que este arquivo pode ser acessado é a partir de uma canal de comunicação UART.

As informações sobre cada acesso devem ser armazenadas em dois bytes seguindo o formato demonstrado pela figura:

Figura 11 – Formato do arquivo de log

Byte 0

Bit	7	6	5	4	3	2	1	0
	USU1	USU0	HR4	HR3	HR2	DS2	DS1	DS0

Byte 1

Bit	7	6	5	4	3	2	1	0
	HR1	HR0	MIN5	MIN4	MIN3	MIN2	MIN1	MIN0

Fonte: Autoria própria.

Onde no Byte 0 os bits USU1:0 representam qual dos usuários realizou o acesso à central de alarme, os bits DS2:0 indicam o dia da semana em binário que o acesso pelo determinado usuário foi realizado.


Já os bits HR4:0 representam o valor em binário da hora em que o acesso foi realizado, por fim nos bits MIN5:0 é armazenado o valor dos minutos em binário. Sendo assim seriam necessário dois bytes para o registro de um acesso, como a EEPROM do ATmega possui 1K byte de memória, seria possível a realização de 500 registros.

2.4 Interrupções

Interrupção é um processo pelo qual um dispositivo externo ou interno pode interromper a execução de uma determinada tarefa do microcontrolador e solicitar a execução de outra. A interrupção permite que um programa responda a determinados eventos no momento em que ocorrem. Dessa forma, interrupções são métodos de aplicação bastante razoável para este projeto.

Todas as interrupções no ATmega328P têm um endereço de atendimento fixo na memória e devem ser habilitadas ou desabilitadas agindo-se sobre um bit específico. Ainda há um bit que pode desabilitar ou habilitar todas as interrupções de uma vez só.

Figura 12 – Endereços e prioridade de interrupções

Vetor	End.	Fonte	Definição da Interrupção	Prioridade
1	0x00	RESET	Pino externo, Power-on Reset, Brown-out Reset e Watchdog Reset	
2	0x01	INT0	interrupção externa 0	
3	0x02	INT1	interrupção externa 1	
4	0x03	PCINT0	interrupção 0 por mudança de pino	
5	0x04	PCINT1	interrupção 1 por mudança de pino	
6	0x05	PCINT2	interrupção 2 por mudança de pino	
7	0x06	WDT	estouro do temporizador Watchdog	
8	0x07	TIMER2 COMPA	igualdade de comparação A do TC2	
9	0x08	TIMER2 COMPB	igualdade de comparação B do TC2	
10	0x09	TIMER2 OVF	estouro do TC2	
11	0x0A	TIMER1 CAPT	evento de captura do TC1	
12	0x0B	TIMER1 COMPA	igualdade de comparação A do TC1	
13	0x0C	TIMER1 COMPB	igualdade de comparação B do TC1	
14	0x0D	TIMER1 OVF	estouro do TC1	
15	0x0E	TIMER0 COMPA	igualdade de comparação A do TC0	
16	0x0F	TIMER0 COMPB	igualdade de comparação B do TC0	
17	0x10	TIMER0 OVF	estouro do TC0	
18	0x11	SPI, STC	transferência serial completa - SPI	
19	0x12	USART, RX	USART, recepção completa	
20	0x13	USART, UDRE	USART, limpeza do registrador de dados	
21	0x14	USART, TX	USART, transmissão completa	
22	0x15	ADC	conversão do ADC completa	
23	0x16	EE_RDY	EEPROM pronta	
24	0x17	ANA_COMP	comparador analógico	
25	0x18	TWI	interface serial TWI – I2C	
26	0x19	SPM_RDY	armazenagem na memória de programa pronta	

Fonte: AVR e Arduino: técnicas de projeto

Toda vez que uma interrupção ocorre, a execução do programa é interrompida: a CPU completa a instrução em andamento, carrega na pilha o endereço da próxima instrução que seria executada (endereço de retorno) e desvia para a posição de memória correspondente à interrupção. O código escrito no endereço da interrupção é executado até o programa encontrar o código RETI (Return from Interruption). Então, é carregado no PC o endereço de retorno armazenado na pilha e o programa volta a trabalhar a partir do ponto que parou antes da ocorrência da interrupção. Uma vez que a interrupção é atendida, todas as outras interrupções habilitadas serão desabilitadas.

Como as interrupções possuem endereços próximos, cada interrupção deve apresentar um comando de desvio para outra posição da memória (RJMP) onde o código para o seu tratamento possa ser escrito. Caso contrário, quando ocorrer a interrupção o código executado não corresponderá ao desejado. Só existe a necessidade do desvio se a interrupção for utilizada.

2.5 Timer

Os *timers* são necessários para fazer todas as temporizações necessárias no projeto, são elas: timeout que é tempo máximo durante o modo de programação, tempo de ativação, tempo da sirene acionada e os 10s ao apertar a tecla R. Para isso o *Timer1* de 16 bits presente no microcontrolador é utilizado no modo comparação gerando assim, uma

interrupção a cada segundo, onde nessas interrupções, incrementam os quatro contadores nela contido.

A inicialização do timer é feita configurando os registradores TCCR1A, TCCR1B e TIMSK1, Figura 14, Figura 13 e Figura 15 respectivamente. Assim como setando o contador de referência TCNT1 (17) iniciando-o em zero e OCR1A (16) conforme a equação:

$$Frequência = \frac{F_{CPU}}{prescaler * (OCR1A + 1)} = \frac{8 * 10^6 [Hz]}{256 * 31249} = 1Hz = 1s$$

Figura 13 – Registrador TCCR1B

TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Fonte: Atmega328P Datasheet

Figura 14 – Registrador TCCR1A

TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Fonte: Atmega328P Datasheet

Figura 15 – Registrador TIMSK1

TIMSK1 – Timer/Counter1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Fonte: Atmega328P Datasheet

Figura 16 – Registrador OCR1A

OCR1AH and OCR1AL – Output Compare Register 1 A

Bit	7	6	5	4	3	2	1	0	
(0x89)	OCR1A[15:8]								OCR1AH
(0x88)	OCR1A[7:0]								OCR1AL
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: Atmega328P Datasheet

Figura 17 – Registrador TCNT1

TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
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: Atmega328P Datasheet

2.5.1 Implementação Timer

No código, é possível observar como foi realizada a configuração nos registradores de inicialização, iniciando o TCCR1A todo em nulo, TCNT1 também iniciando em zero. Já no TCCR1B é habilitado o modo de comparação, onde será possível comparar o TCNT1 com o OCR1A e no registrador TIMSK1 determina que a comparação deverá ser feita com o registrador OCR1A.

Na interrupção, pode-se notar que os quatro contadores necessários para a implementação são incrementados com um, e quando há a necessidade de utilizar-los, eles são zerados e lidos novamente, em razão de que assim que o programa é iniciado eles entram em execução.

```
void timer1_init()
{
    TCCR1A = 0;
    TCNT1 = 0;

    //1 Hz (8000000/((31249+1)*256)) freq = FCPU/(prescaler)*(OCR1A+1)
    OCR1A = 3800; // 31249 Era pra ser isso....

    TCCR1B |= (1 << WGM12); //Habilita modo de comparação
    TCCR1B |= (1 << CS12); //Prescaler 256

    TIMSK1 |= (1 << OCIE1A); //Habilita Comparação com o OCR1A

    sei();
}

ISR (TIMER1_COMPA_vect)
{
    timeoutt++;
    tempoA++;
    tempoR++;
    tempoS++;
}
```


2.6 Conversor Analógico - Digital

A finalidade do conversor analógico digital, como o nome já diz, é converter o sinal analógico proveniente do teclado matricial em digital para que o microcontrolador seja capaz de realizar a leitura dos dados e fazer as instruções necessárias.

No microcontrolador em questão, ATmega328P, a conversão é feita internamente. Para realizar essa conversão dois registradores devem ser ajustados para a função seja realizada. São eles: **ADMUX** e **ADCSRA** ambos com 8 bits. Nas tabelas abaixo é possível entender como estão dispostos cada um dos bits no registrador.

Tabela 5 – ADMUX

	7	6	5	4	3	2	1	0
ADMUX	REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0

Tabela 6 – ADCSRA

	7	6	5	4	3	2	1	0
ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Nos bits 7:6 do **ADMUX** é possível definir qual será a tensão de referência (**REFS**) a ser escolhida, podem ser quatro modos, Tabela 7, para o projeto será definido um tensão de referência **AREF**, ou seja, **REFS** = “00” que irá desabilitar a tensão de referência interna de 1.1V e ler a **AREF**. Quanto ao quinto bit, **ADLAR**, se tiver nível lógico alto, irá alinhar o dado a esquerda nos registradores de dados **ADCR** e **ADCL**, consequentemente se zero, irá alinhar à direita. Portanto, será setado como ‘0’, para que os 8 bits menos significativos no registrador **ADCL** sejam lidos primeiro e em seguida os 2 bits mais significativos do **ADCH**. Para a definir qual porta o conversor irá fazer a leitura dos dados, os três bits menos significativos, **MUX**, devem estar setado conforme a tabela ??, em que ‘0000’ habilita a porta **ADC0** do microcontrolador e a “0111” a **ADC7**. Para calcular o valor do **ADC** utiliza-se a fórmula seguinte, em que o V_{ref} será definido como 5V no pino **AREF**, e V_{in} a tensão lida vinda dos *pushbuttons*, gerando a tabela ?? mencionada anteriormente.

$$ADC = \frac{(V_{in} \cdot 1024)}{V_{ref}} \quad (3)$$

Tabela 7 – Seleção da tensão de referência do ADC

REFS1	REFS0	Seleção da Tensão de Referência
0	0	AREF, tensão interna Vref desligada.
0	1	AVCC. Deve-se empregar um capacitor de 100 nF entre o pino AREF e o GND.
1	0	Reservado
1	1	Tensão interna de referência de 1,1 V. Deve-se empregar um capacitor de 100 nF entre o pino AREF e o GND.

