Universidade de São Paulo
Escola de Engenharia de São Carlos

Registrador de chamadas telefônicas de padrão DTMF

Vinícius Paranaíba Campos

São Carlos – SP

Vinícius Paranaíba Campos

Registrador de chamadas telefônicas de padrão DTMF

Trabalho de Conclusão de Curso apresentado à escola de Engenharia de São Carlos, da Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em Eletrônica

ORIENTADOR: Maximiliam Luppe

São Carlos
2012

Campos, Vinícius Paranaíba
C198r Registrador de chamadas telefônicas de padrão DTMF / Vinícius Paranaíba Campos; orientador Maximiliam Luppe . São Carlos, 2012.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2012.

1. Microcontrolador. 2. Telefone. 3. DTMF. 4. Memória. 5. Registrador. I. Título.

FOLHA DE APROVAÇÃO

Nome: Vinícius Paranaíba Campos

Título: "Registrador de Chamadas Telefônicas de Padrão DTMF"

Trabalho de Conclusão de Curso defendido e aprovado em 23 / 11 / 2012,

com NOTA 9,3 (nove, três), *pela Comissão Julgadora:*

Prof. Dr. Maximiliam Luppe (Orientador)
SEL/EESC/USP

Profa Assistente Luiza Maria Romeiro Codá
SEL/EESC/USP

Prof. Associado Evandro Luís Linhari Rodrigues
SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Homero Schiabel

Dedicatória

Dedico este trabalho a todas as pessoas que sempre me apioaram nessa jornada tão importante em minha vida. Família e amigos em geral.

Agradecimentos

Agradeço a todas as pessoas que sempre me apioaram nessa jornada tão importante em minha vida.

Resumo

Com base no avanço tecnológico existente e no que diz respeito a chamadas telefônicas, percebe-se uma grande dificuldade, por parte do usuário, em controlar as ligações realizadas por meio de seu aparelho telefônico. Em muitas residências, por exemplo, é comum o espanto no fim do mês ao checar a fatura da companhia telefônica. Para solucionar este problema foi desenvolvido um protótipo capaz de registrar as chamadas telefônicas realizadas através de um aparelho telefônico. Fazendo-se uso da tecnologia de sistemas embarcados (microcontrolador, cartão de memória, *display* de cristal líquido, detector de tons *DTMF*), o sistema baseia-se na lógica de usuários e senhas cadastrados para permitir ou não a realização de uma chamada. Dessa forma, além de evitar que pessoas estranhas usem o telefone, o registro de chamadas possui cada ligação relacionada com o usuário que a efetuou. Para a validação deste sistema foram realizados alguns testes de segurança e o sucesso foi obtido. Além disso, melhorias e inovações futuras do projeto também foram abordadas permitindo a continuidade do projeto.

Palavras-chave: Microcontrolador, telefone, DTMF, memória, registrador

Abstract

Considering both the technology advance nowadays and everything related to telephone call, it turns visible the difficulty , faced by the users, to control the dialled calls madden from their telephones. At lots of residences, for instance, it is common dissatisfaction, at the end of the month, when they get their phone bill. Who made which call? How to avoid a number to be dialled ? To solve this problem a prototype was developed and is capable of registering the phone calls madden from a telephone. Considering the existing technology on embedded systems(microcontroler, memory card, crystal liquid display, DTMF detector), the system is based on a logic of users and passwords to allow or not the making of a call. This way, besides avoiding strange people to make phone calls the system also provides a log that consists of each dialled number to the specific user that have made such call. To validate the project some security tests were done and the success obtained. Also, future improvements were considered allowing the project to be continued.

The answer for these types of question is what this article looks forward to, by developing a product capable of registering and controlling the residence phone calls, based on digital systems knowledge and application.

Keywords: Microcontroller, telephone, DTMF, memory, logger

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

1.1 Contextualização

Atualmente, devido à total globalização, a sociedade se move com a necessidade de comunicação, que abrange diversos meios como a Internet e o telefone.

Porém, apesar de a telefonia ter evoluido surpreendentemente nos ultimos anos, as tecnologias para o usário não acompanharam tal avanço. Uma residência, por exemplo, onde não só uma família reside, mas empregados trabalham e amigos e estranhos frequentam, geralmente possui um ou mais contratos com companhias de telefone, possuindo, assim, uma ou mais linhas telefônicas.

Quando o proprietário da linha telefônica não está presente no local, a mesma, de modo geral, fica completamente a disposição de quem quiser utilizá-la. Assim, quando a companhia de telefone cobrar pelos seus serviços, o seu valor pode surpreender o proprietário.

Há formas de evitar esse tipo de problema, mas, de modo geral, ou são de valores exorbitantes para um usuário comum ou não são eficazes quanto à determinação do custo gerado, e, principalmente, de quem efetuou tal ligação.

Desta forma, torna-se interessante um produto que seja acessível ao consumidor de forma geral e simples de ser utilizado. É isto que este projeto propõe. De forma resumida, um sistema irá fazer o controle do uso de uma linha telefônica utilizando a lógica de usuários e senhas, registrando e acusando, assim, qual usuário, dentre os cadastrados, realizou determinada chamada telefônica. Sendo assim, o sistema não só disponibiliza um log de chamadas como também impede que telefonemas sejam realizados por pessoas que não tenham um cadastro.

O custo deste projeto é baixo pelo fato de utilizar componentes de baixo custo em sua composição, e também, por permitir uma economia em gastos com ligação, pois impede a realização chamadas indevidas que antes geravam um custo grande ao fim de cada mês.

1.2 Objetivos do projeto

Este projeto tem como objetivo propor uma solução inovadora no âmbito de telefonia fixa por meio da consolidação de estudos e conhecimentos adiquiridos ao logo do curso de Engenharia Elétrica - Ênfase em Eletrônica.

Tais conhecimentos se restrigem a área de sistemas digitais e sistemas embarcados, fazendo-se uso de circuitos integrados, microcontroladores e afins.

Como produto final espera-se um protótipo capaz de registrar chamadas telefônicas e assim permitir uma gerência e análise destas ao logo do mês/ano em uma residência ou local afim.

1.3 Estrutura da monografia

Esta monografia está organizada em seis capítulos conforme a descrição seguinte.

No capítulo 2 tem-se uma descrição geral do projeto. Logo em seguida, no capítulo 3 é apresentado um estudo teórico dos conceitos mais específicos e relevantes dos assuntos envolvidos no trabalho.

Já no capítulo 4 são tratados as especificações e métodos de implementação utilizados no projeto, abordando a parte de Hardware e de Software.

No capítulo 5 são abordados os resultados dos testes realizadas para garantir a validação do sistema como um todo. Por último, no capítulo 6 são apresentadas as conclusões gerais do trabalho e também idéias para melhorias e inovações futuras.

2 Descrição do projeto

De acordo com o fim proposto deste trabalho (registrar chamadas telefônicas), várias maneiras de implementação poderiam ser levadas em consideração. Porém, restrições foram adicionadas ao seu escopo. A primeira restrição imposta é implementar um sistema de simples utilização, visando usuários que não tenham tantos conhecimentos na área tecnológica, principalmente no manuseio de produtos eletrônicos. A segunda restrição é tornar o sistema de menor custo possível. Para obedecer tal restrição, o sistema foi projetado e desenvolvido utilizando os componentes eletrônicos mais vendidos, e de menor custo. A terceira, e última restrição, é desenvolver um sistema que seja prático e rapido, evitando demoras para a simples realização de uma chamada telefônica por parte do usuário.

Este capítulo trata da especificação do projeto levando-se em consideração todas as limitações e restrições do sistema que serão colocadas e citadas de forma que o projeto fique bem definido dentro de seu escopo.

2.1 Estruturação

O projeto é constituído de dois módulos principais, como pode ser visualizado no diagrama em blocos da **Figura** 1.

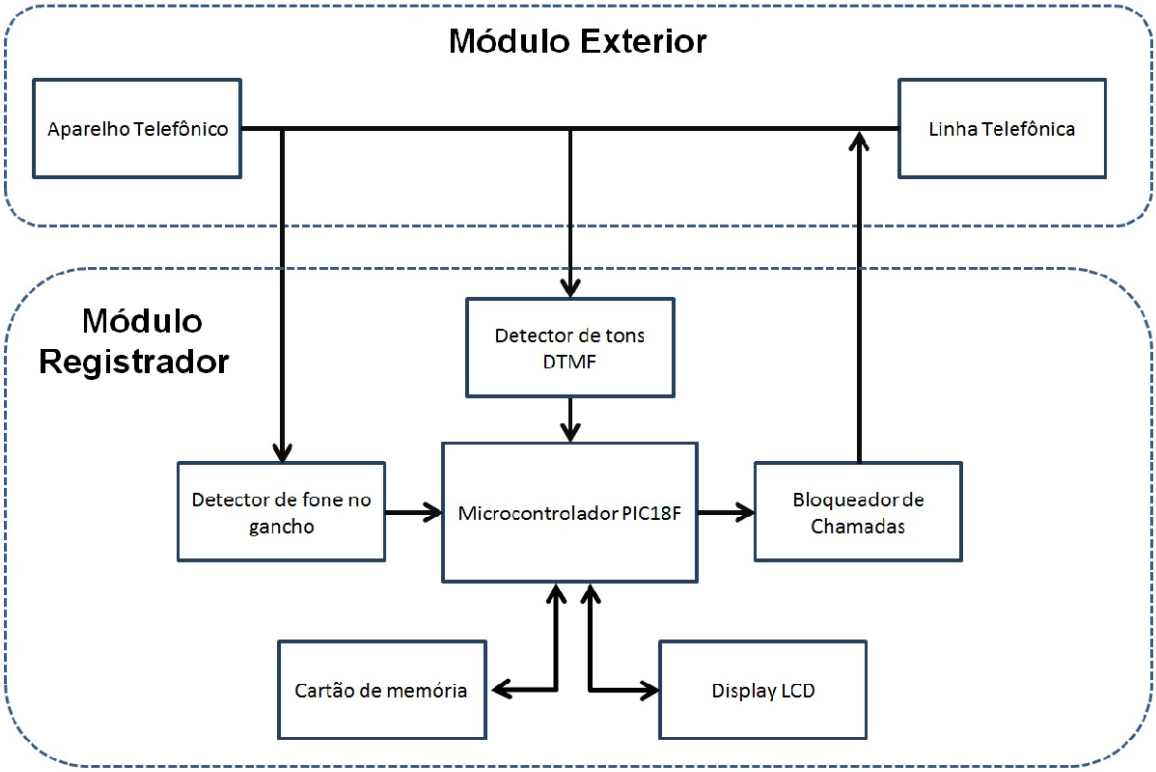


Figura 1: Diagrama em blocos do projeto

2.1.1 Módulo Exterior

O primeiro módulo, o Módulo Exterior, é constituído de elementos que não são parte do projeto, mas são necessários para o seu funcionamento. Sendo assim, a implementação desses elementos não é necessária, mas sim o estudo dos seus comportamentos e das suas funcionalidades.

2.1.2 Módulo Registrador

O segundo módulo, Módulo Registrador, é considerado o coração do sistema. Responsável por realizar a interface homem máquina e e também por gerenciar toda a lógica referente ao fluxo que há entre o usuário retirar o telefone do gancho e completar sua chamada.

Os numeros de usuários e senhas, armazenados no cartão de memória, são utilizados na verificação e validação do sistema para permitir ou não a realização de uma chamada telefônica. Caso o usuário entre com uma senha válida, a linha é liberada e este, então, pode se comunicar normalmente. Caso contrário a linha não é liberada e o acesso negado.

Alem disso, há um registro de ligações realizadas organizadas e relacionadas por usuários. Portanto, a cada término de ligação, uma nova sequência de dados é gerada e então armazenada em um arquivo existente dentro de um cartão de memória.

3 Estudo teórico

Na implementação de todos os módulos, estudos mais aprofundados quanto à funcionalidade e implementação foram necessários. Entretanto, várias decisões foram tomadas quanto ao projeto e implementação deles, em termos de *hardware* e *software*.

3.1 Módulo Exterior

O estudo teórico deste módulo consiste no estudo do funcionamento de uma linha telefônica, considerando-se o aparelho telefônico e a comunicação central-assinante.

3.1.1 Central Telefônica

Um sistema de telefonia pode ser resumido, para fins de compreensão, a uma central telefônica, comunicando-se com varios aparelhos de telefone, de diversos locais, mediante fios elétricos.[4]

A principal função desta central telefônica é estabelecer conexões entre estas linhas telefônicas, seja ela residencial, empresarial, para que duas pessoas distantes possam estabelecer uma conversação.

Para obter-se um padrão na comunicação entre os aparelhos telefônicos, torna-se necessário uma linguagem, ou poderia dizer-se, um conjunto de códigos que sejam entendidos pelos componentes do sistema. Como por exemplo, o tom de linha, os valores de tensão da linha, o sinal de chamada (campainha), etc.[4].

Dentre todos estes sinais presentes na comunicação central-assinante, existem dois que são de suma importância para o projeto, o sinal de chamada(campainha) e o tom de discagem.

O sinal de chamada consiste em um sinal alternado de frequencia 25Hz e valores de tensão entre 70 e 90Vrms(eficazes). Este sinal, quando o telefone está no gancho, faz com

que a campainha soe indicando recepção de uma chamada.[4]

Já o tom de discagem, para que seja produzido pela central, deve detectar uma queda de tensão na linha, que acontece quando o telefone é retirado do gancho. Dependendo da resistência da rede telefônica, a tensão da linha cai de 48V para um valor entre 6 e 14V, DC. Neste momento, a central percebe a intensão de realizar uma chamada , por parte do usuário, e então envia o tom de chamada, permitindo que este digite o número desejado e , posteriormente, se comunique com o destino.[4]

3.1.2 Telefone

O aparelho telefônico é um dispositivo eletrônico responsável pela origem e destino das ligações e que permite a conversação entre assinantes, além de trocar informações com a central telefônica. [6]

De maneira simples pode-se dizer que o telefone possui três circuitos básicos:

- Um circuito de voz
- Um circuito processador de chamadas
- Um circuito de dispositivo sonoro ou campainha

3.1.2.1 Circuito de voz

O circuito de voz consiste , basicamente, em uma cápsula receptora e uma transmissora. A primeira, responsável pela conversão de sinais elétricos em sonoros e a última pelo inverso, ou seja, conversão de sinais sonoros em elétricos. É este circuito que permite a conversação e entendimento dos usuários quando estabelecida uma conexão telefônica.[6] Esta parte não está detalhada devido à pequena importância desta para o projeto como um todo.

3.1.2.2 Processador de chamadas

Um assinante, ao retirar o monofone do gancho, espera o tom de linha originado pela central e então deve digitar o número com o qual deseja estabelecer conexão. Para esta finalidade, há o circuito discador, que pode ser tanto de forma pulsada como em forma de geração de tons DTMF(*Dual Tone Multi-Frequency*).[6]

Existem alguns tipos de "protocolos"de discagem como FSK (*Frequency-shift Keying ou Modulação por chaveamento de frequência*), discagem por pulsos e DTMF, tambem conhecido como discagem de tom. Este sistema, particularmente, foi projetado apenas para atender um tipo de discagem: o de tom, por ser largamente utilizados nos sistemas de telefonia atual, sendo sua utilização , também, bastante simples.

- **Gerador de tons DTMF**

Nos telefones multifrequenciais, quando uma tecla é pressionada, esta ativa a emissão de um par de freqüências na faixa de áudio à central local, que filtra e identifica o par como sendo o código e um número predeterminado. Cada dígito decimal ou tecla possui um par de freqüências específicas, como mostrado na **Tabela** 1

	1209Hz	1333Hz	1447Hz	1633Hz
697Hz	1	2	3	A
770Hz	4	5	6	B
852Hz	7	8	9	C
941Hz	*	0	#	D

Tabela 1: Tabela de frequências - Padrão DTMF. Fonte: `www.ni.com/white-paper/3310/en`

Assim, ao pressionar a tecla 3, por exemplo, as frequências de 1477Hz e 697Hz serão utilizadas para a composição do tom para que possa, então, ser enviado à central.[6]

Os sinais DTMF, por serem confiáveis e facilitarem o projeto de circuitos eletrônicos, têm contribuído para o desenvolvimento de equipamentos sofisticados para automação e controle, como, por exemplo, atendimento automático para saldo bancário, saldo de cartãode crédito, etc. Seguindo esta mesma linha de raciocínio optou-se pela escolha deste padrão para este projeto .

3.1.2.3 Circuito de campainha

O esquema básico, para que se possa entender o funcionamento de um aparelho comum, é mostrado na **Figura** 2.

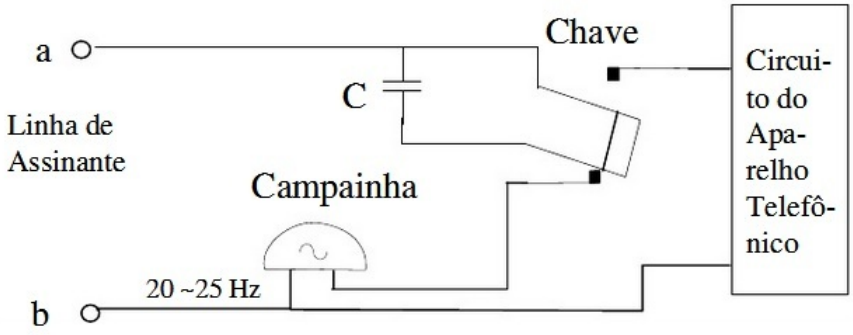


Figura 2: Circuito básico de um aparelho telefônico. Fonte: `pt.scribd.com/doc/38361415/Apostila-Aparelho-Telefone`

Os pontos "a"e "b"do esquema correspondem aos pontos de conexão na rede telefônica.

