CENTRO UNIVERSITÁRIO DE BRASÍLIA - UniCEUB
FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS – FATECS
CURSO DE ENGENHARIA DA COMPUTAÇÃO

JOSÉ CARLOS DA SILVA SANTA CRUZ

AUTOMAÇÃO RESIDENCIAL UTILIZANDO *SMARTPHONE ANDROID, BLUETOOTH* E CONVERSOR RS-485

Orientador: Prof.ª MSc. Maria Marony Sousa Farias

Brasília – DF

2º Semestre de 2013

JOSÉ CARLOS DA SILVA SANTA CRUZ

AUTOMAÇÃO RESIDENCIAL UTILIZANDO *SMARTPHONE ANDROID, BLUETOOTH* E CONVERSOR RS-485

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Orientador: Prof.ª MSc. Maria Marony Sousa Farias

Brasília - DF

2º Semestre de 2013

JOSÉ CARLOS DA SILVA SANTA CRUZ

AUTOMAÇÃO RESIDENCIAL UTILIZANDO *SMARTPHONE ANDROID, BLUETOOTH* E CONVERSOR RS-485

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Engenharia de Computação.
Orientadora: Prof.ª MSc. Maria Marony Sousa Farias

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Prof. Abiézer Amarília Fernandes
Coordenador do Curso

Banca Examinadora:

_____

Prof.ª MSc. Maria Marony Sousa Farias
Orientadora

_____

Prof. nome, titulação.
Instituição

_____

Prof. nome, titulação.
Instituição

_____

Prof. nome, titulação.
Instituição

DEDICATÓRIA

Dedico este trabalho aos meus pais José e Erondina que juntos me mostraram quando criança o caminho que devia percorrer e jamais me desviei dele, à minha esposa Marineide e meus filhos André e Raissa pelos momentos em que estive ausente.

José Carlos da Silva Santa Cruz

AGRADECIMENTOS

Agradeço a Deus pela saúde e por renovar as minhas energias todos os dias ao me levantar, aos meus pais pela educação, pelos exemplos e lições de vida.

Agradeço a minha esposa Marineide pelo apoio aos estudos e por estar sempre cuidando de nossos filhos quando estou ausente.

Aos meus colegas de Faculdade Gilberto de Paula, Ayrton Gissoni, Hugo Molina, Ingred Carvalho, Sérgio Oliveira, Thiago Lima, Marcos Vinícius pela inestimável ajuda.

Aos meus colegas de trabalho Cleiton Ferreira Pinto pelo apoio no desenvolvimento deste projeto e Carlos Alberto Venâncio pela ajuda na formatação deste trabalho.

Aos professores Luciano Duque, Luís Cláudio, João Marcos, Thiago Toríbio, Miguel Archanjo, José Julimá, Francisco Javier e Fabiano Mariath pelo apoio e incentivo aos projetos do Grupo de Robótica.

Á minha orientadora Professora Marony pelos seus conselhos, incentivos e oportunidade dada em ser seu monitor nas disciplinas de engenharia ao longo deste curso.

A todos vocês o meu muito obrigado.

José Carlos da Silva Santa Cruz

“... Fui moço, e agora sou velho; mas nunca vi desamparado o justo, nem a sua semente a mendigar o pão...”.

Salmos 37:25

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE EQUAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ASCII - American Standard Code for Information Interchange

AT - Attention

BCD - Binary-Coded Decimal

Bit – Binary digit

Bps - bit per second

CCP – Capture Compare PWM

CCS – Custom Computer Service

CMD - Command

CMOS - Complementary Metal Oxide Semiconductor

CTS - Clear To Send

DAT - Data

DCE - Data Communications Equipment

DTE - Data Terminal Equipment

DIP - Dual In-Line Package

EEPROM - Electrically-Erasable Programmable Read-Only Memory

EIA - Electronic Industries Alliance

FHSS - Frequency Hopping Spread Spectrum

FSK – Frequency Shift Keying

GFSK – Gaussian Frequency Shift Keying

GND - Ground

GPIO – General Purpose In Out

HVAC - Heating, Ventilation, Air Conditioning

IDE - Integrated Development Environment

I^2C – Inter-Integrated Circuit

IEEE - Institute of Electrical and Electronics Engineers

ICSP - In-Circuit Serial Programming

ISM - Industrial, Scientific and Medical

LAN - Local Area Network

LED - Light Emitting Diode

LSB – Least significant Bit

LLC - Logical Link Control

L2CAP - Logical Link Control and Adaptation Protocol

MAC - Media Access Control

Mbps – Mega bit per second

MCLR – Master Clear Reset

MIPS - Millions of Instructions Per Second

MIT - Massachusetts Institute of Technology

MSB – Most Significant Bit

PC – Personal Computer

PIC - Programmable Interface Controller

PIN - Personnal Identification Number

PWM - Pulse-Width Modulation

RAM - Random Access Memory

RFcomm - Radio Frequency Communication

RTS - Request To Send

RxD - Received Data

SCO - Synchronous Connection Oriented

SPI – Serial Peripheral Interface

SPICE -Simulated Program with Integrated Circuits Emphasis

TIA – Telecommunications Industry Association

TDD-TDMA - Time Division Duplexing-Time Division Multiple Access

TDMA - Time Division Multiple Access

TTL – Transistor-Transistor Logic

TxD - Transmitted Data

UART - Universal Asynchronous Receiver Transmitter

USART - Universal Synchronous Asynchronous Receiver Transmitter

USB - Universal Serial Bus

UTP – Unshielded Twisted Pair

W - Watts

RESUMO

Esse trabalho visa o desenvolvimento de um sistema de automação residencial usando um *smartphone* com *Bluetooth* e Sistema Operacional *Android* para o usuário fazer o acionamento de dispositivos eletroeletrônicos como lâmpadas, sistema de climatização/ventilação, aparelhos de som e TV, abrir ou fechar o portão de entrada de pedestre e de veículos. Os módulos de acionamentos dos equipamentos eletroeletrônicos recebem os comandos no formato ASCII provenientes de outro dispositivo *Bluetooth* com conversor RS-485 instalado na residência que faz a comunicação com um *smartphone Android*. Um cabeamento de par trançado faz a conexão entre os módulos de acionamentos para a transmissão e recepção dos sinais para ligar ou desligar os equipamentos na residência. Cada módulo de acionamento possui o seu próprio conversor RS-485 em conjunto com microcontroladores que codificam/decodificam os comandos para acionamentos dos dispositivos eletroeletrônicos.

Palavras chaves: Automação residencial. *Bluetooth*. Conversor RS-232. Conversor RS-485. Cabeamento de par trançado. *Android*. Microcontroladores. Domótica. *Interface*.

ABSTRACT

This work aims to develop a system for home automation using a smartphone with Android operating system and Bluetooth to the user making the drive electronics devices such as lamps, air conditioning/ventilation systems, stereos and TV, open or close the gate entry pedestrian and vehicles. Modules drive the electronic equipment receive commands in ASCII format from another Bluetooth device with RS-485 converter installed in the home that communicates with an Android smartphone. A twisted pair cabling makes the connection between the drive modules for transmission and reception of signals to switched on or off the equipment in the residence. Each drive module has its own RS-485 converter between with microcontrollers that encode/decode the commands to the drive electronics devices.

Keywords: Home automation. Bluetooth. RS-232 Converter. RS-485 Converter. Unshielded twisted pair cabling. Android. Microcontrollers. Home Automation. Interface.

CAPÍTULO 1 - INTRODUÇÃO

1.1 - Contextualização do Projeto

A automação residencial visa simplificar a vida cotidiana das pessoas, gerando conforto e bem estar. É uma tecnologia que oferece gestão a todos os recursos de uma residência, tais como: iluminação, climatização, segurança, comodidade, irrigação de plantas, acionamento de equipamentos eletroeletrônicos, áudio, vídeo e qualquer dispositivo de acionamento automático.

A automação residencial apresenta os benefícios da vida moderna conectando equipamentos eletroeletrônicos em uma rede, sendo monitorada pelo usuário com interação, de forma que tarefas podem ser feitas automaticamente com ou sem a intervenção humana.

Na Figura 1.1 é mostrada a visão geral do projeto, em que o *Bluetooth* do *smartphone* envia comandos a outro módulo *Bluetooth* que se encarrega de colocar estes sinais em um cabeamento para comandar os módulos de acionamentos para ligar ou desligar equipamentos eletroeletrônicos na residência.

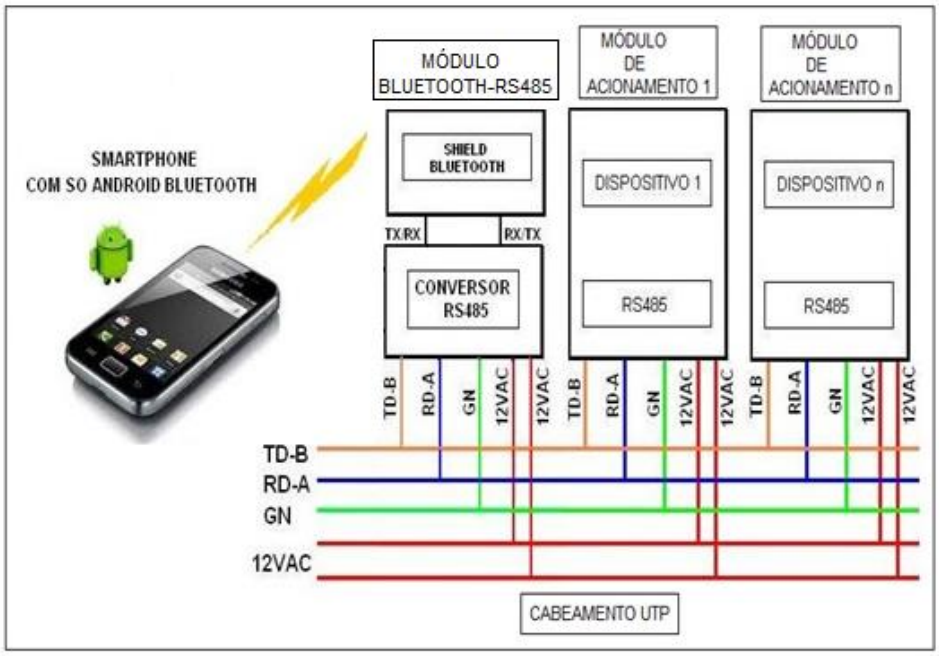


Figura 1.1 - Visão geral do projeto. (Autor: José Carlos)

Cada módulo de acionamento é composto por microcontroladores PIC da Empresa Microchip que são responsáveis pelo controle, codificação e decodificação dos sinais RS-232 TTL para o RS-485 utilizando um circuito integrado conversor de nível que faz a comunicação entre as placas de acionamentos por um cabeamento UTP. Cada placa de circuito impresso (módulo de acionamento) possui um microcontrolador e o circuito integrado conversor de nível MAX485.

1.2 - Objetivo Geral

O objetivo deste trabalho é projetar, desenvolver e implementar uma solução para o acionamento de dispositivos eletroeletrônicos em uma residência, mediante a utilização de um módulo *Bluetooth* externo conectado ao *Bluetooth* de um *smartphone* com Sistema Operacional *Android*, gerenciado por um aplicativo desenvolvido na plataforma MIT *App Inventor*.

1.3 - Objetivos Específicos

A proposta é desenvolver, confeccionar e montar pequenas placas de circuitos impresso (chamados de módulos de acionamentos) para o acionamento de alguns dispositivos eletroeletrônicos no exterior e interior da residência por meio do *smartphone Android* com *Bluetooth*. Estes dispositivos eletroeletrônicos compreenderão o acionamento de lâmpadas (da garagem, da sala, e do quarto), acionamento do fecho elétrico para a abertura do portão de entrada social, acionamento para abrir ou fechar o motor do portão deslizante para entrada de automóvel, leitura da temperatura no interior da residência, ligar ou desligar o sistema de climatização/ventilação e ligar ou desligar o aparelho de som ou TV.

1.4 - Motivação

O telefone celular (*smartphone*) é considerado como uma extensão do corpo de uma pessoa. Suas aplicações vão além do uso na comunicação entre usuários por meio de uma rede telefônica. Personalizações, animações, trabalho, *internet*, acesso às redes sociais, são muito legais, mas o controle automático de um estabelecimento residencial ou comercial é algo futurístico, que parece ter saído de algum filme de ficção científica.

A utilização do *smartphone* com um aplicativo fácil de construir e que melhore a qualidade de vida de seus usuários é muito gratificante.

Este trabalho tem como motivação a elaboração, o desenvolvimento, a construção e montagem de circuitos eletrônicos para proporcionar o conforto e a comodidade do usuário no acionamento de equipamentos eletroeletrônicos em sua residência, utilizando os recursos de um *smartphone Android*.

1.5 - Resultados Esperados

Espera-se com este projeto o desenvolvimento, construção e montagem do hardware para o acionamento de alguns equipamentos eletroeletrônicos no âmbito residencial, mediante o uso de um *smartphone Android* com um aplicativo para a conexão e gerenciamento do *hardware*.

1.6 - Escopo

Acionamento de equipamentos eletroeletrônicos com o aplicativo desenvolvido no MIT *App Inventor*, *Bluetooth*, microcontroladores e um conversor RS-485 com cabeamento UTP. Todo o projeto da parte eletrônica será elaborado e desenvolvido com simulação na IDE ISIS do Proteus, o *firmware* embutido nos microcontroladores serão desenvolvidos na linguagem C da CCS (IDE PIC C *Compiler*). O *layout* das placas de circuito impresso será desenvolvido na IDE ARES do Proteus.

O projeto não contempla o desenvolvimento do aplicativo com senha de usuários, acesso do sistema pela *web*. Essa restrição justifica-se pelo fato do projeto ser apenas para uso residencial e por isso não consta de um banco de dados para cadastro dos usuários e tampouco de senhas para utilização do sistema de automação. O projeto não inclui banco de bateria do tipo *nobreak*, caso haja suspensão do fornecimento de energia elétrica para alimentação dos módulos e acionamentos dos dispositivos eletroeletrônicos. O desenvolvimento do sistema é para o uso em tensão alternada de 220 Volts.

1.7 - Estruturas do trabalho

A estrutura do trabalho se divide nos seguintes capítulos:

O capítulo 2 é apresentado o referencial teórico sobre a automação residencial e *interface* RS-232, RS-485, *Bluetooth* e microcontroladores da série PIC.

No capítulo 3 é apresentado a descrição dos materiais utilizados e o desenvolvimento da construção do protótipo.

No capítulo 4 é apresentado o desenvolvimento de construção do aplicativo para acionamento pelo *Bluetooth* do *smartphone*.

O capítulo 5 apresenta os testes realizados no protótipo.

No capítulo 6 são apresentadas as conclusões do trabalho.

CAPÍTULO 2 - REFERENCIAL TEÓRICO

Este capítulo apresenta todas as bases teóricas para a resolução do problema apresentado no capítulo anterior, apresentando os principais conceitos sobre a comunicação RS-232, RS-485, *Bluetooth* e microcontroladores PIC.