Tabela 8 – Seleção do canal de entrada

MUX[3:0]	Entrada
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	Sensor interno de temperatura
1001-1101	Reservado
1110	1,1V (Tensão fixa para referência)
1111	GND

Quanto ao registrador de controle e status, **ADCSRA**, é definido algumas outras funcionalidades. No bit 7, **ADEN**, é onde a conversão é habilitada, em caso de nível baixo, a conversão **ADC** é desabilitada, então deverá ficar em nível alto. O **ADSC** define quando a conversão deve ser realizada ou não, para o conversão simples ele é alternado entre 0 e 1, ou seja, ele é 1 até que a seja concluída a conversão, e para o caso contínuo poderá ficar sentado sempre em nível alto. O *bit* 4 é uma interrupção que sinalizará quando uma conversão é concluída e o registrador de dados estiver atualizado, porém interrupção a ser utilizada será a interrupção global sinalizada pelo *bit* 3, **ADIE**. Por fim, os últimos 3 LSB's indicarão o divisor de *clock* para o **ADC**, sendo recomendo que esteja em uma faixa de 50kHz a 200kHz, como mostra a tabela abaixo.

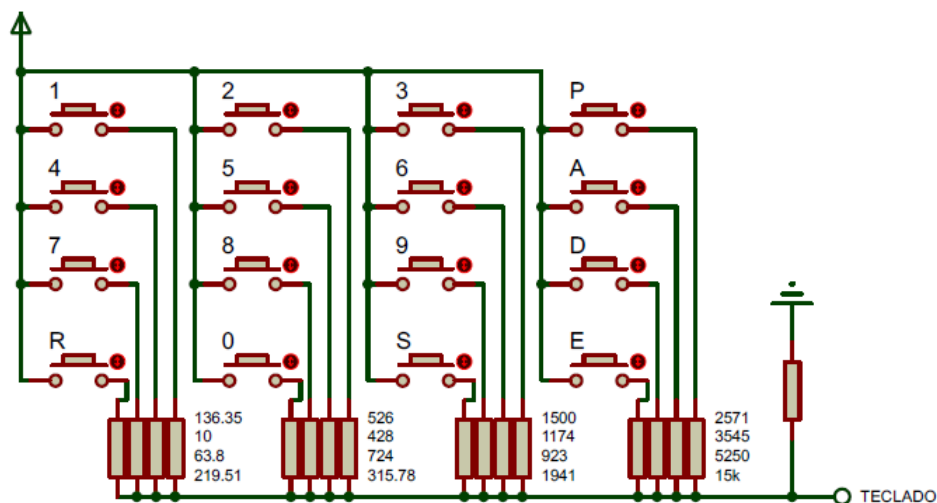
Tabela 9 – Divisor de clock

ADPS2	ADPS1	ADPS0	Fator de Divisão
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

2.7 Teclado 16 Teclas

A fim de atender as necessidades do projeto, é necessária a utilização de um teclado com 16 teclas. Devido a escassez de portas do microcontrolador foi decidido utilizar o teclado através de conversões analógicas-digitais, em que cada tecla representa um valor de ADC distinto, sendo assim foi possível associar cada um do valor do ADC obtido através da conversão a uma única tecla. Como pode-se notar no esquemático.

Figura 18 – Esquemático Teclado de 16 Teclas



Fonte: Proteus 8 Professional

O teclado possui apenas um pino de conexão analógica e toda sua lógica de programação pode ser vista na seção abaixo. A tensão de entrada no ADC0 deve variar de 0V a 5V espaçadas da melhor maneira possível, para que não haja sobreposição de nenhuma tecla, portanto aconselha-se que a tensão entre cada uma das teclas varie em torno de 0,3V. Através da fórmula de divisor de tensão é fácil calcular o valor do resistor a ser alocado.

2.7.1 Implementação Teclado

Para o projeto optou-se em utilizar um teclado baseado em conversões analógicas-digitais, onde se utiliza apenas uma porta do microcontrolador para realizar determinada tecla. Por exemplo, para cada tecla, corresponde a um valor de ADC, por conseguinte um determinado valor de ADC representa um inteiro correspondente a tecla, como pode ser visto na implementação abaixo.

```
int Teclado(){
    int adc_start();
    int conver = adc_start();

    if (conver<10)//1014
        return(17);                                     // Nenhuma Tecla
    else if (conver>1000 && conver<1024)//1014
        return(1);
    else if (conver>930 && conver<990)//962
        return(4);
    else if (conver>890 && conver<920)//901
        return(7);
    else if (conver>760 && conver<790)//778
        return(2);
    else if (conver>700 && conver<730)//717
        return(5);
    else if (conver>660 && conver<690)//671
        return(8);
    else if (conver>560 && conver<610)//594
        return(0);
    else if (conver>500 && conver<550)//533
        return(3);
    else if (conver>460 && conver<490)//471
        return(6);
    else if (conver>390 && conver<430)//410
        return(9);
    else if (conver>830 && conver<860)//840 Tecla R
        return(16);
    else if (conver>330 && conver<360)//348 Tecla S
        return(15);
    else if (conver>260 && conver<300)//287 Tecla P
        return(12);
    else if (conver>210 && conver<240)//225 Tecla A
        return(13);
    else if (conver>140 && conver<210)//164 Tecla D
        return(11);
    else if (conver>10 && conver<140)//64 Tecla E
        return(14);
}
```

Toda a inicialização do ADC pode ser visualizada no código seguinte, em que o foi setado como entrada a porta analógica ADC0, uma tensão de referência de 5V, um divisor de clock por 64 e toda a habilitação da conversão. Na função ADC_start é onde a conversão é feita, sendo chamada sempre que solicitado o valor de uma tecla na função Teclado.

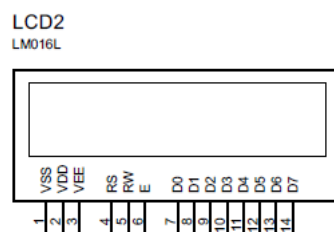
```
void ADC_init() {
    ADMUX &= ~ (1<<MUX3) | (1<<MUX2) | (1<<MUX1) | (1<<MUX0);
    ADMUX &= ~ (1<<REFS1) | (1<<REFS0);
    ADCSRA |= (1<<ADPS2) | (1<<ADPS1) | (1<<ADEN);
    ADCSRA &= ~ (1<<ADLAR) | (1<<ADPS0) | (1<<ADSC);
}

int adc_start() {
    ADCSRA |= (1<<ADSC);
    while(ADCSRA & (1<<ADSC));
    return ADCW;
}
```

2.8 Display LCD

LCD significa *Liquid Crystal Display*, é um dispositivo eletrônico usado para exibição de dados. Os LCDs são preferíveis a sete segmentos e LEDs, pois podem representar facilmente os dados em forma de alfabetos, caracteres, números ou animações. Os LCDs são muito fáceis de programar e tornam o trabalho bastante atraente e simples, vários tipos de LCDs estão disponíveis no mercado, como 16X2, 16X4, 20X2, 20X4, LCDs gráficos (128X64) etc. O LCD que será utilizado para este projeto é um LCD alfanumérico 16X2 LM016L, ele exibe 32 caracteres em duas linhas significa que em uma linha temos 16 caracteres.

Figura 19 – LCD 16x2 - LM016L



Fonte: Própria

2.8.1 Pinagem do LCD

Tabela 10 – Tabela de pinos do LCD

Pinos	I/O	Descrição
VSS	–	Terra
VDD	–	Alimentação
VEE	–	Controle para contraste da tela
RS	In	Registrador Seletor
RW	In	Ler/Escriver
E	In/Out	Habilitador
D0	In/Out	Dado de 8 bits LSB
D1	In/Out	Dado de 8 bits
D2	In/Out	Dado de 8 bits
D3	In/Out	Dado de 8 bits
D4	In/Out	Dado de 8 bits
D5	In/Out	Dado de 8 bits
D6	In/Out	Dado de 8 bits
D7	In/Out	Dado de 8 bits MSB

Enquanto o VDD é conectado a uma voltagem de +5V e o VSS ao terra, VEE é usado para controlar o contraste do LCD dependendo da voltagem.

Existem dois registradores importantes dentro do LCD, o registrador de comando, que vai enviar comandos, e o registrador de dados, na qual é enviado dados para serem mostrados no display. Sendo $RS = 1$ para selecionar o registrador de dados e $RS = 0$ para selecionar o registrador de comando.

A entrada RW permite a leitura ou escrita da informação do LCD dependendo do bit de entrada, $RW = 1$ para leitura e $RW = 0$ para escrita. Como só vai precisar escrever no LCD neste projeto, pode ser deixado a entrada RW conectado ao terra, economizando uma porta do microcontrolador.

O pino E é usado para travar as informações apresentadas aos pinos de dados. Quando os dados são enviados para os pinos, um pulso de $HIGH$ para LOW deve ser aplicado à este pino para que retenha os dados presente nos pinos de dados.

Os pinos de dados de 8 bits, D0-D7 são usados para mandar informações para o LCD ou ler os conteúdos dos registradores internos do mesmo.

Para programar o LCD é possível utilizar 8 portas do microcontrolador, que é mais fácil e prático, porém utiliza mais portas, e utilizando 4 portas, que é mais complexo, porém utiliza menos portas. Como a quantidade de portas utilizadas serão altas, será adotado a utilização de 4 portas de dados, junto com as duas entradas RS e E , somando um total de 6 portas que serão utilizadas para o LCD.

Para forçar o LCD no modo de 4 pinos, é necessário inicializar o LCD com os

comando 32, 33 e 28 em hexadecimal, nas quais configura o *display* para o modo de 4 bits, utilizando assim os dados mais significativos, D4-D7.

2.8.2 Códigos de comando do LCD

Tabela 11 – Códigos de comandos do LCD

Códigos(HEX)	Comando para o LCD
01	Limpa a tela do display
02	Retorna para <i>Home</i>
04	Decremento do cursor(muda cursor para esquerda)
06	Incremento do cursor(muda cursor para direita)
05	Muda display para direita
07	Muda display para esquerda
08	Display off/Cursor off
0A	Display off/Cursor on
0C	Display on/Cursor off
0E	Display on/Cursor piscando
0F	Display on/Cursor piscando
10	Muda a posição do cursor para esquerda
14	Muda a posição do cursor para direita
18	Muda todo o display para a esquerda
1C	Muda todo o display para a direita
80	Força o cursor para o início(1ª linha)
C0	Força o cursor para o início(2ª linha)
28	2 linhas e matriz 5x7(D4-D7, 4 bit)
38	2 linhas e matriz 5x7(D0-D7, 8 bit)

Para enviar comandos para o display, deve-se fazer os pinos *RS* e *RW* = 0 e colocar o número de comando nos pinos *D0-D7*, em seguida mandar um pulso de alto para baixo para o pino *E*. Depois de cada comando, deve-se esperar cerca de 100us para deixar o módulo do LCD realizar o comando, com exceção dos comandos *CLEAR LCD* e do *Return Home*, depois dos comando 0x01 e 0x02, deve-se esperar 2ms.

