UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

ESCOLA DE ENGENHARIA

DEPARTAMENTO DE ELETRÔNICA E DE COMPUTAÇÃO

SISTEMA DE APOIO A DEFICIENTES FÍSICOS ATIVADO POR COMANDOS DE VOZ

Autor _____
Guilherme Santos Vecchi

Orientador _____
Manuel Lois Anido, Ph.D

Co-orientador _____
Carlos José Ribas D'Avila, M. Sc.

Examinador _____
Antônio Petraglia, Ph.D

Dezembro de 2005

Agradecimentos

Gostaria de agradecer a colaboração de pessoas sem as quais a realização deste projeto não seria possível.

Ao meu Orientador Manuel Lois Anido, pela sua atenção, compreensão e motivação, assim como ao meu co-orientador, Carlos José Ribas D'Avila.

Aos meus pais pela paciência e compreensão.

Aos meus colegas de estágio, sempre dispostos a ajudar: Mauricio, Daniel, Vinícius e Raphael.

Aos funcionários do NCE, Nelson e Takano, sempre prontos a ajudar também.

Aos meus companheiros de curso, que em muito contribuíram para a minha formação.

Resumo

Este projeto apresenta a análise e a implementação de um sistema de apoio a deficientes físicos ativado por comandos de voz.

O sistema visa principalmente dar apoio a deficientes físicos que não possam usar as mãos, tais como tetraplégicos, para atender ou fazer ligações telefônicas e também para comandar eletrodomésticos, através de comandos de voz.

O sistema é portátil, de baixo consumo, controlado por um microcontrolador e permite fazer ligações usando telefones celulares ou telefones sem fio domésticos, além de controlar eletrodomésticos, utilizando o protocolo Philips RC5 via infravermelho.

Palavras-chave

- Sistemas Embutidos
- Ativação por comando de voz
- Controle via infravermelho
- Controle de Celulares
- Apoio a Deficientes Físicos

Índice

CAPÍTULO I – Introdução

1.1 Motivação

Este projeto surgiu da necessidade de se criar um sistema capaz de permitir que deficientes físicos, em especial deficientes tetraplégicos, pudessem comandar equipamentos domésticos e, com isso, obtivessem melhor qualidade de vida. Um outro objetivo é que o sistema também pudesse ser usado por qualquer pessoa com dificuldades para discar os números de um telefone, fixo ou celular, ou apertar as teclas de um controle remoto de televisão, como por exemplo, uma pessoa idosa.

O mundo atual oferece-nos uma grande quantidade de equipamentos para desfrutarmos da vida moderna. Temos à disposição telefones que discam por comandos de voz, televisores que se conectam à Internet, microondas operados remotamente, máquinas fotográficas que não usam filmes e automóveis que estacionam sozinhos. No entanto, ainda há muitos problemas por resolver, particularmente na área de sistemas de apoio a deficientes físicos.

Embora existam telefones que discam por comandos de voz pré-gravada, tais telefones não são integrados com controle de dispositivos domésticos, tais como TVs, videocassetes ou som. Da mesma forma, a opção por um telefone celular pode não ser a mais econômica para o usuário, em virtude principalmente do custo das ligações.

Ao analisar ainda mais a fundo as necessidades de um deficiente tetraplégico, observamos que seria também fundamental a eles poder controlar por comando de voz as luzes da casa e a movimentação da cadeira de rodas.

Atualmente, um sistema de baixo custo com estas características não está disponível no mercado. Há espaço, portanto, para o estudo e o desenvolvimento de projetos que possam vir a ser extremamente úteis a deficientes físicos, ou mesmo a pessoas idosas ou em hospitais, onde o controle por comandos de voz possa melhorar substancialmente a qualidade de suas vidas.

1.2 Importância do tema

O desenvolvimento de sistemas embutidos para apoiar deficientes físicos é altamente relevante em virtude da função social que agrega, mas também do ponto de vista de nicho de mercado, onde há grande carência de sistemas de apoio a deficientes.

Este apoio pode dar-se através de vários tipos de projetos. Um exemplo louvável destes projetos é o projeto MOTRIX [15], do NCE/UFRJ, específico para deficientes visuais e sob a coordenação de José Antonio Borges. Este projeto objetiva permitir que deficientes visuais interajam com o computador, lendo e editando textos e navegando pela Internet.

1.3 Revisão Bibliográfica

Dentro das referências bibliográficas deste trabalho, incluímos exemplos de projetos voltados à minimização das limitações causadas por deficiências de diversos tipos.

Em [13] tem-se um exemplo de trabalho para pessoas com dificuldades de locomoção. É um sinalizador sem fio que usa tecnologia *bluetooth*. O deficiente aciona o sinalizador sempre que precisar realizar alguma tarefa complexa para ele. O sinal de rádio deve ser forte o suficiente para atravessar paredes de concreto.

Em [14] o exemplo é de um trabalho para pessoas com deficiência de fala, usando sistemas embutidos. O sistema converte texto digitado em voz. Existem 5 possibilidades de línguas: Inglês Americano, Inglês britânico, Espanhol Sul-americano, Espanhol Castelhano e Alemão. Além disso, há também possibilidades de adaptação de voz para homem, mulher e criança.

Em [15] cita-se um projeto desenvolvido no Núcleo de computação eletrônica (NCE) da UFRJ, chamado MOTRIX. O MOTRIX é um software que permite a pessoas com deficiências motoras graves, em especial tetraplegia e distrofia muscular, a possibilidade de ter acesso a microcomputadores. Isto permite, em especial, com a intermediação da Internet, um acesso amplo à escrita, leitura e comunicação. O acionamento do sistema é feito através de comandos que são falados num microfone.

O uso do Motrix torna viável a execução pelo tetraplégico de quase todas as operações que são realizadas por pessoas não portadoras de deficiência. Mesmo as ações que possuem

acionamento físico complexo, tais como jogos, são viabilizadas através de um mecanismo inteligente, em que o computador realiza a parte motora mais difícil destas tarefas. O sistema pode ser acoplado a dispositivos externos de *home automation* para facilitar, em especial, a interação do tetraplégico com o ambiente de sua própria casa.

A indicação [16] traz como tema os telefones feitos especificamente para deficientes físicos ou idosos. Há telefones que discam por comando de voz, outros cujas teclas são grandes para facilitar a discagem e outros ainda em que basta apertar uma ou duas teclas que a discagem é feita, sem a necessidade apertar todos os dígitos.

Em [17] abordam-se projetos para deficientes físicos que servem para adaptar diversos aparelhos aos deficientes. Os estudantes ouvem os deficientes para saber suas necessidades e assim criam os projetos, com custo zero para os deficientes. Um dos projetos mais bem sucedidos permite que uma pessoa tetraplégica, por comando de voz, ligue e desligue as luzes de casa, permite regular a temperatura e ter um controle do ambiente de sua casa.

Em [18] trata-se do projeto de dois estudantes universitários que permite a deficientes simular o uso do mouse por comando de voz. Os desenvolvimentos deste projeto permitem que os deficientes naveguem na internet mais facilmente. O programa foi escrito em *Visual Basic* e pode ser usado em 3 modos: Modo mouse, mouse opção e modo texto. Em Modo mouse, é permitido ao usuário controlar o cursor do mouse, os clicks do mouse e duplos-click por voz. Em mouse opção, é permitido ao usuário controlar a velocidade do mouse que pode ser: lenta, média e rápida. E no modo texto, é permitido ao usuário comandar quase todas as teclas do teclado por voz.

Em [19] faz-se referência às invenções para deficientes criadas pelo inventor Ray Kurzweil. Seus inventos usam reconhecimento de voz e temos exemplos destes para pessoas com deficiência de leitura, onde texto é convertido em voz, e para pessoas surdas, onde texto é convertido em voz e a voz é convertida em texto, para permitir que uma pessoa surda consiga se comunicar.

1.4 Roteiro do trabalho

No capítulo 1, apresentamos a motivação do trabalho e a relevância do tema. O capítulo 2, por sua vez, traz a abordagem sobre o embasamento teórico necessário ao desenvolvimento do sistema. Em seguida, no capítulo 3, faz-se uma descrição propriamente do sistema, mostrando os seus principais componentes.

No capítulo 4, encontra-se a descrição dos testes realizados para que o sistema funcionasse corretamente, além dos resultados obtidos, que conduziram ao acabamento final do sistema.

Finalmente, no capítulo 5, apresentam-se as conclusões a respeito do projeto, enfatizando os principais problemas enfrentados para sua realização e as melhorias que poderiam ser feitas.

CAPÍTULO II – Embasamento teórico

2.1 Introdução

Este capítulo tem por objetivo apresentar e discutir alguns aspectos das tecnologias utilizadas, de forma a prover um embasamento teórico mínimo necessário para o desenvolvimento do trabalho.

2.2 Microcontroladores

O termo controlador é usado para designar o dispositivo que gerencia um processo ou algum parâmetro do ambiente. O controlador de temperatura do condicionador de ar, por exemplo, liga ou desliga o compressor em função da temperatura ambiente. Antigamente, os controladores usavam lógica discreta e, por isso, tinham um tamanho que dificultava seu emprego em sistemas pequenos. Hoje em dia, usam-se os circuitos integrados microprocessados e todo o controlador pode caber em uma pequena placa de circuito impresso.

O microcontrolador, com o avanço da microeletrônica, recebeu dentro do seu CI uma quantidade de recursos cada vez maior. Isso nos leva à primeira definição: microcontrolador é um CI com alta densidade de integração, que inclui, dentro do chip, a maioria dos componentes necessários para o controlador. É por isso que se usa o apelido: "solução com único chip". Existe uma quantidade expressiva de microcontroladores, porém os mais conhecidos são: 8051, 8096, 68HC705, 68HC11, PIC16FXXX, PIC 8FXXX, etc. A figura 2.1 apresenta o diagrama em blocos de um típico microcontrolador.

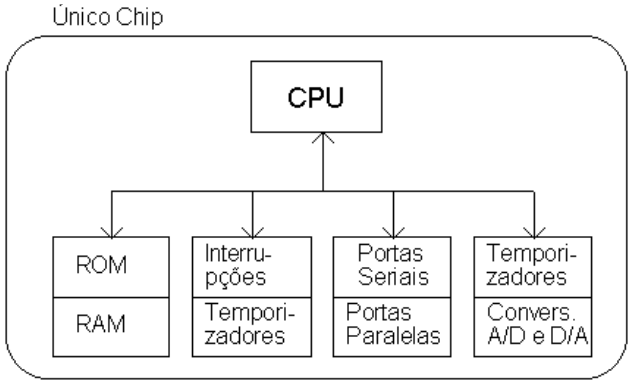


Figura 2.1 – Diagrama de blocos de um típico microcontrolador

Note-se que todos os recursos aqui mostrados estão dentro de um único CI.

A fronteira entre as definições de microcontrolador e de sistema embutido não é clara e muitas das soluções poderiam ser obtidas tanto com microcontroladores quanto com sistemas embutidos. A chave para essa diferença está no tamanho e na complexidade. Os microcontroladores são simples, enquanto que os sistemas embutidos são mais complexos e usam uma grande quantidade de chips.

2.2.1 Os Sistemas Embutidos

De forma rápida e abusando das palavras, podemos dizer que o sistema embutido é um "pequeno computador" que foi embutido em um sistema maior. Podemos ainda dizer que um sistema embutido é um sistema computacional incluído em um outro sistema, com finalidade outra que a de fornecer processamento genérico.

Conforme já afirmamos, a chave para diferenciar um sistema embutido de um microcontrolador está no tamanho. Os microcontroladores usualmente são pequenos e empregam poucos CI's, enquanto que os sistemas embutidos são maiores e mais sofisticados, com uma quantidade maior de CI's. Existem sistemas embutidos de todos os tamanhos: desde pequenas placas com uma CPU 8085, uma ROM e uma RAM, até sofisticados sistemas com CPUs Pentium. Os sistemas mais simples apenas trazem um pequeno programa (Aplicação), gravado em sua ROM ou Flash, enquanto que os mais sofisticados fazem uso de BIOS e trazem um sistema operacional. Para os sistemas embutidos é muito comum o uso do DOS e de sistemas de tempo real, como o Linux, o QNX, etc.

É difícil especificar a anatomia de um sistema embutido, pois ela é ditada pela aplicação. Apesar desta dificuldade, tentamos a caracterização de um sistema embutido típico na figura 2.2.

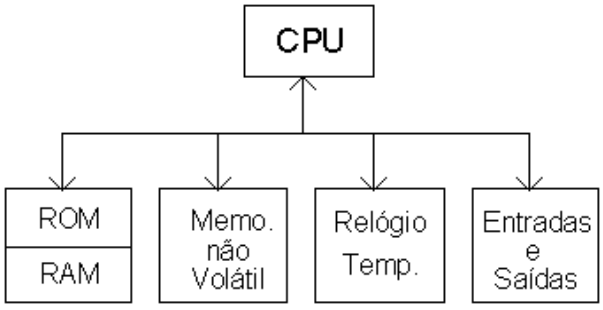


Figura 2.2 – Típico sistema embutido.

2.3 Microcontrolador Atmel AT89C4051

Primeiramente, escolhemos o Atmel AT89C4051 porque era o microcontrolador que já tínhamos à disposição para a execução do projeto juntamente com o gravador de AT89C4051. Também o AT89C2051 encontrava-se disponível, mas enfim usamos o AT89C4051 porque ele tem mais memória de código que o AT89C2051, uma vez que o código final do programa de controle do sistema ficou relativamente grande. No momento da escolha não tínhamos nenhum microcontrolador PIC, nem gravador disponível. Além disso, o AT89C4051 tem a entrada e a saída seriais necessárias para a comunicação serial com o celular e aceita cristal com freqüência de até 24MHz, que é justamente a freqüência necessária para a transmissão do protocolo philips RC5.

O Microcontrolador pode ser entendido como um sistema eletrônico integrado, dotado de "inteligência" programável, utilizado no controle de processos lógicos. Falar em "inteligência" implica em associá-la a ULA (Unidade Lógica e Aritmética), pois é nessa unidade que todos as operações lógicas e matemáticas são efetuadas.

Suas operações são executadas em ciclos, ou seja, todos os sinais necessários para a busca ou execução de uma determinada instrução devem ser gerados dentro de um período de tempo denominado ciclo de máquina.

Microcontroladores assemelham-se a microprocessadores tradicionais, mas possuem características distintas deles, que os particularizam. Entre as principais, citamos:

- Baixo consumo de energia;
- Incorporam memória ROM (PROM ou EEPROM), memória FLASH e RAM (em pequena quantidade);
- Incorporam um ou mais canais de comunicação serial;
- Possuem vários *timers* para contagem de tempo e eventos;
- Suportam vários tipos de canais de comunicação serial, além do tradicional canal serial como o I2C, por exemplo;
- Podem incorporar conversores A/D e D/A;

- Geralmente não possuem gerência de memória nem co-processador de ponto flutuante.

2.3.1 Ciclos de máquina

As operações do microcontrolador são executadas em ciclos, ou seja, todos os sinais necessários para a busca ou execução de uma determinada instrução devem ser geradas dentro de um período de tempo. Este é denominado ciclo de máquina.

Nos microcontroladores da família 8051, o sinal do *clock* é dividido por 12. Portanto, para um *clock* externo de 12Mhz, temos um *clock* interno de 1MHz e, conseqüentemente, cada ciclo de máquina dura 1us.

2.3.2 Memória de programa

Esta é a memória que armazena as instruções. Como vimos, ela possui capacidade de armazenamento de 4kbytes. Esta é uma memória do tipo FLASH. Este tipo de memória pode ser gravada muitas vezes, sem necessidade de apagar a gravação anterior.

2.3.3 Memória de dados

Esta memória é do tipo RAM e tem capacidade para 128 bytes. Ela é dividida em:

- Posições da RAM "com apelidos" para seus endereços. É dividida em quatro bancos e cada um tem os "apelidos" R0, R1,..., R7. Estas posições também podem ser acessadas pelo seu endereço absoluto. Por exemplo: R0 no banco 0 tem o endereço absoluto do byte 00h da RAM. Estes registradores vão da posição 00h até 1Fh de endereço interno.

- Posições da RAM "sem apelido"; isto é, acessíveis apenas pelo seu endereço absoluto, porém com uma propriedade adicional: cada bit de uma posição de memória também tem seu endereço particular, isto é, ela é "bit" endereçável, além de ser "byte" endereçável. Vão da posição de byte 20h até 2fh.

- Registradores de funções Especiais (SFR-Special Function Registers): São aqueles que têm apelido, além do seu endereço absoluto, e possuem também uma função específica dentro do microcontrolador. Por exemplo, o port P1 tem o

endereço do byte 90h e é o espelho do que acontece externamente com o port P1; Alterar os valores do port P1 é alterar os valores desse registrador P1.

2.3.4 Interrupção

Interrupção é um evento externo ou interno que obriga o microcontrolador a suspender suas atividades temporariamente para atender a este evento que o interrompeu. Em resumo, é uma ocorrência que faz o microcontrolador parar a sua rotina e se desviar para outro ponto do software, em que se localiza o serviço de interrupção que foi gerado pela ocorrência. É importantíssimo na estratégia de programação do microcontrolador – caracteriza-se como uma sub-rotina na programação tradicional, só que disparada por um evento externo. Após o serviço de interrupção estar completo, o microcontrolador desvia-se novamente e exatamente para onde estava antes de ter sido interrompido.

Como vimos, a interrupção neste microcontrolador pode ocorrer de 6 fontes independentes. Existem dois pinos para interrupção, que são int0 e int1, acionadas por transição negativa ou por nível.

2.3.5 Portas de entrada/saída

Todas as 15 portas de entrada e saída do Atmel AT89C4051 são bidirecionais e configuráveis por software. Elas se dividem em port1 e port3.

Port 1 – essa porta possui 8 pinos de entrada/saída. Todos os pinos são do padrão CMOS. Os pinos p1.0 e p1.1 precisam de resistores de *pull-up* externos para serem usados como saída, uma vez que estes dois pinos não possuem os resistores de *pull-up* internos, como os demais pinos.

Port 3 – essa porta possui 7 pinos de entrada/saída. Todos os pinos são do padrão CMOS. O port P3 também é utilizado como meio de comunicação externa entre os periféricos e os “Timers” e “serial”. Então os pinos só estarão disponíveis para I/O, dependendo de se utilizar ou não os periféricos desse Microcontrolador.

2.3.6 Timers

O Timer, conhecido como Timer/Counter, é um periférico interno ao microcontrolador. É uma unidade autônoma e poderia, portanto, ser um chip separado daquele. Especificamente no microcontrolador Atmel AT89C4051 temos dois timers.