A chave (chave de gancho) é acionada quando se coloca ou retira o fone do gancho. Esta chave pode manter apenas o circuito da campainha ligado à rede telefônica, o que corresponde ao "fone no gancho", ou então pode deixar o circuito da campainha desligado e o circuito do telefone ligado na rede telefônica, o que corresponde ao "fone fora do gancho".[6][4]

Com o fone no gancho, a campainha fica conectada à linha telefônica, por meio do capacitor "c", cuja função é permitir somente a passagem do sinal alternado de chamada. Este sinal, conforme explicado anteriormente na seção 3.1.1 , faz com que soe a campainha e assim, seja notada a recepção de uma chamada telefônica.[6][4]

Com o fone fora do gancho, o circuito do aparelho telefônico é que fica conectado à rede e, desta forma, devido à resistência da rede, produz uma queda de tensão na linha, fazendo com que esta chegue a um valor de, aproximadamente, 10V, permitindo o funcionamento da eletrônica envolvida em tal circuito.[6][4]

3.2 Módulo Registrador

O Módulo Registrador possui diversos componentes, tais como microcontrolador, circuito integrado para detecção DTMF, cartão de memória SD(*Secure Digital*). Nesta seção é abordado um estudo teórico, separadamente, das características de cada parte deste módulo, dos protocolos de comunicação utilizados, aspectos mais relevantes de *datasheets*, entre outros pontos importantes. Detalhes do *hardware* serão abordados no capítulo 4, seção 4.1.

3.2.1 Microcontrolador PIC18F4550

O PIC(*Peripheral Interface Controller*) é um circuito integrado produzido pela Microchip Technology Inc., pertencente à uma categoria de dispositivos que contém recursos necessários para realizar um completo sistema embarcado. O PIC pode ser visto como um circuito integrado normal, porém dispõe de todos os dispositivos típicos de um sistema microprocessado[7], ou seja:

- CPU (*Central Processing Unit*) - para interpretação e execução de instruções do programa
- EEPROM(*Electrically-Erasable Programmable Read-Only Memory*) - uma memória não-volátil para armazenamento de instruções.
- RAM(*Random Access Memory*) - para armazenamento de variáveis utilizadas pelo programa
- Portas de entrada e saída para controlar/comunicar com periféricos como LCD, sensores, etc.
- Diversos periféricos internos como gerador de relógio,contadores, conversores A/D

Todos estes dispositivos em um espaço extremamente pequeno proporciona grande facilidade de trabalho e enorme vantagem em usar um sistema microcontrolado. Com relativamente poucos componentes externos pode-se obter projetos e sistemas grandes, inteligentes e robustos.

O PIC está presente em uma ampla gama de modelos para melhor adaptar-se às exigências específicas de diferentes projetos específicos, desde os modelos pequenos identificados pela sigla PIC12Fxx com 8 pinos, até chegar a modelos maiores com sigla PIC24xxx, com 40 ou até 60 pinos.[7]

Algumas características peculiares aliadas ao baixo custo do dispositivo fazem com que o PIC18F4550 se torne bastante interessante para o projeto do *Registrador de Chamadas Telefôncias de padrão DTMF*. Dentre estas destaca-se:

- **Módulo SPI**

Necessário para interface com o cartão de memória.

- **Flash de 32Kbytes**

Maior flexibilidade com a implementação do código para o sistema.

- **RAM de 2Kbytes**

Capacidade de armazenamento de variáveis do programa suficiente.

- **Temporizadores de 8 e 16 bits**

Possibilidade de temporização de ações e implementação de relógio para registro das ligações.

- **Frequencia de operação de até 48MHz**

Útil para realização de tarefas que exigem rapidez de funcionamento, assim como, aquisição do número discado pelo usuário.

- **Diversos pinos de entrada e saída e interrupções externas**

Devido à comunicação com diversos dispositivos, a quantidade de entradas e saídas é algo limitante para o projeto. Como pode ser visto na **Figura** 3 este microcontrolador possui uma gama enorme desses tipos de pinos.

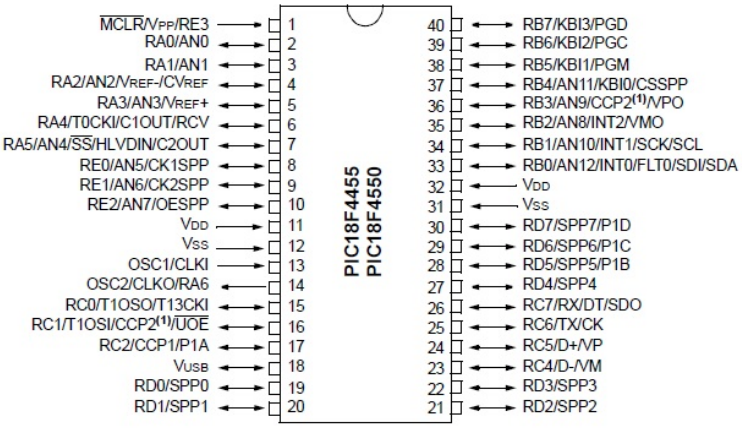


Figura 3: Diagrama de pinos do microcontrolador PIC18F4550. Fonte: Manual do fabricante

Detalhes mais específicos podem ser encontrados no manual do fabricante e, portanto, não serão abordados neste trabalho.

3.2.2 Decodificador DTMF

Com base nos conceitos abordados na seção 3.1.2 deduz-se que, ao contrário de um aparelho telefônico que origina os tons DTMF, um decodificador de tons de padrão DTMF deve fazer simplesmente o oposto, ou seja, receber esses tons e decodificá-los em um número correspondente ao tom..

Muitas são as formas para fazer essa detecção, como implementação de filtros, porém existem no mercado alguns componentes que se encarregam de tal função. Um CI(*Circuito Integrado*) muito utilizado neste ramo é o MT8870.

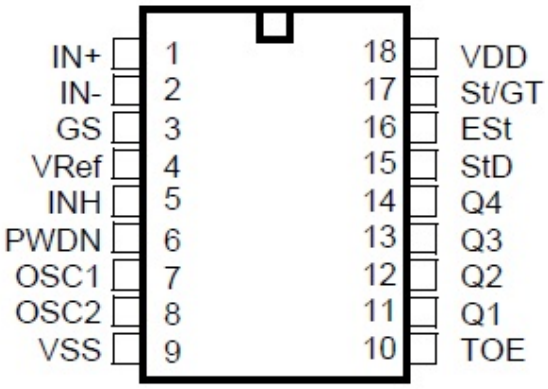


Figura 4: Diagrama de pinos do MT8870. Fonte: Manual do fabricante

O CI mencionado, como pode ser observado na **Figura** 4, possui encapsulamento DIP(*Dual In-line Package*) e 18 pinos para configuração do circuito. O funcionamento deste é explicado com base na **Figura** 5.

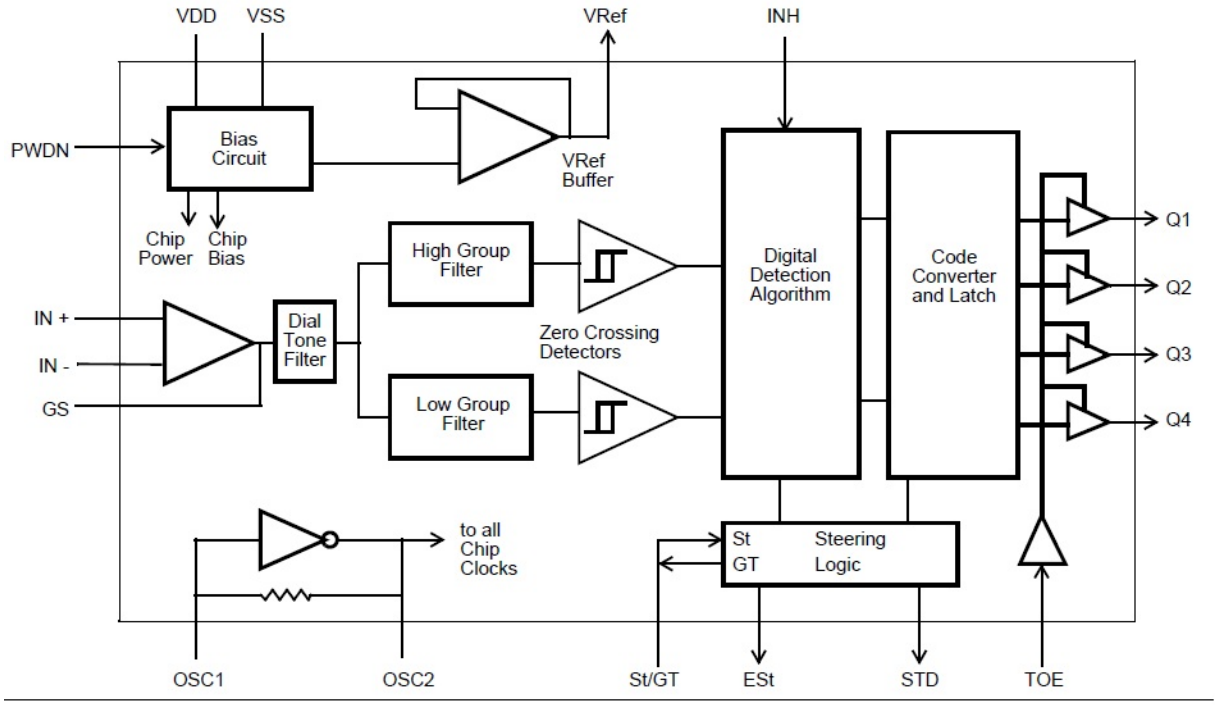


Figura 5: Diagrama de blocos do MT8870. Fonte: Manual do fabricante

Os pinos OSC1 e OSC2 são as entradas correspondentes ao cristal utilizado para o *clock* do sistema. Para compreensão do fluxo de funcionamento da detecção dos tons a abordagem se dará em três etapas:

Seção de Filtros

A separação do grupo de tons baixos e o de tons altos é conseguida através da aplicação do sinal DTMF nas entradas dos dois filtros passa-banda de sexta ordem do circuito.As larguras de banda destes correspondem ao grupo de baixas e altas freqüências considerando o padrão DTMF.[3]

Seção de decodificadores

Após a seção de filtro há um decodificador empregando técnicas digitais de contagem para determinar as freqüências dos tons de entrada e para verificar se eles correspondem a frequências de padrão DTMF. Um algorítmo de cálculo da média protege o sistema contra a simulação de tom por sinais externos, tais como voz, ao mesmo tempo que fornece tolerância a pequenos desvios e variações de freqüência.[3]

Quando o detector reconhece a presença de dois tons válidos a saída EST (*Early Steering*) vai para um estado ativo. Qualquer subsequente perda da condição de sinal fará com que EST assuma um estado inativo .[3]

Steering Circuit

Antes de registrar um par de tons descodificados, o receptor verifica a validade da duração do sinal. Esta verificação é realizada por uma constante de tempo RC externa. O sinal alto na EST provoca uma subida em "vc"(ver **Figura** 6) enquanto o capacitor descarrega. EST permanece elevado durante o período de validação do sinal(tGTP), vc atinge o limite (VTSt) para registar o par de tons, enviando o código de 4 bits correspondente (ver **Figura** 7) para a saída. Neste ponto, a saída GT é activada e leva "vc"a VDD. GT continua em nível alto, desde que EST esteja também em nível alto.

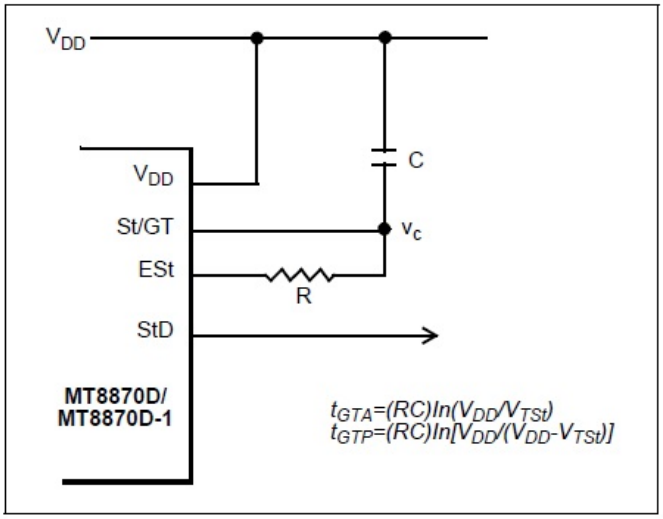


Figura 6: *Steering Circuit.* Fonte: Manual do fabricante

Finalmente a *flag* de saída do *steering circuit* (StD) vai a nível alto, sinalizando que um par de tons foi recebido e registrado. Os conteúdos de saída estarão disponíveis no barramento de saída de 4 bits(Q0 a Q4) caso pino TOE esteja em nível alto.

Digit	TOE	INH	ESt	Q_4	Q_3	Q_2	Q_1
ANY	L	X	H	Z	Z	Z	Z
1	H	X	H	0	0	0	1
2	H	X	H	0	0	1	0
3	H	X	H	0	0	1	1
4	H	X	H	0	1	0	0
5	H	X	H	0	1	0	1
6	H	X	H	0	1	1	0
7	H	X	H	0	1	1	1
8	H	X	H	1	0	0	0
9	H	X	H	1	0	0	1
0	H	X	H	1	0	1	0
*	H	X	H	1	0	1	1
#	H	X	H	1	1	0	0
A	H	L	H	1	1	0	1
B	H	L	H	1	1	1	0
C	H	L	H	1	1	1	1
D	H	L	H	0	0	0	0
A	H	H	L	Se não detectado, saída permanece a mesma da útima detecção			
B	H	H	L				
C	H	H	L				
D	H	H	L				

L= nível baixo **H**= nível alto **Z**= alta impedância **X**= não importa

Figura 7: *Tabela verdade de decodificação*. Fonte: Manual do fabricante

Circuitos exemplos e notas de aplicações são fornecidas pelo fabricante e podem ser encontrados no manual. Estes não serão abordados nesta etapa devido à existência da etapa de projetos, na seção 4.3.1.2, em que o circuito utilizado no projeto é explicitado.

3.2.3 Bloqueador de chamadas

Um bloqueador de chamadas, como o próprio nome sugere, deve impedir a continuidade de uma chamada ou tentativa de chamada, caso algum requesito do sistema não seja cumprido.

Quando uma ligação está em andamento e há o desejo de desligar e realizar uma outra chamada em seguida, o que deve ser feito é simplesmente apertar o "botão"do gancho , esperar um período de tempo necessário para que a central compreenda um término de ligação e depois liberar o "botão"do gancho novamente. Baseado nesta situação e no fato de que , quando o telefone está no gancho, isto significa nada mais que um

"circuito aberto", o que deve-se fazer para elaborar um simples bloqueador de chamadas é utilizar uma chave (*Relay*), abrindo e fechando esta considerando-se o intervalo de tempo mencionado anteriormente.

3.2.4 Cartão de memória SD

O cartão de memória SD é uma memória relativamente simples e barata e que pode ser utilizada para operar em circuitos gerenciados por microcontroladores PIC. Para melhor entendimento deste tipo de memória e aplicação desta ao projeto deve-se levar em conta um aspecto muito importante que é a comunicação SPI.

3.2.4.1 Comunicação SPI

SPI(*Serial to Peripheral Interface*) é um protocolo de comunicação desenvolvido pela Motorola e posteriormente adotado por outras empresas do setor. Às vezes SPI é também chamado barramento serial de "quatro fios".[5]

O protocolo SPI é uma simples interface de comunicação serial a 4 fios usada por muitos microprocessadores / microcontroladores que permite que os controladores e dispositivos periféricos se comuniquem entre si.O barramento SPI, que opera em *full duplex* (sinais que transportam dados em ambas as direções simultaneamente), é um tipo de configuração síncrona de dados com uma interface mestre / escravo e pode suportar até 10Mbps de velocidade. Ambos os protocolos mestre único e multi-mestre são possíveis em SPI, apesar de o multi-mestre ser raramente utilizado e, geralmente, limitado a um único escravo.[5]

Os periféricos mais comumente utilizados são conversores ADC(*Analog-Digital Converter*) e DAC(*Digital-Analog Converter*), módulos de memória EEPROM , sensores de temperatura , sensores de pressão, ou alguns outros dispositivos.[5]

Dados e linhas de controle da SPI e ligação de base:

Um protocolo SPI especifica quatro fios de sinal.

- *Master Out Slave In* (MOSI) - sinal é gerado pelo Mestre, o destinatário é o escravo.
- *Master In Slave Out* (MISO) - escravos geram sinais e o destinatário é o Mestre.
- *Serial Clock* (SCLK or SCK) - sinal SCLK é gerado pelo Mestre para sincronizar a transferência de dados entre o mestre e o escravo.

- *Slave Select* (SS) de mestre para *Chip Select* (CS) de escravo - sinal SS é gerado pelo Mestre para selecionar escravo individual / dispositivos periféricos.