2.1 - Domótica

É a tecnologia responsável pela gestão de todos os recursos habitacionais. Esse termo nasceu da fusão da palavra "*Domus*", que significa casa, com a palavra "Robótica", que está ligada ao ato de automatizar (realizar ações maquinalmente). Tem por objetivo satisfazer as necessidades de comunicação, segurança e comodidade diária das pessoas. **(Fonte: Mundo Educação, www.mundoeducacao.com/informatica/domotica.htm)**.

Há diversos dispositivos desenvolvidos para automatizar as tarefas rotineiras numa casa, interligados entre si gerando um sistema amplo de execução de serviços. Na domótica atual é possível ter controle de diversas tarefas realizadas pelo homem em sua residência como, por exemplo, na iluminação, climatização e segurança. Existem sensores que ao detectarem a presença humana iluminam o ambiente, interage o clima com a temperatura do corpo da pessoa presente na casa em determinado momento, além de controlar quem entra e quem sai por meio de câmeras e detectores de intrusos, entre outros. **(Fonte: Mundo Educação, www.mundoeducacao.com/informatica/domotica.htm)**.

2.1.1 - História da domótica

A história da automação residencial remonta da década de 70 (1975 para ser mais exato) pelos engenheiros da Empresa Pico *Electronics* Ltd., *Glenrothes* na Escócia que desenvolveu o protocolo X10, sendo que o mesmo é utilizado até hoje.

Os primeiros sistemas foram utilizados em edifícios comerciais em 1980, mas tornou-se popular com o advento da *internet* em 1990 no Japão, América do Norte, Europa. A domótica unificou todos os equipamentos eletroeletrônicos numa única rede, criando o que é

chamado de casa digital ou casa inteligente com protocolo de comunicação comum com comandos pelo serviço da *internet*.

2.2 - *Interface* de Comunicação Serial

> O protocolo é um conjunto de regras que controla a comunicação de dados. Representa um acordo entre os dispositivos de comunicação. Sem um protocolo, dois dispositivos podem estar conectados, mas, sem se comunicar. De modo semelhante, uma pessoa que fala francês não consegue entender outra que fala apenas o idioma japonês. **(FOROUZAN, 2008, p. 4)**.

2.2.1 - *Interface* de comunicação RS-232

Segundo Axelson (2007, tradução nossa) esta *interface* é projetada para proporcionar a comunicação entre dois dispositivos. A RS-232 é um nome popular de *interface* serial que se refere ao padrão TIA-232-F, sendo uma *interface* entre o Equipamento Terminal de Dados (DTE = *Data Terminal Equipment*). A norma RS-232 concede 25 linhas ou pinos para a *interface*, sendo o mais comum nove pinos ou linhas.

2.2.2 - *Interface* de comunicação RS-485

Segundo Axelson (2007, tradução nossa) esta *interface* é vulgarmente conhecida como RS-485, sendo definida pelo protocolo TIA-485-A. É uma solução apresentada a indústria para comunicação em longas distâncias e com velocidade superior a da RS-232. O protocolo RS-485 não está limitado a somente dois dispositivos, podendo conectar até 256 computadores por apenas um par de fios.

2.2.3 - Características da *interface* RS-485

De acordo com Axelson (2007, tradução nossa) a rede RS-485 requer apenas uma única fonte de alimentação com 5 Volts necessários para ter nas saídas diferenciais dos *drivers* de 1,5 Volts, não acontecendo com a RS-232 TTL quer requer duas fontes simétricas de 5 Volts. A capacidade de rede no protocolo RS-485 não está limitada à apenas dois dispositivo para conexão sendo uma *interface* multiponto, pode ter vários transmissores (*drivers*) e receptores.

Segundo Axelson (2007, tradução nossa) a taxa de transferência (*bit rate*) segue uma relação com o comprimento do cabo, se a taxa de transferência for baixa o comprimento do cabo aumenta. Com taxas de até 90 kbps o comprimento do cabo é de até 1200 metros. Em taxas mais rápidas (1 Mbps), o comprimento máximo do cabo está em torno de cerca de 120 metros.

2.2.4 - Linhas balanceadas

> Um par trançado consiste em dois fios de cobre encapados, que em geral tem cerca de 1 mm de espessura. Os fios são enrolados de forma helicoidal, assim como uma molécula de DNA. O trançado dos fios é feito porque dois fios paralelos formam uma antena simples. Quando os fios são trançados, as ondas de diferentes partes dos fios se cancelam, o que significa menor interferência.
>
> Os pares trançados podem ser usados na transmissão de sinais analógicos ou digitais. A largura de banda depende da espessura do fio e da distância percorrida, mas, em muitos casos, é possível alcançar diversos megabits/s por alguns quilômetros. **(TANENBAUM 2011, p. 59)**.

O principal cabo utilizado para redes de dados e voz é o cabo de quatro pares trançados não blindado, também conhecido como cabo UTP. Devido à sua construção (trançamento dos pares) e à forma de transmissão empregada (transmissão balanceada), este cabo fornece um bom grau de imunidade a interferências eletromagnéticas, principalmente considerando a aplicação em um ambiente residencial. **(Fonte: Instalações Eletricas, www.instalacoeseletricas.com/download/Automacao_residencial2.pdf)**.

Na Figura 2.1 é mostrado o cabo UTP com quatro pares sem blindagem.

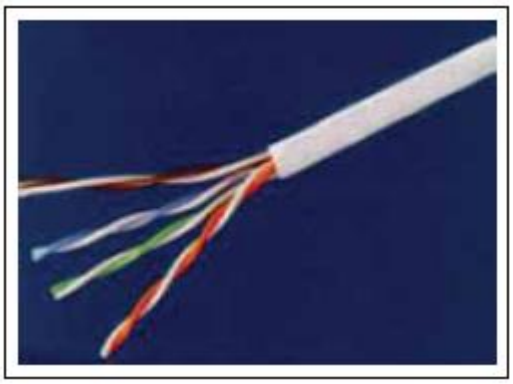


Figura 2.1 – Cabo UTP com quatro pares sem blindagem. (Fonte: Instalações Elétricas, www.instalacoeseletricas.com/download/Automacao_residencial2.pdf)

De acordo com Axelson (2007, tradução nossa) a principal razão para o RS-485 poder transmitir a longas distâncias é o uso de linhas balanceadas, que têm excelente imunidade a ruídos. Cada um dos sinais tem um par de fios dedicado. A tensão sobre um fio é igual ao negativo, ou complemento, de tensão no outro fio. O receptor detecta a diferença entre as tensões. O conversor RS-485 usa uma linha balanceada com dois fios chamada de par diferencial e tem como referência um ponto comum na linha de GND (*Ground*). O par diferencial é designado no protocolo TIA-485 como “A e B”.

Segundo Axelson (2007, tradução nossa) as linhas balanceadas são ótimas no transporte de sinais, pois temos a redução do ruído. Como os dois fios (“A e B”) transportam por iguais os sinais, as correntes em cada linha são opostas, reduzindo o ruído. Se aparecer qualquer tensão induzida em uma das linhas, na outra linha este ruído é cancelado por possuir tensão elétrica igual ao da outra linha.

Na Figura 2.2 são mostrados os sinais diferenciais transmitidos pelas linhas “A e B” na tela de um osciloscópio.

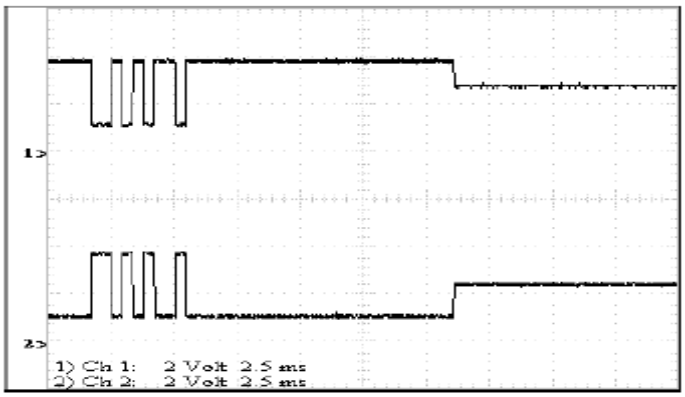


Figura 2.2 - Sinais diferenciais na linha A e B no RS-485 (Fonte: AXELSON, 2007)

2.3 - Porta Serial

A porta serial dos computadores do tipo PC *desktop* mais modernos, utilizam conectores de 9 pinos macho tipo sub-D, conectando um *modem* serial a um computador terminal com o computador remoto. **(Fonte: How Stuff Works, www.informatica.hsw.uol.com.br/portas-seriais2.htm)**.

Na Figura 2.3 é mostrado um conector com 9 pinos macho do tipo sub-D, vulgarmente chamado de DB-9.

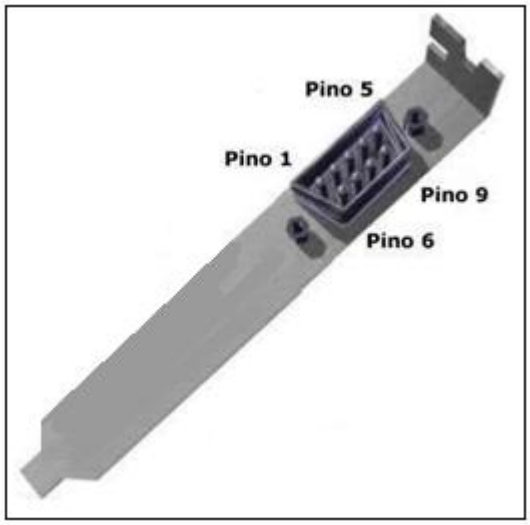


Figura 2.3 – Conector da porta serial com 9 pinos macho Sub-D. (Fonte: Adaptado da How Stuff Works, www.informatica.hsw.uol.com.br/portas-seriais2.htm)

2.3.1 - Comunicação serial

Segundo Axelson (2007, tradução nossa) os níveis lógicos gerados na transmissão a partir do computador terminal no pino TXD, corresponde à tensão de 12 Volts indicando nível lógico 0 e quando se inicia a transmissão temos -12 Volts correspondendo ao nível lógico 1, a sequência de transmissão dos bits se dá a partir do *bit* menos significativo (LSB) para o *bit* mais significativo (MSB).

2.3.2 - Comunicação do PC com dispositivos microcontroladores

Segundo Axelson (2007, tradução nossa) para efetuar a comunicação de um PC com dispositivos microcontroladores é utilizada uma *interface* para a conversão dos protocolos. No PC temos uma *interface* RS-232 com níveis de tensões entre -12 a +12 Volts, enquanto que nos microcontroladores dispomos de um padrão TTL, regidos com nível fixo de tensão de 0 Volt e +5 Volts. A conversão de nível RS-232 para o nível TTL é feita por uma *interface* de conversão com um circuito integrado MAX232.

De acordo com Axelson (2007, tradução nossa) o conversor MAX232 gera os níveis de tensões de -12 a +12 Volts requeridos pela *interface* RS-232 no PC e +5 Volts necessário pelo microcontrolador por meio de um gerador de tensão capacitivo interno. Estes capacitores são conectados nos pinos 1 e 3 (C1+ e C1-), pinos 4 e 5 (C2+ e C2-) e pinos 2 e 6 (V+ e V-), esta técnica de geração de tensão evita que seja utilizada fontes simétricas de -12 Volts, 0 Volt e +12 Volts.

Na Figura 2.4 é mostrada uma representação esquemática de uma *interface* com o MAX232 entre o PC (conector DB-9) e um microcontrolador PIC.

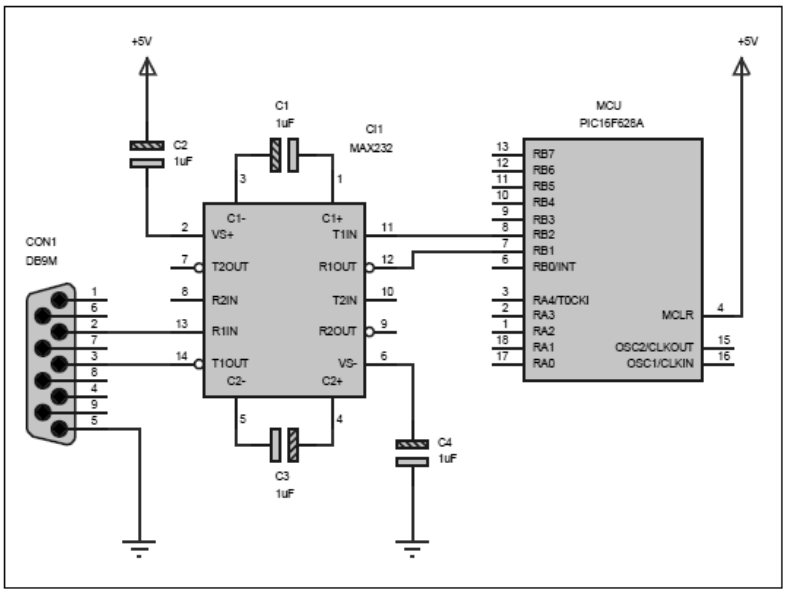


Figura 2.4 – *Interface* MAX232 com microcontrolador PIC. (Autor: José Carlos)

Na Figura 2.5 são mostrados os pinos do circuito integrado MAX232 com 16 pinos, bem como os blocos de circuitos internos que compõem este chip.

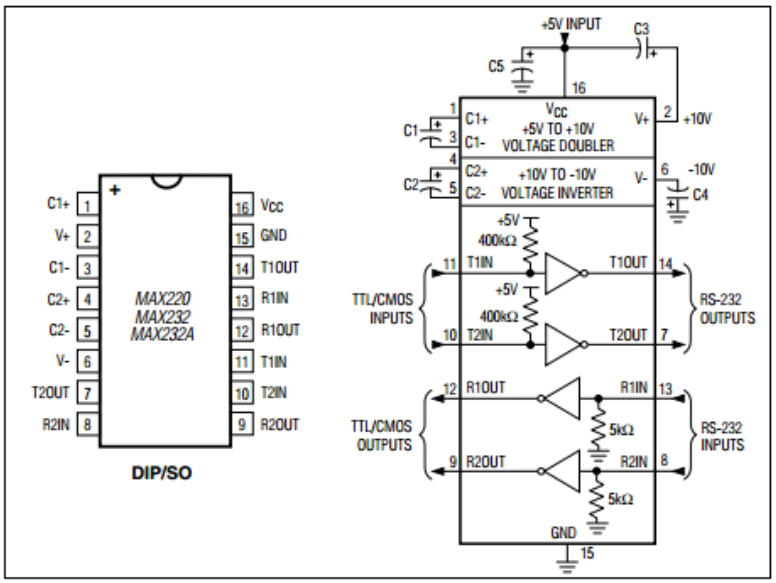


Figura 2.5 – Pinos e circuitos internos do MAX232 (Fonte: Maxim Integrated, www.datasheets.maximintegrated.com/en/ds/MAX220-MAX249.pdf)

Segundo (Oliveira 2006, p.56), depois de completado o circuito para comunicação serial, é necessário conectar com o computador através de um cabo, *cross-over*. É

simplesmente um cabo que coloca o pino de transmissão (TxD) do sistema embarcado conectado com o pino de recepção (RxD) do computador.