Um timer nada mais é que um grupo de *flip-flops* em arranjo de "divisor por dois", que é acionado indiretamente pelo mesmo clock original do microcontrolador, só que esse clock é dividido por 12 antes de entrar nos timers.

O Atmel AT89C4051 possui 2 timers internos: T0 e T1. Estes timers são incrementados internamente a cada período de clock ou a cada múltiplo deste período. Uma grande vantagem da utilização do timer/counter como timer é obtida na contagem de tempo sem envolvimento "pesado" do microcontrolador. O microcontrolador programa o timer para uma determinada contagem, dispara-o e fica liberado para fazer outras tarefas do programa principal. Quando o timer finaliza a contagem de tempo, ele interrompe o microcontrolador que entende, deste modo, que chegou ao tempo programado.

A figura 2.3 ilustra a pinagem externa do Atmel AT89C4051

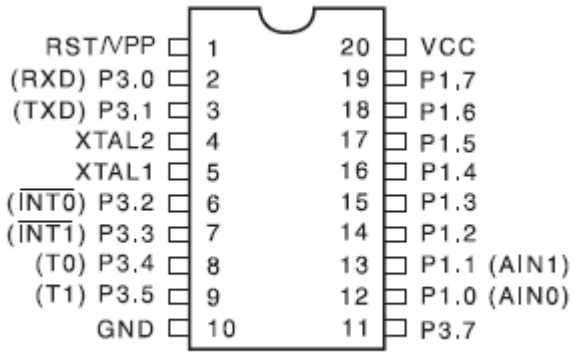


Figura 2.3 – Pinagem externa do Atmel AT89C4051

2.4 Comunicação Serial

2.4.1 Introdução

Dentro do trabalho desenvolvido, tornou-se necessário realizar a comunicação entre o microcontrolador e o dispositivo externo, neste caso, o celular. A escolha do sistema de comunicação mais adequado levou em conta diversos fatores, tais como a velocidade, o custo, a facilidade de programação e a compatibilidade de hardware, entre outros.

As técnicas de comunicação podem ser divididas em duas grandes categorias: serial e paralela. Na comunicação serial, a informação a ser transmitida é dividida em bits, que são enviados ao receptor, um de cada vez, em série, razão pela qual recebe este nome. Podemos citar como exemplos deste tipo de comunicação as interfaces seriais dos computadores – RS-232, USB, protocolos de redes locais (Ethernet), etc. – ou a comunicação série dentro de computadores como I2C e SPI. Já na comunicação paralela, os bits da informação são transmitidos simultaneamente, logo, em paralelo. Incluem-se neste tipo: os barramentos internos dos microprocessadores e microcontroladores, a interface de impressora paralela de computadores, interface Centronics, a interface IDE (integrated Development Enviroment).

Ambas as formas de comunicação possuem vantagens e desvantagens. Ao escolhermos o tipo de comunicação a ser utilizado, levamos em conta os objetivos do trabalho e também a compatibilidade com outros equipamentos utilizados, como o cabo do celular. Este tem em sua extremidade um conector DB9 fêmea e um chip MAX3232 internamente, que converte sinais RS232 para os níveis de tensão do celular, de 0V a 3V. Assim, a escolha pela comunicação serial foi a mais lógica, pois o celular já suportava porta serial assíncrona.

Existem dois modos de comunicação serial e os discutiremos a seguir.

Comunicação Serial Síncrona

Neste modo de comunicação o transmissor e o receptor devem ser sincronizados para a troca de comunicação de dados. Geralmente uma palavra de SINCRONISMO é utilizada para que ambos ajustem o relógio interno. Após a sincronização, os bits são enviados seqüencialmente, sem espaço entre os caracteres, até uma quantidade pré-combinada entre os dispositivos, ou até um caractere indicador de fim de mensagem.

Comunicação Serial Assíncrona

Esta é a forma mais usual de transmissão de dados. Não existe a necessidade desincronização entre os dispositivos, uma vez que os caracteres são transmitidos individualmente e não em blocos como na comunicação síncrona. A transmissão de cada caractere é precedida de um bit de start e terminada por 1 (1/2 ou 2) bit(s) de stop.

Existem basicamente 2 tipos de padrões na categoria comunicação serial:

(i) Comunicação assíncrona universal

(ii) Comunicação síncrona entre dispositivos

A Figura 2.4 mostra a composição típica dos bits de um Byte assíncrono.

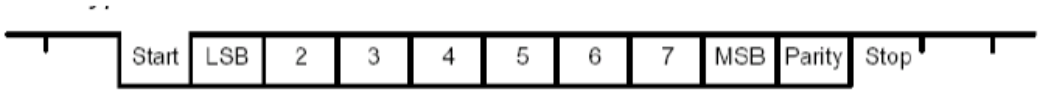


Figura 2.4 – Byte assíncrono típico

Note-se que neste Byte existe um bit de start, 8 bits de dados, um bit de paridade e um bit de parada. A paridade pode ser par ou ímpar. Quando a paridade é par o bit de paridade é gerado de modo que o número de 1s resultante na palavra mais o bit de paridade seja par.

Por exemplo, se a palavra 10001010 está sendo transmitida, ou recebida, e se está utilizando paridade par, o bit de paridade deve ser 1, para que o conjunto palavra + bit de paridade tenha sempre um número par de1s.

Se a paridade usada for ímpar o bit de paridade no exemplo anterior será zero.

No processo de transmissão assíncrona, os dispositivos envolvidos no processo de comunicação devem ter a mesma taxa de transmissão e recepção**.**

2.4.2 Comunicação RS-232

Um dos padrões mais conhecidos para a comunicação serial assíncrona é o RS-232. O padrão define um nível de tensão diferente do TTL e o mais comum é o seguinte:

- Nível Lógico 0: +3 a +12 Volts.

- Nível Lógico 1: -3 a –12 Volts.

O chip MAX232 da MAXIM necessita apenas de fonte +5V e incorpora tanto o conversor TTL-RS232 como o RS232-TTL, conforme mostrado na Figura 2.5.

É importante frisar que também existe o chip MAX3232 da MAXIM que funciona com apenas 3 volts e foi desenvolvido para realizar a interface com telefones móveis (celulares). A pinagem é a mesma do MAX232.

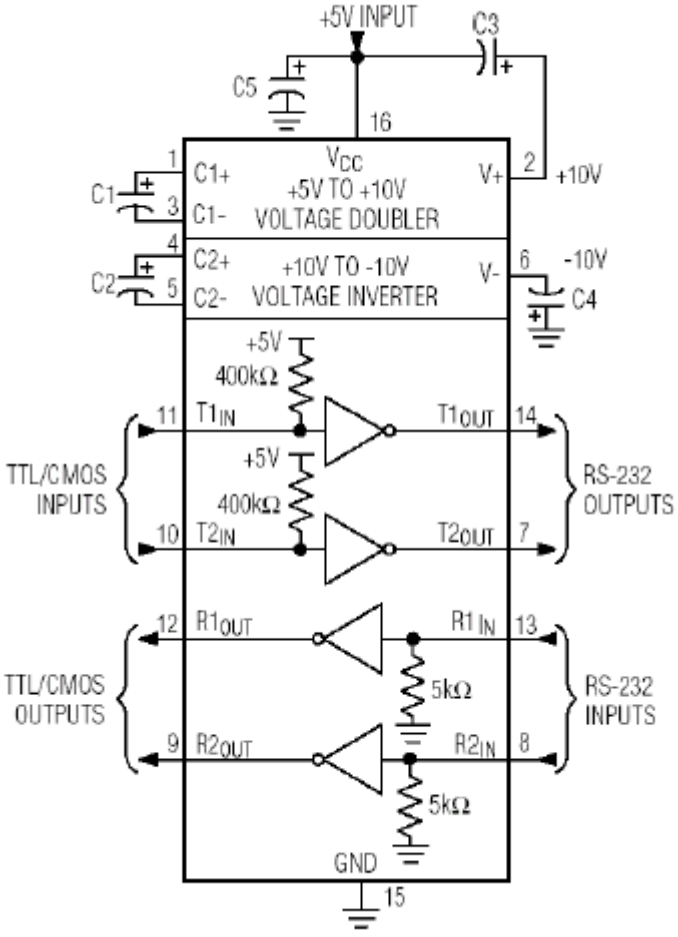


Figura 2.5 – Diagrama de blocos do MAX232

Os capacitores C1, C2, C3, C4 devem valer 10uF.

No caso do MAX3232 os capacitores devem valer 0,1uF.

A Tabela 2.1 mostra os sinais típicos em um conector RS232C. Muitos destes sinais só são empregados em comunicação com dispositivos mais complexos, como no caso de modens

externos. Nos processos mais simples de comunicação apenas os pinos TXD e RXD são utilizados, como neste projeto.

Tabela 2.1 – Sinais Padrão RS232

SIGLA	Descrição
TXD	Dados transmitidos pelo terminal (TX Data)
RXD	Dados recebidos pelo terminal (RX Data)
RTS	Requisição de envio (Request to Send)
CTS	Permissão de envio (Clear to Send)
DSR	Dados prontos (Data Set Ready)
DCD	Detector de portadora (Data Carrier Detected)
DTR	Terminal de dados pronto (Data Terminal Ready)
GND	Terra (ground)

A seguir, apresentamos a descrição dos sinais de uma Interface RS232C.

DTR

O sinal DTR é gerado pela estação de trabalho e avisa ao dispositivo no outro extremo do cabo que a estação está pronta (on) ou não (off). O DTR é normalmente habilitado naturalmente, quando a porta serial é iniciada.

TXD

O sinal TXD transporta os dados transmitidos da estação para o outro dispositivo receptor. Conforme mencionado, a tensão do bit on è interpretada como 1 e a do bit off como 0.

RXD

Analogamente ao sinal TXD, o sinal RXD recebe os dados vindos de outros dispositivos, interpretando-os da mesma forma que a transmissão.

DSR

Este sinal representa a entrada do computador e uma saída do modem indica que o modem está pronto para estabelecer um enlace de comunicação e transferir dados.

GND

Tecnicamente o GND não é um sinal, porém sem ele nenhum dos sinais vai operar. O terra age como uma tensão de referência, de forma a se poder interpretar quais tensões são positivas e quais negativas.

DCD

Este sinal representa uma entrada do computador e uma saída do modem e é indicativa de detecção de portadora; nem sempre é usado.

CTS

O CTS representa uma entrada para o computador e é uma saída do modem e indica se o modem está pronto a enviar dados.

RTS

O RTS representa uma saída do computador e é uma entrada para o modem. Além disso, informa que a estação de trabalho está pedindo ao modem para colocar a portadora na linha para iniciar a transmissão.

Na prática, podemos estabelecer a conexão serial entre o microcontrolador e um celular apenas ligando os sinais TXD, RXD e GND, que são essenciais à comunicação serial.

2.4.3 Conectores DB-9 e DB-25

Há dois tipos de conectores que são os mais utilizados para comunicação serial: DB-9 e DB-25. Nas figuras 2.6 e 2.7 são apresentados os conectores DB-9 e DB-25 e na tabela a seguir os sinais referentes a cada pino destes.

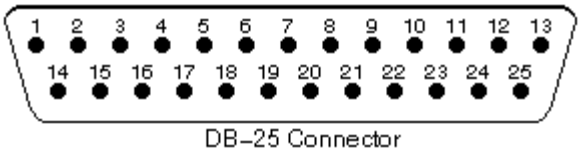


Figura 2.6 – Conector DB-25

Figura 2.7 – Conector DB-9

Tabela 2.2 – Sinais dos pinos do DB-9 e do DB-25

DB-25	DB-9	SIGLA
3	2	RX
2	3	TX
20	4	DTR
7	5	GND
6	6	DSR
4	7	RTS
5	8	CTS

2.5 Controle Remoto de dispositivos usando luz infra-vermelha

2.5.1 Introdução

O intervalo do espectro de luz visível vai de 390nm a 780nm. Abaixo de 390nm temos a luz ultravioleta e acima de 780nm temos a luz infravermelha.

Luz infravermelha (IR) é mais barata, não faz mal à saúde, pode ser usada em larga escala e os componentes são mais simples. Apesar do olho humano não ver luz IR, uma câmera de vídeo ou digital pode "vê-la".

Usa-se alguma técnica de modulação para poder distinguir o sinal de algum ruído que possa aparecer. A maioria dos sinais, da forma como são fornecidos pelo transdutor, não podem ser enviados diretamente através dos canais de transmissão.

Modulação é uma técnica em que se modifica algum sinal básico de modo que se possa codificar informação nele. Para podermos transmitir sinal IR no ar temos que modulá-lo.

A freqüência da portadora varia de 36KHz a 60KHz para consumo doméstico.

Para obter-se um alcance maior do sinal IR, deve-se fazer passar pelo LED IR uma corrente maior do que seria necessário num LED comum. A corrente no LED IR pode variar de 100mA a 1A e os microcontroladores não conseguem fornecer esta faixa de corrente diretamente. Uma solução, então, é usar transistores para amplificar a corrente, conforme mostra a figura 2.8.

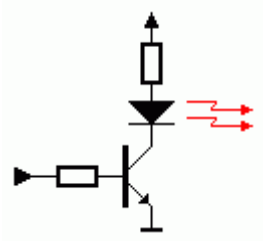


Figura 2.8 – Transistor usado para amplificar a corrente necessária para acionar o LED infravermelho.

2.5.2 Protocolo Infravermelho Philips RC5

Escolheu-se este protocolo devido ao fato de ele ser o protocolo mais usado por hobistas, técnicos, etc., além do fato de permitir uma ampla variedade de controles remotos com baixo custo.

Também foi escolhido porque era possível fazer a modulação na freqüência de 36KHz por software no microcontrolador, utilizando um clock de 24MHz.

As principais características deste protocolo são:

- 5 bits de endereço e 6 bits de comando;
- codificação bifásica;
- freqüência da portadora de 36KHz;
- período de um bit de 1.8ms;
- fabricante: Philips.

2.5.3 Modulação e protocolo RC5

Os bits são considerados “0 “ e “1 “ de acordo com a figura 2.9

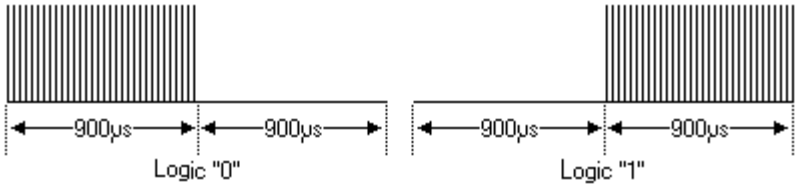


Figura 2.9 – Exemplo de um bit que em “0” e em “1” na codificação RC5.

As principais características deste protocolo são:

- Dois start bits (“1”) – S1 e S2;
- Um *toggle* bit que troca de estado quando se aperta e solta o botão de um controle remoto. Se um botão de controle remoto permanecer apertado o bit “Toggle” não muda e o receptor distingue isso;
- Cinco bits de endereço do dispositivo;
- Seis bits identificadores de comando;
- Tamanho total da mensagem de 14 bits (Duração: 25,2ms).

A figura 2.10 mostra um exemplo de uma mensagem completa. Neste exemplo temos um comando $2B para um endereço $14.

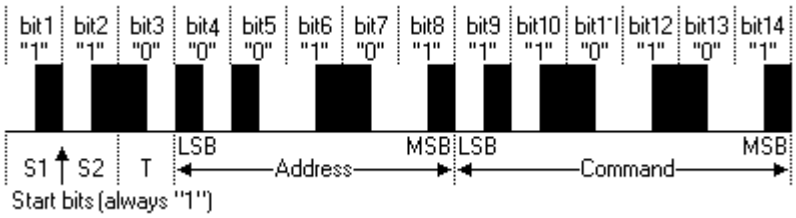


Figura 2.10 – Exemplo de uma mensagem completa

2.5.4 Alguns endereços e comandos pré-definidos

A tabela 2.3 mostra alguns endereços pré-definidos. A televisão usada no projeto tem endereço $00.

Tabela 2.3 – Alguns endereços pré-definidos

RC5 Adress	Device	RC5 Adress	Device
$00	TV1	$09	Câmera
$01	TV2	$05	VCR1
$02	Teletext	$06	VCR2
$08	Sat1	$1A	CDR
$0A	Sat2	$1F	Phone
$03	Vídeo	$14	CD Player

Tabela 2.4 – Alguns comandos RC5 pré-definidos

RC5 Command	TV Command	VCR Command
$00...$09	1,2,2,3,4,5,6,7,8,9	Idem
$0C	Standby	Idem
$10	Volume +	
$11	Volume -	
$32		Fast Rewind
$34		Fast Foward

2.6 Ambiente de Desenvolvimento : O Software da Keil

O sistema Keil é um ambiente de desenvolvimento integrado (IDE) para dar suporte ao desenvolvimento de programas C ou Assembly e também à depuração.

Ao iniciar o software da Keil, tem-se acesso ao ambiente de trabalho global. Trata-se de uma área para a abertura das janelas de trabalho, um menu superior e algumas barras de ferramentas com diversos ícones relativos a funções específicas.

Para que se possa trabalhar dentro desse ambiente não basta o arquivo de código fonte, sendo necessário ter muitas outras informações para que o sistema possa ser compilado e executado. Por isso, o Keil utiliza o conceito de projeto.

Um projeto é um arquivo que guarda todas as informações necessárias ao sistema em desenvolvimento. O importante a ser destacado é que o sistema Keil não funciona adequadamente se um projeto não for aberto.

Uma vez determinado o diretório e o nome do arquivo de projeto, será aberta a tela de edição do projeto. Dentro desta janela são feitas configurações importantes relativas à simulação, como a escolha do microcontrolador, o clock de operação do sistema e a associação do código fonte ao projeto em questão. O Keil tem a opção debug que permite várias facilidades de depuração: Pode-se rodar um programa passo a passo, pode-se estabelecer pontos de parada, pode-se pular algum trecho de código etc., e tudo isto ajuda a encontrar erros de programação. A figura 2.11 mostra uma janela com um projeto aberto e o código fonte na linguagem C associado a este projeto.