Para enviar dados para o display, deve-se setar os pinos *RW* em 1 e *RS* em 0, depois mandar os dados para os pinos *D0-D7* e enviar um pulso de alto para baixo para o pino *E*. Depois de enviar os dados, deve-se esperar cerca de 100us para o módulo do LCD escrever os dados na tela.

Abaixo está a descrição dos endereços de cada posição do display.

Tabela 12 – Endereço das posições do LCD

Linhas	Área de endereços(HEX)					
Linha 1:	80	81	82	83	até	8F
Linha 2:	C0	C1	C2	C3	até	CF

2.9 SPI

Serial Peripheral Interface(SPI) consiste em dois registradores de deslocamento de 8 bits, um para o mestre e o outro para o escravo, ocorrendo assim uma troca de dados entre os dois registradores de forma serial, ou seja, bit a bit, também há um gerador de *clock* no lado mestre que gera *clock* para os registradores. Para este projeto será utilizado a comunicação SPI para obtenção da hora, minuto e dia do CI RTC, que será explicado em um tópico mais a frente.

Foi decidido utilizar o escravo MAX6902, que será detalhado mais a frente, e o SPI do Atmega328P como mestre, significando que o microcontrolador gera o *clock serial* que alimenta os periféricos. Utilizando as portas normais do AVR para o protocolo SPI.

Tabela 13 – Portas do protocolo SPI.

Portas	Função	Significado	I/O
B2	SS1	Slave Select 1	O
B3	MOSI	Master Out Slave In	O
B4	MISO	Master In Slave Out	I
B5	SCK	Serial Clock	O

O protocolo SPI utiliza o *clock serial* para sincronizar a transferência de informações de 1 bit de cada vez, onde o bit mais significativo (MSB) vai primeiro. Durante a transferência, o pino SS1 deve ser setado em *LOW*, pois tem uma lógica invertida, dependendo do escravo que será utilizado, no caso, SS1 para o MAX6902(RTC), e ao final da transferência, deve ser setado em *HIGH*. As informações (endereço e dados) são transferidas entre o microcontrolador e o periférico em grupos de 8 bits, onde o *byte* de endereço é seguido imediatamente pelo *byte* de dados. Para distinguir entre o operações de leitura e gravação, o *bit* D7 do *byte* de endereço é sempre 1 para *write*, enquanto para *read*, o bit D7 é *LOW*.

No AVR, há 3 registradores associados com o SPI, eles são SPSR (*SPI Status Register*), SPCR (*SPI Control Register*) e SPDR (*SPI Data Register*). Abaixo será detalhado cada bit desses registradores.

Tabela 14 – SPSR (SPI Status Register)

SPIF	WCOL	-	-	-	-	-	SPI2X
------	------	---	---	---	---	---	-------

Bit 7 - SPIF (Sinalizador de interrupção SPI): No modo mestre, este *bit* é definido em duas situações: quando uma transferência serial é concluída, ou quando o pino SS é uma entrada e é baixo por um dispositivo externo. Configurando a bandeira

SPIF para um causará uma interrupção se SPIE em SPCR estiver definido e interrupções globais estiverem habilitadas.

Bit 6 - WCOL (*Write Colision Flag*): O bit WCOL é definido se você gravar no SPDR durante uma transferência de dados.

Bit 0 - SPI2X (*Double SPI Speed*): Quando o SPI está no modo master, definir este bit como um, dobra a rapidez SPI.

Observe que tanto o bit WCOL quanto o bit SPIF são apagados quando você lê o SPI e então acessa o SPDR. Alternativamente, o bit SPIF é limpo pelo hardware ao executar o manipulador de interrupção correspondente.

Tabela 15 – SPCR (SPI Control Register)

SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
------	-----	------	------	------	------	------	------

Bit 7 - SPIE (*SPI Interrupt Enable*): Definir este bit como um, habilita a interrupção SPI.

Bit 6 - SPE (*SPI Enable*): Definir este bit como um, habilita o SPI.

Bit 5 - DORD (*Data Order*): Este bit permite que você escolha transmitir MSB e LSB ou vice-versa. O LSB é transmitido primeiro se DORD for um, caso contrário, o MSB é transmitido primeiro.

Bit 4 - MSTR (*Master/Slave Select*): Se você quiser trabalhar no modo mestre, defina este bit como um, caso contrário, o modo escravo é selecionado. Observe que se o pino SS estiver configurado como uma entrada e for direcionado para baixo enquanto MSTR é definido, MSTR será limpo e SPIF será definido.

Bit 3 - CPOL (*Clock Polarity*): Este bit define o valor base do clock quando está ocioso. No CPOL = 0, o valor base do clock é zero enquanto em CPOL = 1 o valor base do clock é um.

Bit 2 - CPHA (*Clock Phase*): CPHA = 0 significa amostra na borda do relógio principal (primeiro), enquanto CPHA = 1 significa amostra no relógio posterior (segundos).

Bits 1, 0 - SPR1, SPRO: SPI *Clock Rate Select* 1 e 0: Esses dois bits controlam a taxa SCK do dispositivo no modo mestre.

Tabela 16 – SPDR (SPI Data Register)

MSB	-	-	-	-	-	-	LSB
-----	---	---	---	---	---	---	-----

O SPI *Data Register* (SPDR) é um registrador de leitura / gravação. Para escrever no registrador de deslocamento SPI, os dados devem ser gravados no SPDR. Para ler o

registrador de deslocamento SPI, você deve ler do SPDR. O *write* no registro SPDR inicia a transmissão de dados. É aconselhado não escrever no SPDR antes que o último *byte* seja transmitido completamente, do contrário, ocorrerá uma colisão. Pode-se ler os dados recebidos antes de outro *byte* de dados é recebido completamente.

2.9.1 RTC MAX6902

O MAX6902 é um circuito integrado RTC(*Real Time Clock*) compatível com SPI que contém um relógio/calendário em tempo real, que provê segundos, minutos, horas, dias, data, mês, ano e século, além de um alarme programável. O relógio opera tanto em formato 24h quanto 12h com indicador AM/PM. O CI opera com uma voltagem de 2V até 5V, com 1MHz e 4MHz para a frequência de *clock*, respectivamente.

Os dados de tempo e calendário são guardados em registradores em formato BCD, como mostrado nas imagens 20 e 21, e a função de alarme está incluída para tempo programado de horários ou intervalos definidos pelo usuário.

Figura 20 – Registradores do MAX6902

		REGISTER ADDRESS								REGISTER DEFINITION									
FUNCTION		A7	A6	A5	A4	A3	A2	A1	A0	VALUE	D7	D6	D5	D4	D3	D2	D1	D0	
CLOCK																			
SECONDS	RD	0	0	0	0	0	0	0	1	00-59	0		10 SEC			1 SEC			
	/W									*POR STATE	0	0	0	0	0	0	0	0	
MINUTES	RD	0	0	0	0	0	0	1	1	00-59			ALM OUT	10 MIN		1 MIN			
	/W									*POR STATE	0	0	0	0	0	0	0	0	
HOURS	RD	0	0	0	0	0	1	0	1	00-23	12/24		10 HR	10 HR		1 HR			
	/W									01-12	1/0	0	A/P	O/1					
										*POR STATE	0	0	0	0	0	0	0	0	
DATE	RD	0	0	0	0	1	1	1		01-28/29			10 DATE			1 DATE			
	/W									01-30									
										0-31									
										*POR STATE	0	0	0	0	0	0	0	0	1
MONTH	RD	0	0	0	1	0	0	1		01-12	0	0	0	10M		1 MONTH			
	/W									*POR STATE	0	0	0	0	0	0	0	0	1
DAY	RD	0	0	0	1	0	1	1		01-07	0	0	0	0	0	WEEK DAY			
	/W									*POR STATE	0	0	0	0	0	0	0	0	1
YEAR	RD	0	0	0	1	1	0	1		00-99			10 YEAR			1 YEAR			
	/W									*POR STATE	0	1	1	1	0	0	0	0	
CONTROL	RD	0	0	0	1	1	1	1					WP	0	0	0	0	0	
	/W									*POR STATE	0	0	0	0	0	0	0	0	
CENTURY	RD	0	0	1	0	0	1	1		00-99			1000 YEAR			100 YEAR			
	/W									*POR STATE	0	0	0	1	1	0	0	1	
Note: *POR STATE defines power-on reset state of register contents.																			

Fonte: MAXIM Datasheet

Figura 21 – Registradores do MAX6902

REGISTER ADDRESS										REGISTER DEFINITION								
FUNCTION	A7	A6	A5	A4	A3	A2	A1	A0		VALUE	D7	D6	D5	D4	D3	D2	D1	D0
ALARM CONFIG	RD	0	0	1	0	1	0	1			0	YEAR	DAY	MONTH	DATE	HOUR	MINUTE	SECOND
	/W									*POR STATE	0	0	0	0	0	0	0	0
RESERVED Do not write to this location.	RD	0	0	1	0	1	1	1			0	0	0	0	0	1	1	1
	/W									*POR STATE	0	0	0	0	0	1	1	1
ALARM THRESHOLDS																		
SECONDS	RD	0	0	1	1	0	0	1		00-59	0	10 SEC			1 SEC			
	/W									*POR STATE	0	1	1	1	1	1	1	1
MINUTES	RD	0	0	1	1	0	1	1		00-59	0	10 MIN			1 MIN			
	/W									*POR STATE	0	1	1	1	1	1	1	1
HOURS	RD	0	0	1	1	1	0	1		00-23 01-12	12/24 1/0	0	10 HR A/P Q/1	10 HR	1 HR			
	/W									*POR STATE	1	0	1	1	1	1	1	1
DATE	RD	0	0	1	1	1	1	1		01-28/29 01-30 01-31	0	0	10 DATE			1 DATE		
	/W									*POR STATE	0	0	1	1	1	1	1	1
MONTH	RD	0	1	0	0	0	0	1		01-12	0	0	0	10M	1 MONTH			
	/W									*POR STATE	0	0	0	1	1	1	1	1
DAY	RD	0	1	0	0	0	1	1		01-07	0	0	0	0	0	WEEK DAY		
	/W									*POR STATE	0	0	0	0	0	1	1	1
YEAR	RD	0	1	0	0	1	0	1		00-99	10 YEAR			1 YEAR				
	/W									*POR STATE	1	1	1	1	1	1	1	1
CLOCK BURST	RD	0	1	1	1	1	1	1										
	/W																	

Fonte: MAXIM Datasheet

Figura 22 – Endereços dos registradores e descrição

WRITE (HEX)	READ (HEX)	DESCRIPTION	POR CONTENTS (HEX)
01	81	Seconds	00
03	83	Minutes	00
05	85	Hours	00
07	87	Date	01
09	89	Month	01
0B	8B	Day	01
0D	8D	Year	70
0F	8F	Control	00
13	93	Century	19
15	95	Alarm Configuration	00
17	97	Reserved	07
19	99	Seconds Alarm Threshold	7F
1B	9B	Minutes Alarm Threshold	7F
1D	9D	Hours Alarm Threshold	BF
1F	9F	Date Alarm Threshold	3F
21	A1	Month Alarm Threshold	1F
23	A3	Day Alarm Threshold	07
25	A5	Year Alarm Threshold	FF
3F	BF	Clock Burst	Not applicable

Fonte: MAXIM Datasheet

Nesse projeto serão utilizadas apenas as informações contidas em alguns desses registradores, são eles, os de dias, horas, minutos e seus respectivos registradores de alarme. Para isso, será usado o *byte* de endereço/comando que especifica quais registradores serão usados, tanto para escrita quanto para leitura.