Na Figura 2.6 é mostrada como é efetuada as ligações TX e RX do cabo *crossover*.

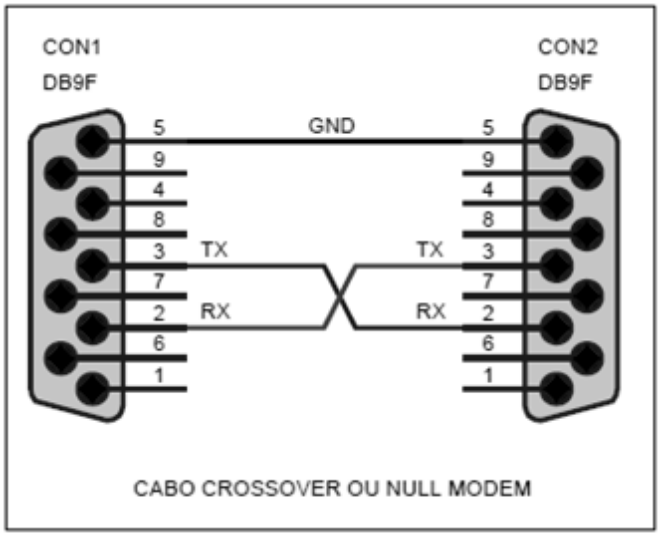


Figura 2.6 - cabo *crossover* ou *null modem* simples. (Autor: José Carlos)

O cabo *crossover* é chamado também de cabo *null modem*, permitindo a comunicação sem utilizar um *modem* entre dois dispositivos seriais, um chamado de *Data Terminal Equipment* (DTE), e o outro chamado de *Data Communications Equipment* (DCE). **(Fonte: How Stuff Works, www.informatica.hsw.uol.com.br)**.

A comunicação é estabelecida entre os dispositivos pelo terminal de transmissão (TxD) conectado ao pino de recepção (RxD) do outro dispositivo. **(Fonte: How Stuff Works, www.informatica.hsw.uol.com.br)**.

Caso haja necessidade de efetuar o *handshaking* entre os dois dispositivos (processo efetuado entre dois computadores, indicando um ao outro que a comunicação pode ser iniciada), o cabo *crossover* em vez de três terminais terá agora cinco terminais em cada conector DB-9 com a adição do terminal RTS (*Request To Send*) de um dispositivo conectado ao terminal CTS (*Clear to Send*) do outro dispositivo, assim é estabelecido uma transmissão de solicitação entre os dispositivos. **(Fonte: How Stuff Works, www.informatica.hsw.uol.com.br)**.

O cabo direto efetua a conexão de um dispositivo DTE com um dispositivo DCE. Neste tipo de cabo os terminais TxD-RxD e RTS-CTS não têm cruzamento como no cabo *crossover*, daí chamado de cabo direto. **(Fonte: How Stuff Works, www.informatica.hsw.uol.com.br)**.

Na Figura 2.7 é mostrada como é efetuada as ligações entre os conectores DB9.

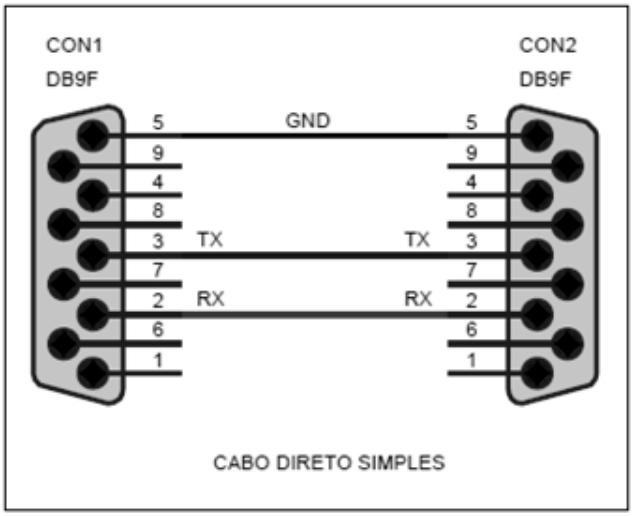


Figura 2.7 - Cabo do tipo direto simples. (Autor: José Carlos)

A porta serial de um PC é chamada de dispositivo DTE, enquanto os *modems* e as impressoras são chamados de dispositivos DCE. Um conversor com *interface* serial como, por exemplo, o circuito integrado MAX232 pode ser tanto um dispositivo DTE ou um DCE, mudando apenas o tipo de conector de terminais ou pinos machos DB-9 para o DTE e o conector fêmea para os dispositivos DCE.

Há a necessidade de prover uma comunicação pelo computador, utilizando qualquer programa de monitoramento da porta serial. O *Windows* XP, possui o *software Hyperterminal*. **(OLIVEIRA, 2006).**

2.3.3 - Tabela ASCII

Segundo Forouzan (2008, p. 1029) números são matéria-prima para computadores, que armazenam caracteres atribuindo um número a cada um deles. Um dos sistemas de codificação mais utilizado foi o ASCII (*American Standard Code for Informati*on

Interchange), contendo 128 códigos (0 a 127), cada um dos quais representado por um número de 7 *bits*. O ASCII pode, de forma satisfatória, lidar com letras maiúsculas e minúsculas, dígitos, sinais de pontuação e alguns caracteres de controle. Foi feita uma tentativa de estender o conjunto de caracteres ASCII para 8 *bits*. O novo código, denominado ASCII Estendido, jamais chegou a ser um padrão internacional.

Na Figura 2.8 é mostrada a representação dos caracteres ASCII de 7 bits.

		Bit Mais Significativo (MSB)							
		000	001	010	011	100	101	110	111
Bit Menos Significativo (LSB)	0000	NULL	DLE	SP	0	@	P	`	p
	0001	SOH	DC1	!	1	A	Q	a	q
	0010	STX	DC2	"	2	B	R	b	r
	0011	ETX	DC3	#	3	C	S	c	s
	0100	EOT	DC4	$	4	D	T	d	t
	0101	ENQ	NAK	%	5	E	U	e	u
	0110	ACK	SYN	&	6	F	V	f	v
	0111	BEL	ETB	'	7	G	W	g	w
	1000	BS	CAN	(	8	H	X	h	x
	1001	HT	EM	)	9	I	Y	i	y
	1010	LF	SUB	*	:	J	Z	j	z
	1011	VT	ESC	+	;	K	[	k	{
	1100	FF	FS	,	<	L	\	l	|
	1101	CR	GS	-	=	M	]	m	}
	1110	SO	RS	.	>	N	^	n	~
	1111	SI	US	/	?	O	_	o	DEL

Figura 2.8 – Representação dos caracteres ASCII de 7 bits. (Autor: José Carlos)

2.4 - Tecnologia *Bluetooth*

Segundo Forouzan (2008, p.434) *Bluetooth* é uma tecnologia para redes LANs sem fio (WLANs) desenvolvida para conectar diversos tipos de dispositivos de diferentes funções, como telefones, *Notebooks*, computadores (*desktop* e *laptop*), câmeras, impressoras, cafeteiras e assim por diante.

2.4.1 - Aplicações da tecnologia *Bluetooth*

As aplicações da tecnologia *Bluetooth* são enormes, sendo destacadas as aplicações:

- Comunicação com dispositivos conectados aos computadores como o *mouse*, teclados, câmeras fotográficas;
- Monitoramento através de sensores de coleta de dados comunicando-se com um telefone celular (*smartphone*);
- Dispositivos no monitoramento com comunicação por sensores utilizados numa UTI de hospital.
- Sincronismo de dados entre os *laptops* num congresso do palestrante com os participantes. **(FOROUZAN, 2008)**.

2.4.2 - A História do *Bluetooth*

No ano de 1994, a empresa Ericsson desenvolveu um sistema sem fio para conectar telefones móveis. Com a associação de mais quatro empresas, como a IBM, Intel, Nokia e Toshiba, formaram o consórcio SIG (*Special Interest Group*). O intuito deste consórcio foi o desenvolvimento de um padrão sem a utilização de fios para conectar dispositivos de computação, comunicação e acessórios. O projeto desenvolvido foi um dispositivo de radiofrequência sem fio de baixa potência e curto alcance. Este projeto ficou conhecido como *Bluetooth*, sendo uma homenagem ao rei viking Harald Blaatand (940-981), um unificador da Dinamarca e da Noruega. **(TANENBAUM, 2011)**.

Segundo Passos (2011, p. 28 e 29) e da mesma forma que o rei teria conseguido unificar suas tribos, o padrão *Bluetooth* procura unir diferentes tecnologias. Uma curiosidade acerca do logotipo é que o símbolo é a união de algumas runas (conjunto de alfabetos relacionados que utilizam letras características) nórdicas, como no caso de:  - H (Hagall) e  - B (Berkanan), que são as letras iniciais do rei viking e o "Dente-Azul" na sua tradução literal deram origem ao símbolo comumente utilizado, o *Bluetooth*.

Na Figura 2.9 é mostrado o símbolo criado para a tecnologia *Bluetooth*, compostas pelas iniciais do rei viking.

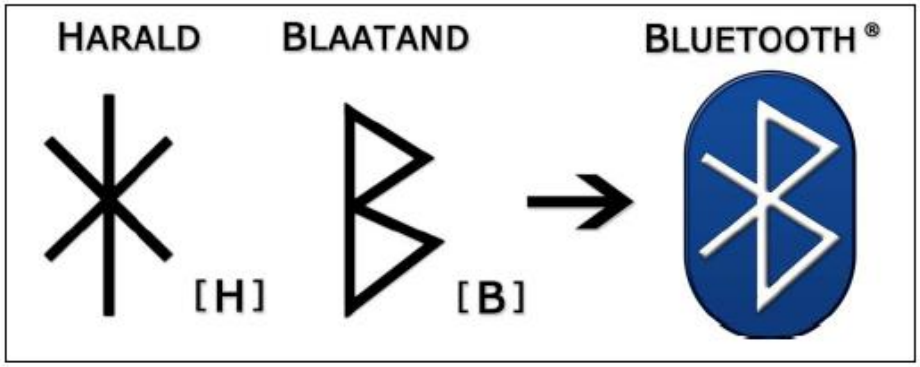


Figura 2.9 – Origem do Logotipo do *Bluetooth*. (Fonte: PASSOS, 2011)

Segundo Tanenbaum (2003, p.244) embora a *idéia* original fosse apenas se livrar dos cabos entre dispositivos, ela logo começou a expandir seu escopo e invadir a área das LANs sem fios. Embora essa mudança torne o padrão mais útil, também cria alguma competição pelo mercado com o 802.11. Para piorar, os dois sistemas também interferem eletricamente um no outro. Também vale a pena notar que a Hewlett-Packard introduziu uma rede de infravermelho para conectar periféricos de computadores sem fios há alguns anos, mas ele nunca obteve realmente um grande êxito.

2.4.3 - Arquitetura *Bluetooth*

2.4.3.1 - Rede *Piconet*

É uma pequena rede *Bluetooth* formada por até 8 estações, sendo que uma delas é chamada de primária e as outras sete estações de secundárias. As estações secundárias estão sincronizadas com a primária. Este tipo de rede possui apenas uma estação primária. A comunicação nesta rede é estabelecida via ponto-a-ponto ou multiponto. **(FOROUZAN, 2008)**.

Segundo Forouzan (2008, p.435) em referências bibliográficas, algumas vezes usam-se os termos mestre e escravo em vez de primária e secundária.

Na Figura 2.10 é mostrada uma rede *Bluetooth* mestre/escravo com arquitetura *piconet*.

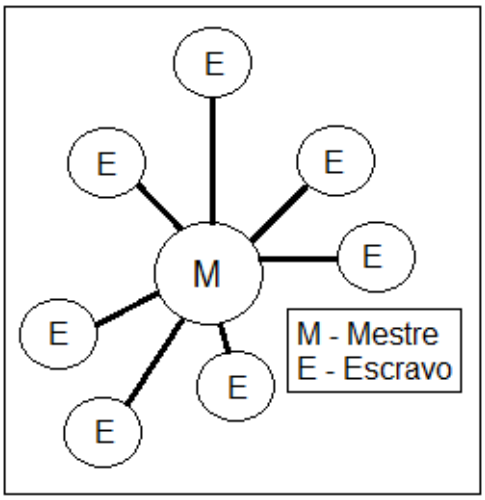


Figura 2.10 – Rede *piconet*. (Autor: José Carlos)

2.4.3.2 - Rede *Scatternet*

Podemos combinar a rede *piconet* para formar uma rede chamada *scatternet*. Uma estação escravo de uma *piconet* pode se transformar em estação mestre de outra *piconet*. Uma estação pode conectar-se a outra e trocar mensagens, como por exemplo, a estação de uma *piconet* onde estava atuando como escravo e agora atuando como mestre pode enviar informações para outros dispositivos escravos da outra rede *piconet*. Neste tipo de rede qualquer dispositivo pode pertencer a duas *piconet*. **(FOROUZAN, 2008)**.

2.4.3.3 - Taxa de dados

Para Forouzan (2008, p.436) todo dispositivo *Bluetooth* tem um transmissor embutido na faixa de radiofrequência de curto alcance. A taxa de dados atual é de 1 Mbps para a faixa de frequências de 2,4 GHz. Isso significa que existem possibilidades reais de interferências entre LANs IEEE 802.11b e as LANs *Bluetooth*.

2.4.3.4 - Modelo de camada do dispositivo *Bluetooth*

O *Bluetooth* dispõe de várias camadas que são diferentes das camadas utilizadas pela *Internet* ou outro tipo de rede. **(FOROUZAN, 2008)**.

Na Figura 2.11 são mostradas as camadas utilizadas pelo dispositivo *Bluetooth*.

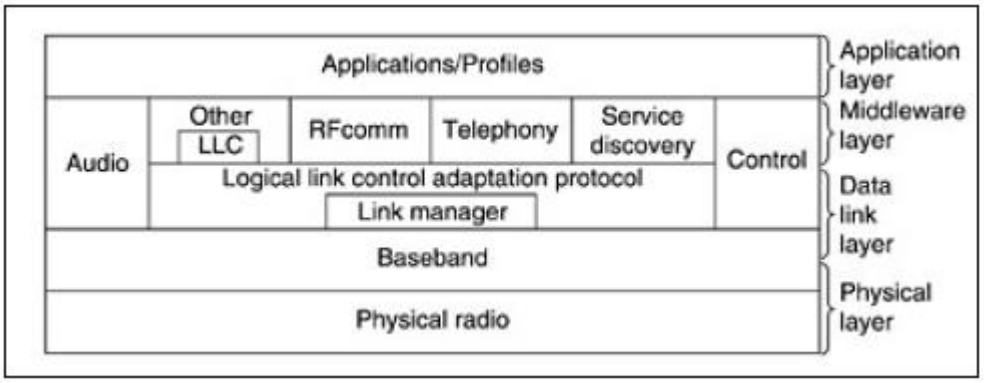


Figura 2.11 – Camadas utilizadas pelo *Bluetooth*. (TANENBAUM, 2003).