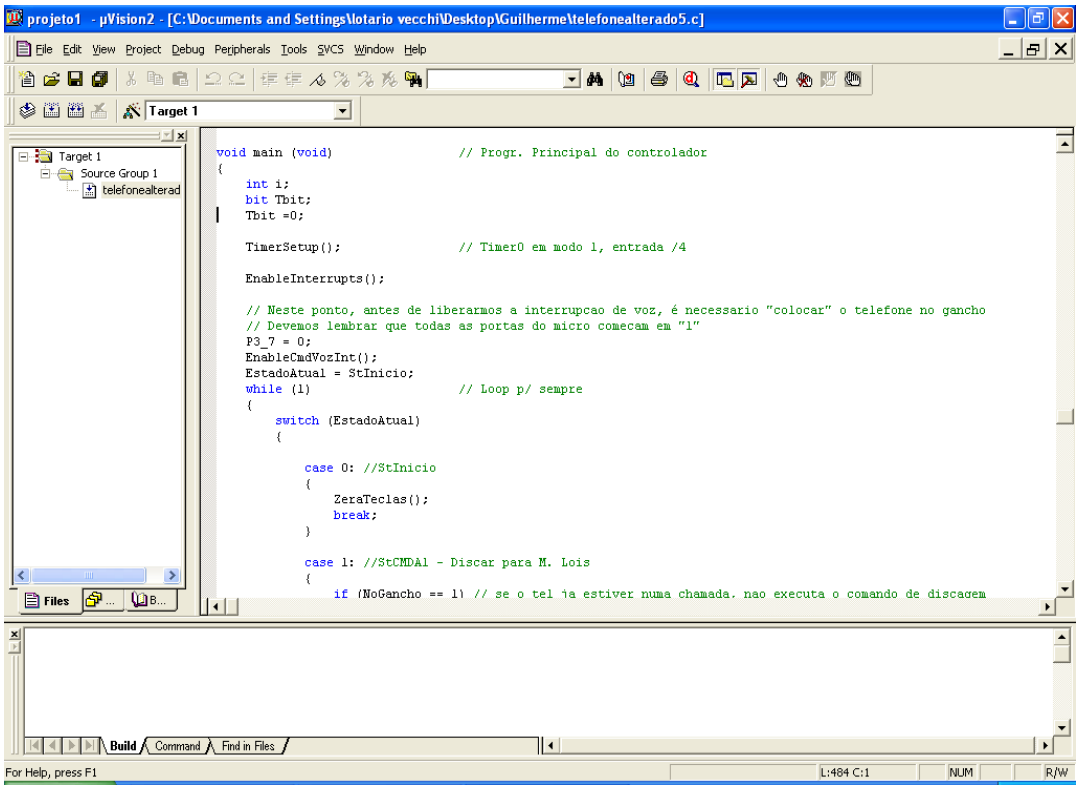


Figura 2.11 – Ambiente de desenvolvimento onde foi feito e compilado o programa do 8051: O software da keil.

A figura 2.12 mostra uma janela com um projeto e um código em assembly associado a este.

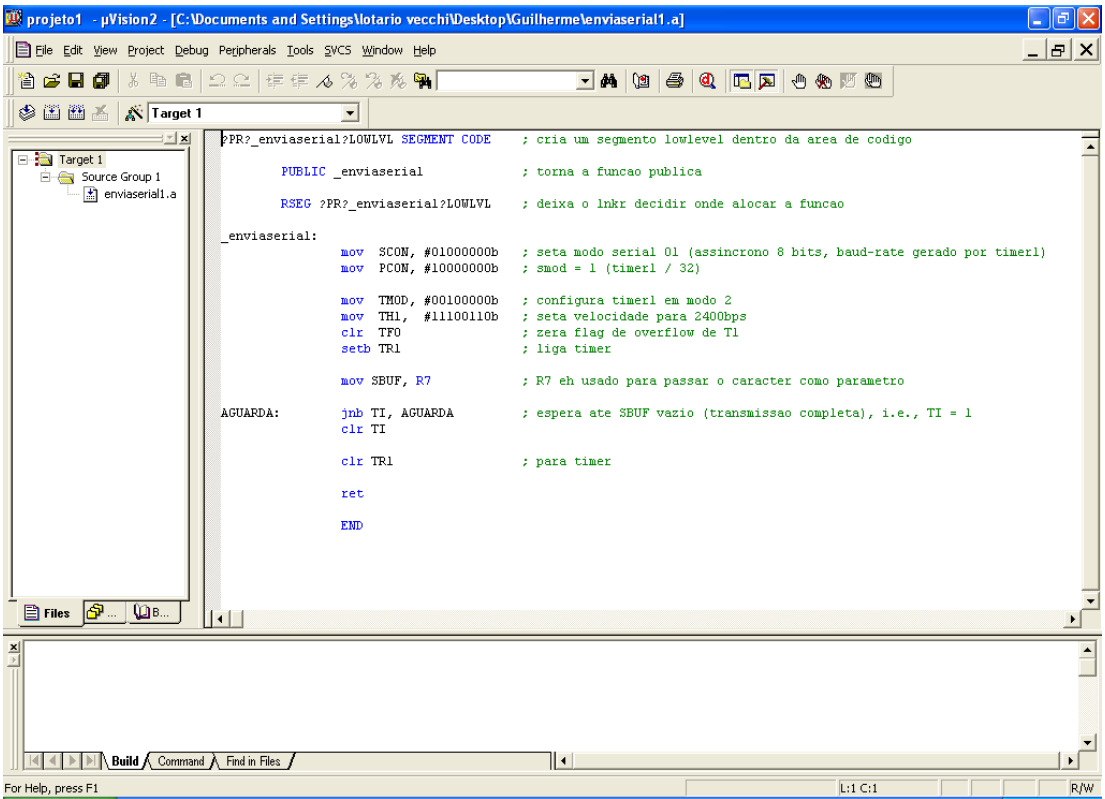


Figura 2.12 – Projeto aberto com um código assembly associado a ele.

A Figura 2.13 mostra um projeto aberto e duas janelas referentes aos ports P1 e P3 abertas. Este microcontrolador usado no projeto tem apenas os ports P1 e P3, mas existem microcontroladores maiores que têm os ports P2 e P4 também.

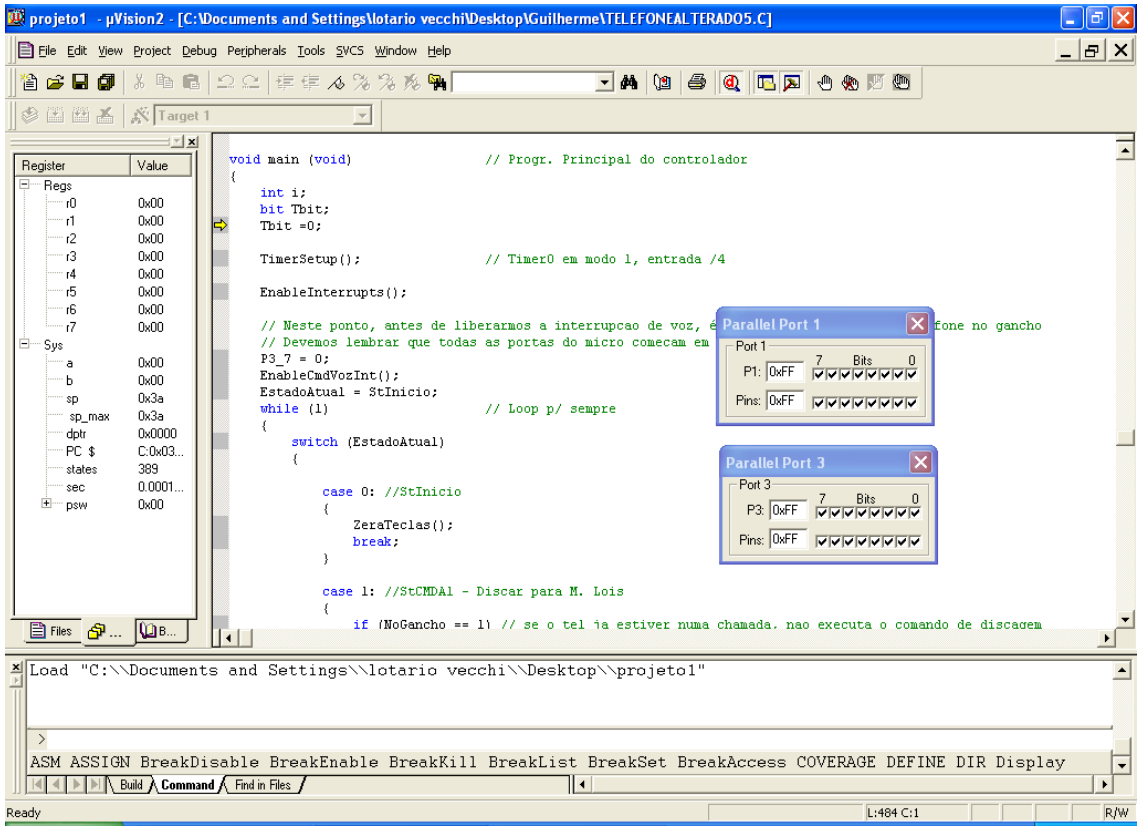


Figura 2.13 – Projeto aberto com as janelas dos ports P1 e P3 abertas

2.7 Características Gerais do sistema de reconhecimento de voz disponível : O Voice direct 364

2.7.1 Introdução

O módulo de reconhecimento de voz permite treinar até 15 palavras, podendo organizá-las em até 3 grupos de comandos. A cada vez que uma das palavras é reconhecida com sucesso, os pinos de saída do módulo vão para nível alto por 1 segundo. Se o módulo for usado de forma adequada, pode-se alcançar uma precisão acima de 99% no reconhecimento das palavras.

2.7.2 Modos de operação

Existem 3 modos de operação: Speaker Dependent (SD), Single Word continuous listening (SCL) e Multi-word continuous listening (MCL), comentados a seguir.

2.7.2.1 Modo de operação SD

Neste modo pode-se gravar até 15 frases de uma palavra.

2.7.2.2 Modo de operação SCL

Neste modo também se pode gravar até 15 frases. A diferença para o modo SD é que as frases têm duas palavras e a primeira palavra de todas as frases é a mesma.

2.7.2.3 Modo de operação MCL

Neste modo de operação, as frases de duas palavras podem ser organizadas em até 3 grupos de até 5 frases cada. A primeira palavra de cada frase pode ser diferente ao mudar-se de grupo de frases.

No Apêndice A tem-se uma descrição mais detalhada do chip reconhecedor de voz.

2.8 Ferramenta EZ Uploader

O código fonte dos programas aplicativos escritos em "C" é salvo no computador com a terminação ".c". Porém o arquivo que realmente será gravado no 8051 é um arquivo com terminação ".hex". Este é o formato binário do código fonte depois de compilado que é efetivamente escrito no 8051. O programa que é usado para gravar o 8051 é o EZ Uploader. A figura 2.14 mostra uma janela deste programa. Pode-se conectar o gravador de 8051 em qualquer porta serial do computador, clicar na opção send e escolher qual arquivo com terminação .hex será gravado no 8051. Para isso deve-se escolher a porta serial adequada.

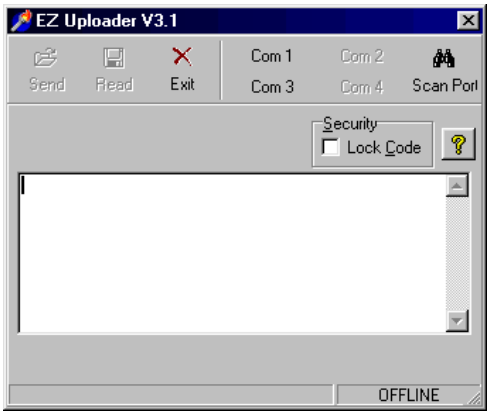


Figura 2.14 – Programa que é usado para gravar o programa no microcontrolador através do gravador de 8051.

No Apêndice B é apresentado detalhadamente o esquema elétrico do gravador de 89C2051 e 89C4051.

CAPÍTULO III – Descrição do sistema de Apoio a Deficientes Físicos

3.1 Introdução

O sistema proposto deve reconhecer comandos de voz, num dos formatos aceitos pela placa reconhecedora de voz (frases), e realizar a discagem telefônica ou realizar o comando de aparelhos eletrodomésticos pré-selecionados.

3.2 A seleção do sistema de reconhecimento de voz

Foi realizada uma busca de sistemas de reconhecimento de voz e chegou-se à conclusão de que o mais adequado seria um sistema com as seguintes características:

- Alta taxa de acerto no reconhecimento dos comandos;
- Baixo custo;
- Baixo consumo de energia, pois é destinado a sistemas portáteis;
- Dimensões reduzidas para ter portabilidade;
- Sistema reconhecedor de frases pré-gravadas pelo usuário e, portanto, dependente de usuário.

Um sistema com estas características é o sistema voice direct 364 – indicação [5] nas referências bibliográficas, cujos detalhes se encontram no Apêndice "A" e que foi brevemente abordado no capítulo 2. A figura 3.1 ilustra seus sinais de entrada e saída.

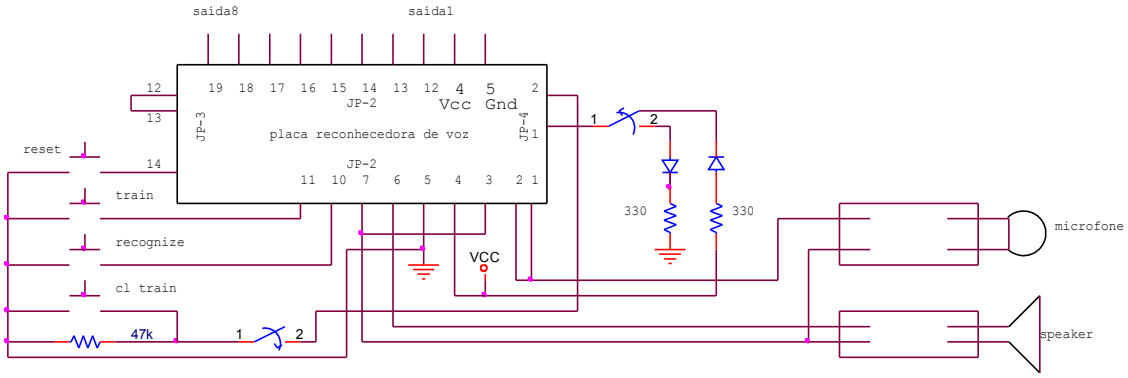


Figura 3.1 – Sistema reconhecedor de frases pré-gravadas

Os sinais de entrada são o microfone (usado para treino e uso normal) e as chaves e botões de configuração de operação (treino, reconhecimento, modos SD, SCL, MCL), os quais são descritos a seguir:

- CL Train – Usado no treino de palavras apenas nos modos SCL e MCL.
- Train – Usado no treino de palavras em todos os modos.
- Recognize – Usado no reconhecimento de palavras em todos os modos.

Os sinais de saída são os que apresentam os pulsos de reconhecimento das frases e um sinal de saída de voz. Este sinal é ligado diretamente a um alto falante e permite orientar o usuário no modo treinamento, indicando as etapas do treinamento e indicando se uma frase foi gravada corretamente ou não.

3.3 Macro Arquitetura do sistema

Tendo em vista que quem aciona o microcontrolador é o chip reconhecedor de voz e que quem controla os equipamentos é o microcontrolador, foi concebido um esquema para atingir o objetivo proposto e é ilustrado na figura 3.2.

A figura 3.2 mostra um esquema geral simplificado do projeto, mostrando como os componentes do projeto se ligam. O chip reconhecedor de voz envia comandos para o microcontrolador e este controla o celular, o telefone fixo e a Televisão.

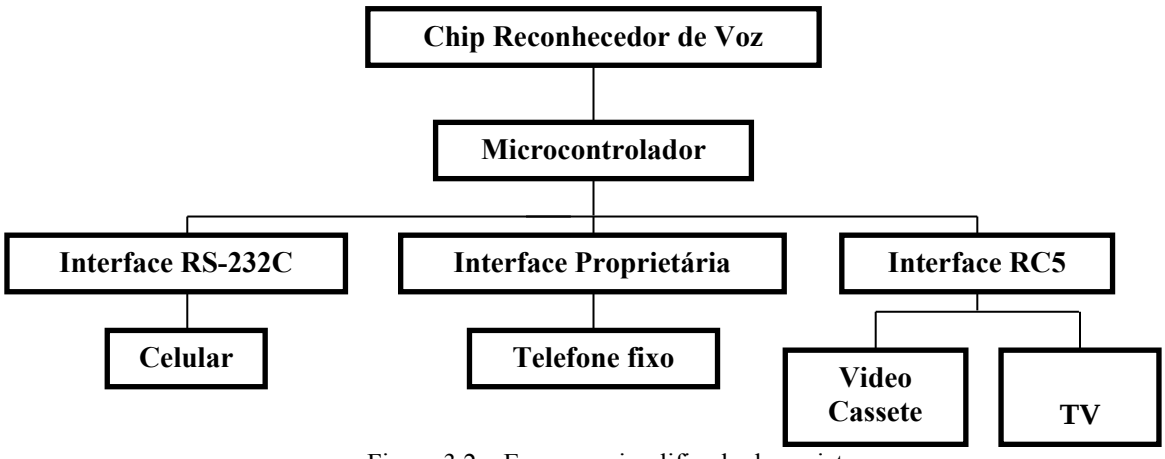


Figura 3.2 – Esquema simplificado do projeto

3.3.1 Interface Entre a Placa reconhecedora de frases e o microcontrolador

Cada frase gera uma combinação de saídas na placa reconhecedora de voz. Só um dos sinais da saída está ativo de cada vez. Isto permitiu utilizar um codificador para diminuir o número de entradas no microcontrolador e é ilustrado pela figura 3.10. Da mesma forma foi utilizado um circuito externo do tipo "nor" para concentrar as interrupções desses sinais em uma única entrada de interrupção ($\overline{\text{INT0}}$) do microcontrolador.

3.4 Comandando um telefone sem fio.

Dentre os equipamentos que podíamos comandar por voz, o telefone demonstrou ser um dos mais interessantes, porque é um aparelho muito usado na atualidade e uma pessoa com deficiência que prejudique ou impossibilite o uso das mãos não consegue discar os números. Um telefone sem fio é mais interessante ainda porque para uma pessoa tetraplégica, que use cadeiras de rodas, ele permite uma maior mobilidade por parte do deficiente. O telefone sem fio utilizado é o modelo General Eletric 27701GE3.

Em relação à figura 3.3, foi utilizado um decodificador para economizar pinos do microcontrolador e quando uma de suas saídas é ativada, o transistor ligado a essa saída através do resistor satura. Se tivéssemos utilizado um microcontrolador com mais pinos não teríamos necessidade do decodificador, mas não tínhamos nenhum disponível para o projeto.