O *bit* 7 do registrador de horas seleciona o modo 12h ou 24h. Quando *HIGH*, o modo de 12 horas é selecionado. No modo 12h, o *bit* 5 é o *bit* AM / PM, lógica *HIGH* para PM. No modo 24 horas, o bit 5 é o segundo bit de 10 horas, lógica *HIGH* para as horas 20 a 23.

2.9.2 Implementação SPI

Abaixo segue o código referente ao SPI. Os dados coletados serão de horas, data e dia coletados através do RTC MAX6902 citado na seção acima. Toda a configuração já foi citada na seção referente ao SPI.

```
void SPI_Init()
{
    DDRB |= (1<<MOSI) | (1<<SCK) | (1<<SS);           //Saída
    DDRB &= ~(1<<MISO);                                //Entrada
    SPCR = (1<<SPE) | (1<<MSTR) | (0<<SPR0) | (0<<SPR1); //Ativa SPI, Mestre, Clock/4
    SPSR &= ~(1<<SPI2X);                                //Desabilita SPI2X
}
```

No código acima é feita a inicialização do SPI, onde são habilitadas as portas de MOSI, SCK, SS como saída e MISO como entrada, assim como habilita a transferência SPI, seta como mestre o microcontrolador e define o *prescaler* de 4, ou seja, trabalhando em uma frequência de 1MHz.

A transferência e recepção dos dados são feitos em uma transferência de dois bytes, sendo assim, primeiro é enviado um byte ao RTC referente ao registrador das horas, minutos ou dia e em seguida outro byte com um valor qualquer, para que seja possível receber a informação do RTC, como é mostrado na implementação abaixo. Na transferência de dados é habilitado o escravo através do SS, e após a recepção ele é desabilitado.

```
void SPI_dado(unsigned data) //Envia dados
{
    PORTB &= ~(1<<SS); //Habilita transferencia
    SPDR = data;        //dado
    while(!(SPSR & (1 << SPIF)));
}

int SPI_ler_horas() //Ler horas
{

```

```

        SPI_dado(0x85);
        SPDR = 0xFF;
        while(!(SPSR & (1<<SPIF)));
        PORTB |= (1<<SS);
        return (SPDR);
    }
    int SPI_ler_minutos()
    {
        SPI_dado(0x83);
        SPDR = 0xFF;
        while(!(SPSR & (1<<SPIF)));
        PORTB |= (1<<SS);
        return (SPDR);
    }
    int SPI_ler_dia()
    {
        SPI_dado(0x8B);
        SPDR = 0xFF;
        while(!(SPSR & (1<<SPIF)));
        PORTB |= (1<<SS);
        return (SPDR);
    }
}

```

A utilização do SPI é chamada apenas quando a senha nos estados de ativar e desativar é satisfeita, em seguida escreve as recepções na EEPROM, conforme mostrado abaixo.

```

horas = SPI_ler_horas();
minutos = SPI_ler_minutos();
dia = SPI_ler_dia();

EEPROM_escrita(endereco, horas);
endereco=endereco+1;
EEPROM_escrita(endereco, minutos);
endereco=endereco+1;
EEPROM_escrita(endereco, dia);
endereco=endereco+1;
EEPROM_escrita(endereco, i);
endereco=endereco+1;
EEPROM_escrita(endereco, modo);
endereco=endereco+1;

```

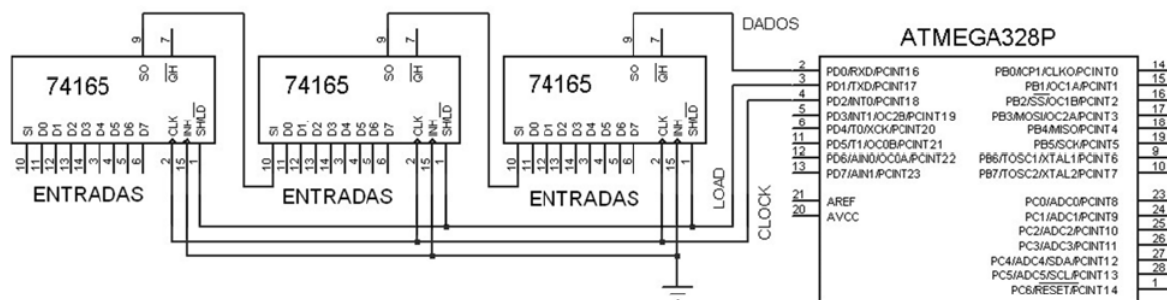
2.10 Expansão de entradas e saídas

Devido ao alto número de pinagens utilizadas neste projeto, faz-se necessária a expansão de dispositivos I/O. Para isso, algumas técnicas podem ser empregadas:

2.10.1 Aumento do número de entradas - Conversão Paralelo-Serial

A conversão de dados paralelo para serial serve para aumentar o número de entradas do sistema. Um CI recomendado para o processo é o 74165. As informações das entradas do CI são transferidas para registradores internos após um sinal de load. Após isso, serão transferidos para o ATmega de acordo com os pulsos de clock gerados. Com um CI 74615 conectado, são necessários 8 pulsos de clock para a obtenção dos dados da sua entrada.

Figura 23 – Conversão paralelo-serial em CI 74165



Fonte: Técnicas de AVR e Projeto

2.10.1.1 Implementação dos sensores

A ideia da utilização do expensor de entradas foi realizada para reduzir as 8 entradas dos sensores para uma entrada serial de 8 bits. Para isso foi realizada uma função que utiliza da lógica do CI 74165 para comunicar com o ATmega.

```
unsigned char sensores(){
    unsigned char sens = 0;
    unsigned char contador_serial = 0;

    PORTD &= ~(1 << CLK_S);
    PORTB &= ~(1 << LOAD);
    PORTB |= (1 << LOAD);
    while(contador_serial<8){
        if(PIND & (1 << SERIAL)){
            sens |= 1 << contador_serial;
            PORTD |= (1 << CLK_S);
        }
        else if(!(PIND & (1 << SERIAL))){
            sens &= ~(1 << contador_serial);
        }
        contador_serial++;
    }
    return sens;
}
```

```

        PORTD |= (1 << CLK_S);
    }
    PORTD &= ~(1 << CLK_S);
    contador_serial = contador_serial + 1;
}
return sens;
}

```

A função acima inicializa o clock do CI 74165 e realiza logo em seguida um load no circuito integrado. Este LOAD carrega o valor dos sensores paralelamente no CI. Com isso o sistema entra em um while que verifica se o bit a ser deslocado é 1 ou 0. De acordo com o valor da saída do CI, o sistema carrega o mesmo bit deslocado na variável que indicará o valor em bit do sensores para o sistema. Com essa interação, a função desloca o próximo bit no CI, através do circuito integrado. Após oito interações, o valor dos sensores é retornado pela função.

2.11 Clock

Devido a taxa de erro, que é necessária ser verificada para evitar perdas de dados no receptor da comunicação UART, o clock escolhido para este projeto ficou em 2 Mhz. Visto que para a comunicação assíncrona utilizando-se 8 bits de dados e nenhum de paridade a resolução máxima de erro segundo a imagem da figura 54 abaixo, fica em torno de mais ou menos 2.0%, sendo assim ficou escolhido um clock de 2 Mhz que gera um erro de 0.2% e caso o o grupo da implementação deseje usar bits de paridade, ainda sim esse clock obedecerá o erro máximo recomendado. A figura 25 na terceira linha no modo assíncrono mostra o erro pra uma taxa de transmissão de 9600 com um clock de 2 Mhz.

Figura 24 – Erro máximo recomendado da taxa de transmissão do receptor para o modo de velocidade normal

D # (Data+Parity Bit)	R _{slow} (%)	R _{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	93.20	106.67	+6.67/-6.8	± 3.0
6	94.12	105.79	+5.79/-5.88	± 2.5
7	94.81	105.11	+5.11/-5.19	± 2.0
8	95.36	104.58	+4.58/-4.54	± 2.0
9	95.81	104.14	+4.14/-4.19	± 1.5
10	96.17	103.78	+3.78/-3.83	± 1.5

Fonte: Datasheet Atmega328p

Figura 25 – Exemplos de configurações UBRRn para frequências de oscilador comumente usadas

Baud Rate (bps)	$f_{osc} = 1.0000\text{MHz}$				$f_{osc} = 1.8432\text{MHz}$				$f_{osc} = 2.0000\text{MHz}$			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	–	–	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	–	–	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	–	–	–	–	–	–	0	0.0%	–	–	–	–
250k	–	–	–	–	–	–	–	–	–	–	0	0.0%
Max. ⁽¹⁾	62.5kbps		125kbps		115.2kbps		230.4kbps		125kbps		250kbps	