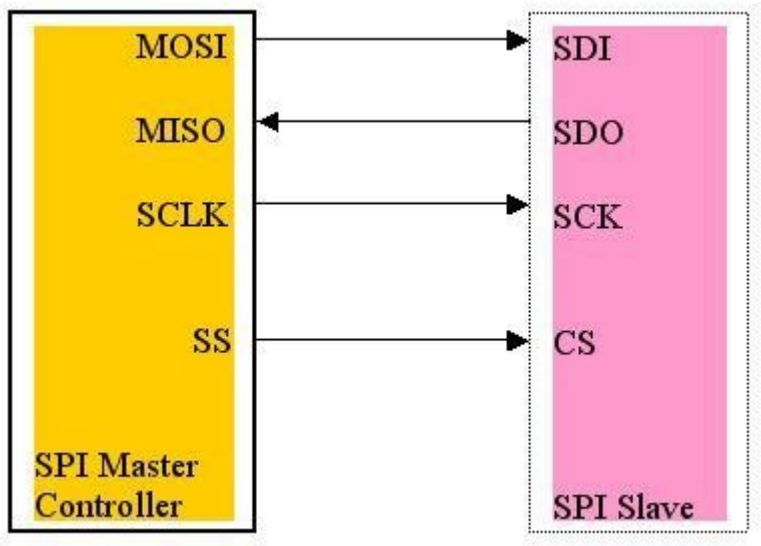


Figura 8: Comunicação SPI. Mestre único, escravo único. Fonte: `www.eeherald.com/section/design-guide/esmod12.html`

Como eles se comunicam:

A comunicação é sempre iniciada pelo mestre. Primeiramente este configura o clock, utilizando uma frequência, a qual é inferior ou igual à frequência máxima que o dispositivo slave suporta. O mestre, em seguida, seleciona o escravo desejado para comunicação setando o *chip select* (SS) de determinado escravo periférico para o estado "baixo". Se um período de espera é necessário o mestre deve, então, esperar por no mínimo um período de tempo equivalente ao de espera para que comece a emitir outros ciclos de clock.[5]

A transmissão de dados pode ocorrer durante cada ciclo de clock. Isso significa que o mestre envia um bit na linha MOSI, o escravo lê este bit na mesma linha e envia um bit na linha MISO, este sendo lido pelo mestre nesta mesma linha.[5]

Significado da polaridade e fase do clock:

Um outro par de parâmetros chamado polaridade de *clock* (CPOL) e fase de *clock* (CPHA) determinam as bordas do sinal de *clock* na qual os dados são conduzidos e amostrados. Isso significa que, além de estabelecer a frequência do *clock*, o mestre também deve configurar o CPOL e CPHA em relação aos dados. Uma vez que o relógio serve como a sincronização da comunicação de dados, existem quatro modos possíveis que podem ser utilizados em um protocolo SPI, com base nestas variáveis.

SPI MODE	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Tabela 2: Diferentes modos de configuração SPI. Fonte: `www.eeherald.com/section/design-guide/esmod12.html`

Se a fase do clock é igual a zero (isto é, CPHA = 0) os dados são transmitidos na subida do clock se CPOL = 0, e, descida se CPOL = 1. Se CPHA = 1, as polaridades são invertidas, dados são trasnmitidos na descida do clock se CPOL = 0, e subida se CPOL = 1. Em geral, os micro-controladores permitem ajustar a polaridade e a fase do clock de acordo com a finalidade de cada projeto.

Vantagens da SPI

- Comunicação *full duplex*
- Escolha arbitrária do tamanho da mensagem, conteúdo e finalidade
- Interface de *hardware* simples
- Escravos usam *clock* do mestre e não precisam de osciladores de precisão
- No máximo um "único"barramento serial por dispositivo (CS), todos os outros são compartilhados

Desvantagens da SPI

- Sem controle de fluxo de *hardware*
- Barramentos multi-mestre são raros e "desajeitados"e são geralmente limitados a um único escravo.
- Sem um padrão formal, validar a conformidade não é possível
- Lida apenas com curtas distâncias se comparada com RS-232, RS-485, ou CAN.

3.2.5 *Display* LCD

Para que toda a interface com usuário seja realizada, utilizou-se um *display* para exibição de mensagens. Mais especificamente, um *display* LCD(*Liquid Crystal Display*) a caractere 20x4(20 caracteres e 4 linhas) e com brilho de fundo.

Este módulo utiliza um controlador próprio(*Control LSI* HCD44780), permitindo uma comunicação com outras placas por meio dos seus pinos.Além de alimentar e conectar os pinos do módulo com a placa do usuário , para que haja o correto funcionamento deste módulo, há um protocolo de comunicação entre as partes, que abrange o envio de *bytes* de instruções e *bytes* de dados. A **Figura** 9 apresenta a descrição dos pinos deste módulo:

Pino	Função	Descrição
1	Alimentação	Terra ou GND
2	Alimentação	VCC ou +5V
3	V0	**Tensão para ajuste de contraste (ver Figura 1)**
4	RS Seleção:	1 - Dado, 0 - Instrução
5	R/W Seleção:	1 - Leitura, 0 - Escrita
6	E Chip select	1 ou (1 → 0) - Habilita, 0 - Desabilitado
7	B0 LSB	Barramento de Dados
8	B1	
9	B2	
10	B3	
11	B4	
12	B5	
13	B6	
14	B7 MSB	
15	A (qdo existir)	Anodo p/ *LED backlight*
16	K (qdo existir)	Catodo p/ *LED backlight*

Figura 9: Lista de pinos e descrições do LCD. Fonte:`www.ece.ufrgs.br/~moreto/files/lcd.pdf`

Percebe-se, assim, a presença de 8 bits de dados para a comunicação com o LCD. Porém, existe a possibilidade de comunicação utilizando-se apenas 4 destes bits. Isto é muito útil quando o microcontrolador do sistema possui poucos pinos de entrada e saída. Nesta configuração, os pinos de 7 a 10 não são conectados ao sistema e os dados e instruções são enviados por *nibles*, com o mais significativo sendo enviado primeiro.[1]

DESCRIÇÃO	MODO	RS	R/W	Código (Hexa)
Display	Liga (sem cursor)	0	0	0C
	Desliga	0	0	0A / 08
Limpa Display com Home cursor		0	0	01
Controle do Cursor	Liga	0	0	0E
	Desliga	0	0	0C
	Desloca para Esquerda	0	0	10
	Desloca para Direita	0	0	14
	Cursor Home	0	0	02
	Cursor Piscante	0	0	0D
	Cursor com Alternância	0	0	0F
Sentido de deslocamento do cursor ao entrar com caracter	Para a esquerda	0	0	04
	Para a direita	0	0	06
Deslocamento da mensagem ao entrar com caracter	Para a esquerda	0	0	07
	Para a direita	0	0	05
Deslocamento da mensagem sem entrada de caracter	Para a esquerda	0	0	18
	Para a direita	0	0	1C
End. da primeira posição	primeira linha	0	0	80
	segunda linha	0	0	C0

Figura 10: Instruções do LCD. Fonte: `www.ece.ufrgs.br/~moreto/files/lcd.pdf`

Analisando a **Figura** 10, por exemplo, caso deseja-se ligar o cursor(função com código em hexa igual a 0E), deve-se enviar primeiramente o *nible* 0000 e depois 1110.

3.2.6 Detector de fone no gancho

O único aspecto relevante de estudo desta parte consiste em entender como funciona um aparelho tefônico e, consequentemente, a comunicação central-usuário, no que diz respeito às características elétricas do circuito em dois estados, telefone no gancho e telefone fora do gancho.

Dado o que ja foi explicado na seções 3.1.1 e 3.1.2, já tem-se embasamento suficiente para elaboração de um circuito que detecte uma queda de tensão na linha, indicando, assim, que o telefone foi retirado do gancho.

4 Metodologia

Nesta parte detalhes mais técnicos com relação à implementação do projeto, tanto de *hardware* quanto de *software*, serão abordados. Estas duas abordagens são subdivididas em especificação, levando em conta todas as variáveis peculiares do projeto, e implementação, baseado nas especificações feitas.

4.1 Especificação de Hardware

4.1.1 Microcontrolador PIC18F4550

Um microcontrolador, para que funcione corretamente e tenha um bom desempenho, requer algumas configurações específicas de *hardware* que, em geral, são padrões e podem ser encontradas no manual do fabricante. Dentre as principais cita-se a utilização de componentes como o cristal, para geração de *clock*, capacitores de acoplamento entre entradas de alimentação, assim como resistores de *pull up* para pino de reset.

Além dessas configurações, este dispositivo, especificamente para o projeto do *Registrador de Chamadas Telefônicas de padrão DTMF*, deverá atender a alguns requisitos como quantidade suficiente de interrupções, sejam estas externas e/ou internas, temporizadores, entradas e saídas digitais em número também suficiente, assim como suporte a diferentes tipos de comunicação para interação com todos periféricos do sistema.

Dessa forma, a escolha do microcontrolador PIC18F4550 se dá de forma ideal. Tendo em vista o que foi abordado na seção 3.2.1, são citados cinco aspectos principais que fazem parte deste dispositivo:

Entradas e saídas digitais

De acordo com o projeto, o que poderá ser entendido mais a fundo conforme o decorrer do documento, são necessários pelo menos 16 pinos de I/O(*Input/Output*), sendo 4 para os bits provenientes do detector de tons DTMF, 7 para comunicação com o *display* LCD

e 4 para interface com o cartão de memórias.

Fácil implementação de *software*

A própria fabricante Microchip disponibiliza uma plataforma de desenvolvimento gratuita e de simples manuseio, o MPLab. Por meio de integração com plugins pode-se implementar *softwares* com linguagem de alto nível em uma linguagem bastante usual, a linguagem C.

Presença de módulo SPI

A comunicação com o cartão de memória é feita utilizando o protocolo SPI de comunicação, sendo assim, a presença deste módulo no microcontrolador é essencial.

Interrupções externas e temporizadores

Tendo em vista a necessidade de temporizar ações do usuário, bem como a implementação de um relógio a presença de dois timers é requerida.

Além disso, existem dois eventos externos que devem ser acoplados à interrupções do microcontrolador, fone no gancho e detecção de tom. Portanto, foram necessárias duas entradas de interrupções externas.

Custo

Ademais de todos aspectos citados, deve-se enfatizar o custo de dispositivos detas família PIC. A Microchip há muito vem crescendo e investindo em tecnologia e seus preços no mercado estão cada vez mais acessíveis no que diz respeito a microcontroladores em geral.

4.1.2 Decodificador DTMF

O Decodificador DTMF é responsável pela identificação da tecla pressionada pelo usuário. Isto é feito mediante a interpretação dos sinais modulados em freqüência existentes na linha telefônica.

Existem no mercado diversas alternativas de componentes que fazem a decodificação de sinais DTMF. Algumas delas são alguns processadores digitais de sinais que possuem suporte a este tipo de decodificação. Porém, o uso deles é complexo, além do custo elevado. Conforme já mencionado na seção 3.2.2, o CI MT8870 foi escolhido.

Um ponto importante a ressaltar é a facilidade com que se pode encontrar materiais(*datasheets, application notes*) para implementação deste módulo.

4.1.3 Bloqueador de chamadas

Para o bloqueador de chamadas, nenhuma especificação muito rigorosa foi feita, tendo em vista que este consiste pura e simplesmente em um *Relay* SPST (*Single Pole Single Throw*) que é, na pratica, uma chave que é acionada por meio de uma saída binária do microcontrolador. A gama de componentes deste tipo é enorme e foi buscado o que melhor atendia à relação custo benefício para o projeto.

4.1.4 Cartão de memória SD

Para registro e armazenamento dos arquivos de senhas e *log* do sistema um cartão de memória SD foi escolhido para o projeto.

Este tipo de memória é bastante simples, muito barata e, por estar presente em diversos outros aparelhos eletrônicos como máquinas fotográficas, calculadoras científicas, celulares, já não é novidade para os usuários e estes não teriam problema quando em contato com esta tecnologia. Com relação a marca e capacidade a flexibilidade é grande, porem optou-se por uma de pequena capacidade por ser mais barata e devido ao dimensionamento do projeto como segue.

Cada ligação gera uma quantidade de, no máximo, quarenta e oito de *bytes*(esta parte está detalhada no seção 4.3.2). Pensando-se em uma residencia com bastante moradores gerando uma média de 10 ligações por dia(média bem alta esta), em um mês, haverá 14400 *bytes* armazenados no cartão. Ao final de um ano, 172800 *bytes*, o equivalente a aproximadamente 170Kbytes. Desta forma, uma memória de 2GBytes é mais que suficiente para o projeto.

Vale ressaltar que, cada protótipo deve, por segurança e garantia, estar atrelado à um único cartão, não sendo garantido o funcionamento caso isto não aconteça. Isto se deve ao fato de cada memória possuir um *offset* de enderaçamento diferente. Com o cartão de memória utiliazdo, por exemplo, percebeu-se que, para a escrita em um endereço X da memória deve-se utilizar, na verdade, o valor de endereçamento de $X + 70144$, obtendo, dessa forma, sucesso de endereçamento tanto na escrita e quanto na leitura. Este valor de *offset*, 70144, foi encontrado empiricamente.

4.1.5 *Display* LCD

Tendo em vista a interface com o usuário do aparelho telefônico torna-se necessário a aquisição de um *display* LCD para a visualização de mensagense afins.

Existem hoje no mercado, vários tipos de Displays LCD, que se diferenciam , substancialmente, em três aspectos: quantidade de caracteres por linha, número de linhas, presença de luz de fundo(blacklight).

Conforme abordado na seção 3.2.5, a utilização de um *display* LCD a caractere com de luz de fundo 20x4(vinte caracteres por linha, quatro linhas). Tais requisitos são essenciais devido à maior flexibilidade com relação às mensagens mostradas no display e melhor leitura destas, facilitando, assim, o manuseio por parte do usuário.

4.1.6 Detector de fone no gancho

O circuito detector de fone no gancho é , também, relativamente simples. Deve-se ter um circuito opto-acoplado de forma que uma queda de tensão na linha seja detectada e , mediante um transistor de chaveamento, uma lógica de nível de tensão(5 volts = Fone fora do gancho; 0 volts = fone no gancho) se mostre presente em uma entrada do microcontrolador.

Sendo assim, optou-se por utilizar o opto-acoplador 4N25 e, juntamente com este, diodos zener de 24V e um led para sinalização. Todos componentes muito fáceis de serem encontrados no mercado e com preços totalmente acessíveis.

4.2 Especificações de Software

Atendendo-se todas as especificações de *hardware* previamente citadas, deve-se pensar, agora, nos requisitos que o *software* necessita. Tendo em vista que este é o responsável por todo o funcionamento e lógica do sistema, muito cuidado deve-se tomar em seu planejamento.

Para que um bom resultado seja atingido, três pontos mais importantes devem ser evidenciados quanto ao funcionamento deste sistema embarcado:

4.2.1 Rapidez e dinâmica de funcionamento

Tendo em vista o propósito deste projeto, visando o registro e controle de chamadas telefônicas, tem-se que quanto menor o tempo expendido pelo usuário entre a retirada do fone do gancho e a discagem do número desejado, melhor é o desempenho deste e consequentemente sua aceitação por parte do usuário. Dessa forma, simplificar rotinas e permitir uma interface simples, com mensagens claras, é , sem dúvida, requisito essencial para implementeção do *software*.

4.2.2 Robustez

Se existe algo totalmente indesejado em qualquer projeto, seja ele de *software* ou não, é a presença de erros que possam impedir a funcionalidade do sistema em determinado momento.

Visando a maior robustez possível vários casos de testes devem ser pensados e testados para garantir que o sistema não falhe, ocasionando erros como, por exemplo, a oportunidade de realizar uma chamada e esta não ser registrada. Os testes e resultados estão abordados mais detalhadamente no capítulo 5.

4.2.3 Rotinas básicas

O projeto de *software* deve atender à implementação de algumas rotinas básicas e importantes , que servirão de alicerce para as demais funções que se mostrem necessárias para agregar valor ao bom funcionamento do *software*. Estas funções estão explicitadas abaixo:

- Leitura e escrita de arquivos do cartão de memória
- Leitura e interpretação dos bits provenientes da decodificação DTMF
- Leitura e escrita de mensagens no *display* LCD

4.3 Projeto

O projeto detalhado nesta parte foi feito de acordo com as especificações mencionadas nas seções anteriores. Estas especificações foram responsáveis pela definição do escopo

do sistema, identificando como e em que direção ele funciona. Nesta fase, tudo que acontece no sistema (eventos, interrupções) é detalhado e projetado. Como o sistema está dividido fundamentalmente em *hardware* e *software*, os projetos destes serão apresentados da mesma maneira.

4.3.1 Projeto de *Hardware*

O projeto de *hardware* e consequente entendimento deste é realizado por meio de análise e descrição do funcionamento dos submódulos tendo como base os desenhos esquemáticos feitos utilizando a ferramenta Altium. Cada parte tem seu desenho mostrado separadamente, tendo em vista que o esquemático completo, com todas as partes integradas, encontra-se no **Apêndice A**.

4.3.1.1 Microcontrolador PIC18F4550

O microcontrolador é o coração do sistema e, por isso, pode-se dizer que este se comunica ou tem ligação com todos outros submódulos.