Existe uma matriz de transistores e cada transistor representa uma tecla. No coletor de cada transistor está ligada uma coluna da matriz do teclado do telefone modificado e no emissor está ligada uma linha. Quando o transistor satura, é quase dado um curto entre uma linha e coluna, o que equivale a apertar uma tecla do telefone. O transistor correspondente à tecla zero e o transistor responsável por tirar o telefone do gancho foram ligados diretamente à saída do microcontrolador conforme mostra a figura 3.3.

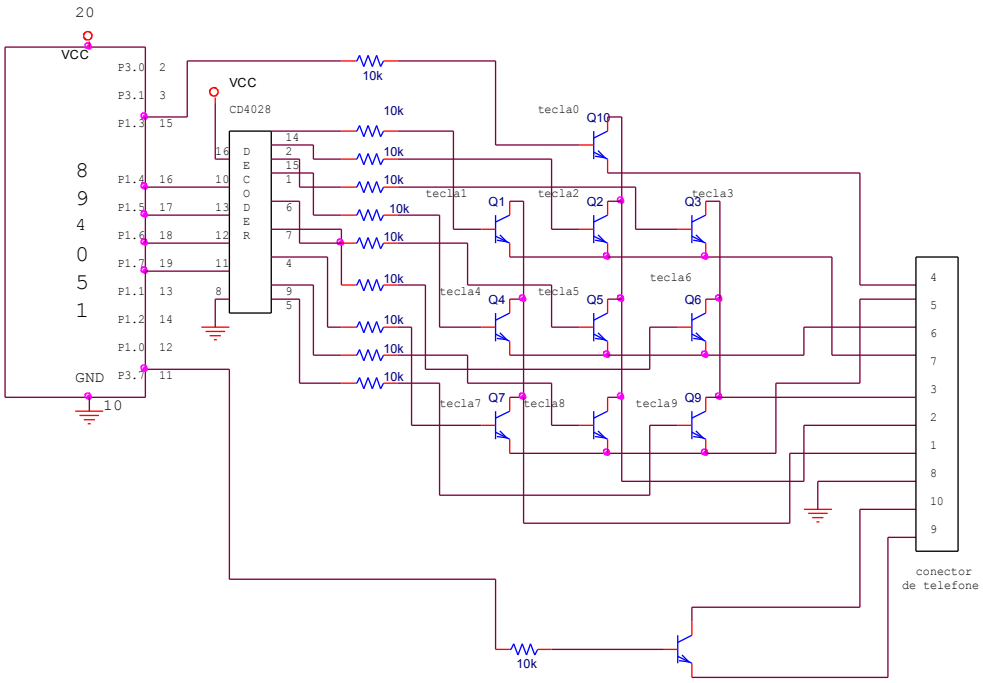


Figura 3.3 – Matriz de transistores

3.5 Comandando Alguns tipos de telefones celulares

Existem alguns tipos de telefones celulares, tais como o Ericsson T68 e o Nokia 3310 que permitem o controle do telefone através de portas de comunicação série. O T68 possui uma interface padrão RS232C, operando a 9600 bps. O telefone Nokia 3310 possui interface e protocolo diferentes do T68, denominados F-bus e M-bus – indicados nas referências bibliográficas pelo número [8] e descritos em mais detalhes no Apêndice C. Existem celulares com interfaces diferentes, como a interface USB que é mais complicada e por isso não foi utilizada.

3.5.1 Comandando o Telefone Nokia 3310

3.5.1.1 Conectando microcontroladores ao Nokia 3310

A maioria dos celulares da Nokia têm as conexões F-Bus e M-bus que podem ser usadas para conectar o celular no computador ou no microcontrolador. A conexão pode ser usada para controlar todas as funções do celular. O popular nokia 3310 tem as conexões para F-Bus e M-bus sob o suporte da bateria conforme mostra a figura 3.6.

Figura 3.4 – Os pinos 1, 2, 3 e 4 são usados para conexão F-BUS e M-BUS

O cabo que se conecta ao telefone entra por debaixo da bateria, conforme mostra a figura 3.5.

Figura 3.5 – O cabo F-Bus entra por debaixo da bateria

As figura 3.6 e 3.7 mostram o cabo e o celular separadamente.

Figura 3.6 – Cabo F-Bus

Figura 3.7 – Nokia 3310

3.5.1.2 As diferenças entre M-BUS e F-BUS

O M-Bus tem apenas um pino bi-direcional que serve tanto para transmissão como para recepção. Ela é mais lenta e é *half-duplex*. Apenas dois pinos do celular são usados. Um para terra e outro para dados. M-Bus roda a 9600bits/s, tem 8 bits de dados, um de paridade e um de stop. O F-Bus usa um pino para transmissão e um para recepção, além do pino de terra, sendo mais parecido com comunicação serial padrão. É mais rápido que o M-bus, usa a taxa de 115200bits/s, 8 bits de dados, sem bit de paridade e um de stop bit.

No Apêndice C temos um detalhamento maior do protocolo F-Bus e seus comandos.

3.5.2 Comandando o telefone Ericsson T68

3.5.2.1 Comandos AT

A tabela 3.4 mostra os comandos AT que foram usados no projeto. Usou-se apenas os comandos ATD, ATA e ATH que são responsáveis por fazer discagem no celular, atender uma ligação e desligar uma ligação. Além disso, existe uma série de comandos AT específicos para o celular T68.

Tabela 3.1 – Comandos AT usados no projeto

Comando AT	Função
ATD	Fazer uma discagem
ATA	Atender uma ligação
ATH	Desligar uma ligação

3.5.2.2 Circuito para comunicação serial entre o microcontrolador e o Ericsson T68

Para acionar o celular usamos comandos AT, que utilizam comunicação serial RS232. Os sinais TX e RX do microcontrolador vão para o chip MAX232, que converte os sinais de TTL para RS232, conforme mostra a figura 3.8. As saídas TX e RX do MAX232 vão para o conector DB-9, que se encontra num dos lados da caixa do sistema.

O cabo do celular tem um conector DB-9 Fêmea na ponta e um chip MAX3232 internamente, convertendo os sinais RS232 para sinais no nível de tensão do telefone, que são de 0V a 3V. Esse conector DB-9 Fêmea do cabo do celular vai se ligar ao conector DB-9 macho, o qual se encontra num dos lados do compartimento onde se situa o sistema.

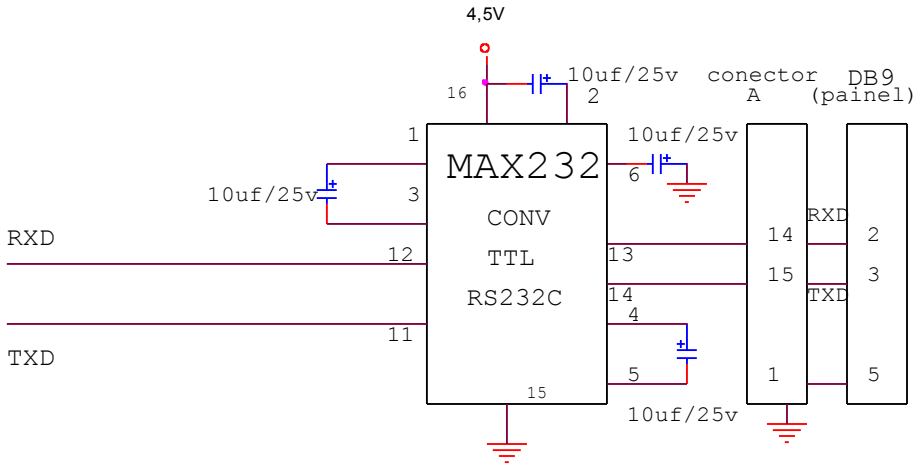


Figura 3.8 – Circuito de comunicação entre o microcontrolador e o celular T68

3.5.2.3 Cálculo do Baud-Rate do Ericsson T68

O cristal utilizado tem freqüência de 24MHz e esta é a freqüência necessária para a transmissão do protocolo RC5. O valor que deve ser carregado no timer para fazer a comunicação do 8051 com o celular depende dessa freqüência e do *baud-rate*. A fórmula abaixo mostra qual o valor que deve ser carregado no timer em função da freqüência do cristal e do *baud-rate*, considerando o *baud-rate* padrão de 9600 bits/s.

"Carga do Timer" (em decimal) = 256 – (Fclock*2^{SMOD}) / (384*Baud-Rate)

Foram usados os seguintes valores na fórmula:

- Fclock = 24MHz
- SMOD = 1
- Baud-Rate = 9600

Para estes valores acima, achamos o seguinte valor para carregar o timer:

- "Carga do Timer" = 243

Carregando o Timer com esse valor tornou-se possível fazer a transmissão serial entre o microcontrolador e o celular de modo correto.

3.6 Comando de Eletrodomésticos através do protocolo infravermelho RC5-Philips

Vários Equipamentos da Philips que utilizam controle remoto usam o mesmo protocolo RC5: TV, Videocassete, som etc. No nosso projeto, utilizamos uma TV, mas poderíamos comandar um videocassete da Philips também, bastando para isso fazer pequenas mudanças no código.

Para comandar a televisão, usamos um Led infravermelho. Não ligamos o Led diretamente no 8051 porque o microcontrolador não tem capacidade de fornecer corrente tão alta. Para isso usamos um transistor NPN que consegue fornecer tal corrente. O transistor utilizado foi um transistor de potência, que agüenta uma corrente mais alta: o BD 137.

Os resistores utilizados no circuito da figura 3.9 foram calculados de forma que o transistor operasse chaveando e quando este estivesse conduzindo, passasse uma corrente de aproximadamente 200mA pelo Led, o que faz o Led ter um bom alcance.

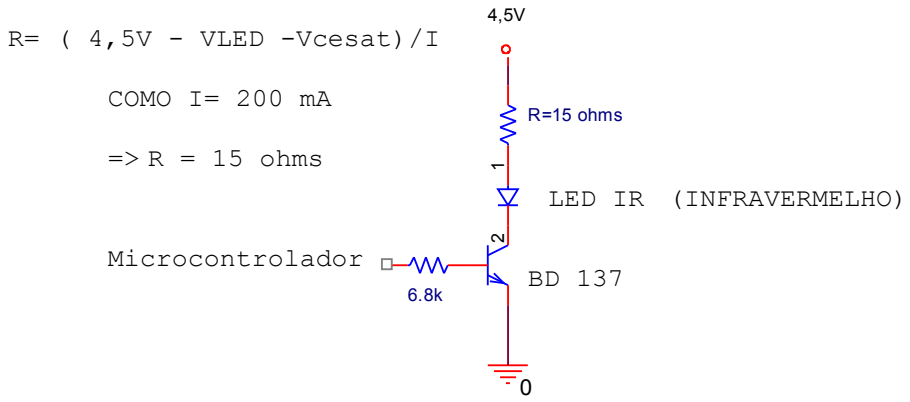


Figura 3.9 – Circuito de ativação do infravermelho

3.7 Esquema eletrônico do circuito

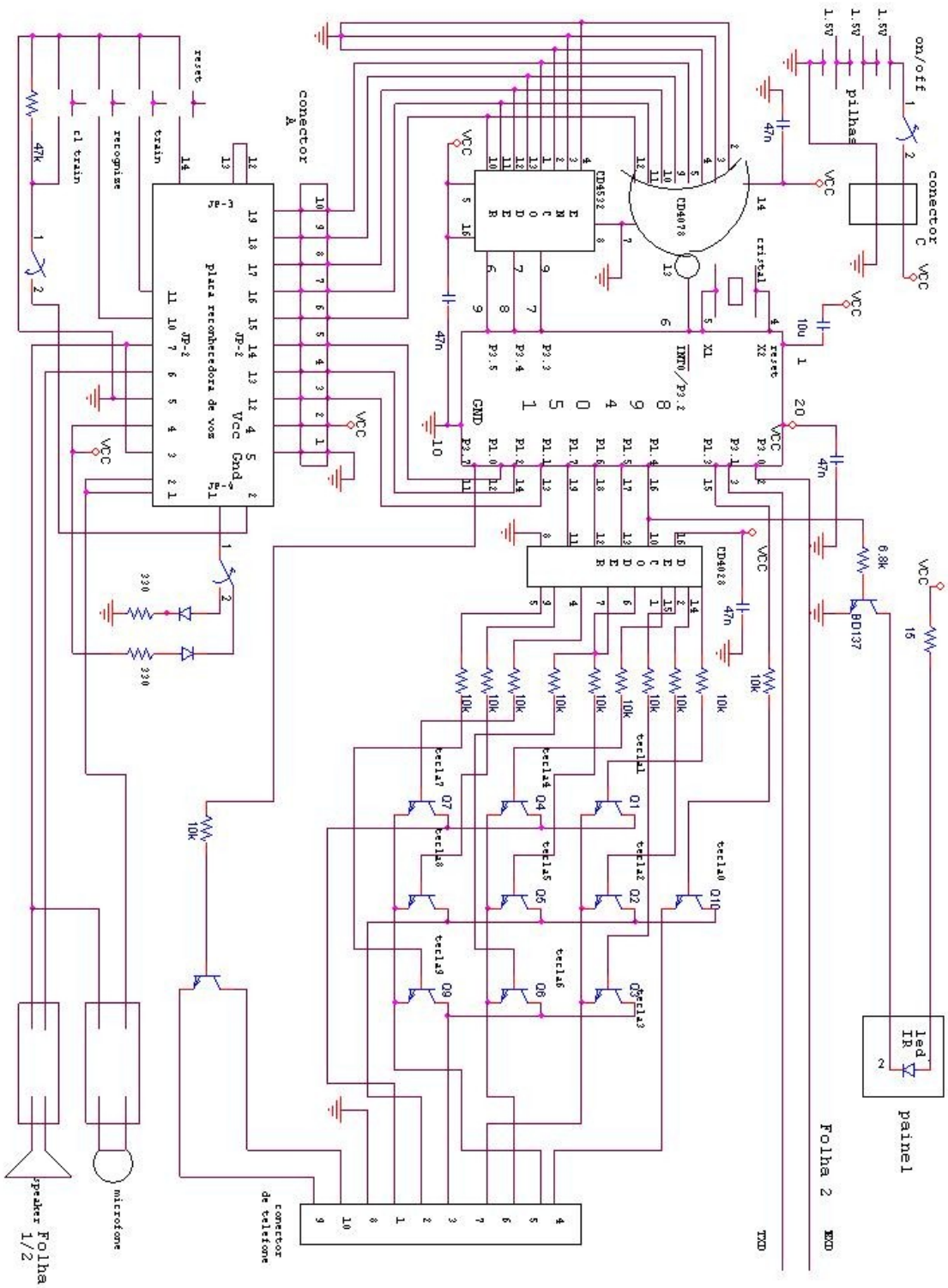


Figura 3.10 – Esquema eletrônico do circuito

3.8 Esquema eletrônico do circuito (Continuação)

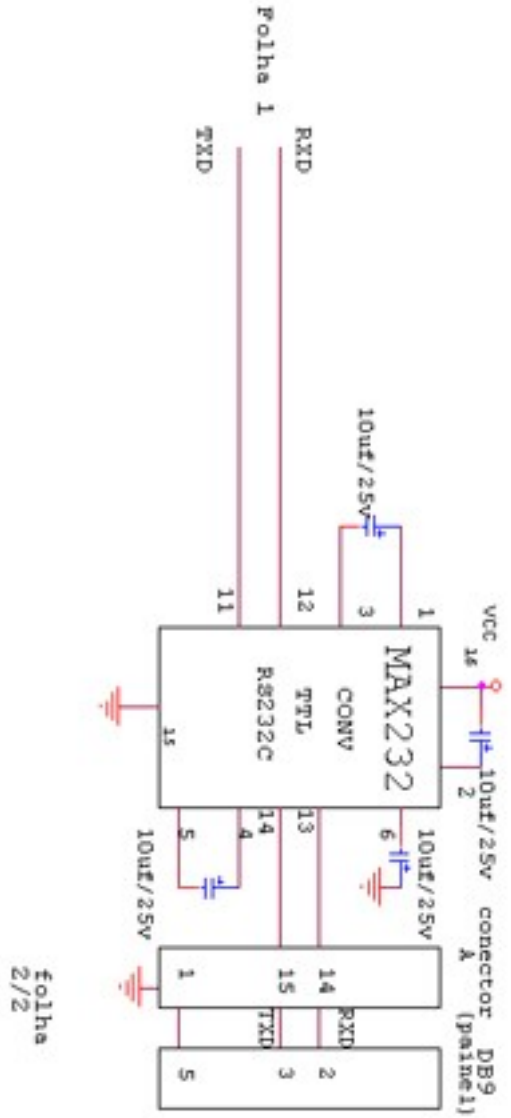


Figura 3.11 – Esquema eletrônico do circuito

O programa de controle do sistema pode ser visto no Apêndice D

CAPÍTULO IV – Testes e resultados

4.1 Introdução

Este Capítulo apresenta uma descrição da montagem final do sistema, onde se ilustra as interconexões dos principais módulos e o empacotamento utilizado.

O Capítulo também descreve os testes que foram realizados com o telefone sem fio, o telefone celular e o comando de uma televisão, utilizando comunicação via raios infravermelho.

4.2 Montagem final do sistema

A figura 4.1 mostra os módulos da caixa do sistema e interconexões.

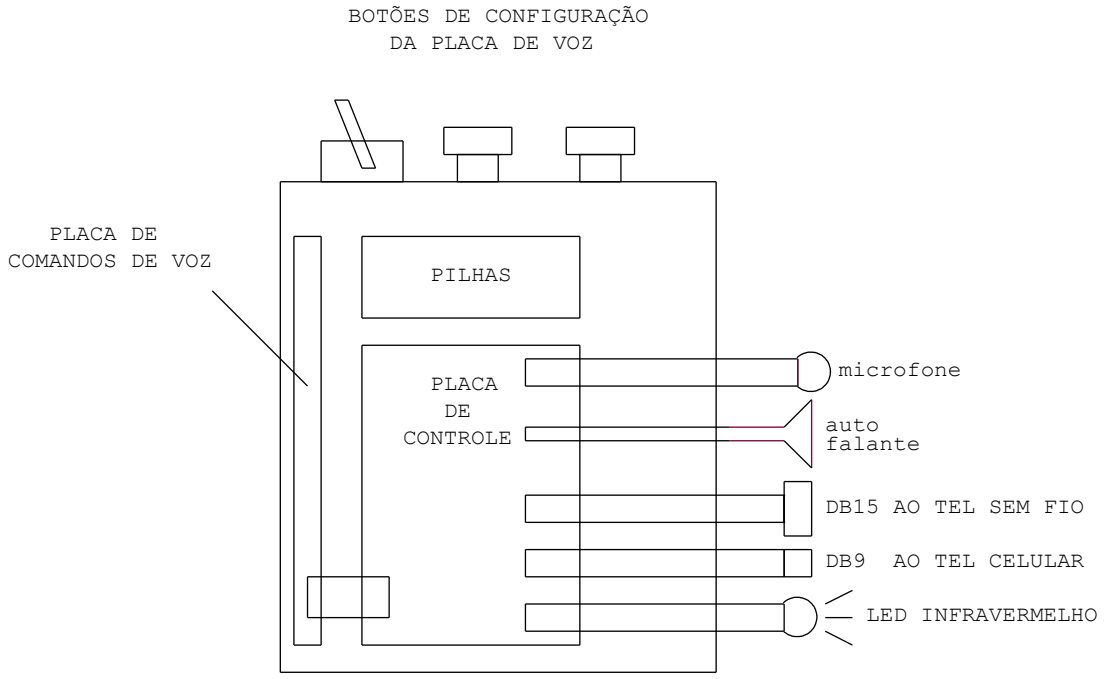


Figura 4.1 – Módulos da caixa do sistema e interconexões

Para que fosse portátil, buscamos construir o sistema com uma embalagem bem pequena e leve e o conseguimos com uma caixa de plástico com dimensões 8.5 cm x 12 cm x 5 cm, tal como ilustrado nas figuras 4.2, 4.3 e 4.4.