2.4.3.5 - Camada de rádio

Segundo Forouzan (2008, p.436) a camada de rádio equivale, a grosso modo, à camada física do modelo *Internet*. Os dispositivos *Bluetooth* são de baixa potência e têm um alcance típico de 10 metros.

A banda de frequências que os dispositivos *Bluetooth* operam, está na faixa de 2,4 GHz, reservada pela ISM (*Industrial, Scientific and Medical*), sendo divididas em 79 canais com largura de banda de 1 MHz. **(FOROUZAN, 2008)**.

Segundo Forouzan (2008, p.437) o *Bluetooth* usa o método de espalhamento espectral de saltos de frequência **FHSS** (*Frequency Hopping Spread Spectrum*) na camada física para evitar interferências com outros dispositivos ou outras redes. O *Bluetooth* realiza 1.600 saltos de frequências por segundo, o que significa que cada dispositivo muda sua frequência de modulação 1.600 vezes por segundo. Um dispositivo usa uma frequência por apenas 625 us (1/1.600 s) antes de saltar para outra frequência; o tempo de permanência em dada frequência é de 625 us.

A modulação dos sinais *Bluetooth* emprega a modulação GFSK (*Gaussian Frequency-Shift Keying*), que é uma modulação FSK (*Frequency Shift Keying*), que utiliza filtros Gaussianos para suavizar os desvios de frequências. **(FOROUZAN, 2008)**.

As frequências de cada canal em MHz podem ser encontradas conforme mostrado na Equação 1.

$$\mathbf{fc} = \mathbf{2402} + \mathbf{n}, \text{ onde } n = 0, 1, 2, 3, \ldots, 78$$

Equação 1 – Frequência de cada canal *Bluetooth*. (FOROUZAN, 2008).

Como exemplo, o primeiro canal da portadora é de 2402 MHz, o último canal é 2402 + 78, sendo igual a 2480 MHz (2,48 GHz). **(FOROUZAN, 2008)**.

2.4.3.6 - Camada banda base

A camada banda base tem sua equivalência com a camada MAC de redes LAN, pelo método TDD-TDMA (*Time Division Duplexing-Time Division Multiple Access*). Com a utilização da técnica *time-slots*, os dispositivos nas estações mestre e escravo efetuam a comunicação de dados entre si. **(FOROUZAN, 2008)**.

O *time-slot* dura por um tempo de 625 microssegundos, sendo chamado tempo de permanência. Este tempo de permanência faz com que o dispositivo mestre de uma estação envie *frames* para um dispositivo escravo ou vice-versa, não podendo o escravo de uma estação enviar informações para outro escravo. **(FOROUZAN, 2008)**.

A comunicação é estabelecida no formato *half-duplex,* tem como característica o envio e o recebimento de informações em momentos distintos. A comunicação se dá utilizando frequências em saltos diferentes. **(FOROUZAN, 2008)**.

2.4.3.7 - Camada L2CAP (*Logical Link Control Adaptation Protocol*)

A camada L2CAP (*Logical Link Control Adaptation Protocol*) recebe os pacotes das camadas superiores, segmentando-os em quadros para serem transmitidos. Na recepção estes quadros são remontados novamente em pacotes. Esta camada aceita pacotes com até 64 kB vindos das camadas superiores. **(TANENBAUM, 2011)**.

A camada L2CAP realiza também a multiplexação e demultiplexação dos dados de vários tipos de pacotes. Na remontagem do pacote, a camada L2CAP distingue em qual o protocolo da camada superior o pacote é entregue, se é para RFcomm (*Radio Frequency Communication*) ou telefonia. **(TANENBAUM, 2011)**.

A gerência da qualidade de serviço (QoS) para os protocolos das camadas superiores é feita pela camada L2CAP. **(TANENBAUM, 2011)**.

Também é negociado em tempo de configuração o tamanho máximo de carga útil permitido, a fim de impedir que um dispositivo de pacotes grandes afogue um dispositivo de pacotes pequenos. Esse recurso é necessário, porque nem todos os dispositivos podem manipular o pacote máximo de 64 kB. **(TANENBAUM , 2011)**.

Os protocolos de áudio e controle em algumas aplicações não necessitam passar pelo protocolo L2CAP. **(TANEMBAUM, 2011)**.

A camada *middleware* contem uma mistura de diferentes protocolos. O LLC (*Logical Link Control*) do 802 foi inserido aqui pelo IEEE para manter a compatibilidade com as outras redes 802. Os protocolos RFcomm, de telefonia e de descobertas de serviços são originais. O protocolo RFcomm é o protocolo que emula a porta serial padrão encontrada nos computadores pessoais para conectar o teclado, o *mouse* e o *modem*, entre outros dispositivos. **(TANEMBAUM, 2011)**.

O protocolo de telefonia é um protocolo de tempo real utilizado pelos três perfis orientados para voz. Ele também gerencia a configuração e o encerramento de chamadas. Por fim, o protocolo de descoberta de serviços é usado para localizar serviços na rede. **(TANEMBAUM, 2011)**.

As aplicações e os perfis se localizam na camada superior. Eles utilizam os protocolos das camadas inferiores para cumprir suas funções. Cada aplicação tem seu próprio subconjunto dedicado dos protocolos. Dispositivos específicos, como um fone de ouvido, em geral, só contém os protocolos exigidos por essa aplicação e nenhum outro. **(TANEMBAUM, 2011)**.

2.4.3.8 - Conexão *Bluetooth*

Uma conexão entre dispositivos *Bluetooth* envolve três estados:

1. *Inquiry*: Quando dois dispositivos *Bluetooth* não sabe nada um sobre o outro, é executado um *inquiry* para tentar descobrir o outro. Um dispositivo envia o pedido *inquiry* e qualquer dispositivo na escuta de tal pedido vai responder com seu endereço e, o seu nome e outras informações;

2. *Paging* (*Connecting*): É o processo de formação de uma ligação entre os dois dispositivos *Bluetooth*. Antes de iniciar está conexão, cada dispositivo tem de saber o endereço do outro (que se encontra no processo de *inquiry*);

3. *Connection*: Depois que um dispositivo tenha concluído o processo de *paging*, ele entra no estado de *Connecting*. Enquanto estiver ligado, um dispositivo pode participar ativamente ou pode ser colocado num modo de baixo consumo de energia chamado de *sleep*:

- Modo *Active*: Este é o modo conectado regular, onde o dispositivo está ativamente transmitindo ou recebendo dados;

- Modo *Sniff*: Este é um modo de economia de energia, onde o dispositivo é menos ativo. Ele entra no modo *sleep* e só ouve as transmissões em um intervalo de tempo definido (por exemplo, a cada 100 ms);

- Modo *Hold*: É um modo temporário de economia de energia, onde um dispositivo entra no modo *sleep* por um período definido e, em seguida, retorna para o modo ativo quando esse intervalo cessar. O mestre pode comandar um dispositivo escravo para o modo *hold*;

- Modo *Park*: É um estado do modo *sleep*. Um mestre pode comandar um escravo para o modo *park*, e o escravo ficará inativo até que o mestre diz-

lhe para sair do modo *sleep*. **(Fonte: Adaptado do site da Sparkfun, www.learn.sparkfun.com/tutorials/bluetooth-basics)**

2.4.3.9 - Emparelhamento

As ligações são criadas por meio de um tempo de um processo chamado de emparelhamento. Quando os dispositivos emparelham-se, eles compartilham seus endereços, nomes e perfis, e, normalmente, armazenando-os na memória. Também compartilham uma chave secreta comum, o que lhes permite a ligação sempre que estão próximos.

O emparelhamento pode ser uma simples operação, como o clique de um botão, isso é comum para dispositivos sem *interface* do usuário, como fones de ouvido. Outras vezes, o emparelhamento envolve códigos numéricos de 6 dígitos correspondentes. O legado mais antigo (versões 2.0 e anterior), os processos de emparelhamento envolvem a introdução de um código PIN comum em cada dispositivo. O código PIN pode variar em tamanho e complexidade de quatro números (por exemplo, “0000” ou “1234”) a uma sequência alfanumérica de 16 caracteres. **(Fonte: Adaptado do site da Sparkfun, www.learn.sparkfun.com/tutorials/bluetooth-basics)**

2.4.4 - Classificação de potência

Um dispositivo *Bluetooth* é especificado pela classe de operação, que define a potência e a distância de transmissão. Existem três classes que estipula a potência de transmissão em relação a distancia máxima de operação. **(Fonte: Adaptado do site da Sparkfun, www.learn.sparkfun.com/tutorials/bluetooth-basics)**.

No Quadro 1 são mostradas as três classes com as suas respectivas potência máxima de transmissão.

Quadro 1 – Classe de operação dos dispositivos *Bluetooth*. (Autor: José Carlos)

Classe do Dispositivo	Máxima Potência de Saída (dbm)	Máxima Potência de Saída (mW)	Distância máxima
Classe 1	20	100	100 m
Classe 2	4	2,5	10 m
Classe 3	0	1	10 cm

Segundo o manual da Itead Studio (2010, p 1), o dispositivo *Bluetooth* utilizado neste projeto é especificado para a classe 2 com potência máxima de saída de transmissão de RF de +4 dbm.

CAPÍTULO 3 - DESENVOLVIMENTO DO PROTÓTIPO

Este capítulo apresenta a aplicação dos conceitos teóricos tratados no capítulo 2, envolvidos na construção do protótipo. Será apresentado o modelo de protótipo desenvolvido, mostrando a descrição da construção, dos materiais utilizados e ferramentas do Ambiente de Desenvolvimento Integrado (IDE).

3.1 - Modelo Proposto para automação de uma residência

O modelo proposto é para uma residência contendo garagem com dois tipos de portões (portão de entrada social e portão deslizante para entrada de automóvel), sala, cozinha e dois quartos.

Todos os módulos de acionamentos são controlados por um aplicativo específico desenvolvido pelo proponente no acionamento dos dispositivos descritos anteriormente. Este aplicativo é instalado no telefone celular (*smartphone*).

Cada módulo de acionamento possui o seu próprio microcontrolador e conversor RS-485, exceto a placa do módulo *Bluetooth*-RS485.

Os microcontroladores PIC da Microchip serão utilizados para comandar os relés para os dispositivos eletroeletrônicos, bem como codificar/decodificar os caracteres ASCII enviados pelo *Bluetooth* do *smartphone*.

Na Figura 3.1 é mostrada a visão geral da automação residencial proposta.

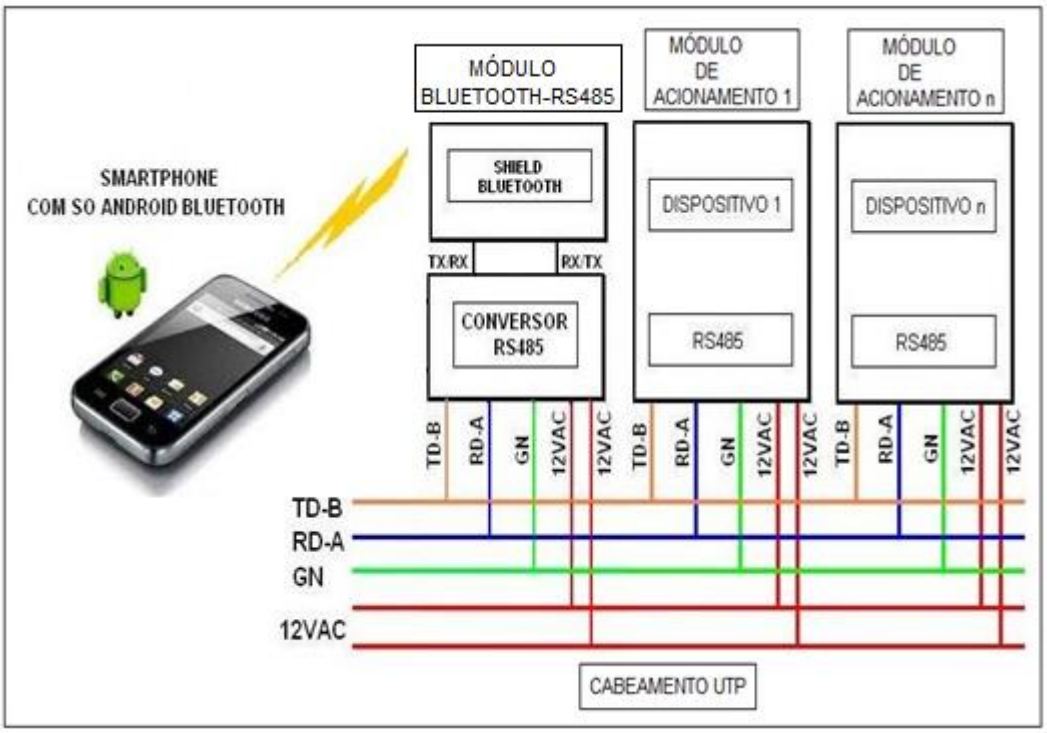


Figura 3.1- Visão Geral da automação residencial. (Autor: José Carlos)

Conforme mostrado na Figura 3.1, o *smartphone* é o responsável pelo envio dos sinais para o acionamento dos dispositivos eletroeletrônicos pelos módulos de acionamentos. Os módulos de acionamentos são interligados um aos outros, pelo cabeamento de par trançado formando um barramento para a comunicação dos sinais de transmissão e recepção.

O cabeamento de par trançado é chamado de UTP, sua utilização em larga escala permite o uso em aplicações, como a transmissão e recepção de sinais analógicos e digitais.

O módulo *Bluetooth*-RS485 é responsável pelo recebimento dos comandos do *smartphone*. O conversor RS-485 nesta placa acopla estes sinais por meio de três fios do cabeamento UTP, com o par diferencial TD-B (*Transmission Data-B*) e RD-A (*Reception Data-A*) e o terminal GN (*GROUND*) que serve como referência para os sinais TD-B e RD-A. O outro par de fios corresponde à alimentação de 12 Volts em corrente alternada (AC) para os módulos de acionamentos.

Cada módulo de acionamento conectado ao cabeamento UTP possui endereçamento codificado em binário, neste projeto 16 dispositivos podem ser acionados,

partindo do módulo de acionamento com endereço 0000 até ao módulo de acionamento com endereço 1111. Esta codificação é proporcionada pelos múltiplos interruptores mecânicos.

3.2 - Descrição dos Materiais Utilizados

3.2.1 - Módulo *Bluetooth* Mestre/Escravo

É uma placa contendo um módulo *Bluetooth* e periféricos interligados para a conexão de configuração em uma porta serial.