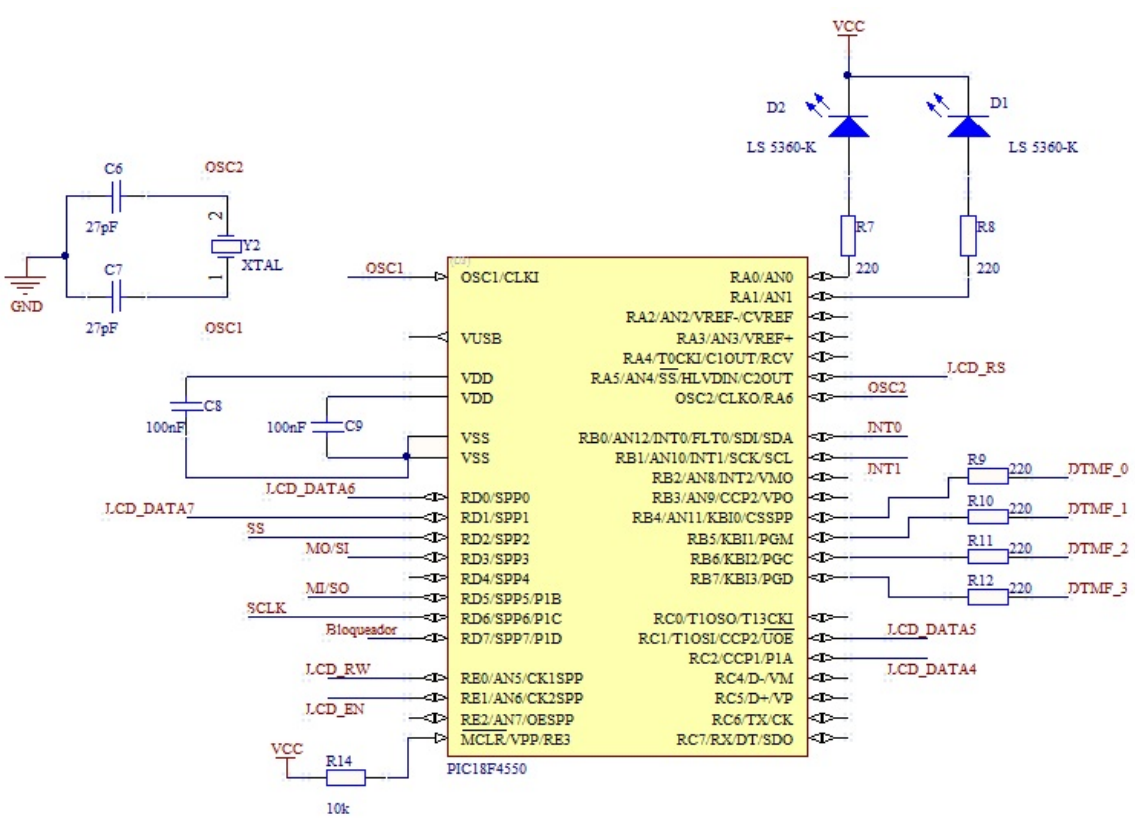


Figura 11: Esquemático do módulo microcontrolador

De acordo com a **Figura** 11 divide-se a análise em duas partes:

Componentes para funcionamento básico

Ao lado esquerdo ve-se um cristal acoplado a dois capacitores. Este é o responsável pela geração do *clock* base para todas atividades realizadas pelo microcontrolador.

Em conjunto tem-se, nos pinos Vss e Vdd, a acoplagem destes por meio dos capacitores de 100nF. Isto se faz necessário para evitar distúrbios provenientes de ruídos.

Finalmente tem-se, na parte inferior, o circuito de *reset* do dispositivo, que consiste em um resistor de 10K ligado ao Vcc e, em série com este, ligado ao Gnd, um botão para acionar o *reset*.

Interface com periféricos e interrupções

Ademais dos componentes básicos necessários para o funcionamento do PIC, nota-se a presença de vários rótulos presentes nas entradas e saídas deste. Esses rótulos indicam a qual periférico e que tipo de dado corresponde. Como exemplo tem-se LCD_DATA4, LCD_DATA5, LCD_DATA6, LCD_DATA7, que são os pinos de saída correspondentes aos dados que são enviados para o *display* LCD.

Por fim, destaca-se a presença de dois circuitos equivalentes compostos por LED em série com uma resistência. Estes são utilizados para sinalização das interrupções, tanto a de tecla digitada quanto a de detecção de estado do fone.

4.3.1.2 Decodificador DTMF

Um decodificador DTMF é o elemento que faz a conversão sinal DTMF para binário. Isto é necessário para que o sistema possa identificar quais teclas do telefone foram pressionadas.

Conforme mencionado no capítulo 4, o CI utilizado para este fim foi o MT8870. A seguir, esquemático deste módulo:

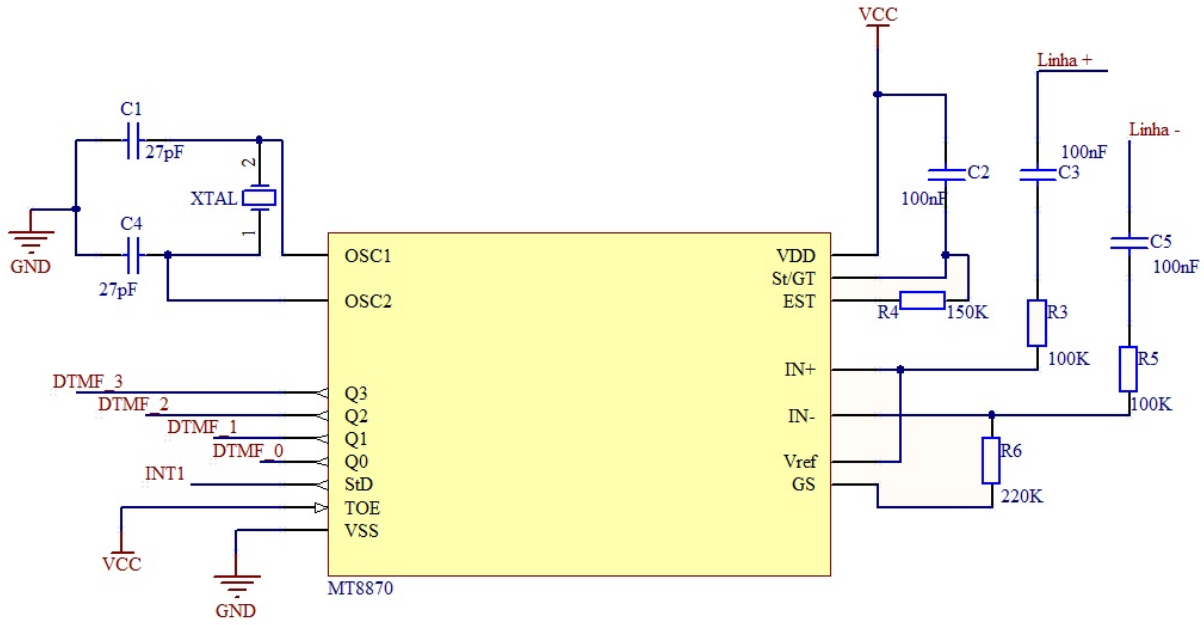


Figura 12: Esquemático de módulo Decodificador DTMF

O circuito apresentado na **Figura** 12 foi baseado no próprio *datasheet* do produto e mostra a simplicidade do MT8870. Houve algumas mudanças nos componentes adjacentes ao CI(capacitores e resistores, basicamente), para melhor adequação ao projeto no que diz respeito à perfeita detecção de um tom DTMF.

Com este circuito é possível fazer a decodificação dos sinais DTMF, obtendo o valor binário nas saídas Q0 a Q3. Tais saídas devem ser ligadas a quatro entradas do microcontrolador para permitir a leitura e interpretação da tecla pressionada no aparelho telefônico. Quando a saída StD estiver ativada, significa que um sinal DTMF foi decodificado e está pronto para ser lido. Este sinal StD pode ser ligado na interrupção externa do microcontrolador, gerando uma interrupção na medida que o usuário do telefone estiver discando.

4.3.1.3 Bloqueador de chamadas

Este módulo, como já mencionado anteriormente, é de uma simplicidade notória. Analisando a **Figura** 13, ve-se que este circuito consite em um *Relay SPST* que é acionado pelo sinal proveniente do microcontrolador, que no caso está representado pelo rótulo "Bloqueador".

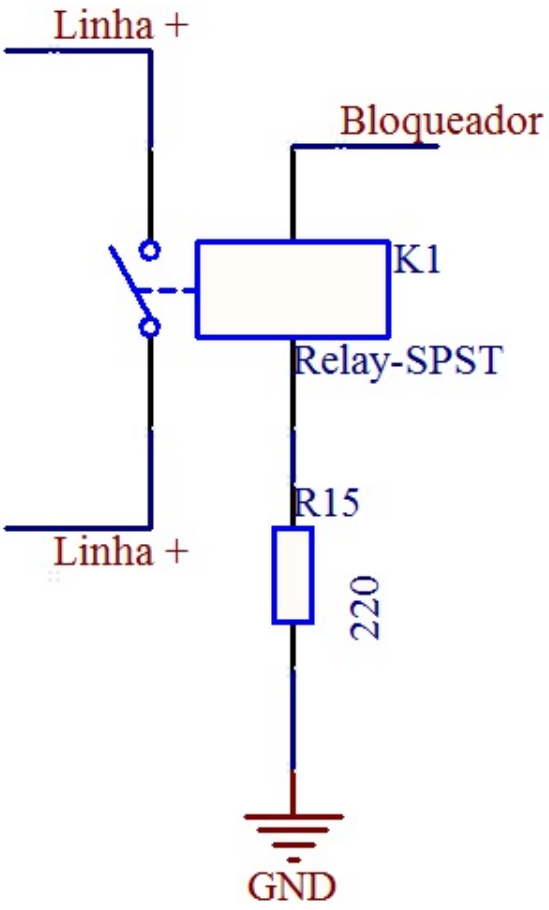


Figura 13: Esquemático de módulo bloqueador de chamadas

Ademais do *Relay*, apesar de este apresentar um valor de resistência intrínseco, tem-se um resistor ligardo em série que chega ao terra, limitando assim a corrente do circuito, já que indutores se portam, na teoria, como curto-circuitos para sinais não alternados.

Este valor de resistência foi encontrado considerando-se a máxima corrente que pode ser fornecida pelo microcontrolador, que gira em torno de 25mA. Sendo assim, considerando o indutor como um curto, temos 5volts aplicados em 220 Ohms, o que gera uma corrente de, aproximadamente, 22,7mA.

4.3.1.4 Cartão de memória SD

A configuração de *hardware* necessária para estabelecer o funcionamento do cartão de memória e também sua comunicação com o microcontrolador consiste, fundamentalmente, na aplicação de divisores resistivos.

Os cartões de memória SD trabalham com uma tensão de aproximadamente 3,3 volts, o que é relativamente menor que a tensão propiciada pelo microcontrolador em suas saídas digitais, que é na ordem de 5 volts.

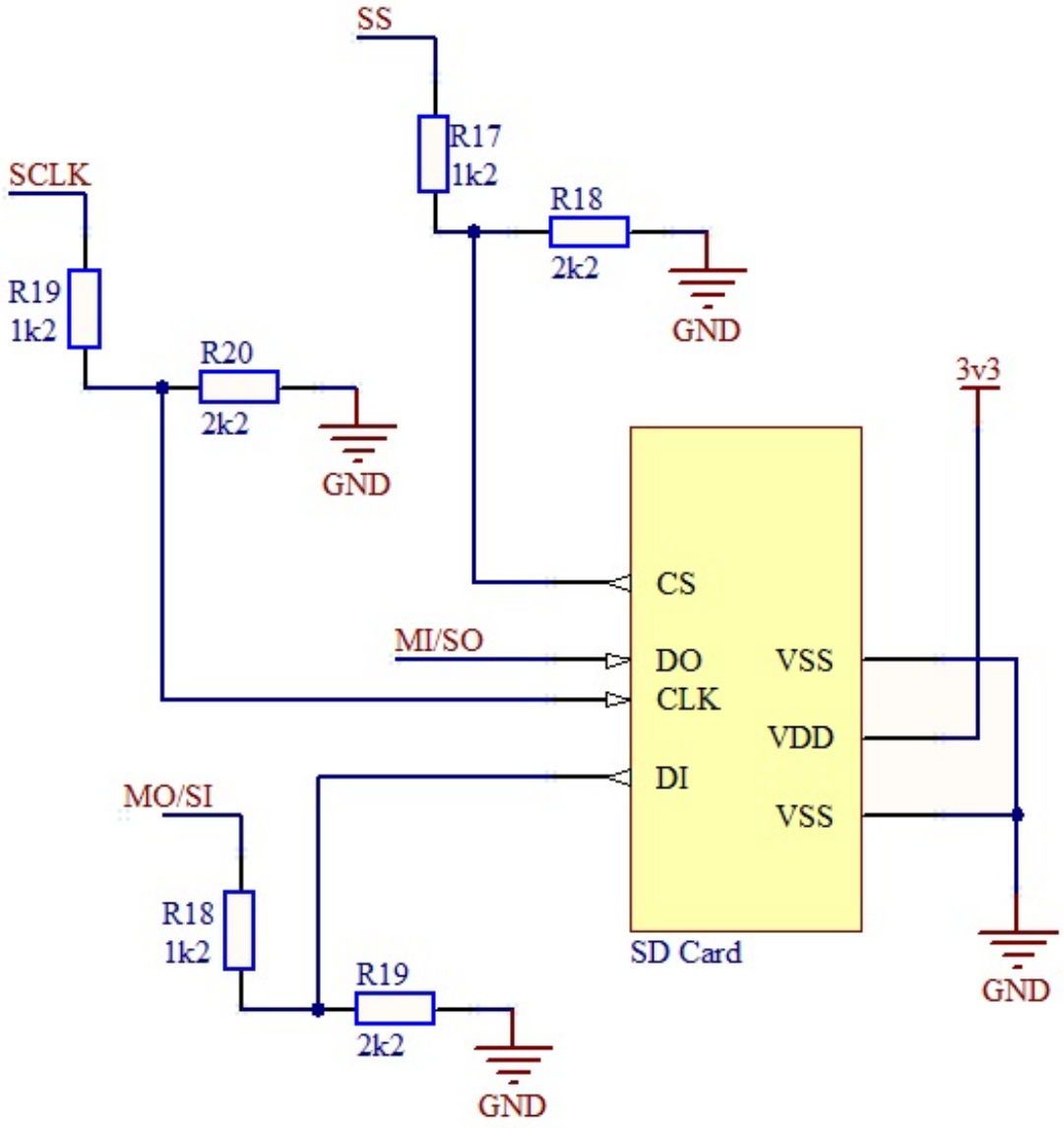


Figura 14: Esquemático de módulo Cartão de memória SD

Para contornar essa situação utilizou-se um divisor resistivo com resistores de 1,2K e 2,2k, conforme pode ser observado na **Figura** 14. Com esta configuração a tensão presente nas entradas do cartão de memória se reduzem a aproximadamente 3,23 volts, tensão esta suficiente para o funcionamento do cartão de memória. Vale ressaltar que para a saída do cartão, pino MI/SO, o divisor resistivo não é necessário. Isto acontece pelo fato de que para o microcontrolador, um valor de 3,3volts em sua entrada já é considerado como nível lógico 1.

4.3.1.5 *Display* LCD

O circuito do *display* LCD é bem simples e consiste basicamente na alimentação deste, ajuste de contraste, pinos para comunicação e recepcção de dados dos microcontrolador, e ajuste de brilho de fundo.

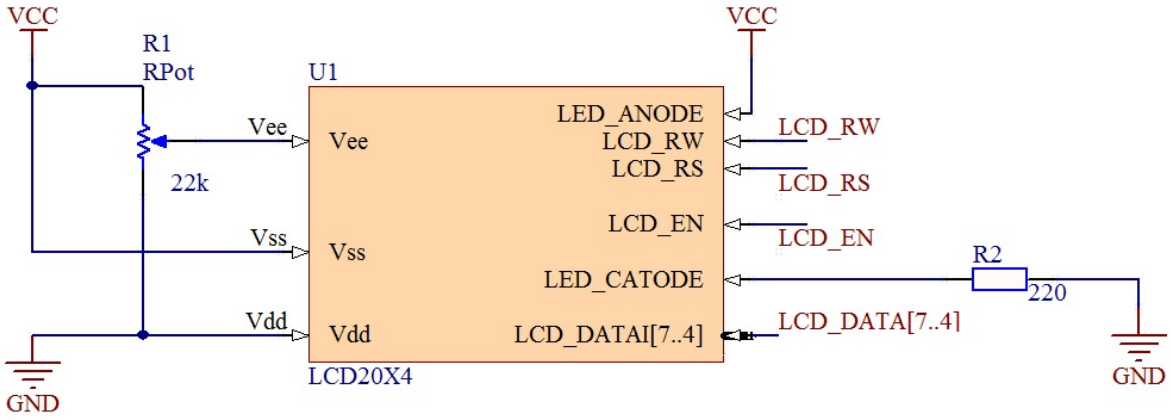


Figura 15: Esquemático de módulo *display* LCD

Os valores de resistência para ajuste de contraste(pino Vee) e brilho de fundo(pinos LED_ANODE e LED_CATODE) foram encontrados experimentalmente realizando variação de um potenciômetro ate encontrar o ponto ótimo de funcionamento.

4.3.1.6 Detector de fone no gancho

O Detector de fone no gancho é um circuito em que, ao colocar uma determinada tensão na sua entrada, sua saída alterna entre nível lógico 0 ou 1 de acordo com o estado do fone (no gancho ou fora do gancho). Esta detecção é feita por meio da verificação da tensão presente na linha telefônica. Com base nos estudos teóricos e em experimentos , contatou-se que a tensão usual é de aproximadamente 48V(corrente contínua) na linha telefônica quando o fone está no gancho e de aproximadamente 7V quando este está fora do gancho. Tais valores de tensão para o segundo estado citado variam de local para local.