A figura 4.2 mostra o compartimento do projeto de lado com o conector DB9 que vai se ligar no cabo do celular.

Figura 4.2 – Compartimento mostrado de lado

A figura 4.2 mostra a tampa do compartimento do projeto, onde se encontra o LED infravermelho que comanda a televisão.

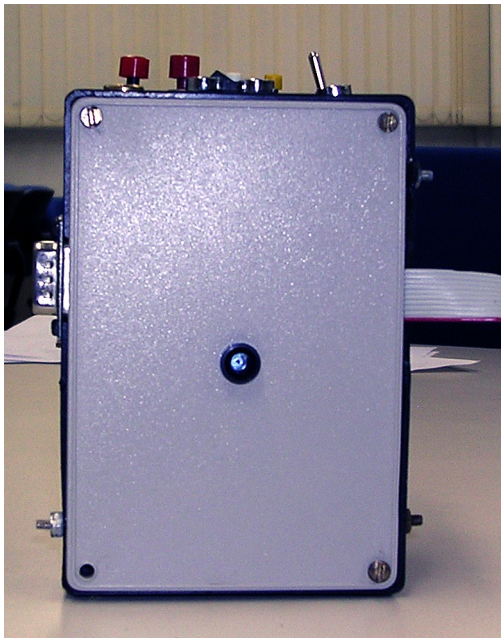


Figura 4.3 – Tampa do compartimento

A figura 4.3 mostra o compartimento por dentro, com todos os seus componentes

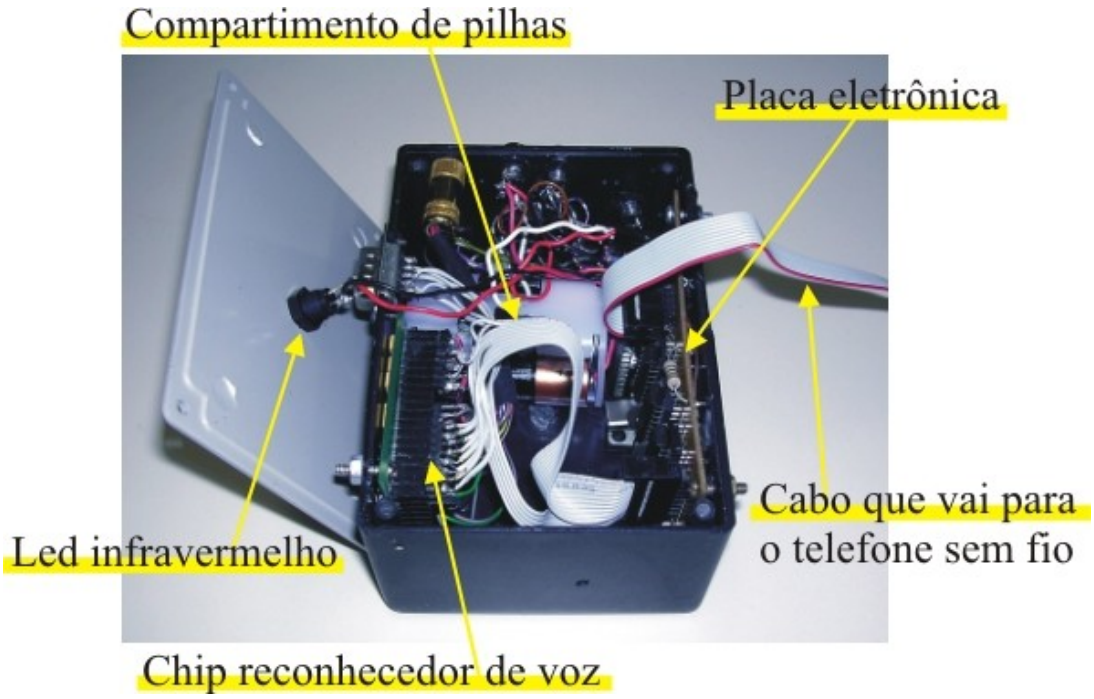


Figura 4.4 – Compartimento mostrado por dentro

A placa de comando do sistema foi construída com uma *protoboard* com fios soldados, ao invés de *wire-wrapp* para obter maior compactação, conforme ilustra a figura 4.5.

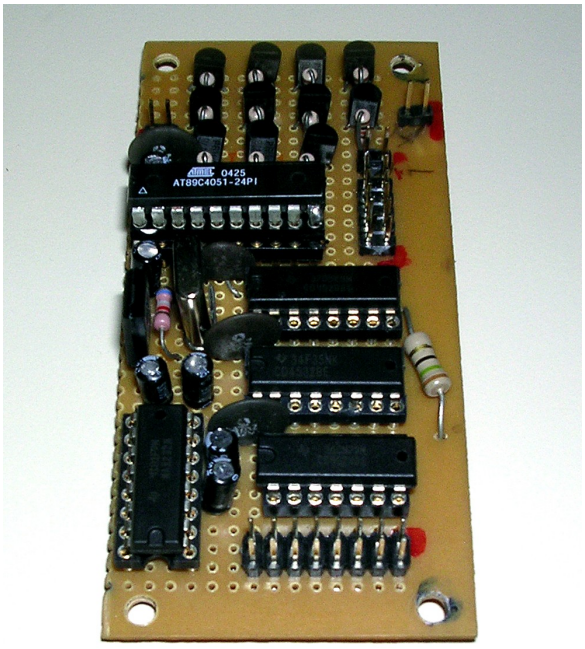


Figura 4.5 – Placa de controle

As conexões entre a placa de controle e os conectores são realizadas por meio de cabos planos (*flat cables*), por sua facilidade de conexão, montagem e compatibilidade (figura 4.4).

A caixa do sistema possui conectores com o telefone sem fio (conector DB15), com o telefone celular (DB9) e conectores com os microfones e fones de ouvido externos. Além disso, há ainda a conexão da placa de controle com o LED infravermelho.

A interconexão entre a placa de controle e a placa reconhecedora de comandos de voz também é feita por meio de um cabo plano – *flat cable*.

4.3 Testes com o Telefone sem fio

A figura 4.6 mostra o telefone sem fio usado no projeto.

Figura 4.6 – Telefone sem fio usado no projeto

A figura 4.7 mostra as palavras que foram gravadas para comandar o telefone sem fio. Por exemplo, para ligar para o telefone de alguém chamado Manuel basta falar "Telefone Manuel" e assim por diante.

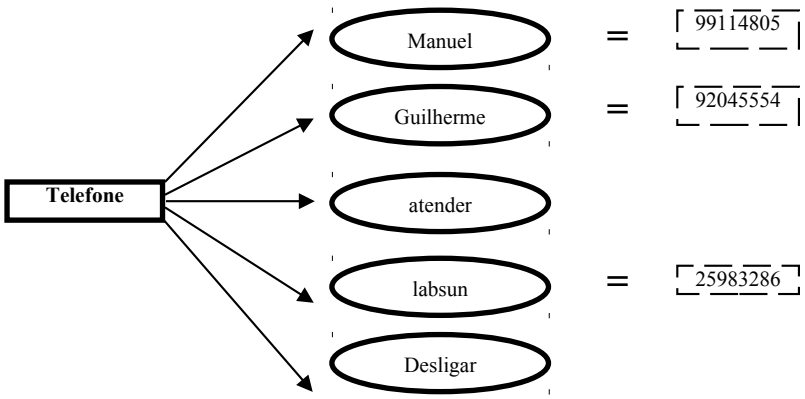


Figura 4.7 – Palavras que foram gravadas para comandar o telefone sem fio

4.4 Testes com o telefone celular

4.4.1 O cabo do celular

A figura 4.8 mostra o esquema eletrônico do cabo. O cabo do celular possui internamente o chip max3232 que converte sinais RS232 para sinais no nível de tensão do celular, que são de 0V a 3V.

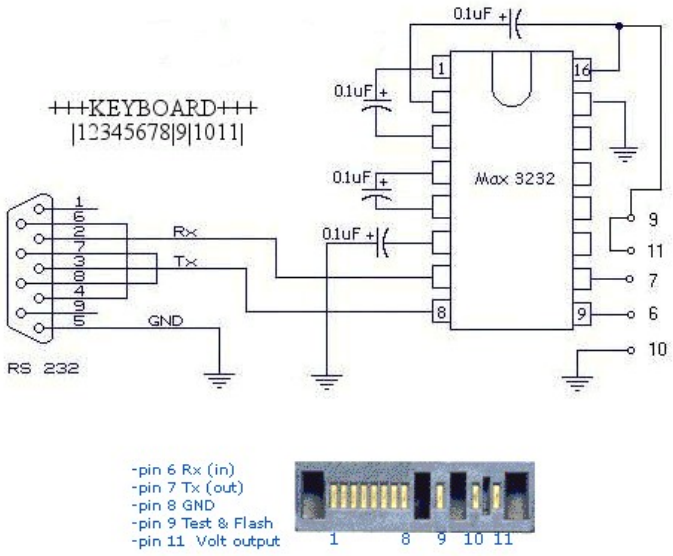


Figura 4.8 – Esquema eletrônico do cabo do celular

4.4.2 Conversão do sinal do 8051 para o celular

A figura 4.9 mostra como o sinal do microcontrolador é convertido para o do telefone, pois os níveis de tensão do microcontrolador são de 0V a 5V e os níveis de tensão do telefone são de 0V a 3V. Primeiro o sinal do microcontrolador é convertido do padrão TTL para o RS232 através do chip MAX232. Esse sinal RS232 será convertido para o nível de

tensão do celular através do chip max3232, que converte sinais RS232 para o nível de tensão do celular.

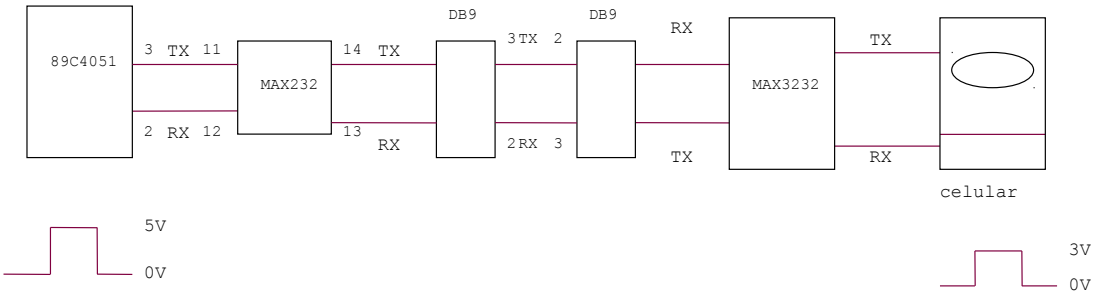


Figura 4.9 – sinal do 8051 convertido para o celular

4.4.3 protocolo de comunicação F-BUS e M-BUS da nokia

Antes de tentar-se a comunicação do 8051 com o aparelho celular T68 da Ericsson usando comandos AT, tentou-se sem sucesso fazer a comunicação do 8051 com o aparelho celular da Nokia, usando protocolo de comunicação F-BUS e M-BUS. O grande problema é que estes protocolos não são abertos do mesmo modo que os comandos AT, pois a Nokia não os divulga e não tínhamos certeza se o material que tínhamos da internet estava correto.

Apesar disso, tínhamos um programa chamado *Gammu* que funcionava para fazer a comunicação do computador com o celular. Para tentarmos descobrir como era o protocolo de comunicação, fizemos o computador se comunicar com o celular através de sua porta serial e em outro computador monitoramos tanto o que o celular estava recebendo como enviando, através das portas seriais desse outro micro, conforme mostrado na figura 4.10.

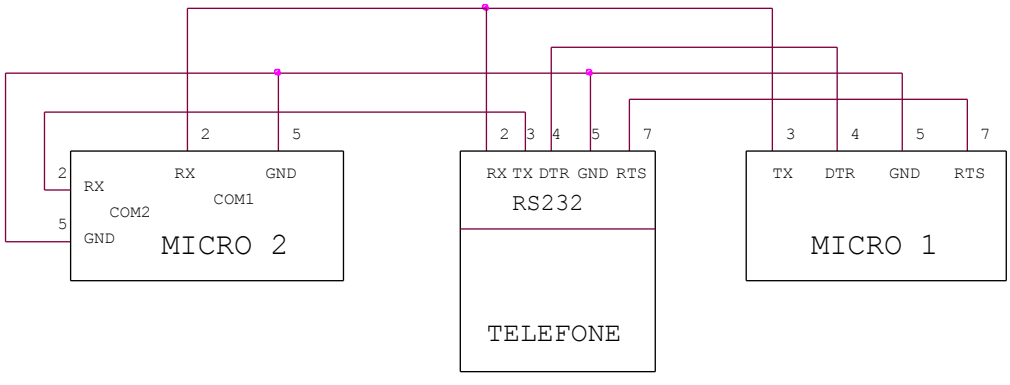


Figura 4.10 – O micro 1 se comunica com o celular e
o micro 2 serve para monitorar o que o celular recebe e envia.

Para ligar o celular tanto no micro, quanto no conector DB-9 do sistema a ser projetado, foi necessário montar um circuito que contivesse o chip max3232, que converte sinais RS232 para o nível de tensão do celular. A figura 4.11 mostra o circuito que foi montado quando fizemos o computador se comunicar com o celular usando F-BUS.

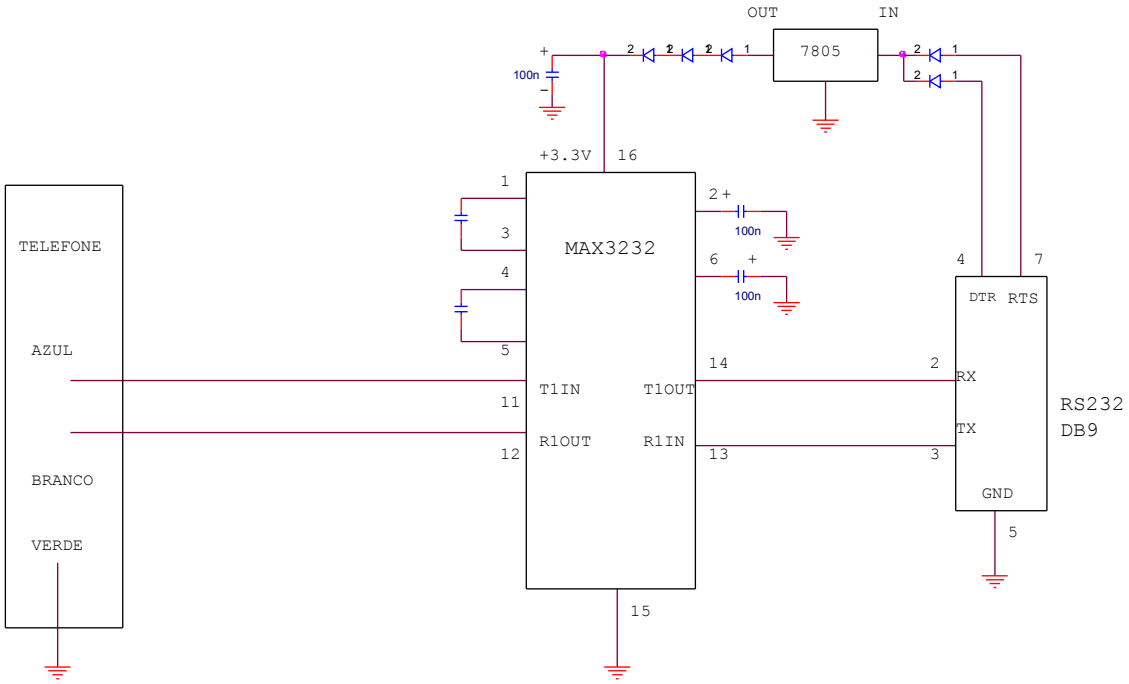


Figura 4.11 – circuito montado para fazer a conversão do sinal RS232 para o celular Nokia 3310.

Contudo, desistimos de fazer o 8051 se comunicar com o celular usando F-BUS porque o *baud-rate* do FBUS era 115200 bits/s e para conseguir-se fazer o 8051 transmitir a essa taxa seria necessário um cristal de freqüência muito acima da máxima permitida pelo microcontrolador usado. Tentamos usar M-BUS, que utiliza a taxa padrão de 9600 bits/s e é *half duplex*. Para isso "curtamos" TX com RX no celular e usamos o mesmo circuito que havia sido montado para o F-BUS. Entretanto, apareceram problemas de comunicação que não conseguimos identificar ao usarmos o programa *Gammu*.

Por conta do cronograma do projeto, acabamos desistindo de usar o celular da Nokia e passamos a usar o celular T68 da Ericsson que aceitava comandos AT. Mas mesmo assim, conseguimos verificar que o protocolo de comunicação da Nokia era realmente o que tínhamos obtido na internet.

4.4.4 Utilizando o Programa Hyper terminal Para auxiliar nos testes

Antes de programarmos o 8051 com o código que continha os comandos AT, fez-se um teste no Hyper terminal do windows, que permite a comunicação serial – entre outras possibilidades –, conforme mostrado na figura 4.12.