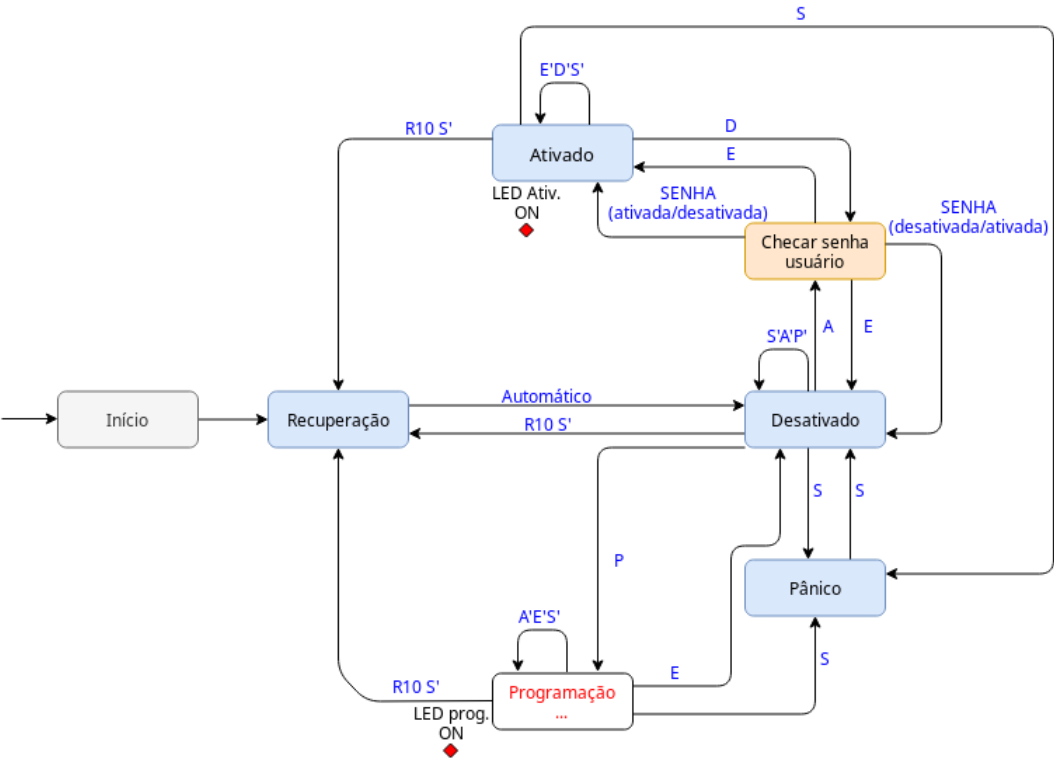
Note: 1. UBRRn = 0, Error = 0.0%

Fonte: Datasheet Atmega328p

2.12 Máquina de estados finitos

A máquina de estados finitos foi dividida em duas partes, a parte da máquina dos modos de operação e a parte da máquina referente as operações de programação. Podemos ver a máquina de estados finitos dos modos de operação abaixo na figura 26

Figura 26 – Máquina de estados finitos de modos de operação

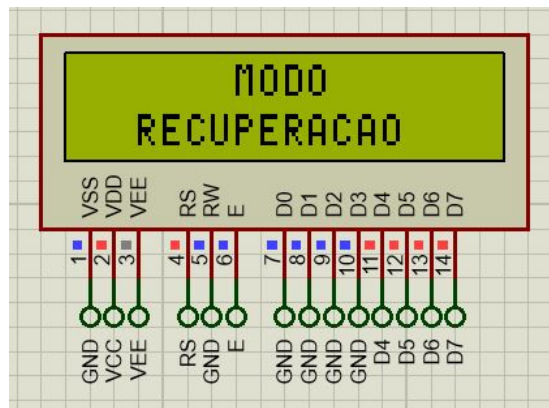


Fonte: Autoria Própria

A máquina dos modos de operação possuirá os seguintes estados

- **Recuperação:** Ao iniciar a primeira vez esse será o primeiro estado acionado. Nesse estado as configurações definidas anteriormente são apagadas e são ativadas as configurações de fábrica. Que são a senha mestre no valor de 1234, o temporizador de ativação em 0, o temporizador de *timeout* em 99 segundos, o temporizador da sirene em 0 segundos e zera as configurações de sensores e zonas. O usuário pode acessar esse modo de qualquer outro modo pressionando o botão R por 10 segundos.

Figura 27 – Tela do modo recuperação



Fonte: Autoria Própria

- **Desativado:** Esse estado é carregado automaticamente após o estado de recuperação, porém também pode ser acessado pelo estado de ativado através do comando D, seguido de senha e da tecla E, caso a senha esteja incorreta o estado retornará para o estado de ativado, se não, ele prossegue para o estado desativado e desliga todos os leds e o alarme caso algum esteja ligado. Nesse estado mesmo que algum dos sensores seja acionado não ocorrerá acionamento da sirene. Esse estado é acessado também pelo estado de pânico pela tecla S e pelo estado de programação por meio da tecla E ou R.

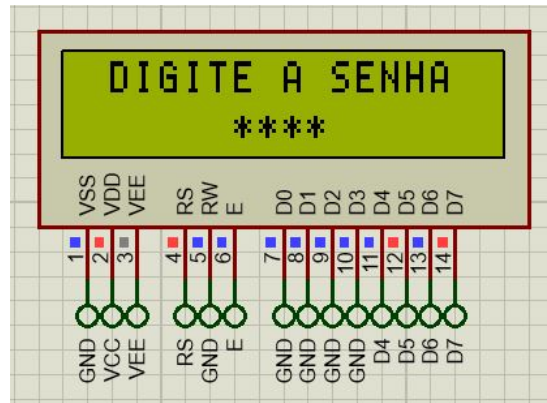
Figura 28 – Tela do modo desativado



Fonte: Autoria Própria

- **Checar senha usuário:** Nesse estado é verificado a senha do usuário para poder acessar o modo ativado/desativado, nesse caso a senha pode ser de qualquer um dos usuários. Um fator importante é que caso o temporizador de ativação seja configurado com um tempo acima de 0, esse estado retardará para entrar no modo ativado, permitindo que o operador possa sair do local que será monitorado sem acionar os sensores.

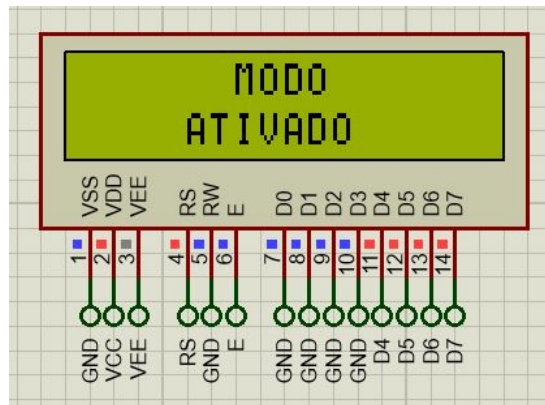
Figura 29 – Tela do modo checar senha usuário



Fonte: Autoria Própria

- **Ativado :** É Neste estado que as configurações definidas pelo usuário ou pela configuração de fábrica se encontram ativas. Para ativar o usuário precisa acessar o estado desativado e pressionar A, digitar a senha do usuário e pressionar E. O sistema irá para o estado checar senha usuário e caso a senha for digitada incorretamente o sistema retornará para o estado de desativado. No modo ativado, caso um sinal seja detectado por um dos sensores habilitado e associado a uma zona, irá disparar a sirene que tocará permanentemente caso o temporizador de sirene seja 0, caso contrário ela ficará ativada até atingir o tempo marcado por esse temporizador, voltando para o estado desativado em seguida. Com relação aos leds de indicação de movimento referentes as zonas, estes serão ativados baseado na associação dos sensores e zonas feita pelo operador, a ativação dos leds se dá de forma cronológica, sendo um por vez, leds de zonas diferentes podem ser ativados e permanecerão ativos até o sistema voltar para o estado desativado.

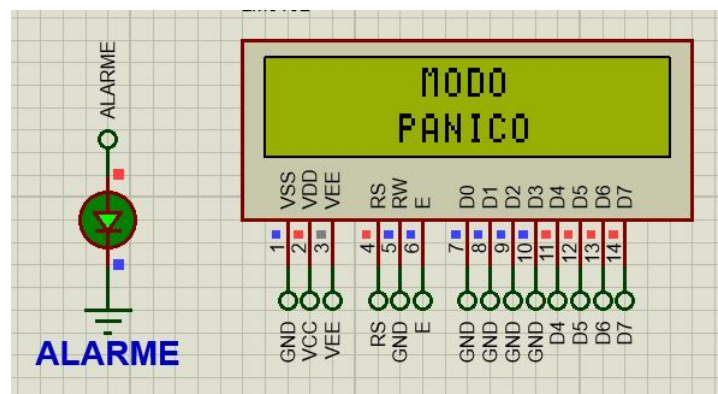
Figura 30 – Tela do modo ativado



Fonte: Autoria Própria

- **Pânico :** O usuário, poderá acessar esse modo através de qualquer modo apertando S. Ao entrar nesse modo a sirene será ativada instantaneamente mesmo que nenhum dos sensores sejam ativados, para retornar para o modo de desativado basta pressionar S. Quando o modo Pânico é ativado, a sirene deve continuar ativada até que a tecla S seja pressionada, para isso, no modo pânico o timer referente ao tempo de atividade da sirene não é levado em consideração, fazendo com que a mesma permaneça ativada até que o operador a desligue manualmente.

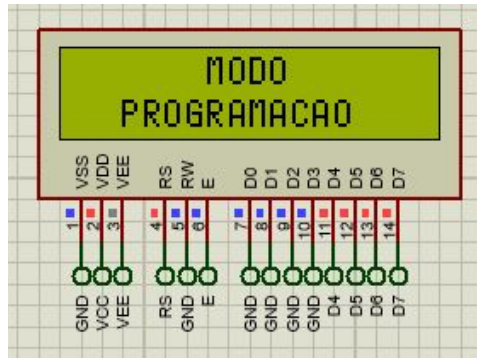
Figura 31 – Tela do modo pânico



Fonte: Autoria Própria

- **Programação :** No estado de programação o usuário define configurações como o tempo de acionamento da sirene, o tempo de *timeout*, Habilitação e Desabilitação de sensores, associação de sensores com zonas, habilitação de zonas e mudança de senhas. O usuário pode acessar o modo de programação através do modo desativado pressionando P seguido da senha mestra e pressionando E, caso a senha seja correta, ele poderá fazer as configurações desejadas enquanto estiver dentro do intervalo do temporizador de *timeout*, quando esse intervalo acaba o sistema volta automaticamente para o estado desativado.