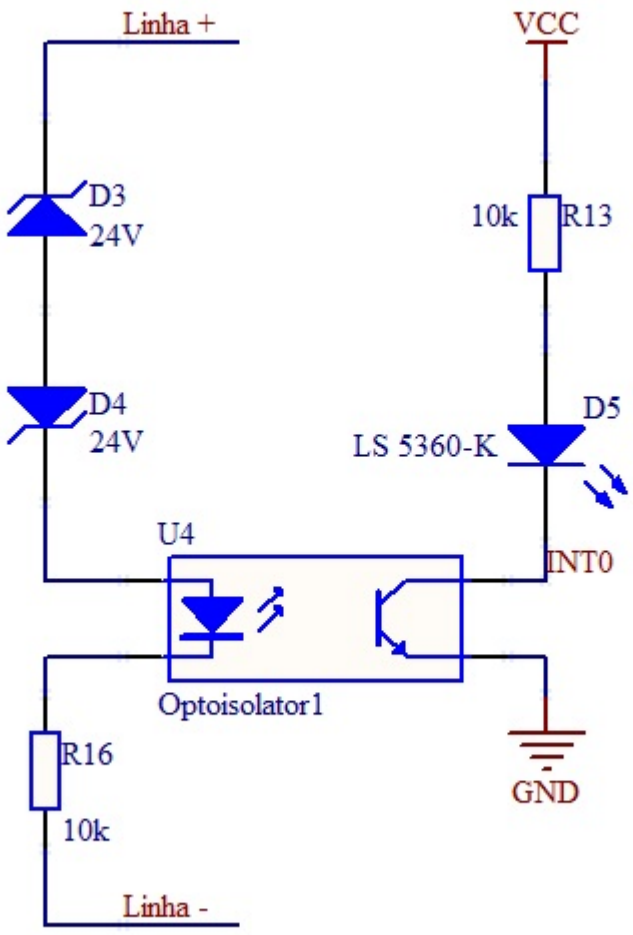


Figura 16: Esquemático de módulo Detector de fone no gancho

Pode-se observar na **Figura** 16 o circuito elétrico do detector de fone no gancho. A tensão de 48V é verificada entre as entradas Linha+ e Linha-. O diodo zener D3 grampeia a tensão em 24V quando esta é superior à 24V. O resistor R16 é limitante da corrente de saída dos zeners.

Calcular esta corrente nem sempre nos garante um bom funcionamento do aparelho, já que o fornecimento de corrente da linha telefônica é muito limitado. Se existe qualquer carga drenando corrente, há uma diminuição de tensão na rede. Por este motivo, o valor do resistor R16 foi encontrado experimentalmente, colocando um potenciômetro em seu lugar. Ajustando o potenciômetro, bastou medir a tensão da saída do fotoacoplador (neste caso no coletor), a mesma saída que vai para a interrupção externa do microcontrolador.

Quando a tensão fosse de aproximadamente 5V, com o fone fora do gancho, e aproximadamente 0V com o fone no gancho, a resistência do potenciômetro poderia ser utilizada. Outro problema que surge trabalhando com linhas telefônicas, é o cuidado com a geração de ruídos que possam atrapalhar o funcionamento do sistema. Por este motivo que se tem o fotoacoplador no circuito, que isola a fonte e o terra usados no transistor dele, com a tensão e o terra da linha telefônica.

Outra característica e utilidade deste circuito é na detecção de uma chamada que está sendo recebida. Tendo em vista que o sinal gerado pela central telefônica para esta situação é um sinal senoidal com amplitude de aproximadamente 70Vrms, a saída deste circuito oscilará entre o nível 0 e 1 em conformidade com o tom de campainha, fazendo com que o *led*(Light-Emitting Diode) fique também alternando entre os estados aceso e apagado ,incando ao usuário o recebimento de uma ligação.

4.3.2 Projeto de Software

O projeto de Software se resume na apresentação de fluxogramas no decorrer da descrição deste tópico. Os fluxogramas descrevem, em um nível alto, a funcionalidade do programa, que possui código desenvolvido na linguagem C para microcontroladores utilizando-se a plataforma MPLAB e o compilador CCS Compiler. Tópicos mais técnicas no que se diz respeito ao código fonte não serão abordados tendo em vista que este pode ser encontrado no **Apêndice B**.

O *software* é o responsável pela interface entre o usuário e o sistema. Por meio dele, o usuário tem mecanismos de informar ao sistema o que ele deseja e de ter seu desejo satisfeito. A interface foi projetada para ser de fácil uso, sem necessidades de muitos conhecimentos e práticas tecnológicas. O fluxograma a seguir apresenta o andamento, de uma forma generalizada, do projeto como um todo. Cada passo é explicado e, se necessário, seu funcionamento detalhado, podendo fazer uso, ou não, de outros fluxogramas.

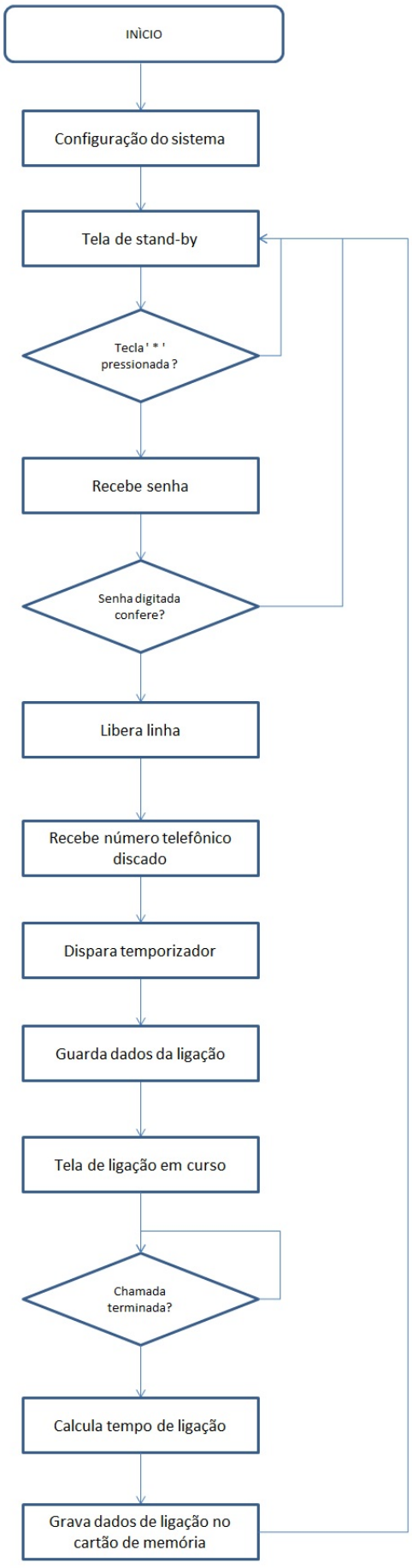


Figura 17: Fluxograma de funcionamento do registrador

Configuração do sistema

A inicialização e configuração do sistema como um todo é a primeira ação que deve ser realizada . Configurar portas, inicializar periféricos, dentre outras, são ações pertinentes a esta etapa do processo. A seguir é apresentado um fluxograma com as tarefas executadas e estas serão mais detalhadas separadamente.

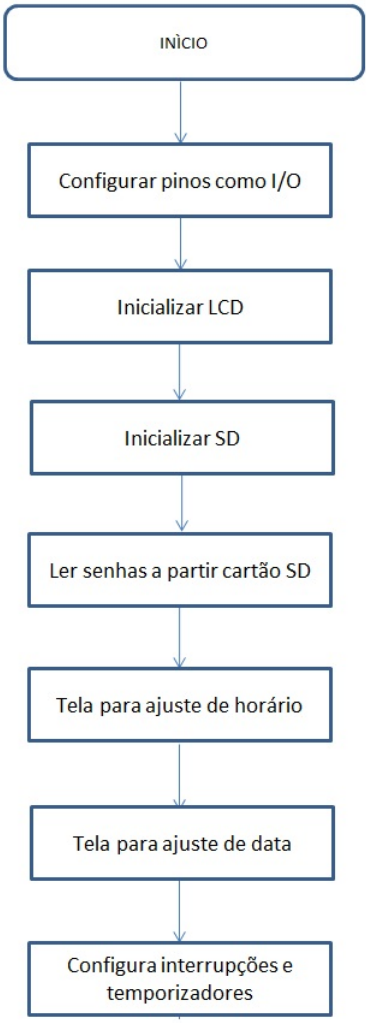


Figura 18: Fluxograma da etapa de inicialização do sistema

- **Configurar pinos como I/O**

  Como o próprio nome da tarefa sugere, neste momento deve-se configurar o *hardware* do microcontrolador e fazer as configurações devidas dos pinos para que estes funcionem corretamente, sejam como entrada(*input*) ou como saída(*output*).

- **Inicializar LCD**

  Função pré definida pela biblioteca de comunicação com LCD necessária para correto gerenciamento do periférico.

- **Inicializar SD**

  Analogamente ao item anterior, 4.3.2, esta tarefa é responsável por inicializar o periférico cartão de memória SD, inicializando a comunicação SPI e algumas variáveis pertinentes ao sistema de arquivos.

- **Ler senhas a partir do cartão SD**

  Após a inicialização do cartão deve-se realizar a leitura e armazenação dos dados presentes no arquivo "users.txt". Estes dados são referentes aos usuários e senhas cadastrados no sistema.

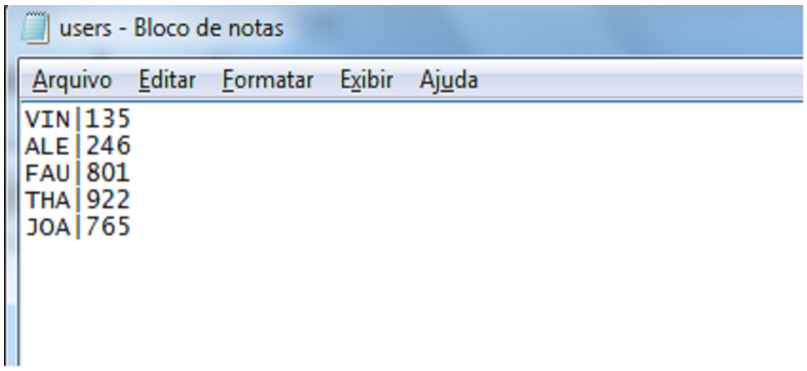


Figura 19: Estrutura do arquivo de usuários e senhas

A **Figura** 19 mostra como o arquivo é estruturado. Cada linha e referente a um usuário diferente, sendo que os dados são nome de usuário e senha, separados pelo caractere '|'. Estes dados são cadastrados previamente pelo usuário fazendo-se uso de um computador. Nada é feito no sistema em si no que diz respeito ao cadastro de usuários e senhas.

Vale ressaltar que o projeto atual permite o cadastro de 10 usuários e senhas. A utilização de um número maior não é recomendada pelo fato de uma maior possibilidade de erros de leitura destas, ocasionando mau funcionamento do sistema.

- **Tela para ajuste de horário**

  Esta parte é bem interessante pois é a responsável pela inicialização das variáveis referentes ao horário do sistema.

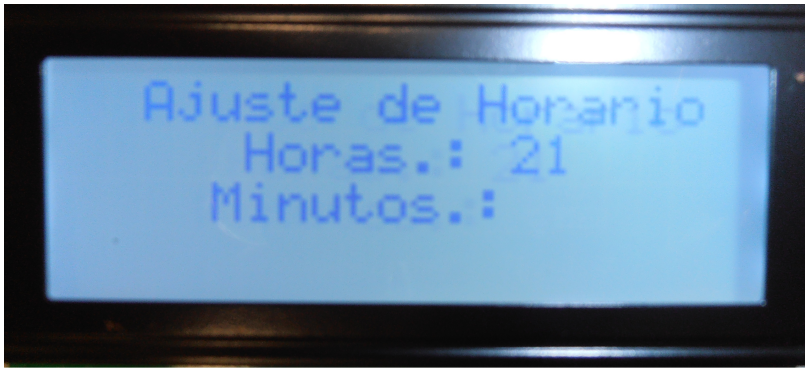


Figura 20: Tela para ajuste de horário

Fazendo-se uso de uma interface com o usuário por meio do *display*, o sistema o leva a ajustar o horário conforme pode ser verificado por meio da **Figura** 20.

- **Tela para ajuste de Data**

  Análogamente ao item anterior, 4.3.2, nesta fase o usuário é induzido a ajustar a data do sistema, como pode ser observado abaixo na **Figura** 21.

Figura 21: Tela para ajuste de data

- **Configurar interrupções e temporizadores**

  Após todas as etapas acima descritas, para finalizar a inicialização, basta configurar corretamente as interrupções e temporizadores atentando a bordas de subida ou descida e valores de registradores para contagem e incremento do relógio.

Tela de *stand-by*

Neste momento mostra-se a tela de *stand-by* do sistema. Esta consiste de uma mensagem orientadora ao usuário indicando a tecla a ser pressionada caso queira realizar um telefonema, e da data e hora do sistema.

Figura 22: Tela de stand-by do sistema

Neste momento, considerando a linha telefônica do usuário, esta funciona normalmente para receber chamadas, pois a interrupção proveniente da corrente de toque em nada influencia no sistema.

O sistema só prosseguirá para a realização de uma chamda caso a tecla correta, o asterisco(*), for pressionada.

Tecla ' * ' pressionada?

O *software*, cada vez que o usuário pressionar uma tecla, estando o sistema nesta parte, ira fazer a verificação permitindo ou não a continuidade do fluxo.

Um aspecto importante a ressaltar é o fato de, primeiramente, ter utilizado a interrupção referente à Detecção do fone no gancho para dar continuidade ao fluxo. Teoricamente seria mais prático, porém teria problemas em administrar essa interrupção, tendo em vista que esta também é acionada pela tom de chamada, variando rapidamente em um curto espaço de tempo.

Recebe senha

Caso o usuário tenha teclado o asterisco, indicando , dessa forma, que deseja realizar uma ligação, deve-se agora preparar o sistema para receber os dígitos referentes à senha deste usuário.

Nada muito diferente de todas as rotinas de recebimento de teclas. O procedimento agora consiste em armazenar os digitos em um vetor de caracteres de tamanho três(sistema projetado para senhas de três digitos).

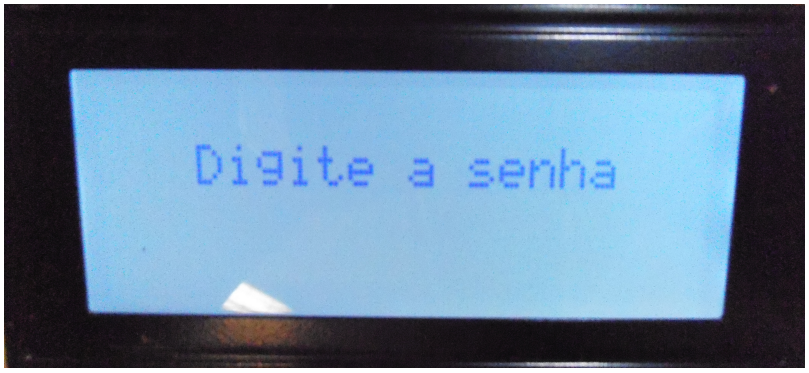


Figura 23: Tela para recebimento de senha

A partir do momento que a detecção e armazenamento da terceira tecla foram feitas, o fluxo segue para a verificação e validação dessa senha.

Senha confere?

Este ponto do *software* é de suma importância, pois é aqui que tem-se o fundamento e proposta do projeto, portanto todo cuidado deve ser tomado para esta validação.

O procedimento para conferência da senha digitada, apesar de toda importância, é bem simples. Busca-se, no vetor de senhas cadastradas existentes, por uma equivalência dos três dígitos, isto fazendo-se uma "varredura"dos caracteres de ambos vetores(senha digitada e senhas existentes). Caso qualquer caractere falhe na equivalência, o sistema intercepta o processo e retorna para o começo de tudo, ponto referente ao item 4.3.2.Se houver, porém, uma equivalência de todos os caracteres, o fluxo do processo continua normalmente.

Antes de partir para o ponto de liberação da linha, o nome de usuário(*login*) referente à senha digitada, é armazenado em uma variável para posterior gravação no arquivo de *log* no cartão de memória.

Libera linha

A partir do momento que todo o fluxo se deu de forma correta até aqui, deve-se liberar a linha para que o usuário possa realizar sua chamada. Conforme mencionado na seção 3.2.3, para que a central entenda um desejo de ligação, o procedimento deve ser um bloqueio(abertura do *relay*) seguido de uma liberação(fechamento do *relay*) da linha. Este perído de tempos foi obtido, experimentalmente, e é de, aproximadamente, 800(oitocentos) milisegundos.

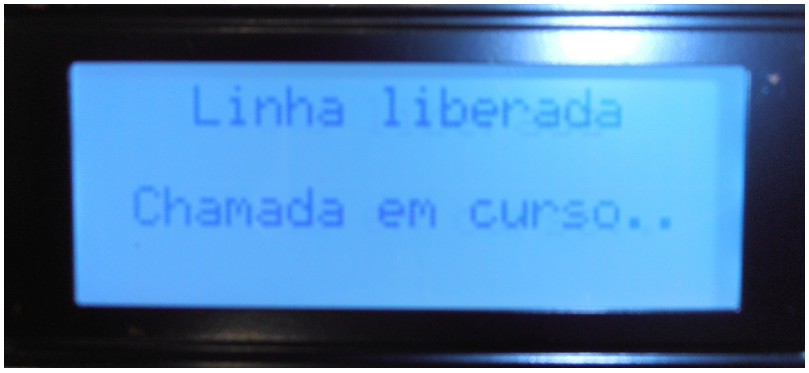


Figura 24: Tela de linha liberada

Dessa forma o usuário receberá o tom de discagem e estará pronto para digitar o número com o qual deseja estabeler conexão.