Para confirmar se o celular aceitava comandos AT digitou-se AT. Como houve um OK como resposta, ficou confirmado que ele aceitava comandos AT. Depois usamos o comando ATD para fazer uma discagem. Quando desligamos o celular enquanto ele estava chamando aquele número, apareceu a mensagem “no carrier”, mas a discagem foi feita com sucesso. Depois ligamos para o celular, esperamos o celular tocar e usamos o comando ATA para atender a chamada. A resposta OK confirmou o atendimento da chamada. Depois usamos o comando ATH para desligar a ligação. Um novo OK como resposta foi obtido e a ligação foi encerrada.

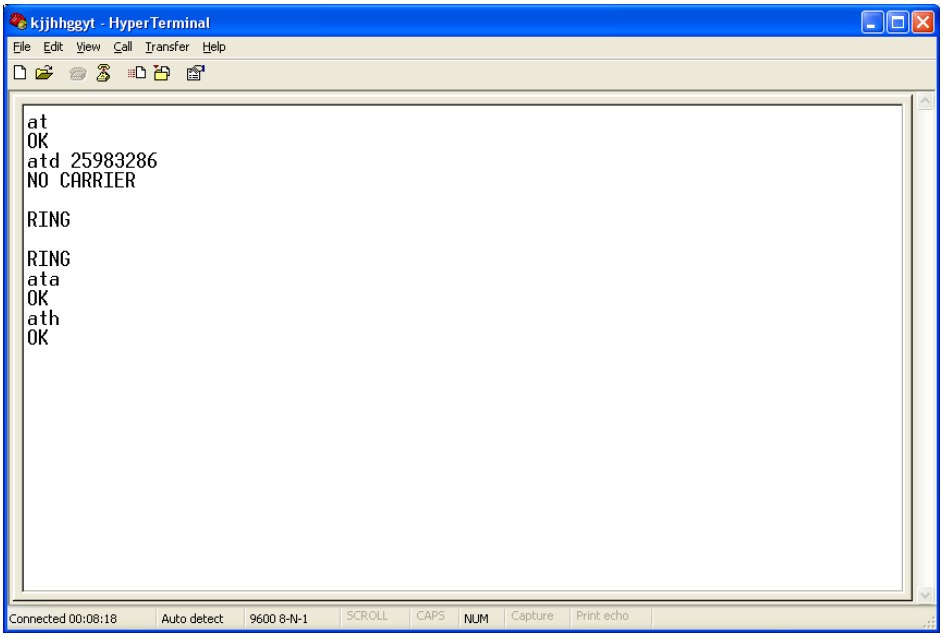


Figura 4.12 – Teste usando o hyper terminal.
Neste teste foram usados os comandos AT ATD, ATH e ATA.
Estes comandos são os que vão ser utilizados no projeto.

4.4.5 Utilizando o Simulador Isis Proteus para Depurar o Sistema

O Proteus Isis Professional é um software com a finalidade de simular circuitos com microcontroladores entre outros componentes eletrônicos. Isto evita a gravação do microcontrolador desnecessariamente.

O Circuito da figura 4.13 foi montado no Proteus para testar o programa do 8051 que seria gravado no chip. A montagem deste circuito permitiu ligar o celular na porta serial do computador e verificar se o programa do 8051 funcionava usando comandos AT, pois direcionamos o conector DB-9 da figura para a porta serial do computador. Este procedimento evitou as várias gravações do chip, o que aumentou sua vida útil, além de reduzir o trabalho.

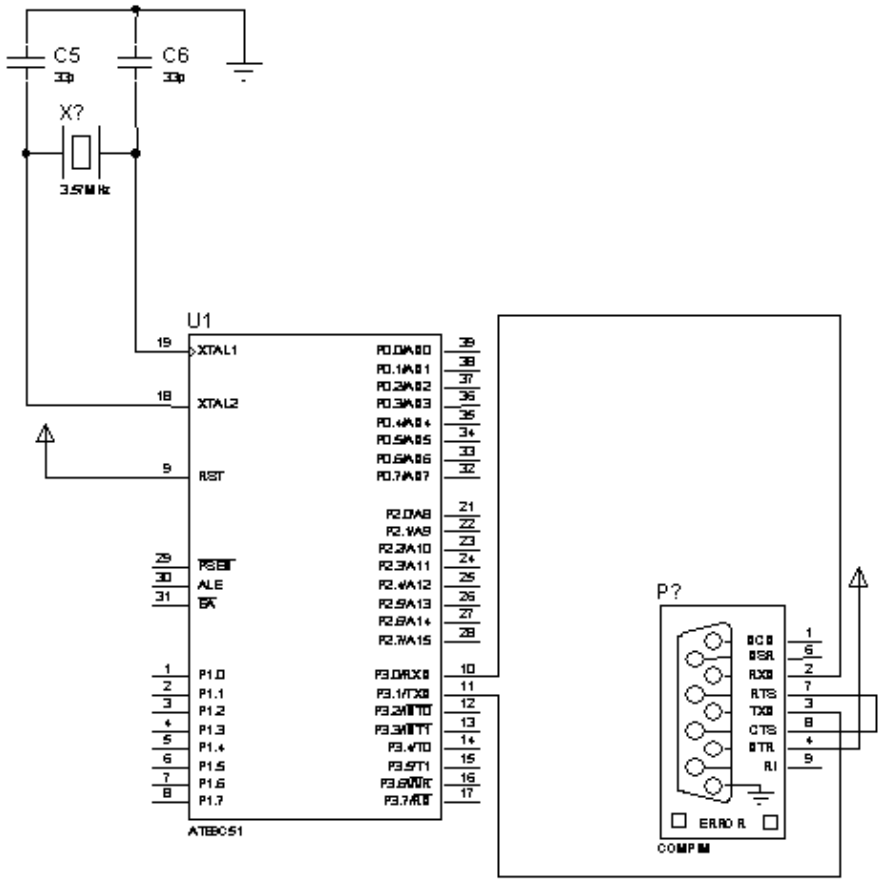


Figura 4.13 – Circuito montado no proteus que foi usado para testar o programa do 8051 que usava comandos AT.

A figura 4.14 mostra o celular usado no projeto

Figura 4.14 – Ericsson T68: O Celular usado no projeto

A figura 4.15 mostra as palavras que foram gravadas para comandar o celular. Para ligar para o Manuel pelo celular basta falar: “Celular Manuel”. Para desligar uma ligação basta falar: “Celular desligar” e assim por diante.

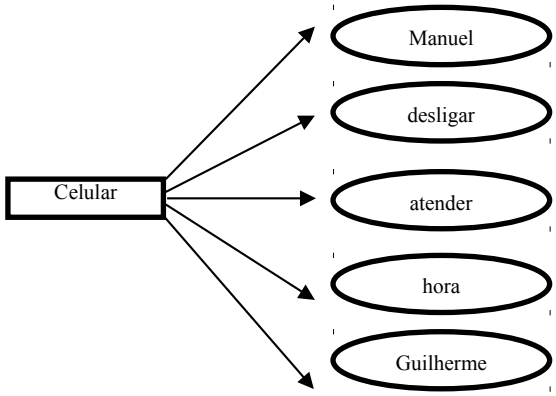


Figura 4.15 – Palavras que foram gravadas para comandar o Celular

4.5 Testes com infravermelho

O circuito da figura 4.16 foi montado no Proteus para verificar se o protocolo de comunicação do led infravermelho estava sendo enviado corretamente e se freqüência da portadora era mesmo de 36KHz. Para isso usamos o osciloscópio do próprio Proteus.

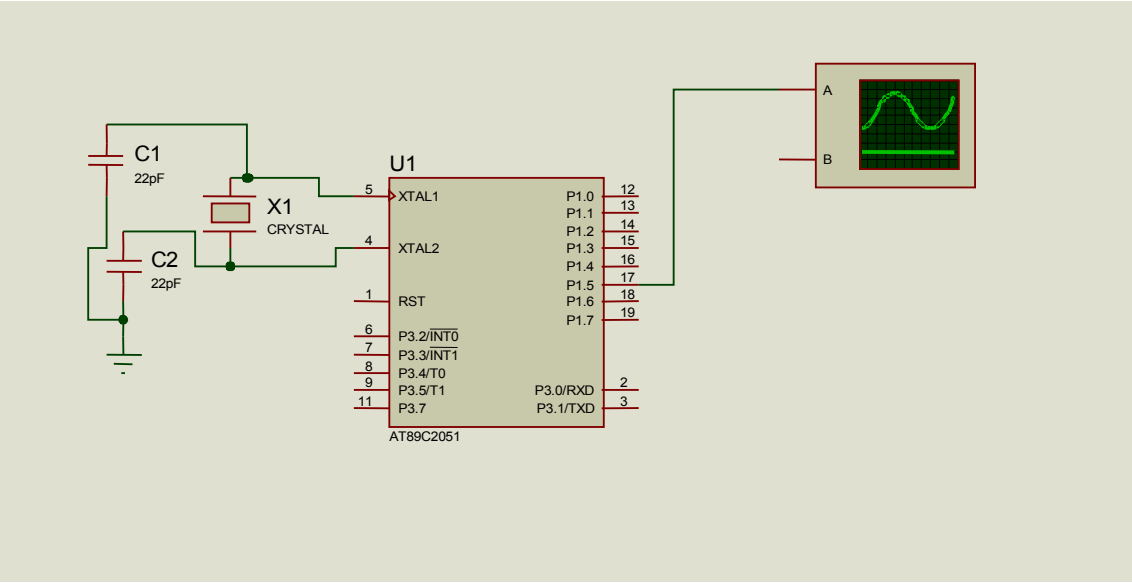


Figura 4.16 – circuito montado no proteus para testar o envio correto
do protocolo do infravermelho

A figura 4.17 mostra o sinal modulado no protocolo RC5. Foi enviada uma seqüência de "1" lógicos. O período medido de um bit foi aproximadamente 1.8ms, como esperava-se.

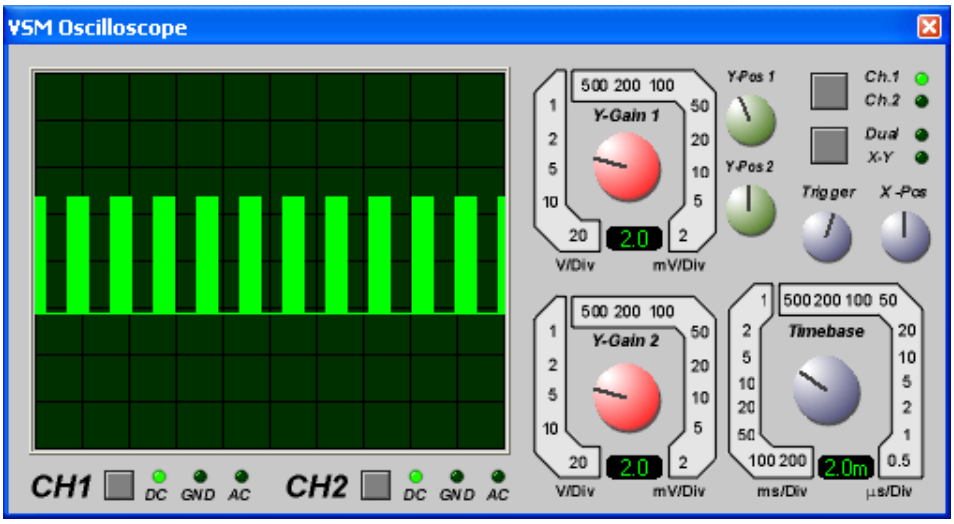


Figura 4.17 – Sinal medido no osciloscópio modulado no protocolo RC5

A figura 4.18 mostra o mesmo sinal que a figura 4.9, só que a escala de tempo foi alterada para que fosse possível calcular a freqüência da portadora.

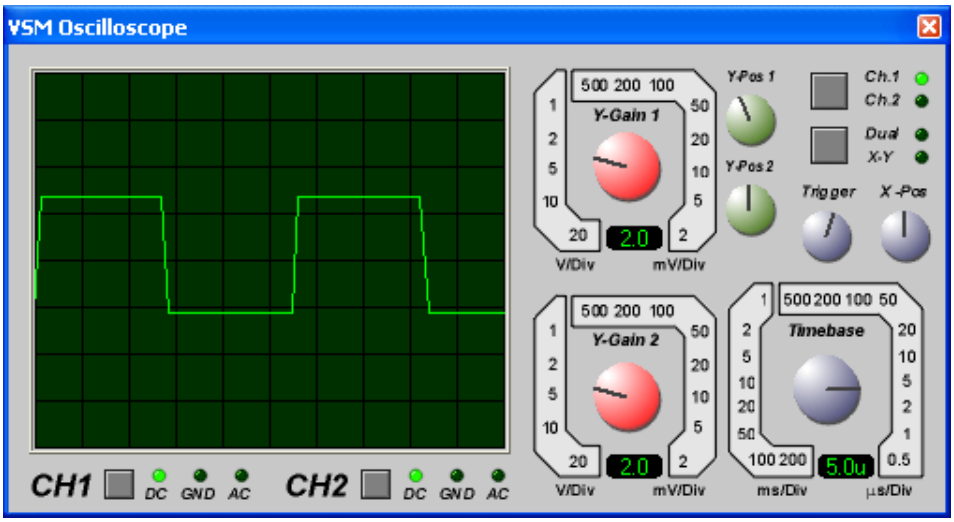


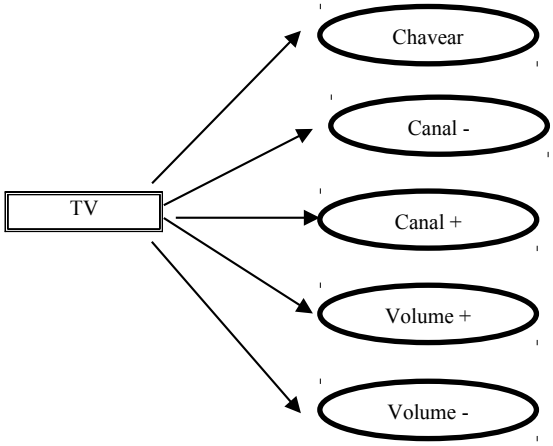
Figura 4.18 – O período medido da portadora, em torno de 28us, correspondendo a uma freqüência de aproximadamente 36KHz.

A TV utilizada no projeto foi uma TV da marca Philips de 20 polegadas, modelo 20PT120A /78R, conforme mostra a figura 4.19.

Figura 4.19 – TV utilizada no projeto

A figura 4.20 mostra as palavras que foram gravadas para comandar a televisão. Para ligar ou desligar a televisão basta falar: “TV Chavear”. Para diminuir um canal basta falar: “TV Canal - ” e assim por diante.

Figura 4.20 – Palavras que foram gravadas para comandar a televisão


TV
Chavear
Canal -
Canal +
Volume +
Volume -

CAPÍTULO V – Conclusões

Este projeto apresentou a análise e implementação de um sistema de apoio a deficientes físicos ativado por comandos de voz, ou mais especificamente ativado através do reconhecimento de frases pré-gravadas pelo usuário.

O fato de o usuário ter que gravar antecipadamente as frases a serem reconhecidas não representa uma restrição do sistema, pois a programação é muito fácil e é uma garantia que o sistema dificilmente obedecerá os comandos de outra pessoa.

Todos os objetivos propostos no início deste trabalho foram plenamente alcançados, com exceção da utilização de telefones Nokia 3310, o que não representa um problema pois o telefone T68 atende perfeitamente e possui conexão externa.

Nos testes realizados , a placa reconhecedora de frases pré-gravadas mostrou ter bom desempenho e o próximo passo será colocá-la em uso real para posteriormente aprimorar o sistema.

Do ponto de vista didático, este trabalho proporcionou a oportunidade de travar conhecimento com diversas tecnologias e ferramentas de desenvolvimento, tais como microcontroladores e programação em linguagem de alto nível " C " , interface serial RS232C, controle de eletrodomésticos através de infravermelho. Esta variedade de temas abordados no projeto final contribuiu sisnificativamente para a consolidação e aprimoramento do aprendizado no curso de Engenharia Eletrônica do Del/UFRJ e no Núcleo de computação Eletrônica (NCE/UFRJ).

Quanto a trabalhos futuros, o projeto deve ser revisado para utilizar um microcontrolador PIC para eliminar a lógica externa hoje existente e utilizar uma freqüência de operação menor para baixar o consumo de energia. Esta última questão será possível utilizando o modulador PWM do PIC, dispensando a necessidade de um cristal de alta freqüência para gerar os tempos da modulação RC5 da Philips.

Um outro aprimoramento do sistema será testar o uso de outros tipos de telefones celulares para tornar o sistema mais utilizável.

Também é possível expandir o número de comandos que a placa reconhecedora de frases aceita sob controle do microcontrolador (admite até 64 comandos). Isto permitirá controlar mais dispositivos como por exemplo cadeira de rodas, luzes, etc.

Para torná-lo um produto comercial é necessário realizar uma reengenharia do produto, tornando-o menor, mais leve, com menor consumo e mais confiável.

Referências Bibliográficas

[1] Adel S .Sedra ,Kenneth C .Smith.
Microeletronics Circuits, Oxford University Press, Fourth Edition, 1998.

[2] Microcontrolador 8051 – detalhado
Denys E. C. Nicolosi, Érica.

[3] Embedded.C ,
Michael.J.Rant.Addsan – Addison-Wesley.

[4] Microcontroladores 8051
Salvador . P.Gimenez, Prentice Hall.

[5] Manual do chip reconhecedor de voz: Voice Direct 364
http://www.selectronic.fr/includes_selectronic/pdf/Sensory/Voicedirect364.pdf

[6] Schildt, H –C completo e total,São Paulo, Mcgraw Hill, 1990.