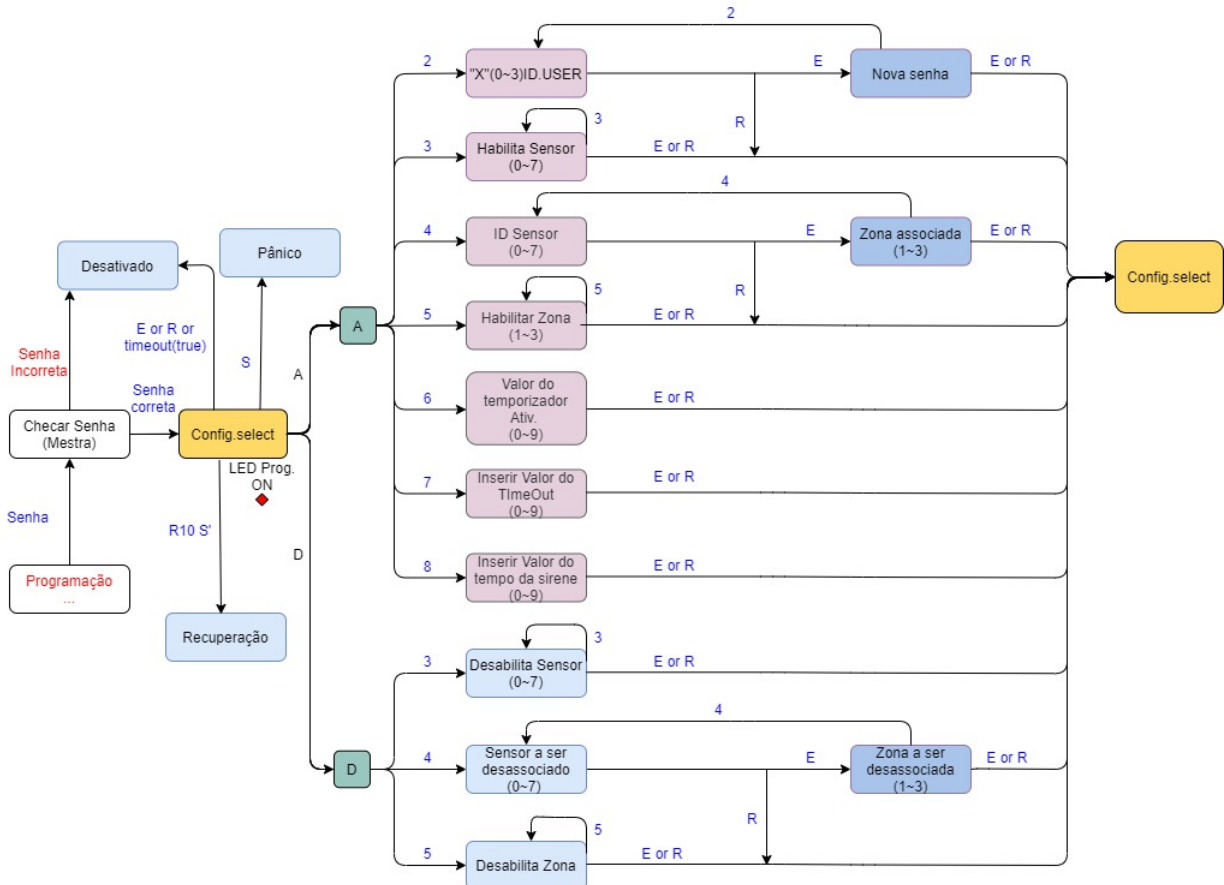
Figura 32 – Tela do modo programação



Fonte: Autoria Própria

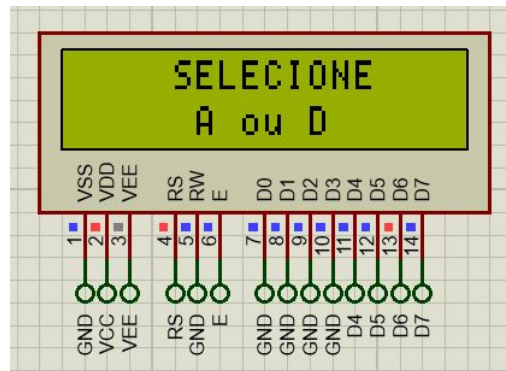
Já na etapa de programação nossa máquina será um pouco mais extensa. Após o usuário seguir os passos informado no item anterior referente ao estado de programação, ele obtem acesso a várias funções do alarme. Nesse modo, teremos a opção do usuário digitar A ou D. A qualquer momento o usuário pode apertar R para cancelar a operação, S para ir para o modo de pânico ou manter R pressionado para ir para o modo de recuperação, como podemos observar na figura 33 abaixo.

Figura 33 – Máquina de estados finitos do modo de programação



Fonte: Autoria Própria

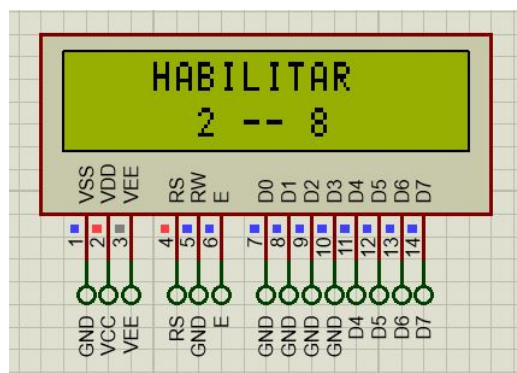
Figura 34 – Tela do estado de config select



Fonte: Autoria Própria

Se o usuário optar por escolher A após a etapa de checagem de senha, ele terá as seguintes opções.

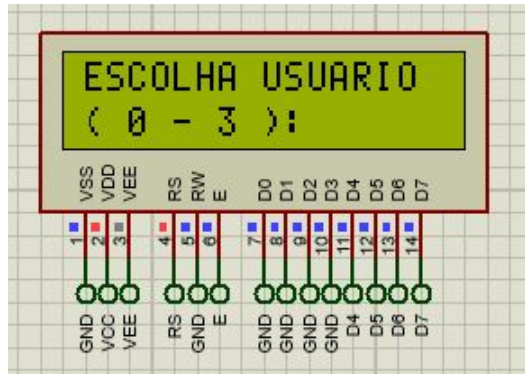
Figura 35 – Tela do estado de habilitação



Fonte: Autoria Própria

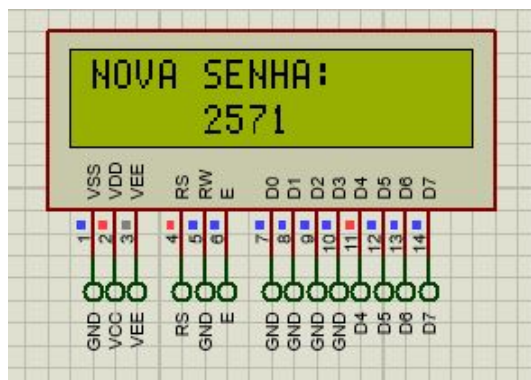
- **Trocar a senha do usuário :** Escolhendo o dígito 2 , o operador poderá alterar a senha de algum dos usuários, sendo necessário informar a identificação do usuário em questão e a nova senha, caso o usuário deseje cadastrar ou alterar a senha de outro usuário, ele pode apertar 2, após digitar a nova senha do usuário escolhido atualmente. Após selecionar o usuário e adicionar um valor para a nova senha, o usuário também tem a opção de confirmar apertando E, onde irá retornar para o estado select.config salvando a nova senha e a opção de cancelar, apertando R para voltar para o estado select.config sem salvar a nova senha;

Figura 36 – Tela do estado de escolha do usuário



Fonte: Autoria Própria

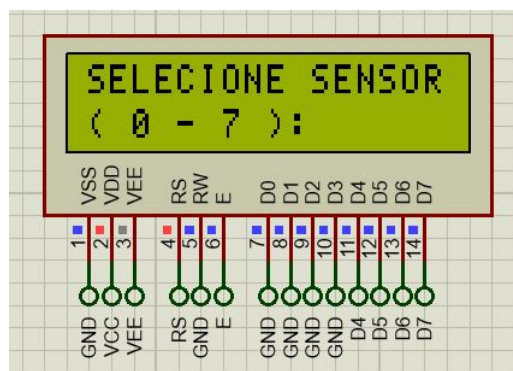
Figura 37 – Tela do estado da nova senha



Fonte: Autoria Própria

- **Habilitar Sensor:** Na escolha do dígito 3, o operador entra no modo de desabilitação de sensores, podendo habilitar um dos oito sensores, e caso ele deseje poderá habilitar outros pressionando a tecla 3 após a escolha do sensor desejado. Nesse caso ele também tem a opção de pressionar E para confirmar ou R para cancelar e voltar para o estado select.config em ambos os casos;

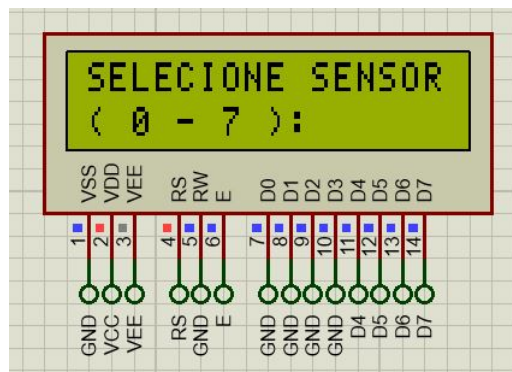
Figura 38 – Tela do estado de habilitação do sensor



Fonte: Autoria Própria

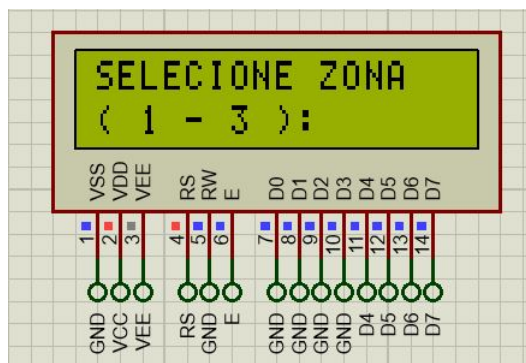
- **Associar o sensor a uma zona:** Na escolha do dígito 4, o operador entra no modo de associação de sensores a zonas, podendo associar um ou mais sensores a uma zona específica. Primeiro ele seleciona o sensor que deseja associar e pressiona E, em seguida escolhe a zona, após isso ele poderá apertar 4 salvando a configuração atual e voltando para a opção de escolher um outro sensor para fazer a associação. Nesse caso ele também tem a opção de pressionar E para confirmar ou R para cancelar e voltar para o estado select.config em ambos os casos;