A comunicação serial é feita por intermédio de um *software* serial como o *Hyperterminal* do *Windows*. A configuração deste módulo é através dos comandos AT (*ATention Commands*), onde são configurados os parâmetros de comunicação serial, nome do dispositivo, senha de pareamento, modo de operação (mestre ou escravo).

Na Figura 3.2 é mostrado o módulo *Bluetooth* v2.2 mestre/escravo.

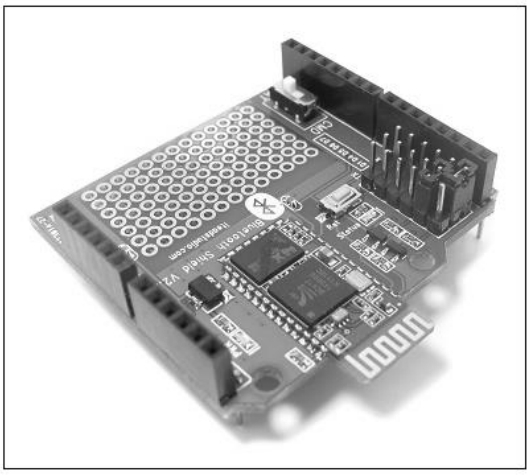


Figura 3.2 – Módulo *Bluetooth* da *Itead Studio*. (Fonte: www.imall.iteadstudio.com/im120417010.html)

O módulo *Bluetooth* v2.2 mestre/escravo é alimentado com uma tensão de 5 Volts sob uma corrente máxima de 40 mA quando não há pareamento com outro dispositivo. O módulo será responsável pela recepção das informações enviadas pelo telefone *smartphone*, através do pino de recepção RX do *Bluetooth* no conversor RS-485 que coloca estas informações no cabeamento de par trançado. O pino transmissão TX do *Bluetooth* envia as informações dos módulos de acionamentos ao telefone *smartphone*.

3.2.2 - Especificações do módulo *Bluetooth* v2.2 mestre/escravo

No Quadro 3 são mostradas algumas especificações para o módulo *Bluetooth* v2.2 mestre/escravo.

Quadro 3 - Especificações do módulo *Bluetooth* v2.2 mestre/escravo adaptado da *Itead Studio*. (Fonte: www.imall.iteadstudio.com/im120417010.html)

Microprocessador	CSR BC 417
Tamanho da PCI (em mm)	53,3 x 47 x 1,6
LEDs indicadores	Alimentação (PWR) e Status
Alimentação	5 volts DC
Pinos de I/O	6 pinos
Protocolo de comunicação	UART/Bluetooth 2.0

3.2.3 - *Hardware* do módulo *Bluetooth* v2.2 mestre/escravo

O módulo *Bluetooth* possui pinos configuráveis para a comunicação serial com um microcontrolador ou com o PC/*notebook*. Possui ainda um interruptor deslizante para configuração no modo de dados (DAT = *DATA*) ou comandos AT (CMD = *Command*), onde estabelecemos as características de funcionamento do módulo. Na Figura 3.3 são mostrados os principais pinos e o interruptor de modo.

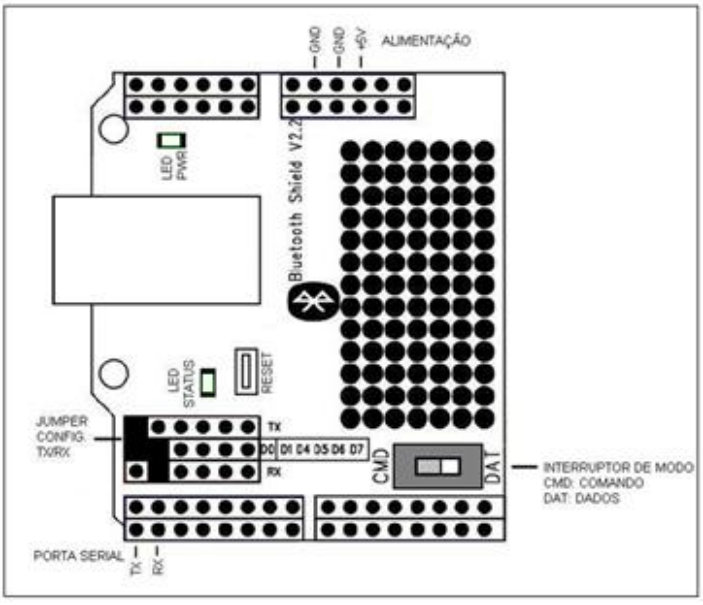


Figura 3.3 – *Hardware* do módulo *Bluetooth* com os principais pinos (Fonte: José Carlos)

3.2.4 - Configurações básicas de funcionamento do módulo *Bluetooth*

Os *jumpers* de configuração TX-RX no módulo *Bluetooth* são ajustados como mostrado na Figura 3.4. O módulo Conversor *Bluetooth*-RS485 possui o MAX232 para fazer a configuração pelo programa terminal no PC ou *notebook*, através de um cabo conversor serial com conectores USB e DB-9.

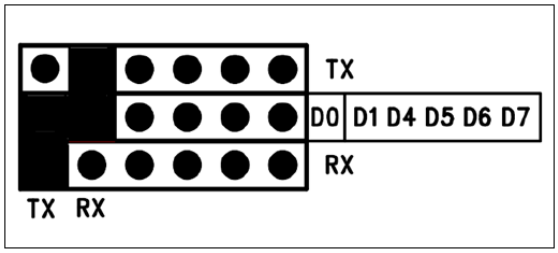


Figura 3.4 - Configuração TX-RX do módulo no modo Comando AT. (Fonte: José Carlos)

Ao alimentar o módulo com 5 Volts DC, este entra no modo de estado de comando (interruptor no modo CMD). A taxa de transmissão da *interface* de comunicação serial (*Hyperterminal*) com o módulo *Bluetooth* é de 38400 bps. O interruptor no modo DAT faz com que o módulo se comunique com um dispositivo pareado, enviando e recebendo informações.

3.3 - Microcontroladores PIC

3.3.1 - Microcontrolador PIC16F73

O PIC16F73 possui uma arquitetura de 8 *bits* baseado na tecnologia CMOS *FLASH* em um invólucro de 28 pinos, 22 pinos são para uso geral de entrada/saída (GPIO). Tem 5 canais de conversão Analógico para Digital (A/D) de 8 *bits* com 2 temporizadores (t*imers*), 2 periféricos para o modo *Capture*/*Compare*/PWM (CCP), porta serial síncrona configurável para 3 pinos SPI, 2 pinos para o barramento I^2C e uma *UART*. **(Fonte: Adaptado da Microchip, ww1.microchip.com/downloads/en/DeviceDoc/30325b.pdf).**

Na Tabela 1 são mostrados alguns periféricos do microcontrolador PIC16F73.

Tabela 1 – Descrição dos periféricos do microcontrolador PIC16F73. (Fonte: Adaptado da Microchip, www.microchip.com/www.products/Devices.aspx?dDocName=en010218)

Programa de Memória	***Flash***
Memória de Programa	7 kB
Velocidade da CPU	5 MIPS
RAM	192 *Bytes*
Periféricos Digitais de Comunicação	1-*UART*; 1-A/E/*USART*; 1-SPI; 1-I2C – 1-SSP (SPI/I2C)
Periféricos *Capture*/*Compare*/PWM	2 CCP
Timers	2 de 8 *bits* e 1de 16 *bits*
ADC	5 canais de 8 *bits*
Faixa de Temperatura	-40 a 125 grau Celsius
Faixa de Tensão de Operação	2 à 5,5 Volts
Quantidade de Pinos	28

O microcontrolador PIC16F73 possui 3 portas que podem ser configuradas como entradas e saídas para uso geral, são designadas por:

- Port A possui 6 pinos designados como RA0, RA1, RA2, RA3, RA4 e RA5;
- Port B possui 8 pinos designados como RB0, RB1, RB2, RB3, RB4, RB5, RB6 e RB7;
- Port C possui 8 pinos designados como RC0, RC1, RC2, RC3, RC4, RC5, RC6 e RC7.

Na Figura 3.5 é mostrado o microcontrolador PIC16F73 com invólucro de 28 pinos no formato DIP (*Dual In Package*).

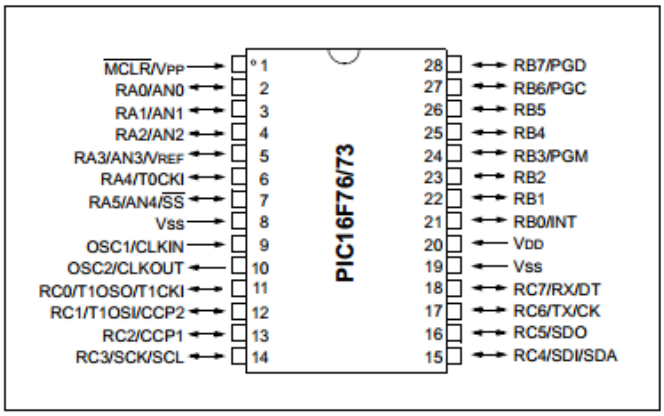


Figura 3.5 - PIC16F73 com invólucro de 28 pinos formato DIP. (Fonte: Microchip, ww1.microchip.com/downloads/en/DeviceDoc/30325b.pdf)

3.3.2 - Microcontrolador PIC16F628A

O PIC16F628A é baseado numa arquitetura de 8 *bits* com tecnologia CMOS *FLASH* em um invólucro de 18 pinos. Dos 18, 16 pinos são para uso geral de entrada/saída (GPIO). Possui 2 temporizadores (*timers*), 1 periférico para o modo *Capture*/*Compare*/PWM (CCP), porta serial *USART*. **(Fonte: Adaptado da Microchip, ww1.microchip.com/downloads/en/devicedoc/40044f.pdf).**

Na Tabela 2 são mostradas algumas características do microcontrolador PIC16F628A.

Tabela 2 – Descrição dos periféricos do microcontrolador PIC16F628A. (Fonte: Adaptado da Microchip, www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010210)

Programa de Memória	***Flash***
Memória de Programa	3,5 kB
Velocidade da CPU	5 MIPS
RAM	224 *Bytes*
Periféricos Digitais de Comunicação	1-*UART*; 1-A/E/*USART*;
Periféricos *Capture*/*Compare*/PWM	1 CCP
Timers	2 de 8 *bits* e 1de 16 *bits*
Comparadores	2
ADC	5 canais de 8 *bits*

Faixa de Temperatura	-40 a 125 grau Celsius
Faixa de Tensão de Operação	2 à 5,5 Volts
Quantidade de Pinos	18

O microcontrolador PIC16F628A possui 2 portas que podem ser configuradas como entradas e saídas para uso geral, são designadas por:

- Port A possui 8 pinos designados como RA0, RA1, RA2, RA3, RA4, RA5, RA6 e RA7;
- Port B possui 8 pinos designados como RB0, RB1, RB2, RB3, RB4, RB5, RB6 e RB7.

Na Figura 3.6 é mostrado o invólucro de 18 pinos no formato DIP (*Dual In Package*) do microcontrolador PIC16F628A.

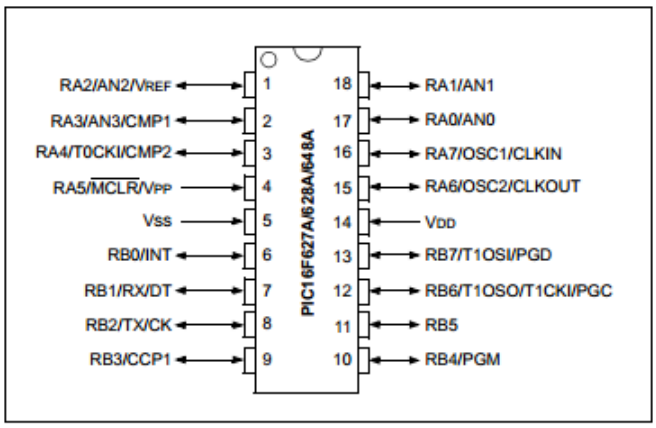


Figura 3.6 - PIC16F628A com invólucro de 18 pinos formato DIP. (Fonte: Microchip, ww1.microchip.com/downloads/en/devicedoc/40044f.pdf)

3.3.3 - Microcontrolador PIC12F675

O PIC12F675 possui uma arquitetura de 8 *bits* com tecnologia CMOS *FLASH* com invólucro de 8 pinos. Dispõe de 6 pinos para uso geral de entrada/saída (GPIO). A porta do microcontrolador é designada por:

- Port A possui 6 pinos designados como GP0, GP1, GP2, GP3, GP4 e GP5.

Possui 2 temporizadores (*timers*), um temporizador de 8 e um de 16 *bits*, um canal de comparação, 4 canais para conversão analógica para digital (A/D) de 10 *bits*, 128 *bytes* de memória de dados *EEPROM*.

Na Tabela 3 são mostradas as características do microcontrolador PIC12F675.

Tabela 3 – Descrição dos periféricos do microcontrolador PIC12F675. (Fonte: Adaptado da Microchip, www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010114)

Programa de Memória	***Flash***
Memória de Programa	1,75 kB
Velocidade da CPU	5 MIPS
RAM	64 *Bytes*
***EEPROM* de dados**	128 *Bytes*
Comparadores	1
Timers	1 de 8 *bits* e 1de 16 *bits*
ADC	4 canais de 10 *bits*
Faixa de Temperatura	-40 a 125 graus Celsius
Faixa de Tensão de Operação	2 à 5,5 Volts
Quantidade de Pinos	8

Na Figura 3.7 é mostrado o microcontrolador PIC12F675 com invólucro de 8 pinos no formato DIP (*Dual In Package*).

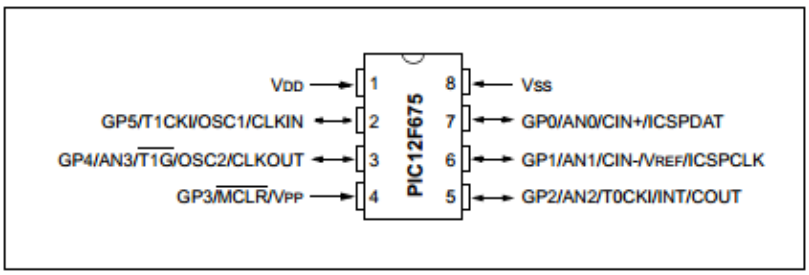


Figura 3.7 - PIC12F675 com invólucro de8 pinos formato DIP. (Fonte: Microchip, ww1.microchip.com/downloads/en/DeviceDoc/41190G.pdf)

3.3.4 - Circuito integrado MAX232

O MAX232 é um transmissor/receptor duplo que inclui um gerador de tensão capacitivo para fornecer os níveis de tensão TIA/EIA-232-F a partir de uma única alimentação

de 5 Volts. Cada receptor converte entradas TIA/EIA-232-F a níveis TTL/CMOS com 5 Volts. **(Fonte: Adaptado da Texas Instruments, www.ti.com/lit/ds/symlink/max232.pdf).**

Na Figura 3.8 é mostrado o encapsulamento DIP de 16 pinos e uma *interface half-duplex* utilizando o MAX232.