Recebe número telefônico discado

Linha com acesso liberado e sistema pronto para receber e depois armazenar o número telefônico discado.

Neste ponto o *software* esperará o pressionamento da tecla asterisco, novamente, para entender que é o fim do número discado. Isto se faz necessário pois os números diferem de tamanho de acordo com o tipo de ligação realizada, seja esta interurbana ou local. Portanto, se o usuário deseja ligar para o número local 34135589, após a tecla nove, deve pressionar a tecla asterisco para que o sistema dê continuidade ao fluxo.

Dispara temporizador

Neste ponto deve-se disparar um temporizador para controlar o tempo despendido durante a ligação.

Guarda dados da ligação

Depois que um número foi discado, deve-se memorizar as variáveis referentes à ligação(data, hora, usuário, número discado) para posterior armazenamento na memória SD.

Tela de ligação em curso

Esta tela do sistema é bem semelhante à mencionada no item 4.3.2. Consiste, além da mostragem de data e hora a contagem do tempo de ligação para que o usuário esteja ciente deste.

Figura 25: Tela de ligação em curso

Enquanto a chamada não for finalizada, esta é a tela que ficará sendo mostrada.

Chamada terminada?

O único evento que carcteriza finalização da chamda neste momento é o fone ser colocado no gancho. Quando isto acontece, uma interrupção é gerada na porta do microcontrolador e este entende que deve prosseguir para o último passo a ser realizado no sistema.

Grava dados de ligação no cartão de memória

Para finalizar todo o processo e fluxo deve-se agora obter todas as informações recebidas e armazená-las no cartão de memória.

O arquivo de log presente na memória é do formato csv (*Comma Separated Values*) e tem nome "log.csv". Este formato foi escolhido devido à simplicidade de manuseá-lo tanto por parte do usuário como por parte do desenvolvedor do software que deve ater-se à construção do *frame* de forma correta neste arquivo.

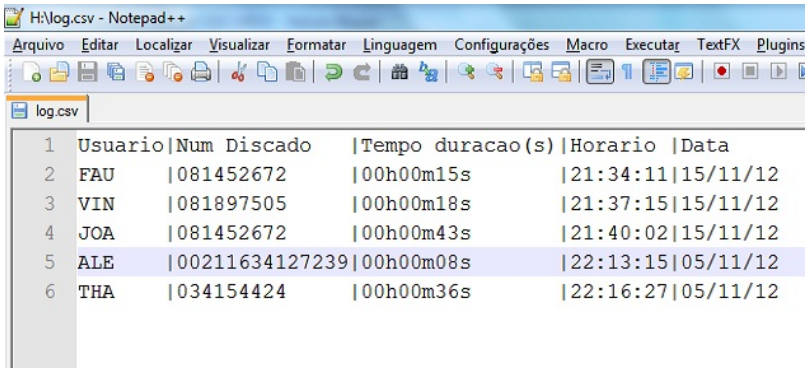


Figura 26: Arquivo de log do sistema - log.csv

Por meio da análise da **Figura** 26, pode-se perceber facilmente a estrutura adotada para construção do *frame* referente ao registro de cada ligação.

Os valores das variáveis(Usuário, Número Discado, Tempo de ligação, Horário, Data) são separados pelo carctere '|'. Dessa forma, ao utilizar *softwares* de leitura deste tipo de arquivo, como Excel, estes valores serão separados em colunas de acordo com o carctere mencionado.

Sendo assim, fica bastante fácil e prático a análise deste log, permitindo buscas filtradas por qualquer especificação.

Após a gravação do *frame* , como explicitado anteriormente, o sistema simplesmente retornará à posição de espera, referente ao item 4.3.2, finalizando, entao, o loop geral.

5 *Testes e resultados de validação*

Neste capítulo serão descritos todos os testes realizados para a consequente validação do projeto. A apresentação se dará de forma pontual, item a item, especificando os processos mais importantes.

5.1 Testes de segurança

Nesta parte serão explicitados testes relacionados à segurança do sistema, isto é, à vulnerabilidade deste a determinadas ações que possam, por ventura, burlar seu fluxo.

Em geral estes testes visam garantir a não utilização do aparelho por indivíduos não cadastros e também, a não obtenção de sucesso em chamadas que possam ser concluídas e não registradas.

Telas de ajuste de Horário e Data

Testar um período de tempo maior que quinze segundos para digitar uma variável necessária para o ajuste, seja ela dia, mês, ano, hora ou minuto.

Para cada caso deve-se esperar que uma mensagem de "Tempo esgotado"seja mostrada e o sistema reinicie, voltando ao início da configuração.

Tela de *stand-by*

Testar o pressionamento de outras teclas, diferentes de '*'(asterisco). Como estas teclas, neste ponto do fluxo, não sao permitidas, deve-se esperar que o sistema não execute nenhuma ação além da referente ao bloqueio e desbloqueio da linha.

Tela chamada em curso

Testar uma demora maior que quinze segundos para digitar o número completo a ser telefonado, isto é, não pressionar '*'(asterisco). Caso isto ocorra, o sistema deve bloquear a linha e reiniciar o fluxo, fazendo com que o usuário tenha que digitar o número novamente.

Verificação de senhas

Testar cinco diferentes tipos de tentativas de acesso com senhas inexistentes:

- Senha com nenhum caractere correto
- Senha com somente um caractere correto
- Senha com dois caracteres corretos
- Senha com todos três caracteres corretos
- Senha em que cada caractere, individualmente, seja igual a um único caractere de uma outra senha existente

À exceção do último item listado , todos outros casos devem resultar em insucesso e, dessa forma, a não continuidade do fluxo do programa.

Vale ressaltar a existência de um problema devido à presença de dois usuários distintos com mesma senha. Caso isto seja verificado, o sistema indicará, sempre, o primeiro usuário da lista como realizador da chamada. Dessa forma, é totalmente recomendado que cada usuário tenha sua própria e única senha, diferente das demais.

5.2 Testes de funcionamento geral

Nesta parte serão listados alguns testes pertinentes à verificação do fluxo e funcionamento do programa como um todo, sem se ater a problemas específicos de segurança e afins.

Receber uma ligação

Testar o recebimento de uma chamada e consequente possibilidade de comunicação com o outro usuário que está do "outro lado da linha".

Chamada com duração curta

Testar a realização de uma ligação telefônica que dure menos de quinze segundos. Esta não deve ser levada em consideração pelo sistema devido a situações como "chamar, chamar e ninguém atender", desistência por parte do usuário em realizar tal ligação. Dessa forma, nenhum registro deverá ser adicionado ao arquivo de *log*.

Chamada com duração relevante

Diferentemente do item 5.2, testar a realização de chamadas com durações relevantes, ou seja, maiores que 15 segundos.

Fazer ligações para números distintos considerando ligações interurbanas, locais . Todas devem ser tratadas de forma igual e devem encontrar-se registradas no arquivo de *log*.

Qualidade da conexão

Testar a qualidade da conexão tomando como base ruídos, volume de ligação tanto em chamadas recebidas quanto nas realizadas.

Interrupção do fluxo por reposição do fone

Testar o interrompimento do fluxo colocando o fone no gancho em diferentes pontos do fluxo do sistema. Salvos à excessão o período de configuração e a tela de *stand-by* do sistema, em todos outros pontos como tela de espera de senha, tela de chamada em curso, a reposição, por parte do usuário, do fone ao gancho, deve interromper o fluxo e retornar à tela de *stand-by* do sistema, reconhecendo dessa maneira uma desistência do usuário em realizar a chamada.

5.3 Discussão e resultados

Considerando-se todos os testes abordados anteriormente o sucesso na realização destes foi obtido.

No que diz respeito à interação com o usuário o sistema se mostrou totalmente intuitivo e simples. Todas as telas existentes para a interface possuem mensagens claras e objetivas resultando em fácil compreensão e atingindo assim, um dos objetivos do projeto , que é a facilidade de manuseio do produto.

Com relação à segurança, em nenhum momento conseguiu-se *burlar* o sistema quanto ao acesso de pessoas não cadastradas, porém para a realização de ligações não registradas isso sim ocorreu. Esta última restrição pode ser *burlada* pelo fato de haver uma espera de quinze segundos para concretização de determinadas ações como ajuste de horário, ajuste de data. Nestes momentos, pelo fato de a linha se encontrar liberada, é sim possível a ligação para números que não serão registrados, porém, na pior das hipóteses essas ligações não durarão mais do que quinze segundos, o que não acarreta problemas para o

funcionamento do projeto.

Por último , em relação ao fluxo geral do processo o sistema também se mostra bastante robusto. Um único detalhe é com relação à gravação dos dados no cartão de memória. Devido à variação da quantidade de dígitos de um número telefônico, diferindo devido aos tipos de ligação existentes(interurbanas, locais) e também à uma má administração da memória do microcontrolador por parte do compilador, observou-se certos casos de superposição de valores das variáveis envolvidas no registro, como pode ser notado na **Figura** 27.

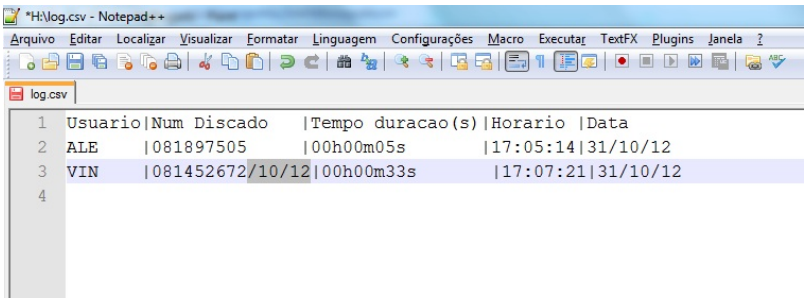


Figura 27: Arquivo de *log* com registro danificado

No segundo campo do segundo resgistro nota-se, em destaque, que pedaço do valor da data do sistema foi concatenado ao valor da variável que armazena os dígitos do telefone, situação totalmente indesejada para o projeto.

Para contonar este problema bastou uma simples inicialização da variável que armazena os dígitos. Dessa forma as gravações se procederam de forma correta sem nenhum problema adicional.

6 Conclusões e trabalhos futuros

6.1 Conclusões

A proposta deste trabalho é desenvolver um dispositivo que permitisse a gerência e análise de chamadas telefônicas realizadas em um aparelho por meio do registro dessas ligações. Os resultados obtidos por meio do desenvolvimento e dos testes realizados mostraram que o projeto foi desenvolvido com sucesso e que pode-se pensar em uma outra etapa que seria a de confecção do produto propriamente dito.

Tendo em vista que os testes foram realizados apenas com protótipos, é necessário enfatizar a necessidade de novas baterias de testes para o produto final, abordagem essa que não foi feita neste trabalho.

No âmbito pessoal, este projeto foi de grande valia para o autor contribuindo para o aprimoramanto de seus conhecimentos na área de projeto e desenvolvimento de sistemas embarcados, considerando-se tanto a parte de *hardware* como a parte de *software*. Além da experiência em desenvolvimento adquirida, foram também acrescentados ao conhecimento do autor, novos conceitos relacionados a telefonia e também no que diz respeito à tecnologia da informação, que é hoje um ramo bastante explorado e em grande crescimento.

6.2 Dificuldades encontradas

Durante todo o período de desenvolvimento do projeto alguns obstáculos foram encontrados e, dentre todos, tem-se alguns principais:

Biblioteca de comunicação SPI

As bibliotecas utilizadas na implementação do *softwares* para comunicação SPI existentes e disponíveis gratuitamentes possuem alguns erros. Um bom tempo foi despendido para o entendimento e busca de solução destes pequenos erros, buscas estas realizadas

primordialmente em *blogs* e fóruns relacionados ao assunto. A seção B.5 presente no **Apêndice** B retrata esta dificuldade mediante a necessidade de implementação de uma nova rotina para a biblioteca.

Uma abordagem estremamente válida também, é a preferência por utilizar-se uma biblioteca que promove essa comunicação SPI sem a utilização do módulo SPI já existente no microcontrolador, sendo emulada por *software*. Apesar de esta ser menos robusta o projeto não fica *hardware* restrito à pinagem específica do PIC.

Compilador CCS

O compilador CCS se mostrou muito vantajoso por apresentar uma linguagem simples, se encarregando de vários procedimentos relacionados a configurações de registradores, interrupções, de maneira automática, sem que fosse necessário preocupar com registradores e *flags* de um modo geral. Porém, vale ressaltar a limitação que *softwares* deste tipo possuem no que diz respeito à programação para microcontroladores. Quando o assunto é administração de memória RAM, por exemplo, alguns tipos de variáveis, como *struct* devem ser evitados pois a execução do programa, algumas vezes, não se dá de forma correta.

6.3 Trabalhos futuros

Como já abordado anteriormente, uma implementação futura a curto prazo é a confecção do protótipo final considerando-se todas características, inclusive físicas, como tamanho, peso, design de envoltório para armazenamento,etc.

Abordando o tema de melhorias e inovação, em se tratando de um projeto de sistema embarcado e da utilização de um microcontrolador tecnologicamente avançado, a gama de inovações futuras é enorme. Dentre todas, destacam-se:

- **Aplicativo de supervisão**

  Os dados registrados no cartão de memória, para que sejam manipulados, deve-se utilizar *softwares* de edição de arquivos já existentes como *Excel, Notepad++*. Isto torna o produto, como um todo, sem identidade e fidelidade com o usuário.

  Portanto, como aplicaçao futura, pode-se desenvolver um *software* que possua recursos para tratamento desses dados de maneira mais específca, abordando funcionalidades como geração de relatórios, tempo médio gasto ao telefone, número mais

discado, entre outras.

- **Tarifação**

  O projeto, assim como o é hoje, apenas registra e não aborda o quesito da tarifação das chamadas telefônicas. Um estudo deve ser realizado no sistema de tarifação utilizado pelas operadoras e regulamentado pela ANATEL(*Agência Nacional de Telecomunicações*) e , dessa forma, esta funcionalidade pode ser acrecentada ao produto, agregando um valor muito grande, pois, além da independência quase que total para com as operadoras, o usuário poderá fazer, também, cálculos e controles de custos.

- **Comunicação USB(*Universal Serial Bus*)**

  O microcontrolador PIC18F4550 já possui interface para comunicação USB. Sendo assim, pode-se usufruir desta característica por meio da implementação de *softwares* de configuração do sistema, podendo, por exemplo, gravar usuários e senhas diretamente na EEPROM, sem a necessidade desses dados estarem presentes no cartão de memória.

- **Melhorias de *hardware/software***

  - **RTC(*Real Time Clock*)**

    Para a temporização e contagem do tempo do relógio do sistema utilizou-se o *timer* do próprio microcontrolador. Este se dá de forma correta porém possui a desvantagem de ser dependente da energia elétrica do sistema. Caso haja uma queda, os valores do tempo são perdidos e o usuário deve fazer ajuste novamente.

    Com a utilização de componentes externos, os chamados RTC, uma contagem de tempo real mais robusta pode ser implementada. Estes componentes são facilmente encontrados no mercado e sua utilização é bastante simples.

  - **Reconhecimento de informações recebidas da central**

    A central telefônica disponibliza algumas informações na linha telefônica durante uma ligação. Um exemplo típico é o número de telefone que está ligando para o aparelho.

    Portanto um estudo desse tipo de informação deve ser abordado e sua implementação feita agragando ainda mais valor ao projeto.

Dessa forma, ligações à cobrar podem ser levadas em consideração no momento do registro de chamdas.

- **Codificação e cadastro de senhas e usuários**

  Da maneira como está implementado o projeto não oferece nenhuma proteção com relação às senhas e usuários do sistema. Para evitar possíveis falhas deve-se implementar um método de codificação destes dados para que somente o sistema reconheça somente caracteres que possuam esta codificação.

  Além disso, deve-se implementar um cadastro ou mudança de senhas e usuários através do próprio protótipo, utilizando o teclado do telefone, sem a necessidade de um computador.

Referências

[1] Miyadaira, A. N. Microcontroladores PIC18 - Aprenda e Programe em Linguagem C.Érica, 2009.

[2] MICROCHIP. PIC18F2455/2550/4455/4550Datasheet.2004.

[3] MITEL. MT8870D/MT8870D-1Datasheet.1995.