Figura 39 – Tela do estado de selecionar sensor para associar a uma zona



Fonte: Autoria Própria

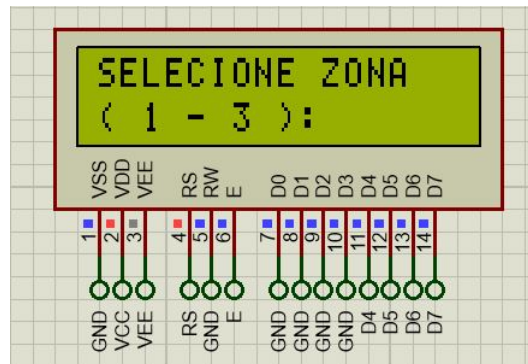
Figura 40 – Tela do estado de selecionar zona a qual sensor será associado



Fonte: Autoria Própria

- **Habilitar Zona:** Na escolha do dígito 5, o operador tem a acesso ao modo de desabilitação de zonas, podendo habilitar uma das três zonas, e caso ele deseje poderá habilitar outras pressionando a tecla 5 após a escolha da zona desejada. Nesse caso ele também tem a opção de pressionar E para confirmar ou R para cancelar e voltar para o estado select.config em ambos os casos;

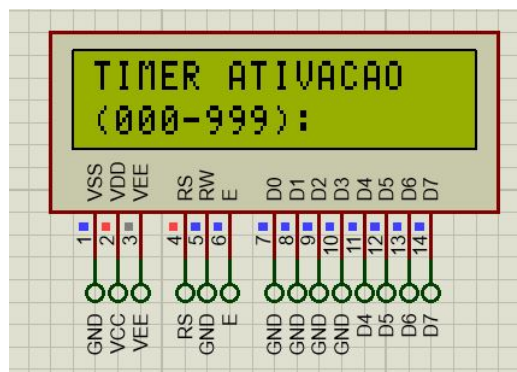
Figura 41 – Tela do estado de habilitar zona



Fonte: Autoria Própria

- **Ajustar valor do temporizador de ativação:** escolhendo o dígito 6, o operador entra no modo de ajuste do temporizador de ativação, este timer pode ser configurado por valores entre 000 e 999. Nesse caso ele também tem a opção de pressionar E para confirmar ou R para cancelar e voltar para o estado select.config em ambos os casos;

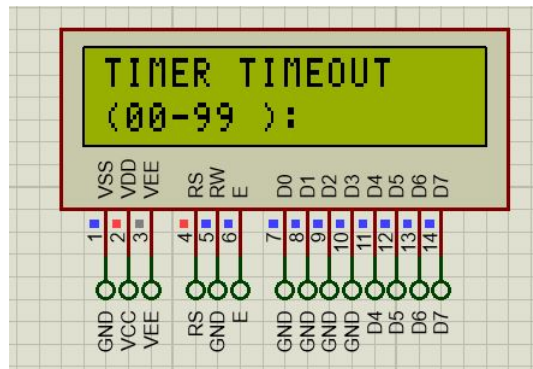
Figura 42 – Tela do estado do temporizador de ativação



Fonte: Autoria Própria

- **Ajustar valor do *timeout*:** Na escolha do dígito 7, o operador tem acesso ao modo de ajuste do temporizador do timeout, este ajuste recebe valores entre 00 e 99. Nesse caso ele também tem a opção de pressionar E para confirmar ou R para cancelar e voltar para o estado select.config em ambos os casos;

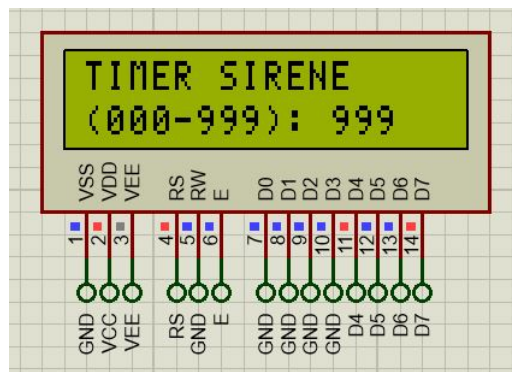
Figura 43 – Tela do estado do temporizador de timeout



Fonte: Autoria Própria

- **Ajustar valor do tempo da sirene:** Ao selecionar o dígito 8, o operador entra no modo de ajuste do temporizador da sirene, este ajuste pode ser feito com valores entre 000 e 999. Nesse caso ele também tem a opção de pressionar E para confirmar ou R para cancelar e voltar para o estado select.config em ambos os casos;

Figura 44 – Tela do estado do temporizador da sirene



Fonte: Autoria Própria

No caso do operador entrar no modo de programação e pressionar a tecla D ao invés da tecla A, a central entra no modo de programação referente as funções de: desabilitar sensores, desabilitar zonas e desassociar sensores a zonas específicas.

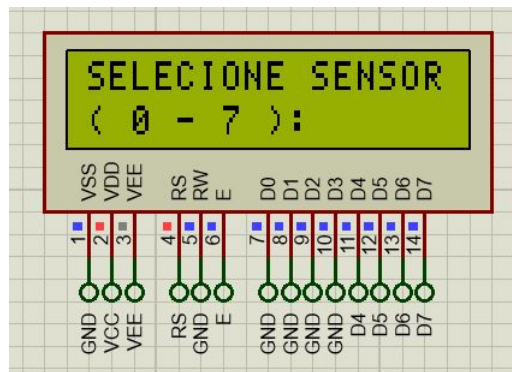
Figura 45 – Tela do estado de desabilitação



Fonte: Autoria Própria

- **Desabilitar sensor:** Na escolha do dígito 3, o operador entra no modo de desabilitação de sensores, podendo desabilitar um dos oito sensores, e caso ele deseje poderá desabilitar outros sensores pressionando a tecla 3 após a escolha do sensor desejado. Nesse caso ele também tem a opção de pressionar E para confirmar ou R para cancelar e voltar para o estado select.config em ambos os casos;

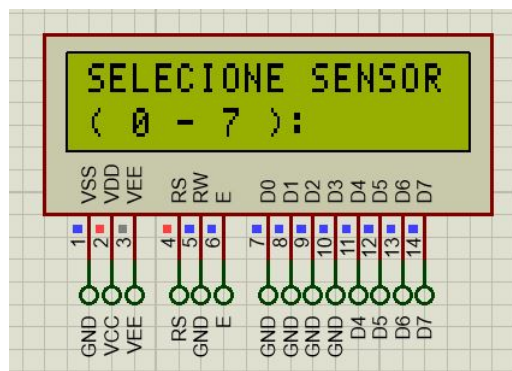
Figura 46 – Tela do estado de desabilitação do sensor



Fonte: Autoria Própria

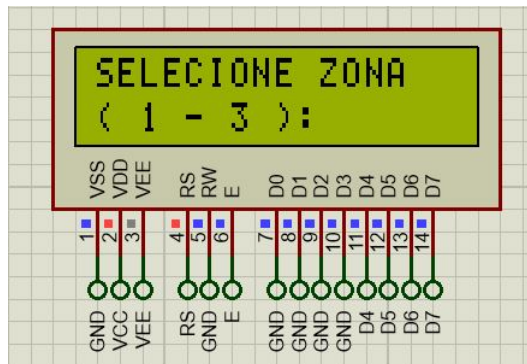
- **Desassociar sensor a zona :** Na escolha do dígito 4, o operador entra no modo de desassociação de sensores a zonas, podendo desassociar um ou mais sensores a uma zona específica. Primeiro ele seleciona o sensor que deseja desassociar e pressiona E, em seguida escolhe a zona, após isso ele poderá apertar 4 salvando a configuração atual e voltando para a opção de escolher um outro sensor para fazer a desassociação. Nesse caso ele também tem a opção de pressionar E para confirmar ou R para cancelar e voltar para o estado select.config em ambos os casos;

Figura 47 – Tela do estado de selecionar sensor para desassociar de zona



Fonte: Autoria Própria

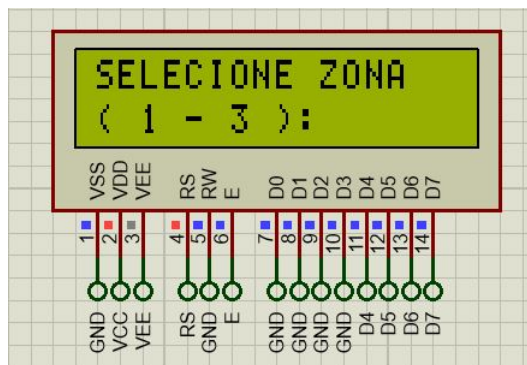
Figura 48 – Tela do estado de selecionar zona a qual sensor será desassociado



Fonte: Autoria Própria

- **Desabilitar Zona:** Na escolha do dígito 5, o operador tem a acesso ao modo de desabilitação de zonas, podendo desabilitar uma das três zonas, e caso ele deseje poderá desabilitar outras pressionando a tecla 5 após a escolha da zona desejada. Nesse caso ele também tem a opção de pressionar E para confirmar ou R para cancelar e voltar para o estado select.config em ambos os casos;

Figura 49 – Tela do estado de desabilitar zona



Fonte: Autoria Própria

O estado **Config Select** presente na MDE do modo de configuração, esta espelhado mas se trata do mesmo estado, o motivo do espelhamento é apenas para fins de facilitar o entendimento das conexões entre os estados internos do modo de programação.

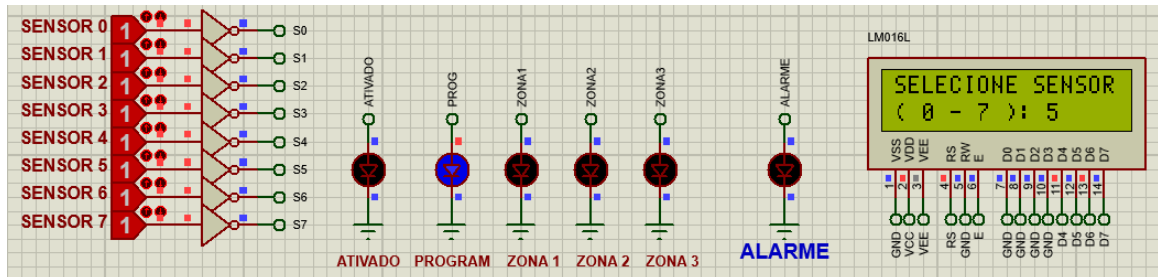
O melhor entendimento dos ajustes das funções presentes no modo de configuração, pode ser feito por meios dos infográficos disponíveis neste documento em anexos.