[7] Mobile phone T68 Developers' Guideline
AT comands Online Reference disponível no endereço:
http://quiezent.dyndns.org/files/t68_at_commands_dg_ver1_external.pdf

[8] http://www.embedtronics.com/nokia/fbus.html

[9] http://www.gadgets.demon.co.uk/nokia21xx/protocol.html

[10] http://www.mzeditora.com.br/artigos/embut.htm

[11] http://users.pandora.be/davhomepage/index.htm

[12] http://www.ustr.net/infrared/infrared1.shtml

[13] http://www.hindawi.com/journals/es/si/portable.html

[14] http://www.ece.auckland.ac.nz/~p4p_2005/archive/reports2003/category_ems.htm

[15] http://www.tmaa.com/tts/News/Fonix/zygo.txt

[16] http://intervox.nce.ufrj.br/motrix/

[17] http://www.stakes.fi/include/i1_ch_17.html

[18] http://atp.caeds.eng.uml.edu/projects.html

[19] http://ostc.thaiembdc.org/newsletter/0503_01.html

[20] http://www.businessweek.com/1998/08/b3566022.htm

Apêndice

A

Documentação Sobre o chip Reconhecedor de voz

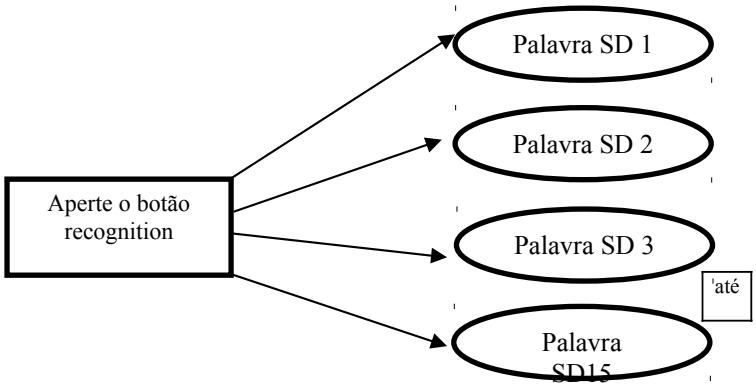
Apêndices

A - Documentação sobre o chip reconhecedor de voz

Modos de Operação:

Existem 3 modos de Operação: Speaker Dependent (SD), Single Word Continuous listening (SCL) e Multi-word continuous listening (MCL).

- **Modo de Operação SD**:



Modo de operação SD

O roteiro abaixo é um exemplo de como gravar palavras no modo SD

Gravando Palavras no modo SD:

Ação	pressione o botão train	começa a gravar
Voice direct 364	"say Word one"	gravando 1ª palavra
Usuário	"ligue"	
Voice direct 364	"repeat"	
Usuário	"ligue"	
Voice direct 364	"say word two"	gravando 2ª palavra
Usuário	"desligue"	
Voice direct 364	"repeat"	
Usuário	"desligue"	
Voice direct 364	"say word three"	gravando 3ª palavra
Usuário	"luzes"	
Voice direct 364	"repeat"	
Usuário	"luzes"	
Ação	pressione o botão train	encerra a gravação

Obs: Voice direct 364 continuará pedindo novas palavras até que um máximo de 15 palavras seja gravado – o que corresponde a 15 posições de memória. Para parar a gravação, basta pressionar train novamente. Pode-se continuar gravando palavras, de onde se parou, a qualquer momento, bastando para isso apertar train ou cl train (modos SCL e MCL).

As palavras podem ter uma pausa no meio, contanto que esta seja menor que 0.5s. Por exemplo, a palavra "Luiz Paulo" pode ser programada contanto que o intervalo de tempo entre "Luiz" e "Paulo" seja menor que 0.5s. O intervalo total de cada palavra pode ser no máximo de 2.5s.

Saídas que são ativadas no modo SD

Palavra SD	Saída 1	Saída 2	Saída 3	Saída 4	Saída 5	Saída 6	Saída7	Saída 8
Palavra SD 01	A							
Palavra SD 02		A						
Palavra SD 03			A					
Palavra SD 04				A				
Palavra SD 05					A			
Palavra SD 06						A		
Palavra SD 07							A	
Palavra SD 08								A
Palavra SD 09	A							A
Palavra SD 10		A						A
Palavra SD 11			A					A
Palavra SD 12				A				A
Palavra SD 13					A			A
Palavra SD 14						A		A
Palavra SD 15							A	A

A indica que a saída está ativada.

Reconhecendo palavras no modo SD:

Quando o botão recog for pressionado, o reconhecimento de palavras começa. – o Led verde permanece apagado.

O Voice direct 364 dirá: "say a word". Se for dita uma palavra não programada, será dita uma mensagem "word not recognized". Se a palavra dita for uma das programadas, uma ou duas das 8 saídas serão ativadas (conforme mostra a tabela 2.1) e será dita uma mensagem indicando o número da palavra.

Tanto durante a gravação como durante o reconhecimento de palavras em qualquer modo – SD, SCL, MCL – alguns erros podem acontecer. Se acontecer um erro, este erro será falado: “spoke to soon”(falou cedo demais), “please talk louder” (por favor fale mais alto). Se depois do Voice direct 364 indicar *repeat*, for dita uma palavra diferente da primeira, o programa dirá a mensagem: “training error” etc. Quando três erros forem cometidos seguidamente, a gravação termina.

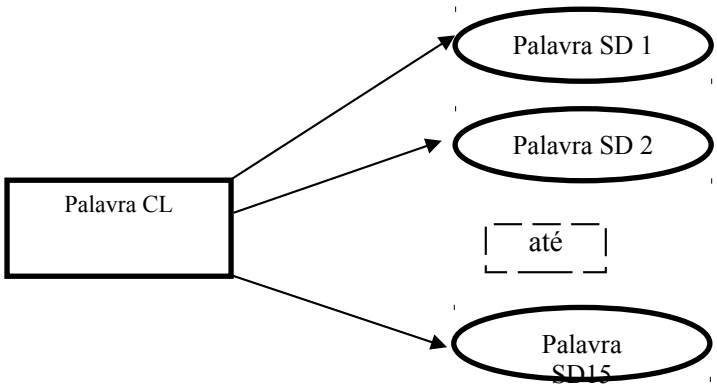
O botão reset:

Sempre que estiver gravando em um modo e desejar-se gravar em outro modo, é necessário pressionar o botão *reset*. Por exemplo: Se estiver gravando no modo SCL e desejar-se fazer gravação de palavras no modo MCL, é necessário colocar a chave do circuito na posição correspondente ao modo MCL e pressionar o botão reset.

Apagando palavras programadas:

Para apagar uma programação, basta pressionar os botões *regog* e *train* simultaneamente por pelo menos 100ms. Não é possível apagar uma determinada palavra ou um determinado subconjunto de palavras do conjunto total de palavras gravadas. É possível apenas apagar o conjunto total de palavras gravadas. Neste caso é dita a mensagem: “memory erased” (memória apagada).

- **Modo de operação SCL:**



modo de operação SCL

O roteiro abaixo é um exemplo de como gravar palavras no modo SCL

Gravando palavras no modo SCL:

Ação	pressione o botão cl train	começa a gravação CL
Voice direct 364	"say Word one"	gravando palavra CL
Usuário:	"Luiz Paulo"	
Voice direct 364	"repeat"	
Usuário:	"Luiz Paulo"	
Ação	Aperte o botão train	começa a gravação SD
Voice direct 364	"Say Word one"	gravando 1ª palavra SD
Usuário:	"começo"	
Voice direct 364	"repeat"	
Usuário:	"começo"	
Ação	pressione o botão train	continua gravação SD
Voice direct 364	"say Word two"	gravando 2ª palavra SD
Usuário:	"desligue"	
Voice direct 364	"repeat"	
Usuário:	"desligue"	
Ação	pressione o botão train	continua gravação SD
Voice direct 364	"say word three"	gravando 3ª palavra SD
Usuário:	"luzes"	
Voice direct 364	"repeat"	
Usuário:	"luzes"	
Ação	pressione o botão train	encerra a gravação

O usuário pode sair do modo gravação a qualquer momento pressionando o botão *train* ou *recog*, ou não respondendo a mensagem "Say Word X" ou "repeat". Neste último caso é dita a mensagem "training complete". Quando todas as 15 posições de memória forem usadas o modo gravação também é encerrado.

Reconhecendo palavras no modo SCL:

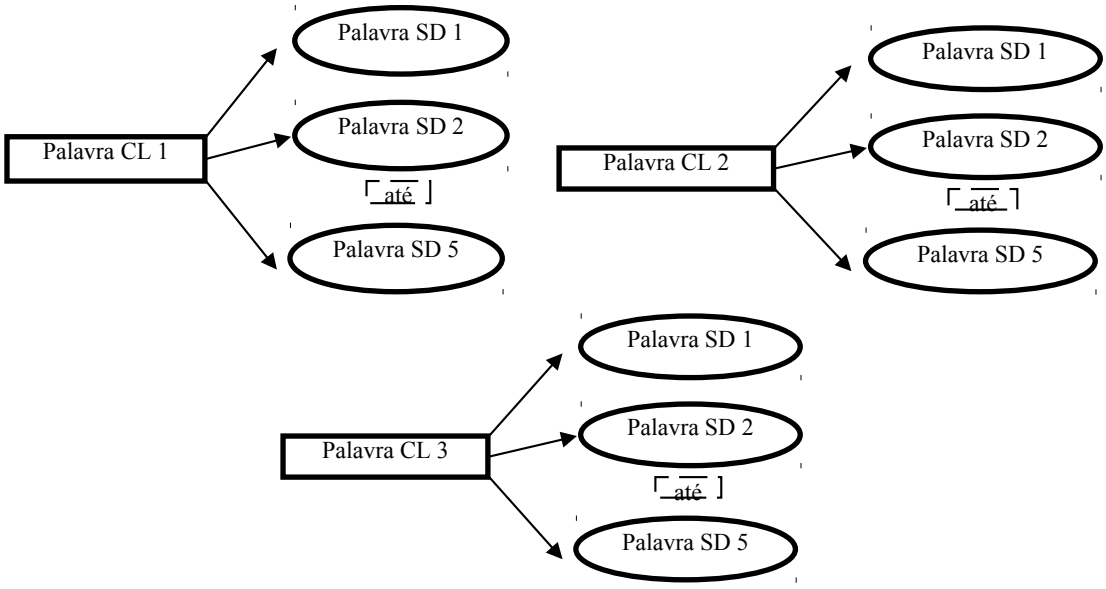
Após pressionar o botão recognition e verificar se o Led verde (talk led) está aceso, o Voice direct 364 está preparado para ouvir palavras. Após sucesso no reconhecimento da palavra CL, o Led verde pisca. Em seguida, é dado um tempo de 3s para que se fale a palavra SD. Se novamente for obtido sucesso no reconhecimento da palavra SD, uma ou duas das 8 saídas irá ficar ativa por 1s – conforme indica a tabela a seguir – e Voice direct 364 dirá uma mensagem indicando o número da palavra.

Saídas que são ativadas no modo SCL

Palavra Cl + palavra SD	saída 1	saída2	Saída3	Saída4	Saída5	Saída6	Saída7	Saída8
CL + palavra SD 01	A							
CL + palavra SD 02		A						
CL + palavra SD 03			A					
CL + palavra SD 04				A				
CL + palavra SD 05					A			
CL + palavra SD 06						A		
CL + palavra SD 07							A	
CL + palavra SD 08								A
CL + palavra SD 09	A							A
CL + palavra SD 10		A						A
CL + palavra SD 11			A					A
CL + palavra SD12				A				A
CL + palavra SD13					A			A
CL + palavra SD14						A		A
CL + palavra SD 15							A	A

A indica que a saída está ativada.

Modo de operação MCL:



Modo de operação MCL

O roteiro abaixo é um exemplo de como gravar palavras no modo MCL

Gravando palavras no modo MCL:

Ação	pressione o botão cl train	começa a gravar palavras do 1º conjunto
Voice direct 364	“say Word one”	gravando palavra CL do 1o conjunto de palavras
Usuário:	“futebol”	
Voice direct 364	“repeat”	
Usuário:	“futebol”	
Ação	Pressione o botão train	começa a gravação SD
Voice direct 364	“Say Word one”	começa a gravar 1ª palavraSD do 1o conjunto depalavras
Voice direct 364	“Say Word one-one”	
Usuário:	“Flamengo”	
Voice direct 364	“repeat”	
Usuário:	“Flamengo”	
Ação	pressione o botão train	continua gravação SD
Voice direct 364	“Say Word one-two”	começa a gravar 2ª palavra SD do 1o conjunto de Palavras
Usuário:	“Botafogo”	
Voice direct 364	“repeat”	
Usuário:	“Botafogo”	
Ação	pressione o botão CL train	começa gravação do 2º Conjunto de palavras
Voice direct 364	“Say Word two”	começa a gravar palavra CL do 2o conjunto de palavras
Usuário	“veículo”	
Voice direct 364	“repeat”	
Usuário	“veículo”	
Ação	pressione o botão train	começa gravação SD do2º conjunto de palavras
Voice direct 364	“Say word two-one”	começa a gravar 1ª palavra SD do 2o conjunto de palavras
Usuário	“carro”	
Voice direct 364	“repeat”	
Usuário	“carro”	
Ação	pressione o botão train	continua gravação SD
Voice direct 364	“Say Word two-two”	começa a gravar 2ª palavra SD do 2o conjunto de palavras
Usuário:	“ônibus”	
Voice direct 364	“repeat”	
Usuário:	“ônibus”	
Ação	pressione o botão train	encerra a gravação

Reconhecendo palavras no modo MCL:

Após pressionar o botão *recog* e verificar se o Led verde está aceso, o Voice direct 364 está pronto para reconhecer palavras. Se for dita uma das três palavras CL programadas o Led verde pisca. Após isso, é dado um tempo de três segundos para se falar uma das cinco palavras SD correspondentes à palavra CL. Se for obtido sucesso no reconhecimento da palavra SD, duas das oito saídas ficarão ativas por um segundo e o Voice direct 364 dirá uma mensagem indicando o número da palavra, conforme é mostrado na tabela 2.3.

Observamos que se pode gravar apenas uma ou duas palavras CL e menos que cinco palavras SD para cada palavra CL.

Saídas que são ativadas no modo MCL

A indica saída ativada.

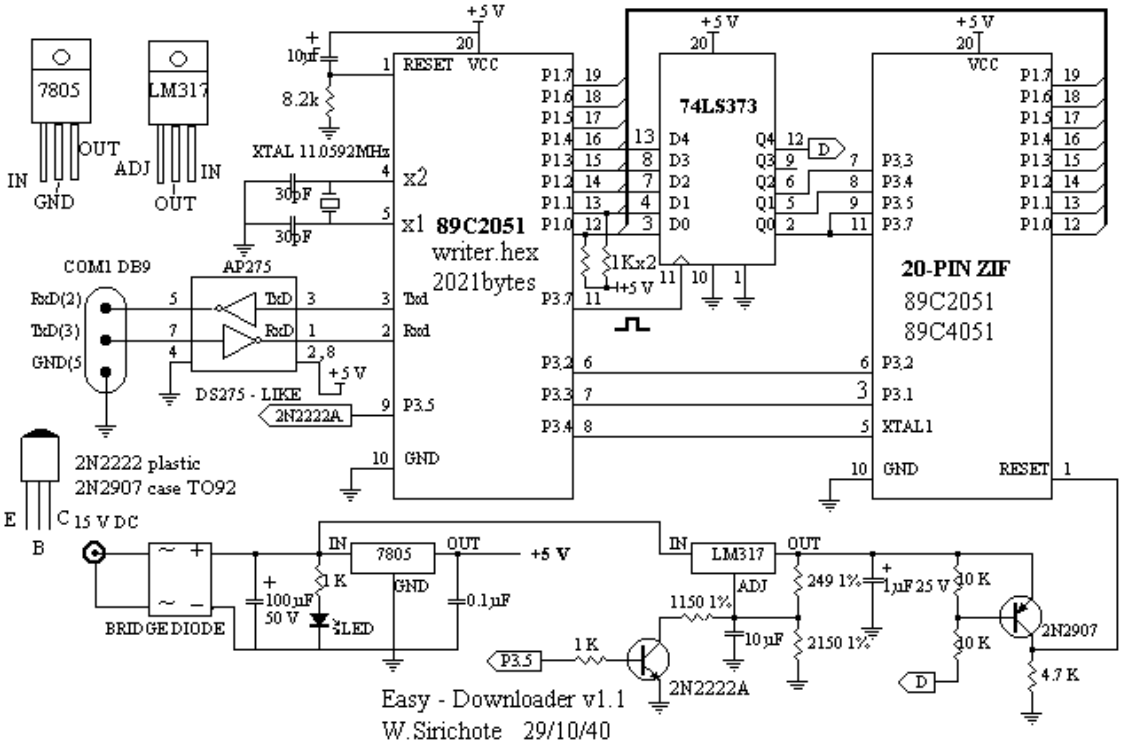
Palavra CL	Palavra SD	Saída1	Saída2	Saída3	Saída4	Saída5	Saída6	Saída7	Saída8
Palavra CL A	Palavra SD 01	A			A				
Palavra CL A	Palavra SD 02	A				A			
Palavra CL A	Palavra SD 03	A					A		
Palavra CL A	Palavra SD 04	A						A	
Palavra CL A	Palavra SD 05	A							A
Palavra CL B	Palavra SD 01		A		A				
Palavra CL B	Palavra SD 02		A			A			
Palavra CL B	Palavra SD 03		A				A		
Palavra CL B	Palavra SD 04		A					A	
Palavra CL B	Palavra SD 05		A						A
Palavra CL C	Palavra SD 01			A	A				
Palavra CL C	Palavra SD 02			A		A			
Palavra CL C	Palavra SD 03			A			A		
Palavra CL C	Palavra SD 04			A				A	
Palavra CL C	Palavra SD 05			A					A

Apêndice

B

Esquema do Gravador 89C2051 e 89C4051

B - Esquema elétrico do gravador de 89C2051 e 89C4051



O circuito usa um 89C2051 gravado, um *latch* de 8 bits 74LS373, um 7805, um LM317 e dois transistores: 2N2222A e 2N2907A. O *latch* de 8 bits 74LS373 proporciona um sinal para selecionar os modos de programação. Um Byte a ser programado ou lido de volta é enviado / recebido através de P1. A voltagem de programação pode ser obtida de 0V, 5V e 12V por sinal apropriado de P3. A voltagem de alimentação do gravador deve ser de aproximadamente 15Vdc. O programa usado para gravar o microcontrolador é o EZ Uploader e pode ser encontrado em http://chaokhum.kmilt.ac.th/~kswichit/easy1/easy.htm.

Apêndice

C

Protocolo F-Bus e Comandos

C - Protocolo F-Bus e comandos

Informações mais detalhadas podem ser encontradas nos endereços eletrônicos http://www.gadgets.demon.co.uk/nokia21xx/protocol.html e também em http://www.embedtronics.com/nokia/fbus.html.

O cabo serial precisa ser alimentado corretamente e para isso os pinos DTR e RTS precisam ser ajustados de modo adequado. O DTR precisa ser alimentado com uma tensão entre +3 e 12V e o RTS precisa ser alimentado com uma tensão entre –3 e –12V. Essas tensões podem ser obtidas do próprio Max232. Pode-se ligar o pino DTR no pino V+ do MAX232, que deve valer aproximadamente +10V e ligar o pino RTS no pino V- do Max232, que deve valer uns –10V.

O próximo passo é sincronizar a UART no celular com o microcontrolador ou PC. Isso é feito enviando uma string de 0x55 ou "U" 128 vezes. Após isso o celular está pronto para receber comandos.