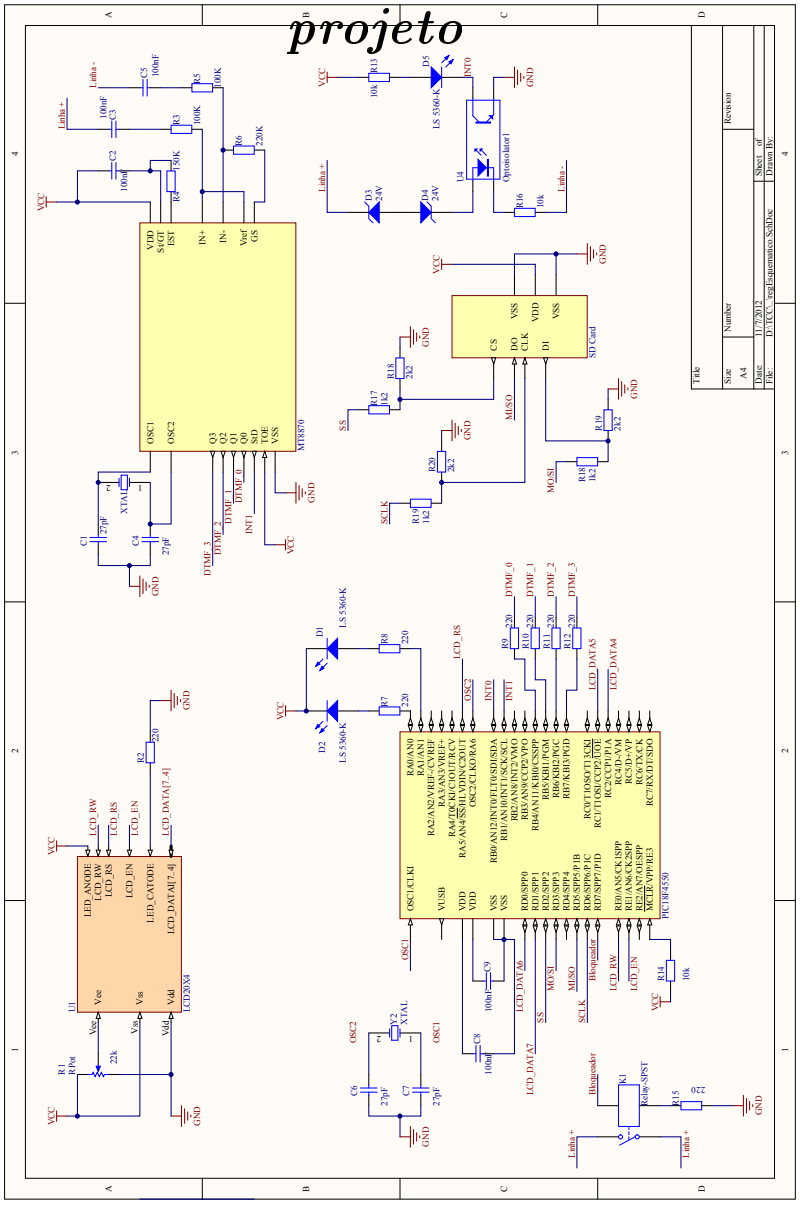
[4] NETO, V. S. ; CARVALHO, F. T. A. Tecnologia de Centrais telefônicas.Érica, 1999.

[5] SPI Bus interface. ; Disponível em http://www.eeherald.com/section/design-guide/esmod12.html. Acesso em : 06/09/2012.

[6] ALENCAR, M. S. Telefonia Digital. Érica, 1999.

[7] Site Oficial da Microchip Technology Inc. Disponível em http://www.microchip.com. Acessado em: 28/08/2012.

APÊNDICE A – Desenho esquemático do projeto



APÊNDICE B – Código fonte do sistema

B.1 Arquivo 'main.c'

```
//#device PIC18f4550 *=16
#include <18F4550.h>
#device CCS4
#device *=16
#device PASS_STRINGS = IN_RAM


#FUSES NOWDT                    //No Watch Dog Timer
#FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
#FUSES PLL1                     //Divide By 12(48MHz oscillator input)
#FUSES CPUDIV4                  //System Clock uses external crystal
#FUSES USBDIV                   //USB clock source comes from PLL divide by 2
#FUSES HSPLL                      //Crystal osc <= 4mhz for PCM/PCH , 3mhz to 10 mhz for ↩
    PCD
#FUSES FCMEN                    //Fail-safe clock monitor enabled
#FUSES IESO                     //Internal External Switch Over mode enabled
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOBROWNOUT               //No brownout reset
#FUSES BORV20                   //Brownout reset at 2.0V
#FUSES VREGEN                   //USB voltage regulator enabled
#FUSES PBADEN                   //PORTB pins are configured as analog input channels on ↩
    RESET
#FUSES LPT1OSC                  //Timer1 configured for low-power operation
#FUSES MCLR                     //Master Clear pin enabled
#FUSES STVREN                   //Stack full/underflow will cause reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for↩
     I/O
#FUSES ICPRT                    //ICPRT enabled
#FUSES NOXINST                  //Extended set extension and Indexed Addressing mode ↩
    disabled (Legacy mode)
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NOCPB                    //No Boot Block code protection
#FUSES NOCPD                    //No EE protection
#FUSES NOWRT                    //Program memory not write protected
#FUSES NOWRTC                   //configuration not registers write protected
#FUSES NOWRTB                   //Boot block not write protected
```

```
#FUSES NOWRTD                   //Data EEPROM not write protected
#FUSES NOEBTR                   //Memory not protected from table reads
#FUSES NOEBTRB                  //Boot block not protected from table reads

#use delay(clock = 24000000)


#include <math.h>
#include <VariaveisGlobais.h>
#include "lcd_plus.h"
#include "mmc_spi.h"
#include "stdlib.h"
#include "string.h"
#include "stdio.h"
#include "fat_mmc_spi.h"
#include "funcoes.h"
#include "Interrupcoes.h"

#ZERO_RAM
void main (void){




   InicializaSistema();


   while(1){
      switch(flag_tela){
         case 1:
            tela1_standby();
            break;
         case 2:
            tela2_senha();
            break;
         case 3:
            tela3_ligacao();
            break;
         case 4:
            tela4_grava_sd();
            break;
         default:
            tela1_standby();
            break;
      }

   }
}
```

B.2 Arquivo 'VariaveisGlobais.h'

```
//Variaveis Globais
int senha_digitada[3];

int16 lig_duracao = 0;
int16 lig_duracao_h = 0;
int16 lig_duracao_m = 0;
int16 lig_duracao_s = 0;

int16 lig_hora = 0;
int16 lig_minuto = 0;
int16 lig_segundo = 0;
int16 lig_dia = 0;
int16 lig_mes = 0;
int16 lig_ano = 0;

int num_teclado;
int aux_fator10 = 0;

int16 segundo = 0;
int16 minuto = 0;
int16 hora = 0;
int16 dia = 0;
int16 mes = 0;
int16 ano = 0;

int cont_senha = 0;
int cont_num_discado = 0;
int cont_horario = 0;
int cont_data = 0;
int conta = 0;
int glb_tot_users = 0;
int user_indice = 0;
int lig_num_chamado[13];
int16 cont_timer2 = 0;

int flag_tela = 0;

SHORT INT flag_ajuste_horario = 0;
SHORT INT flag_ajuste_data = 0;
SHORT INT flag_senha = 0;

SHORT INT flag_fone_gancho = 0;
SHORT INT flag_ligacao = 0;

SHORT INT flag_hora = 0;
SHORT INT flag_minuto = 0;
SHORT INT flag_realizar_chamada=0;
SHORT INT flag_bloqueia = 0;
SHORT INT flag_dia = 0;
SHORT INT flag_mes = 0;
```

```
SHORT INT flag_ano = 0;
SHORT INT flag_lig_valida = 0;

#define led1 PIN_A0
#define led2 PIN_A1
#define blq_linha output_low(PIN_D7)
#define dsblq_linha output_high(PIN_D7)

#define LCD_DB4   PIN_C1
#define LCD_DB5   PIN_C2
#define LCD_DB6   PIN_D0
#define LCD_DB7   PIN_D1
#define LCD_RS    PIN_A5
#define LCD_RW    PIN_E0
#define LCD_E     PIN_E1

#define MMC_CLK   PIN_D6
#define MMC_DI    PIN_D5
#define MMC_DO    PIN_D3
#define MMC_CS    PIN_D2
```

B.3 Arquivo 'Interrupcoes.h'

```
#int_timer1
void timer1(void) {

   static int conta;

   set_timer1(0x56AD + get_timer1()); //20 MHz
   conta++;
   if (conta == 25) { // Se estourou 25 vezes = 1 segundo
   segundo++;
   conta = 0;
   atualiza_horario();
   }

}

#int_timer2
void timer2(void) {

   static int16 contador = 0 ;

   contador ++;

   if(contador>=1000){
      contador = 0;
      cont_timer2 ++;
   }
}

#int_EXT

void  fone_gancho(void)
{


   clear_interrupt(INT_EXT);
   clear_interrupt(INT_EXT1);
// clear_interrupt(INT_TIMER1);
// clear_interrupt(INT_TIMER2);

   disable_interrupts(INT_EXT);
   disable_interrupts(INT_EXT1);
   disable_interrupts(INT_TIMER1);
   disable_interrupts(INT_TIMER2);

   flag_fone_gancho = 1;

   output_high(led2);
   delay_ms(20);
   output_low(led2);
```

```
    enable_interrupts(INT_EXT);
    enable_interrupts(INT_EXT1);
    enable_interrupts(INT_TIMER1);
    enable_interrupts(INT_TIMER2);

}

#int_EXT1

void  dtmf(void)
{

    clear_interrupt(INT_EXT);
    clear_interrupt(INT_EXT1);

    disable_interrupts(INT_EXT);
    disable_interrupts(INT_EXT1);
    disable_interrupts(INT_TIMER1);
    disable_interrupts(INT_TIMER2);

    if(flag_ajuste_horario){
        if(!ajuste_horario()){
            flag_ajuste_horario = 0;
        }
    }
    if(flag_ajuste_data){
        if(!ajuste_data()){
            flag_ajuste_data = 0;
        }
    }
    if(flag_senha){
        if(!ler_senha_digitada()){
            flag_senha = 0;
        }
    }
    if(flag_ligacao){
        if(!ler_num_discado()){
            flag_ligacao = 0;
        }
    }
    if(flag_realizar_chamada){
        ler_portb();
        if(num_teclado == 11){
            flag_realizar_chamada = 0;
        }else{
            flag_bloqueia = 1;
        }
    }

    enable_interrupts(INT_EXT);
    enable_interrupts(INT_EXT1);
    enable_interrupts(INT_TIMER1);
    enable_interrupts(INT_TIMER2);
```

```
}
```