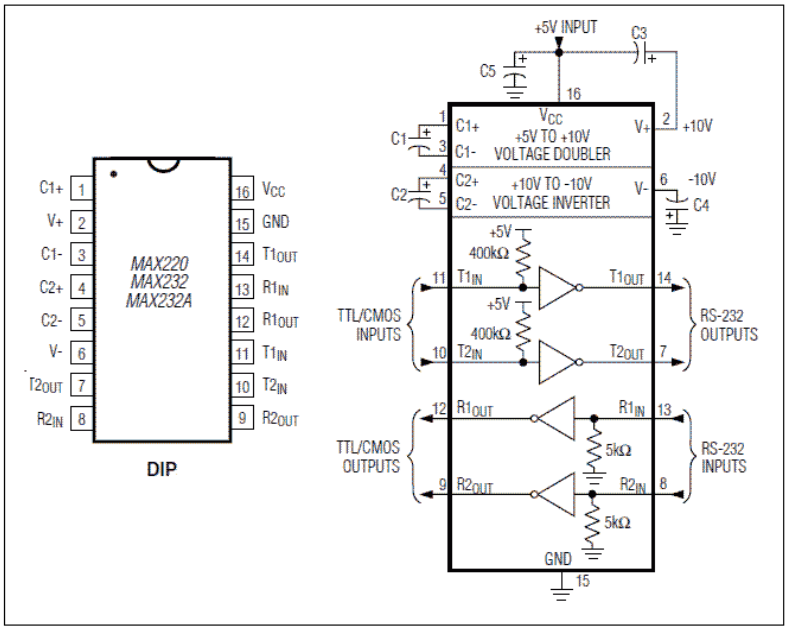


Figura 3.8 – Invólucro DIP e interface *Half-Duplex* com o MAX232. (Fonte: Maxim Integrated, www.maximintegrated.com).

3.3.5 - Circuito integrado MAX485

O circuito integrado MAX485 é um transceptor de baixa potência para comunicação RS-485. É responsável pela transmissão e recepção dos sinais no cabeamento UTP. O circuito integrado possui um transmissor e um receptor.

Aplicações para o RS-485 incluem redes de controle de processos, automação industrial, terminais remotos, automação predial, tais como aquecimento, ventilação, ar condicionado (HVAC), sistemas de segurança, controle de motores e controle de movimento. **(Fonte: Adaptado da Analog Devices, www.analog.com)**.

Na Figura 3.9 é mostrada a nomenclatura dos pinos do circuito integrado MAX485.

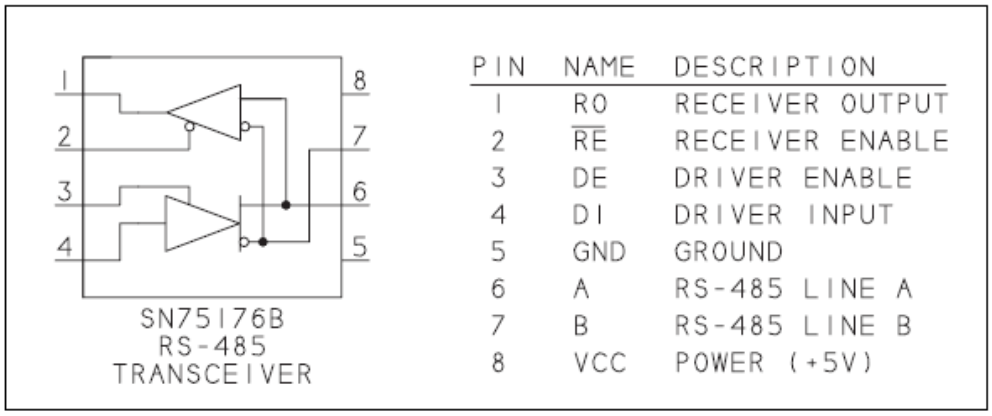


Figura 3.9 – Nomenclatura dos pinos do Circuito Integrado MAX485. (Fonte: Axelson, 2007).

Na Figura 3.10 é mostrada uma aplicação para uma rede *half-duplex* com o circuito integrado MAX485. A linha de transmissão "A" e "B" possui resistores terminadores de 120 Ω na entrada da rede e no final, para o balanceamento da linha.

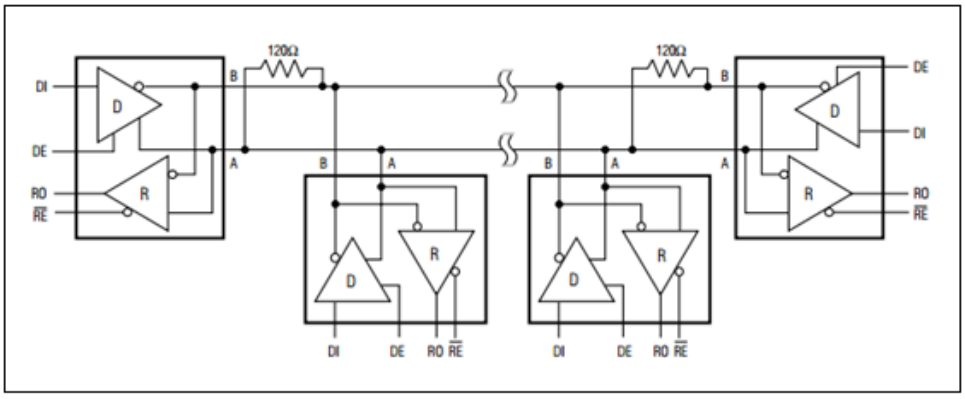


Figura 3.10 – Aplicação de uma Rede Típica Half-Duplex com o MAX485. (Fonte: Site da Maxim Integrated, www.datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf).

3.3.6 - *Dip Switch*

É um componente utilizado em circuitos eletroeletrônicos, sendo formado por múltiplos interruptores encapsulado num invólucro DIP (*Dual In Package*), com um conjunto de terminais posicionados em duas linhas para serem soldados em uma placa de circuito impresso.

O par de terminais de cada interruptor é acionado por uma alavanca que ao ser movido para a posição *ON*, fecha o contato elétrico e a corrente elétrica pode circular pelo circuito.

O *Dip Switch* permite a configuração de dispositivos eletrônicos usando uma codificação geralmente em binário. Neste projeto é responsável pelo endereçamento dos módulos de acionamentos.

Na Figura 3.11 é mostrado um *Dip Switch* com quatro interruptores para montagem nas placas de circuitos impressos.

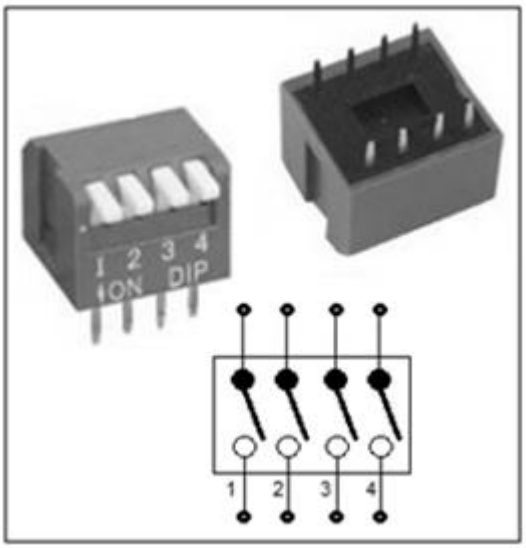


Figura 3.11 – *Dip Switch* com 4 interruptores. (Autor: José Carlos)

3.3.7 - Cabo conversor USB RS-232

O cabo conversor USB RS-232 é utilizado para fazer a configuração dos comandos AT necessário para o funcionamento do módulo *Bluetooth*. A configuração é feita utilizando um programa terminal (*Hyperterminal*) instalado em um PC ou *notebook*.

Este cabo possui um conector USB do tipo A em uma das extremidades do cabo, na outra um conector macho DB-9.

Na Figura 3.12 é mostrado o cabo conversor USB RS-232 com conector USB tipo A e DB-9 e um adaptador de conexão DB-9 para DB-25.

Figura 3.12 – Cabo conversor USB RS232. (Fonte: Hu Infinito, www.huinfinito.com.br/conversores/197-conversor-usb-rs232-cabo.html)

3.3.8 - Sensor de temperatura LM35

O LM35 é um circuito integrado sensor de temperatura de precisão, com uma tensão de saída linear proporcional ao valor de temperatura em graus centígrados.

O sensor de temperatura possui uma resolução de 10 mV por cada 1 ºC, a entrada analógica do PIC12F675 tem uma resolução de 10 *bits*, que corresponde à 2^{10}, resultando em 1024 valores lido pelo canal de conversão analógica para digital do PIC. Como cada grau corresponde a 10 mV, a expressão de temperatura em função do valor lido na entrada analógica do microcontrolador.

Na Equação 2 é mostrado o cálculo da temperatura em função dos valores da conversão analógica para digital feita pelo microcontrolador PIC.

$$\mathbf{Temperatura} = \left(\mathbf{Valor\ ADC} * \left(\frac{\mathbf{Vdd}}{\mathbf{Resolução}}\right)\right) * \mathbf{100}$$

Equação 2 – Cálculo da temperatura em função dos valores da conversão analógica para digital (Autor: José Carlos)

Onde:

Valor ADC: resultado da conversão analógica para digital;

Vdd: tensão de alimentação do microcontrolador (5 V, quando a fonte de referência ADC for externa);

Resolução: quantidade de *bits* utilizados na conversão analógica para digital.

Condutores elétricos quando submetidos à passagem de uma corrente elétrica dissipa uma energia em forma de calor. Para que a leitura de um sensor seja confiável, a temperatura lida tem que ser igual ou próxima ao do valor real. A energia dissipada na forma de calor tem que ser a mais baixa possível, sendo uma das mais importantes características para a escolha do sensor de temperatura.

O LM35 possui um aquecimento interno muito baixo, dissipando a energia em forma de calor de 0.1ºC sob um regime de corrente de 60 μA.

Na Figura 3.13 é mostrada a disposição dos pinos do sensor de tempertura LM35.

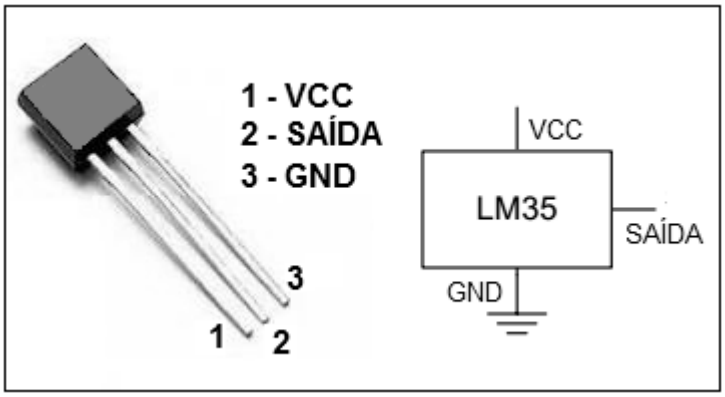


Figura 3.13 – Disposição dos pinos do sensor de temperatura LM35. (Autor: José Carlos)

3.3.9 - Teclado matricial 4 x 3

O teclado 4 x 3 é utilizado para o usuário ligar ou desligar os dispositivos eletroeletrônicos na residência. É parte integrante do módulo de acionamento manual, composto por dez teclas numéricas em um arranjo matricial com quatro linhas e três colunas.

O microcontrolador PIC16F73 gerencia as linhas e colunas do teclado detectando qual tecla é pressionada, quando uma tecla é pressionada o microcontrolador envia um caractere em ASCII através do conversor MAX485 no módulo de acionamento de usuário. Os módulos de acionamentos dos dispositivos recebem o sinal pelo cabeamento UTP, decodificam qual caractere em ASCII foi enviado ligando ou desligando o dispositivo no qual contém o código da tecla pressionada no teclado.

Na Figura 3.14 é mostrado o teclado 4 x 3 composto por teclas de membranas flexíveis.

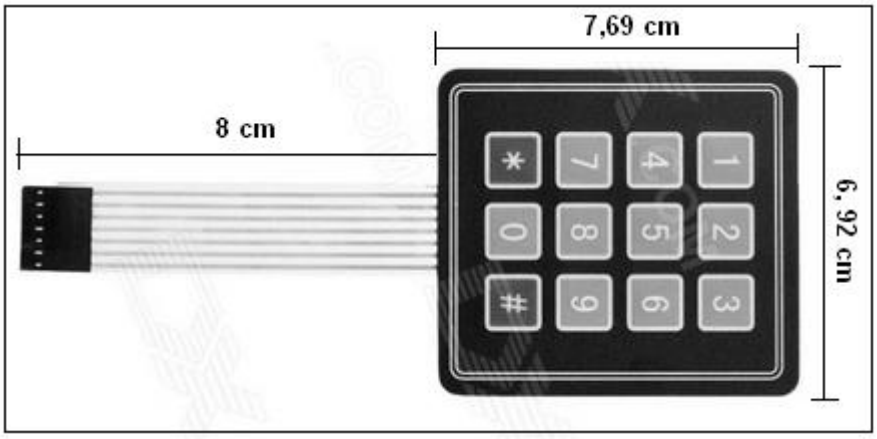


Figura 3.14 – Teclado matricial 4 x 3. (Fonte: Adaptado da Deal Extreme, www.dx.com/pt/p/3x4-matrix-12-key-membrane-switch-keypad-keyboard-117718)

Conforme mostrada na Figura 3.14, o teclado possui as dimensões de 6,92 x 7,69 centímetros e um cabo com um conector de 8 centímetros.

A tecla sustenido (#) foi programada para desligar todos os dispositivos eletroeletrônicos da residência, para o fecho elétrico do portão de entrada social, portão de entrada de veículos e a persiana, esta tecla não tem atuação. A tecla asterisco (*) foi programada para ligar as luzes da residência.

O teclado 4 x 3 é feito para ser colado em uma superfície lisa, como a de uma caixa para abrigar a placa do módulo de acionamento de usuário.

3.3.10 - Fecho elétrico de 12 Volts

Utilizado para a abertura de portas ou portão de entrada social por meio de um acionamento elétrico, com um botão de comando ou controle remoto. Possui uma bobina eletromagnética que ao ser acionada por corrente elétrica contínua ou alternada dispara um mecanismo para a abertura da porta ou portão.

O fecho elétrico é alimentado com tensão contínua ou alternada de 12 Volts com corrente de 1 Ampère.

Na Figura 3.15 é mostrado o fecho elétrico de 12 V, marca HDL, modelo FEC91 com espelho fixo.

Figura 3.15 – Fecho elétrico FEC 91. (Fonte: Leroy Merlim, www.leroymerlin.com.br/fecho-eletrico-fec91-espelho-fixo-hdl_87566073)

3.3.11 - Contactor

O contactor é considerado um relé para acionar cargas com corrente elétrica intensa. Na Figura 3.16 é mostrado o contactor utilizado para o acionamento do ar-condicionado, sendo que a bobina eletromagnética é acionada por uma tensão de corrente alternada de 220 Volts.