A seguir segue um exemplo de um comando enviado para o nokia que serve para obter a versão de hardware e software do aparelho da Nokia.

Byte: 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15

Dados: 1E 00 0C D1 00 07 00 01 00 03 00 01 60 00 72 D5

Byte0: todos os comandos enviados por cabo irão começar com o caracter 0x1E(infravermelho seria 0x1C)

- Byte1: Este é o endereço de destino. Quando enviamos dados, é o byte de identificação do celular. No nosso caso, é sempre 00 para o telefone.
- Byte2: É o endereço de origem. Quando enviamos dados, é o byte de identificação do PC. No nosso caso, será sempre 0x0C.
- Byte3: É o tipo de mensagem ou 'comando'. 0xD1 é o byte para obter a versão do hardware e software do aparelho.

- Bytes 4 e 5: Bytes 4 e 5 são o comprimento da mensagem. No nosso caso, comprimento de 7 bytes. byte 4 é o mais significativo e byte 5 é o menos significativo.

- Bytes 6 a 13: O segmento de dados começa no byte 6 e vai até o byte 12. Como o nokia é um telefone de 16 bits, ele requer um número par de bytes. Como o nosso comprimento é ímpar, o último byte será um byte de enchimento e a mensagem acabará na localização 13.

- Bytes 14 e 15: O byte 14 é o resultado da operação *XOR* sobre todos bytes ímpares e o byte 15 é o resultado da operação *XOR* sobre todos bytes pares.

Se o telefone recebeu este frame, ele deverá mostrar a seguinte resposta:

1E 0C 00 7F 00 02 D1 00 CF 71

1E 0C 00 D2 00 26 01 00 00 03 56 20 34 2E 34 35 0A 32 31 2D 30 36 2D 30 31 0A 4E 48 4D 2D 35 0A 28 63 29 20 4E 4D 50 2E 00 01 41 3F A4

A primeira linha é um frame de reconhecimento, confirmando que recebeu o frame anterior: Agora os bytes de origem e destino estão trocados, pois então é o celular respondendo para o PC. A mensagem tem comprimento de 2 bytes. Os últimos 2 bytes são bytes de *Checksun*. O primeiro é a operação *XOR* de todos os bytes ímpares do frame. O último é a operação *XOR* de todos os bytes pares do frame. Após o envio desses dois *frames*, o nokia 3310 ficará esperando um frame de reconhecimento do PC. Se o frame de reconhecimento não é enviado, o nokia 3310 irá repetir enviar os dados. O Nokia 3310 irá enviar dados apenas 3 vezes e depois desistirá.

O segundo frame do Nokia são os dados requeridos. O tipo de mensagem é 0xD2, que corresponde ao recebimento da versão do hardware e software do nokia. A mensagem tem tamanho de 38 bytes. Como todos frames padrão do *F-bus*, os últimos dois bytes no frame são bytes de *Checksum*.

Tudo que é requerido agora é enviar um frame de reconhecimento de volta para o Nokia para dizer, ’Eu peguei!’

1E 00 0C 7F 00 02 D2 01 C0 7C

0x7F é o byte de reconhecimento do frame. A mensagem contém o byte que é o tipo de mensagem é o byte D2. O *Checksum* precisa ser calculado e enviado.

Apêndice

D

Programa de Controle do Sistema

D - Programa de controle do sistema

```
/*---------------------------------------------------------------------------
TELEFONE.C
---------------------------------------------------------------------------*/
#include <at892051.H>        /* special function register declarations   */
#include <string.h>          /* for the intended 802051 derivative       */
#include <stdio.h>           /* prototype declarations for I/O functions */
#include <intrins.h>
// comandos do usuario
sbit CMDA = 0xB3; // T0 P3_3
sbit CMDB = 0xB4; // P3_4
sbit CMDC = 0xB5; // P3_5
sbit CLA = 0x91; // P1_1
sbit CLB = 0x92; // P1_2
sbit CLC = 0x90; // P1_0
void confdelay900us();
void sendzerophysical();
void sendonephysical();
void sendonelogic();
void sendzerologic();
void DisableTimerInt();
void EnableTimerInt();

// a,b,c,d são entradas do decodificador
sbit Tecla0 = 0x93; // P1_3 Para acionar a tecla 0 usou-se diretamente um pino do microcontrolador
sbit a = 0x94; // P1_4
sbit b = 0x95; // P1_5
sbit c = 0x96; // P1_6
sbit d = 0x97; // P1_7

sbit NoGancho = 0xB7; // INT0 P3_7

// variaveis de estado
int EstadoAtual;
#define StInicio            0;
#define StCMDA1             1;
#define StCMDB1             2;
#define StCMDC1         3;
#define StCMDD1         4;
#define StCMDE1         5;
#define StCMDA2         6;
#define StCMDB2         7;
#define StCMDC2         8;
#define StCMDD2         9;
#define StCMDE2         10;
#define StCMDA3         11;
#define StCMDB3         12;
#define StCMDC3         13;
#define StCMDD3         14;
#define StCMDE3         15;
#define StFim           16;
```

```
// variaveis de tempo - tempos definidos para um clock de 24 MHz
#define UmQuartodeSeg           6; // contagem relativa a aproximadamente 1/4 de segundo
#define UmSeg   28;       // contagem relativa a aproximadamente 1 segundo
#define DoisSeg   58;       // contagem relativa a aproximadamente 2 segundos

//Variaveis gerais
int Contador;


void confdelay900us(){
          TMOD=0X01;  //Timer0 no modo 1
   TH0=0xF8;
          TL0=0xF7;
}
void sendonephysical()
{
 int cont=0;
 confdelay900us();
 TR0=1;          //Liga o timer
 while(!TF0)    //Fica esperando overflow
 {
  a=0;
          cont=3;
          while (--cont);
          a=1;
          cont=2;
          while (--cont);
          _nop_();
          _nop_();
          _nop_();
          _nop_();
          _nop_();
 }
 TR0=0;
 TF0=0;
}
void sendzerophysical()
{
 int cont=0;
 confdelay900us();
 a=1;
 TR0=1;          //Liga o timer
 while(!TF0);
 TR0=0;
 TF0=0;
}
void sendonelogic()
{
 sendzerophysical();
 sendonephysical();
}

void sendzerologic()
{
 sendonephysical();
```

```
  sendzerophysical();
}
void ResetTimerOverflow(void)
{
        OV = 0;
}

void TimerSetup(void) // seta configuracoes do modo do timer
{
        // Setando o modo do contador pra 1
//      TMOD = (T0_M0_ | T1_M0_ );
   TMOD = 1;
        // Colocando o valor inicial do contador em 0
        TL0 = 0;
        TH0 = 0;
}

void EnableTimerInt(void)
{
        ET0 = 1;
        Contador = 0;
        TR0 = 1; //inicia a contagem
}

void DisableTimerInt(void)
{
        ET0 = 0;
        TR0 = 0; //parando a contagem
}

void TimerInt(void) interrupt 1  // Interrompe qdo timer overflow
{
        Contador++;
        ResetTimerOverflow();   // Reseta Ovf do timer
}

void atender()
{
        printf("ATA\n\r");
}

//Efetua comando de discagem para voz (atd finalizado em \")
void ligar(char *numero)
{
        printf("ATD%S;\n\r",numero);
}

//Comando AT para Hang-up
void desligar()
{
        printf("ATH\n\r");
}

void horario(char *hora)// muda a data do celular
{
```

```
        printf("AT+cclk=\"%S\";\n\r",hora);
}

void command(int bit1,bit2,bit3,bit4,bit5,bit6,bit tBit)//envia o sinal no protocolo RC5
{
  int cont;
  int i;
  //Start bits
 sendonelogic();
 sendonelogic();
   //Toggle bit
if (tBit == 0) sendzerologic();else sendonelogic();
 // Address
for (cont =0;cont <5;cont++)
 sendzerologic();

 // Command

if (bit1 == 1) sendonelogic(); else sendzerologic();
if (bit2 == 1) sendonelogic(); else sendzerologic();
if (bit3 == 1) sendonelogic(); else sendzerologic();
if (bit4 == 1) sendonelogic(); else sendzerologic();
if (bit5 == 1) sendonelogic(); else sendzerologic();
if (bit6 == 1) sendonelogic(); else sendzerologic();

for (i=0; i < 39; i++){
        sendzerophysical();
}
}

void configurar_interface()
{
        //Configuracao da interface serial

  PCON = 0x80;
        SCON  = 0x50;                   /* SCON: modo 1, 8-bit UART, com recepção          */
  TMOD = 0x20;            /* TMOD: timer 1, modo 2(8-bit com reload)         */
  TH1   = 243;            /* TH1:  valor de timer para 9600 com clock 3.57Mhz   */
  TR1   = 1;              /* TR1:  Liga timer1                               */
  TI    = 1;              /* TI:   set TI to send first char of UART         */
}

//Buffer para receber retorno do celular

// rotina usada para esperar determinados tempo como o de tom por exemplo
void espera(int ciclos)
{
        PT0 = 1;
        EnableTimerInt();          // zerando contador
        while((Contador*2) < ciclos);  // esperando dois segundo
        DisableTimerInt();
        PT0 = 0;
}

void EnableCmdVozInt(void)
```

```
{
        EX0 = 1;    // interrupcao do reconhecedor do comando de voz INT0
        IT0 = 1;    // interrupcao por flanco de descida
}

void ex0_isr(void) interrupt 0
{
        // analisando a porta a qual chega o sinal da interrupcao, seta o bit que indica qual o comando
a processar
        if (CMDA == 0 && CMDB == 0 && CMDC == 0 && CLA == 1)
        {
                EstadoAtual = StCMDA1;
        }
        else if (CMDA == 1 && CMDB == 0 && CMDC == 0 && CLA == 1)
        {
                EstadoAtual = StCMDB1;
        }
        else if (CMDA == 0 && CMDB == 1 && CMDC == 0 && CLA == 1)
        {
                EstadoAtual = StCMDC1;
        }
        else if (CMDA == 1 && CMDB == 1 && CMDC == 0 && CLA == 1)
        {
                EstadoAtual = StCMDD1;
        }
        else if (CMDA == 0 && CMDB == 0 && CMDC == 1 && CLA == 1)
        {
                EstadoAtual = StCMDE1;
        }
        else if (CMDA == 0 && CMDB == 0 && CMDC == 0 && CLB == 1)
        {
                EstadoAtual = StCMDA2;
        }
        else if (CMDA == 1 && CMDB == 0 && CMDC == 0 && CLB == 1)
        {
                EstadoAtual = StCMDB2;
        }
        else if (CMDA == 0 && CMDB == 1 && CMDC == 0 && CLB == 1)
        {
                EstadoAtual = StCMDC2;
        }
        else if (CMDA == 1 && CMDB == 1 && CMDC == 0 && CLB == 1)
        {
                EstadoAtual = StCMDD2;
        }
        else if (CMDA == 0 && CMDB == 0 && CMDC == 1 && CLB == 1)
        {
                EstadoAtual = StCMDE2;
        }
        else if (CMDA == 0 && CMDB == 0 && CMDC == 0 && CLC == 1)
        {
                EstadoAtual = StCMDA3;
        }
   else if (CMDA == 1 && CMDB == 0 && CMDC == 0 && CLC == 1)
        {
```

```
                EstadoAtual = StCMDB3;
        }
   else if (CMDA == 0 && CMDB == 1 && CMDC == 0 && CLC == 1)
        {
                EstadoAtual = StCMDC3;
        }
        else if (CMDA == 1 && CMDB == 1 && CMDC == 0 && CLC == 1)
        {
                EstadoAtual = StCMDD3;
        }
else if (CMDA == 0 && CMDB == 0 && CMDC == 1 && CLC == 1)
        {
                EstadoAtual = StCMDE3;
        }
        ResetTimerOverflow();
}

void EnableInterrupts(void)
{
        // Habilita interrupcoes
        EA = 1;
        //seta as prioridades
        IP = 0x01;    // PX0 = 1
}

void ZeraTeclas(void)
{
Tecla0 = 0;
a = 0;
b = 0;
c = 0;
d = 0;

}

void gancho(void)//tira ou coloca o telefone sem fio no gancho
{
NoGancho =1;
espera(26);
NoGancho =0;
espera(26);
}
void Tc0(void)//serve para acionar a tecla 0
{
Tecla0 = 1;
espera(26);
Tecla0 = 0;
espera(26);
}
void Tc1(void)//serve para acionar a tecla 1
{
a=1;
espera(13);
a=0;
espera(13);
```

```
}
void T2(void)//serve para acionar a tecla 2
{
b=1;
espera(13);
b=0;
espera(13);
}
void T3(void)//serve para acionar a tecla 3
{
a=1;b=1;
espera(13);
a=0;b=0;
espera(13);
}
void T4(void)//serve para acionar a tecla 4
{
c=1;
espera(13);
c=0;
espera(13);
}
void T5(void)//serve para acionar a tecla 5
{
a=1;c=1;
espera(13);
a=0;c=0;
espera(13);
}
void T6(void)//serve para acionar a tecla 6
{
b=1;c=1;
espera(13);
b=0;c=0;
espera(13);
}
void T7(void)//serve para acionar a tecla 7
{
a=1;b=1;c=1;
espera(13);
a=0;b=0;c=0;
espera(13);
}
void T8(void)//serve para acionar a tecla 8
{
d=1;
espera(13);
d=0;
espera(13);
}
void T9(void)//serve para acionar a tecla 9
{
a=1;d=1;
espera(13);
a=0;d=0;
```

```
espera(13);
}
void main (void)                        // Progr. Principal do controlador
{
   int i;
   bit Tbit;
        Tbit =0;

        TimerSetup();           // Timer0 em modo 1, entrada /4
        EnableInterrupts();

        // Neste ponto, antes de liberarmos a interrupcao de voz, é necessario "colocar" o telefone no
gancho
        // Devemos lembrar que todas as portas do micro comecam em "1"
        P3_7 = 0;
        EnableCmdVozInt();
        EstadoAtual = StInicio;
        while (1)                       // Loop p/ sempre
        {
                switch (EstadoAtual)
                {
                        case 0: //StInicio
                        {
                                ZeraTeclas();
                                break;
                        }
                        case 1: //StCMDA1 - Discar para M. Lois
                        {
                                if (NoGancho == 1) // se o tel ja estiver numa chamada, nao executa o
comando de discagem
                                {
                                        EstadoAtual = StFim;
                                }
                                else
                                {
                                        // devemos "desapertar as teclas" antes de tirar o tel do gancho
                                        // lembrando denovo que as portas do micro comecam em "1"
                                        ZeraTeclas();

            gancho();
                                   Tc0();
                                        espera(26);
                                        T9();
                                        T9();
                                        Tc1();
                                        Tc1();
                                        T4();
                                        T8();
                                        Tc0();
                                T5();
                                        EstadoAtual = StFim;
                                }
                                break;
                        }
                        case 2: //StCMDB1 -  Discar para Guilherme
```

```
                        {
                                if (NoGancho == 1) // se o tel ja estiver numa chamada, nao executa o
comando de discagem
                                {
                                        EstadoAtual = StFim;
                                }
                                else
                                {
                                        // devemos "desapertar as teclas" antes de tirar o tel do gancho
                                        // lembrando denovo que as portas do micro comecam em "1"
                                        ZeraTeclas();
            gancho();
                                        Tc0();
                                        espera(26);
                                        T9();
                                        T2();
                                        Tc0();
                                        T4();
                                        T5();
                                        T5();
                                        T5();
                                        T4();

                                        EstadoAtual = StFim;
                                }
                                break;
                        }
                        case 3: //StCMDC1 - atender
                        {

          gancho();

EstadoAtual = StFim;

                                break;
                        }
                        case 4: //StCMDD1 ligar labsun
                        {

           gancho();
                                espera(13);
                                T3();
                                T2();
                                T8();
                                T6();
EstadoAtual = StFim;
                                break;

                        }

                    case 5: //StCMDE1 - desligar
                                {
                                   gancho();
                                        EstadoAtual = StFim;
                                        break;
```

```
                        }

        case 6: //StCMDA2 ligar Manuel pelo celular
                        {
                         configurar_interface();
         ligar("99114805");

EstadoAtual = StFim;

                                break;
                        }

        case 7: //StCMDB2 desligar uma ligação do celular
                        {
        configurar_interface();
        desligar();

        EstadoAtual = StFim;

                                break;
                        }
        case 8: //StCMDC2 atender uma chamada no celular
                        {
                                 configurar_interface();
           atender();

EstadoAtual = StFim;
                                break;
                        }
case 9: //StCMDD2 muda a data do celular
                        {
configurar_interface();
horario("05/05/18,22:00:00-12");

EstadoAtual = StFim;
                                break;
}
case 10: //StCMDE2 ligar guilherme
                        {
         configurar_interface();
         ligar("92045554");

EstadoAtual = StFim;
                                break;
}
case 11: //StCMDA3 ligar/desligar TV
                        {
         for (i=0;i<5;i++){
                        command(0,0,1,1,0,0,Tbit);}
         Tbit = (!Tbit);

EstadoAtual = StFim;
                                break;
}
case 12: //StCMDB3 canal -
```

```
                                {
            for (i=0;i<3;i++){
                                 command(1,0,0,0,0,1,Tbit);}
            Tbit = (!Tbit);

EstadoAtual = StFim;

                                        break;
}
case 13: //StCMDC3 canal +
                                {
             for (i=0;i<3;i++){
                                  command(1,0,0,0,0,0,Tbit);}
             Tbit = (!Tbit);

EstadoAtual = StFim;
                                        break;
}
case 14: //StCMDD3 volume +
                                {

            for (i=0;i<8;i++){
                                command(0,1,0,0,0,0,Tbit);}
            Tbit = (!Tbit);

EstadoAtual = StFim;
                                        break;
}
case 15: //StCMDE3 volume -

                                {
            for (i=0;i<8;i++){
                                command(0,1,0,0,0,1,Tbit);}
                        Tbit = (!Tbit);
EstadoAtual = StFim;
                                        break;
}
            case 16: //Stfim
                                {

            // - antes de reabilitar a interrupcao de voz, devemos esperar o tempo a qual a interrupcao fica
            //   ativa para prevenir que nao se trate uma mesma interrupcao mais de uma vez.
            // - note que esta espera é necessaria pois no caso do comando de desligar, o programa pode
                                        //          acabar tratando duas vezes a mesma interrupcao.
                                        espera(28);
            // EnableCmdVozInt(); como o desabilitamento esta sendo feito por hardware, so preciso
habilitar no inicio
                                        EstadoAtual = StInicio;
                                }
        }
            }
}   // End main
```