B.4 Arquivo 'Funcoes.h'

```
//Funções utilizadas

void ler_portb(void){

      int valor_portb = 0;
      valor_portb = input_b();

      num_teclado = (valor_portb)/16;

      if(num_teclado == 10){
         num_teclado = 0;
      }
      output_high(led2);
      delay_ms(20);
      output_low(led2);
}

int read_file_senha(){

   FILE stream;
   char fname[10];
   int i = 0;
   int j = 0;
   int l = 0;
   int linha = 0;
   int aux_int = 0;
   char str[160];

// for(i=0; i < 300 ; i++){
//    str[i] = "";
// }
   fname = "/users.txt";

   if(fatopen(fname, "r", &stream) != GOODEC)
   {
      return 1;
   }

   while(1){
      str[i] = fatgetc(&stream);
//    printf(lcd_putc,"\f%c",str[i]);
//    delay_ms(350);
      if(str[i] == '|'){

         for(j = i-3; j < i; j += 1){

            write_eeprom(i+l,str[j]);

            delay_ms(10);
            l++;
```

```
            }
            l = 0;
        }

        if(str[i] == '\r'){

            for(j = i-3; j < i; j += 1){

                aux_int = (int)(str[j]) - (int)(0x30);

                write_eeprom(i+l, aux_int);
                delay_ms(10);

                l++;
            }
            l = 0;


        }
        if(str[i] == '\n'){
            linha++;
            i--;
        }
        if((str[i] == -1)){
            if(linha<1){
                linha++;
            }
            break;
        }
        i++;
    }

    glb_tot_users = linha;

    if(fatclose(&stream) != GOODEC)
    {
        return 1;
    }

    return 0;
}

int AppendFile( char *appendstring)
{
//    int i = 0;
    FILE stream;
    char fname[10];

    fname = "/log.csv";


    if(fatopen(fname, "a", &stream) != GOODEC)
    {
        printf(lcd_putc,"file not found");
```

```
            delay_ms(500);
        return 1;
    }

    if(fatputs(appendstring, &stream)){

        return 1;
    };


    if(fatclose(&stream) != GOODEC)
    {
      return 1;
    }

    return 0;
}

void atualiza_data(void){

    switch(dia){
        case 29:
            if((mes % 4)!=0){
                mes++;
                dia = 1;
            }
            break;
        case 30:
            if(mes == 2){
                mes++;
                dia = 1;
            }
            break;
        case 31:
            if((mes == 4)||(mes == 6)||(mes == 9)||(mes == 11)){
                mes++;
                dia = 1;
            }
            break;
        case 32:
            if((mes != 2)||(mes != 4)||(mes != 6)||(mes != 9)||(mes != 11)){
                mes++;
                dia = 1;
            }
        default:
            break;
    }
    if(mes>=13){
        ano++;
        mes = 1;
    }


}
```

```
int ajuste_data(){

   ler_portb();
   cont_data++;
   if(cont_data == 2){
      flag_dia = 1;
      num_teclado += aux_fator10;
   }else{
      if(cont_data == 4){
         num_teclado += aux_fator10;
         flag_mes = 1;

      }else{
         if(cont_data == 6){
            num_teclado += aux_fator10;
            flag_ano = 1;
            cont_data = 0;
            return 0;
         }else{
            aux_fator10 = 10 * num_teclado;
         }
      }
   }
   return 1;
}

void atualiza_horario(void){

      if (segundo==60){ // testes para cronometrar

         segundo=0;
         minuto++;

      }
      if (minuto==60){

         minuto=0;
         hora++;
      }
      if (hora==24){

         hora=0;
         dia++;
         atualiza_data();
      }
}

int ajuste_horario(){

   ler_portb();
   cont_horario++;
   if(cont_horario == 2){
      flag_hora = 1;
```

```
      num_teclado += aux_fator10;
   }else{
      if(cont_horario == 4){
         num_teclado += aux_fator10;
         flag_minuto = 1;
         cont_horario = 0;
         return 0;
      }else{
         aux_fator10 = 10 * num_teclado;
      }
   }
   return 1;
}
void tela_ajuste_data(void){

ajus_data1:

   dsblq_linha;
   flag_dia = 0;
   flag_ajuste_data = 1;
   num_teclado = 0;
   cont_data = 0;
   flag_mes = 0;
   flag_dia = 0;
   flag_ano = 0;

   printf(lcd_putc,"\f    Ajuste de Data    ");
   delay_ms(50);
   printf(lcd_putc,"\n       Dia: ");
   delay_ms(50);

   cont_timer2 = 0;
   while(flag_dia == 0){
      enable_interrupts(INT_TIMER2);
      if(cont_timer2>10){
         lcd_gotoxy(1,2);
         printf(lcd_putc, "    TEMPO ESGOTADO    ");
         blq_linha;
         delay_ms(800);
         dsblq_linha;
         cont_timer2 =0;
         goto ajus_data1;
      }
   }


   if(num_teclado>31   || num_teclado==0){
      lcd_gotoxy(1,2);
      printf(lcd_putc, "    Valor invalido    ");
      delay_ms(400);
      goto ajus_data1;
   }

   dia = num_teclado ;
```

```
   printf(lcd_putc,"%Lu",dia);
   delay_ms(50);

ajus_mes:

   flag_dia = 0;

   num_teclado = 0;

   lcd_gotoxy(1,3);
   printf(lcd_putc,"       Mes: ");
   delay_ms(50);

   cont_timer2 = 0;

   flag_mes = 0;
   while(flag_mes == 0){
      enable_interrupts(INT_TIMER2);
      if(cont_timer2>10){
         lcd_gotoxy(1,3);
         printf(lcd_putc, "   TEMPO ESGOTADO    ");
         blq_linha;
         delay_ms(800);
         dsblq_linha;
         cont_timer2 =0;
         lcd_gotoxy(1,3);
         printf(lcd_putc, "                     ");
         goto ajus_data1;
      }
   }

   if(num_teclado>12 || num_teclado==0){
      lcd_gotoxy(1,3);
      printf(lcd_putc, "   Valor invalido    ");
      delay_ms(400);
      lcd_gotoxy(1,3);
      printf(lcd_putc, "                     ");
      goto ajus_data1;
   }

   mes = num_teclado ;
   printf(lcd_putc,"%Lu",mes);
   delay_ms(50);

ajus_ano:

   flag_mes = 0;

   num_teclado = 0;

   lcd_gotoxy(1,4);
   printf(lcd_putc,"       Ano.: ");
   delay_ms(50);
```

```
   cont_timer2 = 0;

   flag_ano = 0;
   while(flag_ano == 0){
      enable_interrupts(INT_TIMER2);
      if(cont_timer2>10){
         lcd_gotoxy(1,4);
         printf(lcd_putc, "   TEMPO ESGOTADO   ");
         blq_linha;
         delay_ms(800);
         dsblq_linha;
         cont_timer2 =0;
         lcd_gotoxy(1,4);
         printf(lcd_putc, "                    ");
         goto ajus_data1;
      }
   }

   if(num_teclado==0){
      lcd_gotoxy(1,4);
      printf(lcd_putc, "   Valor invalido   ");
      delay_ms(400);
      lcd_gotoxy(1,4);
      printf(lcd_putc, "                    ");
      goto ajus_data1;
   }

   flag_ano = 0;
   ano = num_teclado ;
   num_teclado = 0;
   printf(lcd_putc,"%Lu",ano);
   delay_ms(500);
   printf(lcd_putc,"\f\n    Data ajustada   ");
   delay_ms(400);



}

void tela_ajuste_horario(void){

ajus_horario1:
   dsblq_linha;
   flag_hora = 0;
   flag_ajuste_horario = 1;
   cont_horario = 0;
   flag_hora = 0;
   flag_minuto = 0;
   num_teclado = 0;

   printf(lcd_putc,"\f  Ajuste de Horario ");
   delay_ms(100);
   printf(lcd_putc,"\n     Horas.: ");
   delay_ms(100);
```

```
   enable_interrupts(INT_TIMER2);

   flag_hora = 0;
   while(flag_hora == 0){

      if(cont_timer2>10){
         lcd_gotoxy(1,2);
         printf(lcd_putc, "   TEMPO ESGOTADO   ");
         blq_linha;
         delay_ms(800);
         dsblq_linha;
         cont_timer2 =0;
         goto ajus_horario1;
      }
   }

   if(num_teclado>23){
      lcd_gotoxy(1,2);
      printf(lcd_putc, "   Valor invalido   ");
      delay_ms(400);
      goto ajus_horario1;
   }




   hora = num_teclado ;


   printf(lcd_putc,"%Lu",hora);

ajus_minuto:

   delay_ms(50);
   flag_hora = 0;
   num_teclado = 0;

   lcd_gotoxy(1,3);
   printf(lcd_putc,"    Minutos.: ");
   delay_ms(50);

   cont_timer2 = 0;
   //enable_interrupts(INT_TIMER2);

   flag_minuto = 0;
   while(flag_minuto == 0){

      if(cont_timer2>10){
         lcd_gotoxy(1,3);
         printf(lcd_putc, "   TEMPO ESGOTADO   ");
         blq_linha;
         delay_ms(800);
         dsblq_linha;
```

```
         cont_timer2 =0;
         lcd_gotoxy(1,3);
         printf(lcd_putc, "                    ");
         goto ajus_horario1;
      }
   }

   if(num_teclado>59){
      lcd_gotoxy(1,3);
      printf(lcd_putc, "   Valor invalido   ");
      delay_ms(400);
      lcd_gotoxy(1,3);
      printf(lcd_putc, "                    ");
      goto ajus_horario1;
   }

   flag_minuto = 0;
   minuto = num_teclado ;
   num_teclado = 0;

   printf(lcd_putc,"%Lu",minuto);
   delay_ms(500);

   printf(lcd_putc,"\f\n  Horario ajustado");
   delay_ms(500);
   //printf(lcd_putc,"\f");

   blq_linha;
}

int SD_init(){

   int8 i = 0;
   int8 f = 1;
   setup_spi(FALSE);

   printf(lcd_putc,"\fInicializando SD..");
   delay_ms(550);

   while(f){
         delay_ms(150);

         f = fat_init();
      i++;
      delay_ms(150);
      if(i>=50){
         return 1;
      }
   }

   printf(lcd_putc,"\fSD OK..");
   delay_ms(300);
```

```
   return 0;
}
void tela1_standby(void){

   flag_bloqueia = 0;
   flag_realizar_chamada = 1;
   clear_interrupt(INT_EXT);
   disable_interrupts(INT_EXT);
   ext_int_edge( 0, L_TO_H);
   delay_ms(200);
   enable_interrupts(INT_EXT);

   delay_ms(150);

   printf(lcd_putc,"\f(*) --> Telefonar \n");
   while(flag_realizar_chamada == 1){
      if(flag_bloqueia){
         flag_bloqueia = 0;
         blq_linha;
         delay_ms(800);
         dsblq_linha;

      }

      lcd_gotoxy(1,3);
      printf(lcd_putc,"   Data: %2Lu/%2Lu/%2Lu   ",dia,mes,ano);
      lcd_gotoxy(1,4);
      printf(lcd_putc,"   Hora: %2Lu:%2Lu:%2Lu   ",hora,minuto,segundo);

   }

   flag_tela = 2; //vai para tela esperar senha ser digitada

}

int valida_senha(){

   int i = 0;
   int j = 0;
   int qtd_igual = 0;
   char eeprom_r;
   int add_rom = 0;
   for(i = 0; i<glb_tot_users ; i++){
   add_rom = (int)(8*i + 7);


      for(j = 0; j<3 ; j++){

         eeprom_r = read_eeprom(add_rom);

         if(senha_digitada[j] == eeprom_r){
            qtd_igual++;

         }
```

```
         add_rom++;
      }

      if(qtd_igual == 3){
         user_indice = (i);

         return 0;
      }else{
         qtd_igual = 0;

      }
      if((qtd_igual == 0)&&(i == (glb_tot_users -1))){
         return 1;
      }
   }
}

void tela2_senha(void){

   ext_int_edge(0 , H_TO_L);

   flag_senha = 1;
   printf(lcd_putc,"\f\n   Digite a senha   \n");

   flag_fone_gancho = 0;
   while(flag_senha && (flag_fone_gancho == 0)){}
   if(flag_senha == 0){


      if(valida_senha()){
         printf(lcd_putc,"\f\n  SENHA INCORRETA  ");
         delay_ms(600);
         flag_tela = 1; // volta pra tela do standby
      }else{
         printf(lcd_putc,"\f\n   Senha correta    ");
         delay_ms(250);
         flag_tela = 3;//vai pra tela e libera ligação
      }
   }else{
      if(flag_fone_gancho){
         flag_tela = 1;
         flag_fone_gancho = 0;
         flag_senha = 0;
      }
   }

}

void tempo_duracao(){
   static int16 aux =0;

   if(lig_duracao>=3600){

      lig_duracao_h = (int16)(floor(lig_duracao/3600));
```

```
      aux = (int16)(lig_duracao % 3600);
      lig_duracao = (int16)(aux);
   }

   if(lig_duracao>=60){

      lig_duracao_m = (int16)(floor(lig_duracao/60));
      aux = (int16)(lig_duracao % 60);
      lig_duracao =(int16)(aux);

   }
   if(lig_duracao<60){

         lig_duracao_s = lig_duracao;
      }
   }


int ler_senha_digitada(){

   ler_portb();
   senha_digitada[cont_senha] = (int)(num_teclado);

   cont_senha++;

   if(cont_senha>=3){
      cont_senha = 0;
      return 0;
   }else{
      return 1;
   }
}

void tela4_grava_sd(){


   int i,l = 0;
   int j = 2;
   int contador = 0;
   int indice = 0;
   int aux = 0;
   int add_rom = 0;
   char string[30] = "";
   char aux_string[5] = "";
   char eeprom_r = "";

   for(i=0; i < 30 ; i++){
      string[i] = "";
   }

   if(flag_lig_valida){

      printf(lcd_putc,"\f\nRegistrando chamada");
```

```
clear_interrupt(INT_EXT);
clear_interrupt(INT_EXT1);
clear_interrupt(INT_TIMER2);

disable_interrupts(int_ext);
disable_interrupts(int_ext1);
disable_interrupts(int_timer2);

blq_linha;

//comeca a escrever no arquivo

//usuario
string[l]="\r";
l++;
string[l]="\n";
l++;

indice = user_indice;

add_rom = (int)(8*indice + 3);

for(i=0; i<3; i++){

   eeprom_r = read_eeprom(add_rom);
   string[l] = eeprom_r;
   add_rom++;
   l++;
}

string[l] = "|";
l++;

//numero discado
contador = cont_num_discado;
cont_num_discado = 0;


for(i=0; i < contador ; i++){

   aux = ((int)(lig_num_chamado[i])  + (int)(0x30));

   string[l] = (char)(aux);

   delay_ms(350);
   l++;

}

string[l] = "|";
l++;

//calculo do tempo de duracao
```

```
      tempo_duracao();

      //Horas
      aux_string = "";
      j=2;

      if(lig_duracao_h<10){
         j = 1;
         string[l] = "0";
         l++;
      }
      itoa(lig_duracao_h, 10, aux_string);

      for(i=0; i < j ; i++){

         string[l] = aux_string[i];

         l++;
      }

      string[l] = "h";
      l++;
//    printf(lcd_putc, "\f%s",string);
//    delay_ms(500);
//


//    Minutos
      aux_string = "";
      j=2;

      if(lig_duracao_m<10){
         j = 1;
         string[l] = "0";
         l++;
      }
      itoa(lig_duracao_m, 10, aux_string);


      for(i=0; i < j ; i++){

         string[l] = aux_string[i];
         l++;
      }

      string[l] = "m";
      l++;

      if(AppendFile(string)){
         printf(lcd_putc, "\fDado nao gravado");
         delay_ms(350);
         goto fim_gravacao;
      }else{
         printf(lcd_putc, "\fAguarde...");
```

```
            delay_ms(150);
        }

        l = 0;
        for(i=0; i < 30 ; i++){
            string[i] = "";
        }

//      Segundos
        aux_string = "";
        j=2;

        if(lig_duracao_s<10){
            j = 1;
            string[l] = "0";
            l++;
        }
        itoa(lig_duracao_s, 10, aux_string);

        for(i=0; i < j ; i++){

            string[l] = aux_string[i];
            l++;
        }

        string[l] = "s";
        l++;

        //string[l] = "|";
        //l++;



        string[l] = "|";
        l++;

//      //Horario
//      //Horas
        aux_string = "";
        j=2;

        if(lig_hora<10){
            j = 1;
            string[l] = "0";
            l++;
        }
        itoa(lig_hora, 10, aux_string);

        for(i=0; i < j ; i++){

            string[l] = aux_string[i];
            l++;
        }
```

```
      string[l] = ":";
      l++;

//    //Minutos
      aux_string = "";
      j=2;

      if(lig_minuto<10){
         j = 1;
         string[l] = "0";
         l++;
      }

      itoa(lig_minuto, 10, aux_string);


      for(i=0; i < j ; i++){

         if(aux_string[i] == -1){
            break;
         }

         string[l] = aux_string[i];
         l++;
      }
   // delay_ms(2550);
      string[l] = ":";
      l++;


//    //Segundos
      aux_string = "";
      j=2;

      if(lig_segundo<10){
         j = 1;
         string[l] = "0";
         l++;
      }

      itoa(lig_segundo, 10, aux_string);

      for(i=0; i < j ; i++){

         if(aux_string[i] == -1){
            break;
         }

         string[l] = aux_string[i];
         l++;
      }

      string[l] = "|";
      l++;
```

```
//     printf(lcd_putc, "\f%s",string);
//     delay_ms(500);



//     //Data
//     //Dia

       aux_string = "";
       j=2;

       if(lig_dia<10){
          j = 1;
          string[l] = "0";
          l++;
       }

       itoa(lig_dia, 10, aux_string);

       for(i=0; i < j ; i++){

          if(aux_string[i] == -1){
             break;
          }

          string[l] = aux_string[i];
          l++;
       }

       string[l] = "/";
       l++;


       //Mes
       aux_string = "";
       j=2;

       if(lig_mes<10){
          j = 1;
          string[l] = "0";
          l++;
       }

       itoa(lig_mes, 10, aux_string);

       for(i=0; i < j ; i++){

          string[l] = aux_string[i];
          l++;
       }

       string[l] = "/";
       l++;
```

```
//     //Ano

       aux_string = "";
       j=2;

       if(lig_ano<10){
          j = 1;
          string[l] = "0";
          l++;
       }

       itoa(lig_ano, 10, aux_string);

       for(i=0; i < j ; i++){

          string[l] = aux_string[i];
          l++;
       }

//     printf(lcd_putc, "\f%s",string);
//     delay_ms(500);
       if(AppendFile(string)){
          printf(lcd_putc, "\fDado nao gravado");
          delay_ms(350);
          goto fim_gravacao;
       }else{
          printf(lcd_putc, "\fNumero registrado");
          delay_ms(100);
       }



//     //habilita as interrupcoes novamente
fim_gravacao:

       ext_int_edge( 1, H_TO_L);

       enable_interrupts(INT_EXT1);
       enable_interrupts(int_ext1);
       enable_interrupts(int_timer2);

       delay_ms(200);
       dsblq_linha;
       delay_ms(150);

    }

    flag_tela = 1; //volta pro início

}

int ler_num_discado(){
```

```
   ler_portb();

   if(num_teclado != 11 ){

      lig_num_chamado[cont_num_discado] = (int)(num_teclado);
      cont_num_discado++;
      return 1;
   }else{

      return 0;
   }


}

void tela3_ligacao(void){
   //inverte a borda da interrupcao do fone no gancho
ligacao:
   ext_int_edge(0 , H_TO_L);

   clear_interrupt(int_ext);
   clear_interrupt(int_ext1);

   disable_interrupts(int_ext);
   disable_interrupts(int_ext1);

   blq_linha;
   delay_ms(800);
   dsblq_linha;
    delay_ms(100);

   enable_interrupts(int_ext);
   enable_interrupts(int_ext1);
   delay_ms(50);
   flag_ligacao = 1;

   printf(lcd_putc,"\f   Linha liberada   ");
   printf(lcd_putc,"\n\n Chamada em curso.. ");
   delay_ms(300);

   cont_timer2 = 0x00;

   flag_fone_gancho=0;
   while((flag_ligacao)&&(flag_fone_gancho==0)){

      enable_interrupts(INT_TIMER2);
      if(cont_timer2>15){
         printf(lcd_putc,"\f   Tempo Esgotado   ");
      goto ligacao;
      }
   }
   if(flag_fone_gancho){
      flag_tela = 1;
      flag_fone_gancho=0;
```

```
      return;
   }

// memoriza o momento da chamada
   lig_hora = hora;
   lig_minuto = minuto;
   lig_segundo = segundo;
   lig_dia = dia;
   lig_mes = mes;
   lig_ano = ano;

   //dispara timer2 para contagem do tempo de ligacao
   clear_interrupt(INT_TIMER2);
   disable_interrupts(INT_TIMER2);
   cont_timer2 = 0x00;
   enable_interrupts(INT_TIMER2);

   delay_ms(300);
   //enquanto o telefone nao for desligado

   printf(lcd_putc,"\f");
   flag_fone_gancho = 0;
   while(flag_fone_gancho == 0){

      lcd_gotoxy(1,1);
      printf(lcd_putc,"   Tempo(s): %4Lu   ",cont_timer2);
      lcd_gotoxy(1,3);
      printf(lcd_putc,"   Data: %2Lu/%2Lu/%2Lu   ",dia,mes,ano);
      lcd_gotoxy(1,4);
      printf(lcd_putc,"   Hora: %2Lu:%2Lu:%2Lu   ",hora,minuto,segundo);

   }


   clear_interrupt(INT_TIMER2);
   if(cont_timer2>=15){

      lig_duracao = 0x00;
      lig_duracao = (cont_timer2 - 15);

      //zera contador de timer2
      cont_timer2 =  0x00;
      flag_lig_valida = 1;

   }else{
      cont_timer2 =  0x00;
      flag_lig_valida = 0;
   }

   flag_tela = 4;//vai pra gravacao de sd

}

void InicializaSistema(void){
```

```
int i=0;
set_tris_a(0b00000000);
set_tris_b(0b11110011);
set_tris_c(0b00000000);
set_tris_d(0b00100000);
set_tris_e(0b00000000);

for(i=0;i<14;i++){
   lig_num_chamado[i] = 0;
}

blq_linha;

clear_interrupt(global);
clear_interrupt(INT_EXT);
clear_interrupt(INT_EXT1);
clear_interrupt(INT_TIMER1);
clear_interrupt(INT_TIMER2);

setup_spi(false);
lcd_init();
delay_ms(200);

printf(lcd_putc,"\fInicializando..");
delay_ms(500);
printf(lcd_putc,"\f\n   REGISTRADOR DE   ");
printf(lcd_putc,"\n   CHAMADAS - 1.0   ");
delay_ms(1000);

if(SD_init()){
   printf(lcd_putc,"\fSD nao inicializado");
   while(1){}
};
printf(lcd_putc,"\nConfigurando...");
if(read_file_senha()){
   printf(lcd_putc,"\fErro lendo Senha");
   while(1){}
}

delay_ms(100);
enable_interrupts(global);

ext_int_edge( 0, L_TO_H);

ext_int_edge( 1, H_TO_L);
enable_interrupts(INT_EXT1);
setup_timer_2 ( T2_DIV_BY_4, 0x6A, 10);

tela_ajuste_horario();
tela_ajuste_data();

setup_timer_1 (T1_INTERNAL | T1_DIV_BY_4);
set_timer1(0x566D);
enable_interrupts(INT_TIMER1);
```

```
    blq_linha;
    delay_ms(800);
    dsblq_linha;
    delay_ms(100);
}
```

B.5 Função mmc_write_block_new em 'Mmc_spi.h'

Esta função foi implementada para escrita correta de dados no cartão de memória. A lógica utilizada é a seguinte:

- **Buscar ponteiro de posição de memória**
- **Copiar os dados do início do setor ate o endereço do ponteiro**
- **Concatenar com os dados a serem inseridos**
- **Realizar a escrita de todo o buffer, considerando os 512bytes**

```
MMC_EC mmc_write_block_new(int32 address, int32 size, int8 *ptr){

   //Função criada para escrita correta de dados no cartão de memória
   //A escrita deve ser feita sempre com 512bytes no início de cada setor
   //Portanto, devemos sempre bufferizar os dados
   float32 arredondamento=0;
   int32 new_address = 0;
   float32 indice = 0;
   int32 i = 0;
   int32 f = 0;
   int32 nadd_size = 0;
   int32 nsize = 0;
   int  MMCBuffer_new[512];


   arredondamento = floor(address/512);
   new_address = (int32)arredondamento;
   new_address = new_address*512;

   indice = fmod(address,512);
   i = (int32)indice;


   mmc_read_block(new_address, i, &MMCBuffer_new);

   delay_ms(300);

      for(f=i;f<i+size;f++){

      MMCBuffer_new[f]=ptr[f-i];

      }

   nadd_size = address + size ;

   nsize = 512 - i;
```

```
    nsize = nsize - size;

    mmc_read_block(nadd_size, nsize, &MMCBuffer_new[i+size]);

    if(mmc_write_block(new_address, 512, &MMCBuffer_new)==0){

        return(MMC_EC_OK);
    }else{

        return(MMC_EC_BLOCK_NOT_WRITTEN);
    }
}
```