Figura 3.16 - Contactor. (Fonte: Loja do Circuito Elétrico, www.circuitoeletrico.com.br/loja/product.php?id_product=45)

Na Figura 3.17 é mostrado o diagrama elétrico do contactor. A bobina eletromagnética é representada pelos terminais A1 e A2, os contatos de potência para acionamento da carga estão representados por T1, T2 e T3, enquanto as linhas de alimentação da carga são indicadas por L1, L2 e L3. Os contatos NO são contatos auxiliares para o acionamento de uma bobina eletromagnética de outro contactor.

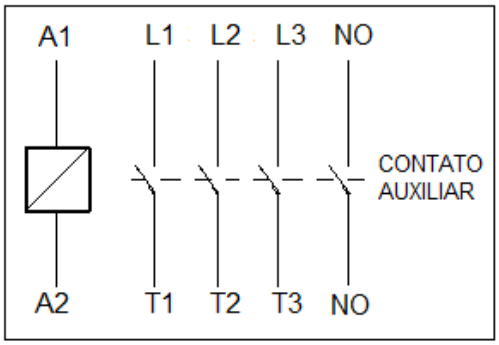


Figura 3.17 - Diagrama elétrico do contactor. (Autor: José Carlos)

3.3.12 - Módulo de alimentação dos circuitos

O cálculo do dimensionamento do transformador é feito a partir da quantidade de módulos do projeto, baseando-se na corrente de acionamento da bobina eletromagnética dos relés dos módulos de acionamentos e módulo sensor de temperatura. Cada relé drena da fonte cerca de 30 mA, o módulo *Bluetooth*-RS485 drena 40 mA quando não está conectado ao *Bluetooth* do *smartphone*, o módulo de acionamento manual no pior caso quando todos os LEDs estão acionados drenam 80 mA.

No Quadro 2 são mostrados a corrente de consumo de cada módulo do projeto para o dimensionamento da corrente do transformador.

Quadro 2 – Corrente de consumo dos módulos do projeto. (Autor: José Carlos)

DESCRIÇÃO	QUANT.	CORRENTE DE OPERAÇÃO	CORRENTE TOTAL
MÓDULOS DE ACIONAMENTOS	05	30 mA	150 mA
MÓDULO SENSOR TEMPERATURA	01	30 mA	30 mA
MÓDULO *BLUETOOTH*-RS485	01	40 mA	40 mA
MÓDULO DE ACIONAMENTO MANUAL	01	80 mA	80 mA
	SOMA DA CORRENTE TOTAL = 300 mA		

A corrente do transformador pode ser estipulada como duas vezes o valor da soma total de corrente dos módulos. A escolha recai em um transformador de 6 + 6 Volts com corrente de 1 Ampère.

O fecho elétrico é alimentado também por outro transformador de 6 + 6 Volts e 1 Ampère de corrente. Na Figura 3.18 é mostrado o transformador para os módulos e para o fecho elétrico.

Figura 3.18 – Transformador para os módulos e fecho elétrico. (Fonte: Site da Hu Infinito, www.huinfinito.com.br/indutores/756-transformador-abaixador-6v6v-1a.html)

3.4 - Desenvolvimento dos módulos de acionamentos na IDE Proteus 7.8 SP2

O Ambiente de Desenvolvimento Integrado (IDE) Proteus, possui o *software* ISIS para captura esquemática e simulação *SPICE* (*Simulation Program With Integrated Circuit Emphasis)*, o software ARES utilizado para roteamento manual ou automático das trilhas de circuito impresso das placas, visualização da placa em 3D.

O IDE Proteus foi desenvolvido pela Empresa *Labcenter Electronics*, sendo uma ferramenta completa no desenvolvimento de projetos de eletrônica analógico-digital e eletrônica embarcada. Sua biblioteca compreende uma enorme quantidade de componentes, incluindo algumas famílias de microcontroladores.

3.4.1 - Módulos de acionamentos dos dispositivos eletroeletrônicos

Cada módulo de acionamento possui um microcontrolador PIC16F628A para a codificação e decodificação dos caracteres ASCII e acionamento de cargas.

Um modelo básico dos módulos de acionamentos desenvolvido no ambiente de simulação do Proteus/ISIS recebe os comandos enviados pelo teclado do computador codificados em ASCII. Os terminais RB1/RX e RB2/TX do PIC16F628A fazem a comunicação serial.

Na Figura 3.19 é mostrada a representação do modelo básico para os módulos de acionamentos. O terminal *virtual* (*VIRTUAL TERMINAL*) foi utilizado para simular o canal de comunicação serial entre o *Bluetooth* e os microcontroladores.

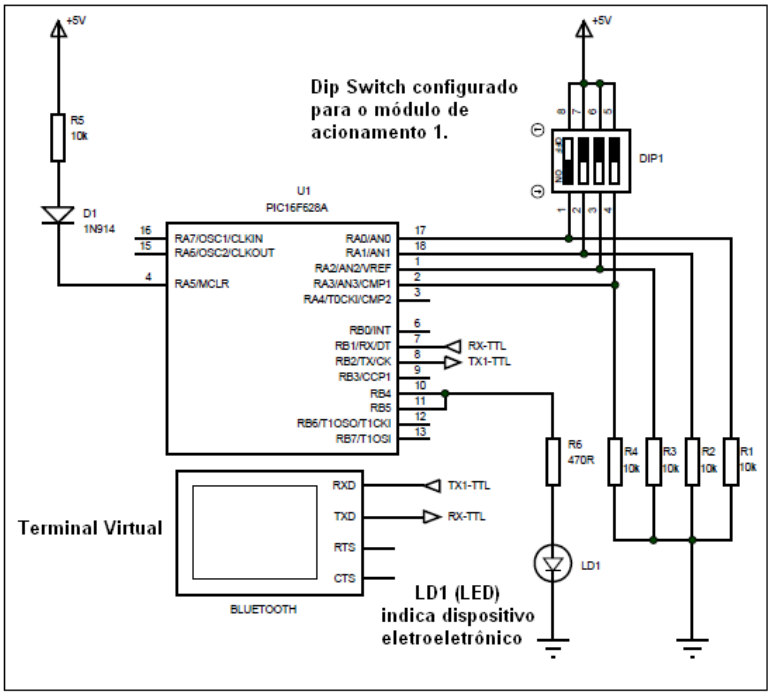


Figura 3.19 – Modelo básico dos módulos de acionamentos. (Autor: José Carlos)

O *Dip Switch* (DIP1) faz a codificação ou endereçamento dos módulos de acionamentos dos dispositivos eletroeletrônicos. Utilizando quatro interruptores do *Dip Switch*, pode-se endereçar até 16 módulos de acionamentos, com um código em BCD, do endereço 0000 até ao endereço 1111.

No Quadro 3 são mostradas as possíveis combinações do *Dip Switch* com quatro micros interruptores para a codificação de até 16 módulos de acionamentos.

Quadro 3 – Codificação dos módulos de acionamentos pelo *Dip Switch* (Autor: José Carlos)

***DIP SWITCH* (BCD)**					
4	**3**	**2**	**1**	**VALOR DECIMAL**	**MÓDULO ACIONADO**
0	0	0	0	0	MÓDULO DE ACIONAMENTO 0
0	0	0	1	1	MÓDULO DE ACIONAMENTO 1
0	0	1	0	2	MÓDULO DE ACIONAMENTO 2

0	0	1	1	3	MÓDULO DE ACIONAMENTO 3
0	1	0	0	4	MÓDULO DE ACIONAMENTO 4
0	1	0	1	5	MÓDULO DE ACIONAMENTO 5
0	1	1	0	6	MÓDULO DE ACIONAMENTO 6
0	1	1	1	7	MÓDULO DE ACIONAMENTO 7
1	0	0	0	8	MÓDULO DE ACIONAMENTO 8
1	0	0	1	9	MÓDULO DE ACIONAMENTO 9
1	0	1	0	10	MÓDULO DE ACIONAMENTO 10
1	0	1	1	11	MÓDULO DE ACIONAMENTO 11
1	1	0	0	12	MÓDULO DE ACIONAMENTO 12
1	1	0	1	13	MÓDULO DE ACIONAMENTO 13
1	1	1	0	14	MÓDULO DE ACIONAMENTO 14
1	1	1	1	15	MÓDULO DE ACIONAMENTO 15

O desenvolvimento do programa do microcontrolador para a comunicação serial dos caracteres em ASCII pelo teclado do PC/*notebook* utiliza a linguagem C para microcontroladores com o IDE PIC C *Compiler* da Empresa CCS.

Na Figura 3.20 é mostrado um trecho do código do programa para a configuração da *UART* do microcontrolador PIC16F628A.

A diretiva: #use rs232(baud=9600, xmit=PIN_B2, rcv=PIN_B1) faz a configuração do canal de comunicação serial RS-232 TTL do microcontrolador, taxa de transmissão em 9600 bps, “xmit” estabelece que o pino RB2 é para transmissão serial e “rcv” com o pino RB1 para recepção dos comandos seriais.

```
/*****************************************************************************************************/
#include <16f628A.h>
#fuses INTRC_IO,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT
#use delay(clock=4000000)
#use standard_io(b)
#use rs232(baud=9600, xmit=PIN_B2, rcv=PIN_B1)

/*
velocidade de transmissão 9600 bps, sem paridade, 8 bits de dados.
RB2 pino de transmissão TTL/RS232.
RB1 pino de recepção TTL/RS232.
*/

/*****************************************************************************************************/
```

Figura 3.20 - Trecho do código do programa da configuração da *UART* do PIC. (Autor: José Carlos)

Na Figura 3.21 é mostrado o trecho do programa para o módulo de acionamento. Na linha 81 é mostrada a codificação feita com o *Dip Switch*, conectado aos pinos RA3, RA2, RA1 e RA0 do microcontrolador PIC. O endereçamento ou a codificação de cada módulo de acionamento é feito com o comando *switch*(Endereco) na linha 90, e o comando case 0b00000001: linha 91.

```
78
79   do
80   {
81    Endereco = PORT_A & 0b00001111;  //Máscara nos pinos de entrada do Port A RA<3:0>.
82
83 //Cada módulo de acionamento tem endereço configurado de 0000 a 1111 no Dip Switch.
84
85     if(Tecla_Pressionada!=0x00)  //Verifica se alguma tecla foi pressionada.
86     {
87
88 /*************************************Módulo de Acionamento 1**********************************/
89
90      switch(Endereco)
91      {
92       case 0b00000001: //Módulo de Acionamento 1.
93       {
94        Envia_Comando(); //Função de envio de caracteres pela serial.
95       } //Fim do case endereço.
96
97       Tecla_Pressionada=0x00;
98      } //Fim switch endereço do RX1.
99
100 /*************************************Módulo de Acionamento 2**********************************/
101
102     switch(Endereco)
103     {
104      case 0b00000010: //Módulo de Acionamento 2.
```

Figura 3.21 - Trecho do código para o módulo de acionamento 1 (Autor: José Carlos)

Para ligar e desligar o dispositivo eletroeletrônico conectado ao módulo de acionamento 1 é utilizada a função Envia_Comando() com o *case* ‘1’ (linha 38) para ligar e *case* ‘a’ (linha 48), para desligar. Na Figura 3.22 é mostrado o trecho do programa da função Envia_Comando().

```
27 void Envia_Comando()
28 {
29 /*****************************Codificação do Módulo de Acionamento 1*****************************/
30
31 //Cada módulo de acionamento tem endereço configurado de 0000 a 1111 no Dip Switch.
32   switch(Endereco)
33   {
34    case 0b00000001:  //Endereço do Módulo de Acionamento 1.
35    {
36     switch(Tecla_Pressionada) //Cada tecla acionada controla um dispositivo.
37     {
38      case '1': //Valor do número "1" em ASCII (hex) = 31 convertido p/ decimal = 49.
39      printf("\f1");  //Envia p/ serial o número "1".
40      output_high(pin_B5);
41      delay_ms(100);
42      output_high(pin_B0);  //Desabilita TX e habilita RX MAX485.
43      printf("\fLIGADO");
44      delay_ms(250);
45      output_low(pin_B0);   //Habilita TX e desabilita RX MAX485.
46      break;
47
48      case 'a': //Valor de "a" em ASCII (hex) = 61 convertido p/ decimal = 97.
49      printf("\fa");
50      output_low(pin_B5);
51      delay_ms(100);
52      output_high(pin_B0);  //Desabilita TX e habilita RX MAX485.
53      printf("\fDESLIGADO");
54      delay_ms(250);
55      output_low(pin_B0);   //Habilita TX e desabilita RX MAX485.
        break;
```

Figura 3.22 – Trecho do programa da função Envia_Comando(). (Autor: José Carlos)

Na Figura 3.23 é mostrado o diagrama do circuito elétrico para a simulação no Proteus/ISIS.

O *LED* LD1 simula o dispositivo eletroeletrônico que será ligado ou desligado. O terminal *virtual* (*Virtual Terminal*), rotulado como *Bluetooth*, recebe os caracteres do teclado do computador, quando for pressionado o caractere 1, o computador converte este número em ASCII: 0110001 (decimal 49), o terminal virtual recebe esta informação pelo pino TX1-TTL e envia ao pino 7 (RB1/RX) do microcontrolador PIC16F628A, fazendo com que o *LED* LD1 seja acionado emitindo luz.

Para desligar o *LED* LD1 o caractere “a” é convertido em ASCII para: 1100001(decimal 97).

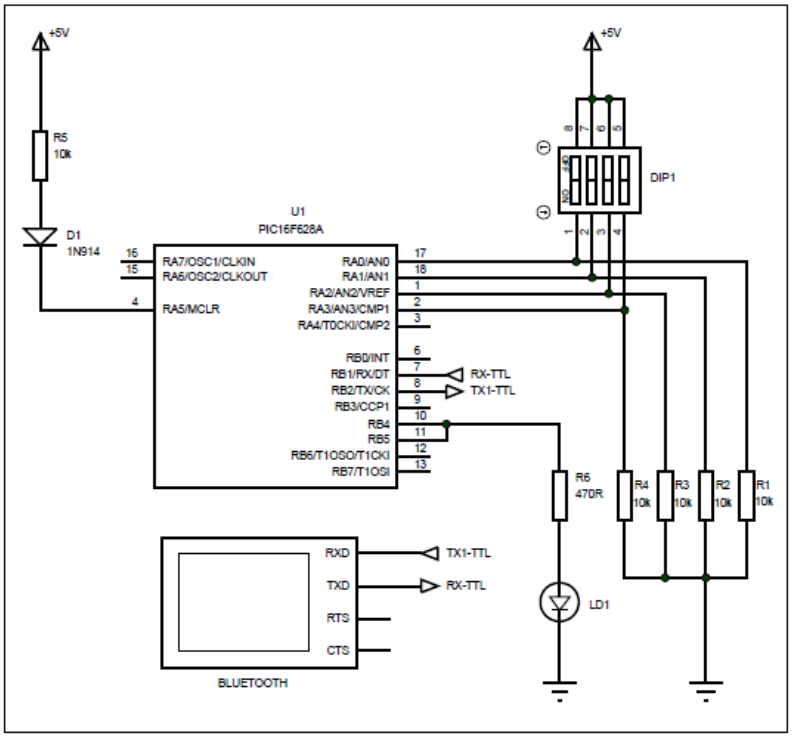


Figura 3.23 - Simulação para os módulos de acionamentos (Autor: José Carlos)

3.4.2 - Módulo *Bluetooth*-RS485

O circuito do módulo *Bluetooth*-RS485 não foi simulado no ambiente Proteus/ISIS, porque a versão demo 7.8 SP2 do IDE Proteus não possui em sua biblioteca o conversor de nível MAX485 para a simulação, sendo que o mais próximo que esta biblioteca possui é o MAX487. Este circuito foi montado numa *Protoboard*. Na Figura 3.24 é mostrada a montagem do circuito na *Protoboard*.