2.13 Exemplo de operação na central de alarme

Abaixo iremos simular um passo a passo que irá mostrar como associar um sensor a uma zona e como ele será detectado no modo ativado do alarme.

Primeiramente para chegar na tela abaixo é necessário que o usuário esteja no modo desativado, pressione P e coloque a senha mestre corretamente pressionando E em seguida. Após isso aperta A e 4 chegando na tela da figura 50. Foi selecionado o sensor 5.

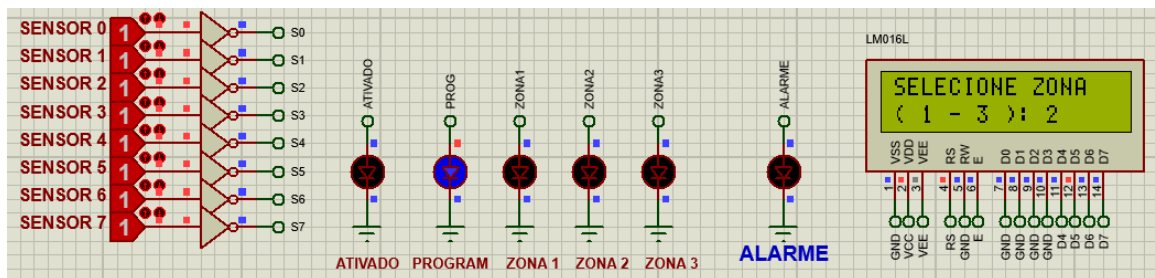
Figura 50 – Passo 1: Selecionando o sensor 5



Fonte: Autoria Própria

Pressionando E vamos para a próxima tela que será a mostrada na figura 51. Foi selecionada a zona 2.

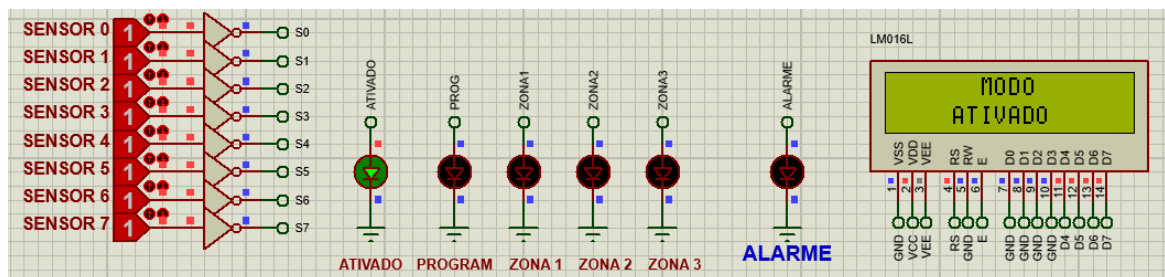
Figura 51 – Passo 2: Selecionando a zona 2



Fonte: Autoria Própria

Pressionando E e em seguida R ou E, vamos para o modo desativado. Pressionando A e colocando alguma senha correta entramos no modo ativado, mostrado na figura 52.

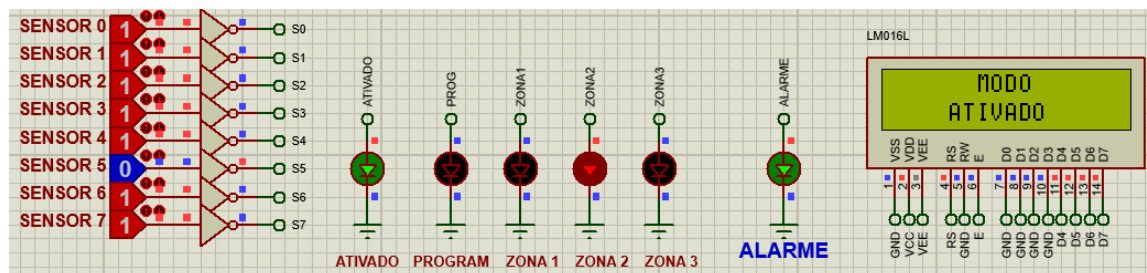
Figura 52 – Passo 3: Modo ativado monitorando os sensores



Fonte: Autoria Própria

Caso seja detectado algo no sensor 5, a sirene é ativada assim como o led da zona 2. O resultado é visto na figura 53.

Figura 53 – Passo 4:



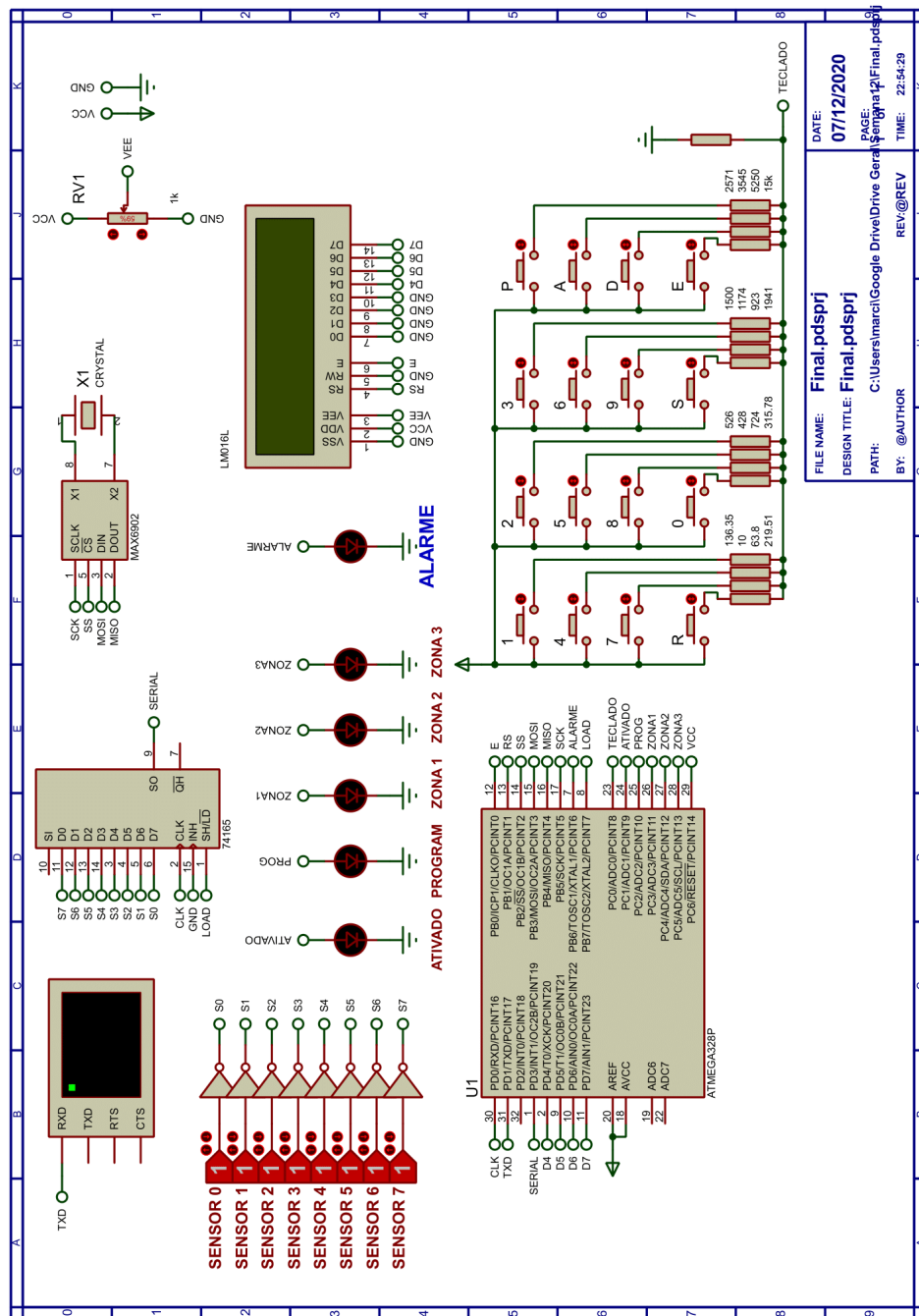
Fonte: Autoria Própria

3 Resultados

Os resultado das simulações feitas da nossa central de alarme estão apresentadas no vídeo referente a este relatório. Para acessar a gravação, disponibilizamos o link de acesso: <<https://youtu.be/gxGGSU8JYaM>>

Além disso, para total visualização do código utilizado na produção da implementação, disponibilizamos o link de acesso: <<https://tinyurl.com/y4egd7dl>>

Figura 54 – Circuito final da Central de Alarme



Fonte: Autoria própria

4 Conclusão

A experiência da implementação do projeto foi frutífera para implementação e fixação de todos os conteúdos aprendidos durante o curso de Sistemas Digitais. Dentre eles temos, comunicações seriais de formas distintas, como UART, SPI e conversão paralelo-serial. Além disso, ainda foi aplicado os conteúdos aprendidos acerca da leitura de dados analógicos, através de conversões analógico-digitais, realizadas e configuradas pelo ATMega328P. Com a utilização de uma diversidade de elementos externos atrelados ao microprocessador, podemos concluir que houve a fixação dos conteúdos citados acima.

Além disso, é basilar citar que além de algumas definições feitas na máquina de estado, a maior alteração feita no projeto passado foi a utilização de uma malha de divisores de tensão para transformar o sinal do teclado em um valor analógico, devido a complexidade de utilizar uma comunicação serial para realizar essa coleta de dados, como proposto anteriormente. Com isso, foi utilizado o ADC para verificar o valor recebido pelo teclado. Em adição, o clock do sistema também foi alterado de 2MHz para 8MHz, tendo em vista que a simulação do circuito apresentou uma performance melhor nessa frequência de oscilação.

Por fim, vale citar que, durante as simulações, a função referente a comunicação UART não funcionou de acordo com o esperado. Entretanto, acreditamos que o problema na simulação foi devido a configuração do Software de simulação.

Referências

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.

MAX6902 Datasheet. Maxim Integrated Products Complete Datasheet. [S.l.], 2003.

Lima, Charles Borges; Villança, Marco valério Miorim. *AVR e Arduino : técnicas de projeto*. 2. ed. Florianópolis, Ed. dos autores, 2012.

Mazidi, Muhammad; Naime, Sarmad; Naimi, Serphr. *The AVR microcontroller and embedded system using assembly and c*. New Jersey Pearson: 2011.