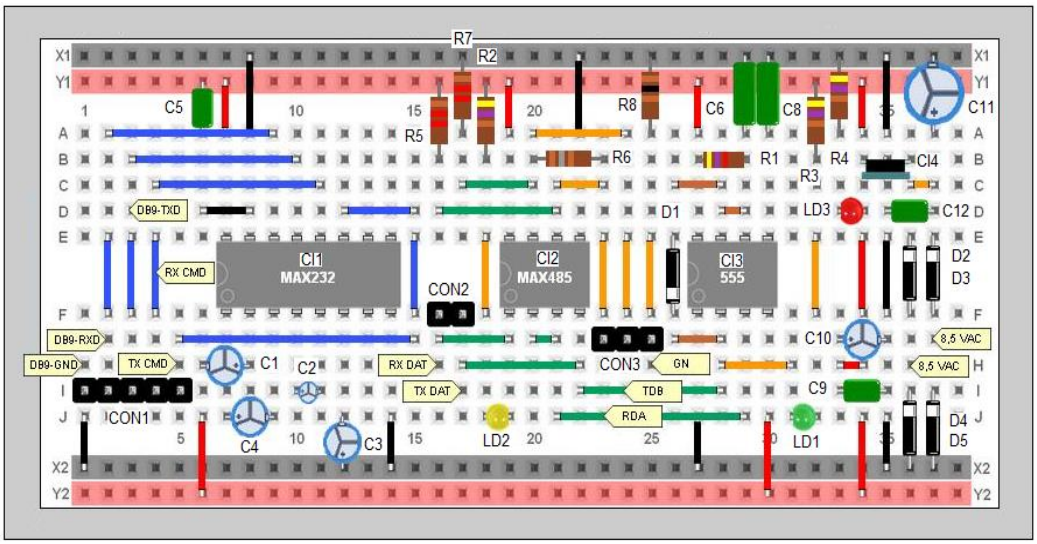


Figura 3.24 – Circuito *Bluetooth*-RS485 montado na *Protoboard*. (Autor: José Carlos)

Depois de montado o circuito foi revisado, garantindo que não há ligações erradas através dos fios entre os blocos do circuito na *Protoboard*.

Efetuou-se alguns testes preliminares de funcionamento no circuito do módulo *Bluetooth*-RS485 montado na *Protoboard*, testes como a configuração dos comandos AT no *Bluetooth* e a conexão entre o *Bluetooth* e um *smartphone*. Este teste visa possíveis correções elétricas e de funcionamento para o desenvolvimento da placa de circuito impresso.

3.4.3 - Módulo de acionamento manual

O módulo de acionamento manual possui dois microcontroladores:

- Um PIC16F73 gerencia o teclado matricial 4 x 3, o *buzzer* e o conversor RS-485. O teclado matricial é utilizado para o acionamento dos dispositivos eletroeletrônicos, caso o usuário não queira utilizar o *smartphone*. O *buzzer* é um dispositivo eletroacústico que faz a emissão de som quando uma tecla for pressionada pelo usuário no teclado 4 x 3.
- Um PIC16F628A faz o controle dos *LEDs*, indicando o acionamento de dispositivos eletroeletrônicos.

Na Figura 3.25 é mostrado o diagrama de circuito elétrico do módulo de acionamento manual para a simulação no Proteus ISIS.

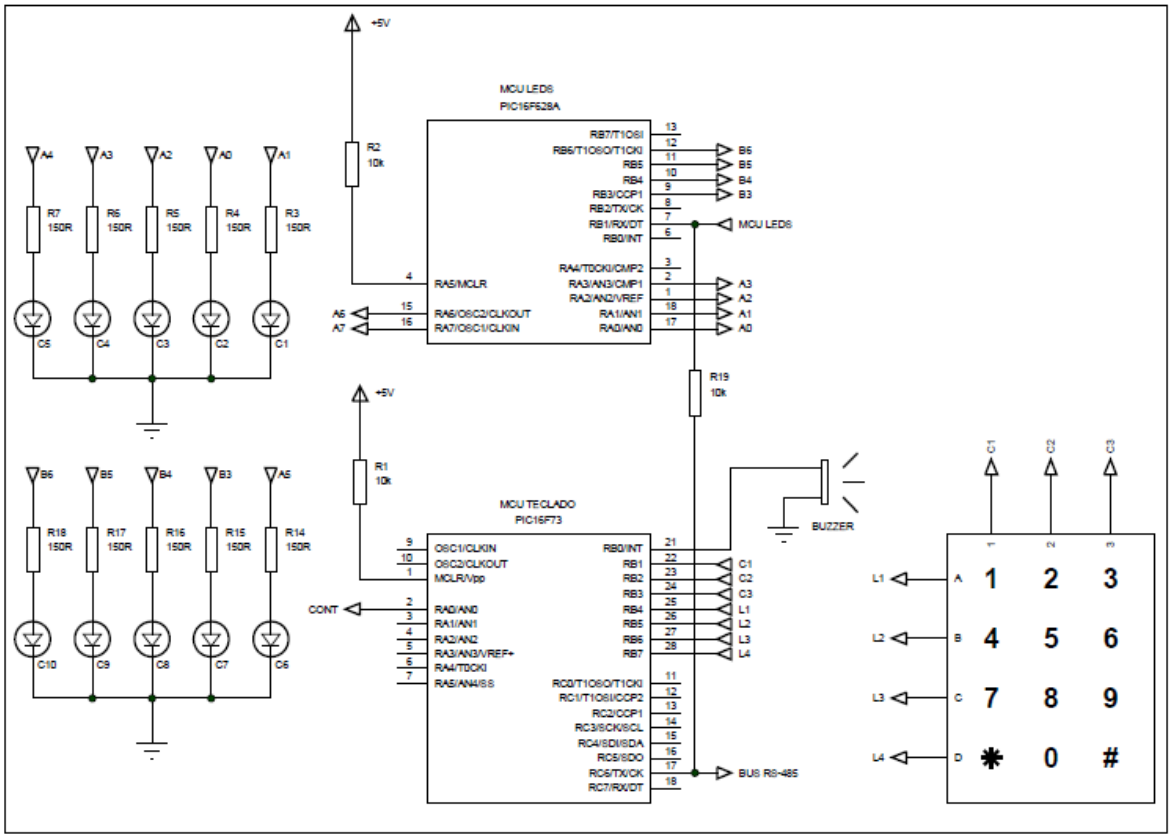


Figura 3.25 - Simulação do módulo de acionamento manual. (Autor: José Carlos)

Na Figura 3.26 é mostrado o trecho do programa da função principal para ligar e desligar o dispositivo eletroeletrônico por meio da tecla 1 e tecla 2.

```
do
{
 while(Tecla_1()) //Tecla 1.
 {
  Gera_Ton(1000,100); //Função gera som (1kHz duração=100ms), caso uma tecla seja pressionada.
  output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
  printf("\f1"); //Liga carga 1.
  delay_ms(150);
  output_low(pin_A0);  //Habilita RX e desabilita TX MAX485.

   while(Tecla_1()) //Tecla 1.
   {
    delay_ms(50);
    output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
    printf("\fa"); //Desliga carga 1.
    delay_ms(50);
    output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
   }
  }

   while(Tecla_2()) //Tecla 2.
   {
    Gera_Ton(1000,100); //Função gera som (1kHz duração=100ms), caso uma tecla seja pressionada.
    output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
    printf("\f2"); //Liga carga 2.
    delay_ms(50);
    output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
/*
    while(Tecla_2()) //Tecla 2.
```

Figura 3.26 – Trecho do programa da função principal para ligar e desligar o dispositivo eletroeletrônico. (Autor: José Carlos)

Conforme mostrado na Figura 3.26, a linha 90 com a função Tecla_1() chama outra função, Gera_Ton(1000,100), que emite o som de *bip* para a tecla 1 ao ser pressionada pelo usuário, em seguida o pino RX do conversor MAX485 é desativado e o pino TX é ativado para enviar o comando da linha 94 printf("\f1") que aciona o módulo de acionamento 1 para ligar a luz da garagem. Na linha 96 o pino RX do conversor é novamente ativado.

Para desativar o módulo de acionamento 1 e desligar a luz da garagem a mesma função Tecla_1() na linha 98 conforme mostrado na Figura 3.26, tem um retardo de 50 ms (linha 100), que significa que o usuário deve manter a tecla premida por pelo menos 50 ms para evitar falso acionamentos (*bouncing*). A luz é desligada com o envio do comando na linha 102 printf("\fa").

O mesmo princípio é válido para as outras teclas do módulo de acionamento manual, a tecla sustenido (#) é responsável por desligar todos os dispositivos eletroeletrônicos da residência, exceto para o portão de entrada social, portão de entrada veicular e a cortina do tipo persiana.

Na Figura 3.27 é mostrado o trecho do código do programa para o acionamento da tecla sustenido (#), com envio de comandos para desligar os dispositivos eletroeletrônicos.

```
275    while(Tecla_12()) //Tecla # desliga todas as cargas acionadas.
276    {
277     Gera_Ton(1000,500); //Função gera som (1kHz duração=0,5s), caso uma tecla seja pressionada.
278     output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
279     delay_ms(50);
280     printf("\fa");
281   // printf("\fb");
282   // printf("\fc");
283     printf("\fd");
284     printf("\fe");
285   // printf("\ff");
286     printf("\fg");
287     printf("\fh");
288     printf("\fi");
289     printf("\fj");
290     delay_ms(50);
291     output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
292    }
293
294  delay_ms(100);
295  } while(true);
296 }
297
298 /*****************************************************************************************************/
299
```

Figura 3.27 – Trecho do código do programa para o acionamento da tecla sustenido. (Autor: José Carlos)

Conforme mostrado na Figura 3.26 (linha 92) e na Figura 3.27 (linha 277), é utilizada a função Gera_Ton(1000,100) do compilador PIC C *Compiler* da CCS. O único

parâmetro que é alterado para ser utilizado com o PIC16F73 é a linha 39 que configura o pino RB0 para gerar o som no *buzzer*.

O parâmetro na função 1000 é o valor da frequência (*bip*), correspondendo a 1000 Hz = 1 kHz é o valor passado 100 corresponde ao tempo de duração do *bip*, que é de 100 ms, estes dois parâmetros foram ajustados pelo autor.

Na Figura 3.28 é mostrado um trecho da função Gera_Ton(1000,100) do compilador C da CCS.

```
37 //#define TONE_PIN  PIN_B0
38 //#define BUZZER  PIN_B2 //Modificado p/ buzzer no pino RB2 do PIC16F877A.
39 #define BUZZER PIN_B0 //Modificado p/ buzzer no pino RB0 do PIC.
40
```

Figura 3.28 - Trecho da função Gera_Ton do compilador C CCS. (Adaptado do PIC C *Compiler* da CCS)

Os pinos de entrada do microcontrolador são controlados pelos pinos das colunas do teclado, enquanto que os pinos de saída do microcontrolador controlam as linhas do teclado matricial. Quando é feito o cruzamento entre uma linha é uma coluna, uma tecla é ativada e o contato elétrico é estabelecido, neste procedimento a função retorna “1” e quando nenhuma tecla é pressionada retorna “0” na função.

Na Figura 3.29 é mostrada a função para as teclas sustenido (#) e asterisco (*), nas linhas 47, 48, 60 e 61 o retorno de função, indicando se uma tecla é pressionada ou não.

Para as outras teclas, o procedimento é o mesmo adotado para as teclas sustenido (#) e asterisco (*).

```
37 ************************************Função para tecla #*************************************/
38
39 int Tecla_l2() //Tecla #.
40 {
41  output_B(0bllllllll0); //RB<7:1> em 1.
42  output_low(L4); //L4 desativada conectada ao pino RB7.
43  input(C3); //C3 entrada conectado ao pino RB3.
44  delay_ms(l0); //Tempo de l0ms.
45
46    if(input(C3)) //Testa entrada de C3...
47    return(0); //Retorna 0 se RB3 = 1.
48    return(1); //Retorna 1, RB3 = 0 indicando tecla pressionada.
49 }
50
51 /************************************Função para tecla* *************************************/
52
53 int Tecla_ll() //Tecla *.
54 {
55  output_B(0bllllllll0); //RB<7:1> em 1.
56  output_low(L4); //L4 desativada conectada ao pino RB7.
57  input(Cl); //Cl entrada conectado ao pino RBl.
58  delay_ms(l0); //Tempo de l0ms.
59
60    if(input(Cl)) //Testa entrada de Cl...
61    return(0); //Retorna 0 se RBl = 1.
62    return(1); //Retorna 1, RBl = 0 indicando tecla pressionada.
63 }
64
65 /************************************Função para tecla 0*************************************/
```

Figura 3.29 - Função para as teclas sustenido e asterisco. (Autor: José Carlos)

3.4.4 - Módulo sensor de temperatura

O módulo sensor de temperatura utiliza um microcontrolador PIC12F675 em conjunto com um sensor de temperatura LM35. O módulo sensor de temperatura é responsável pela a leitura do valor de temperatura no interior da residência pelo sensor LM35. Essa leitura é enviada pelo cabeamento por meio do conversor RS-485 até ao módulo *Bluetooth*-RS485, que novamente envia ao *smartphone*.

O usuário pode interagir a partir do valor lido na tela do *smartphone* e decidir se liga ou desliga o sistema de ventilação ou climatização proporcionada por um ventilador elétrico ou um ar-condicionado.

O LM35 envia o valor de tensão para a conversão analógica para digital no pino GP0/AN0 do microcontrolador PIC12F675. O valor de temperatura é mostrado no terminal virtual que simula o conversor RS-485.

Na Figura 3.30 é mostrado o circuito para a simulação do módulo sensor de temperatura no Proteus/ISIS.

Ao enviar o caractere “t” pelo teclado, o valor de temperatura é retornado no terminal virtual.

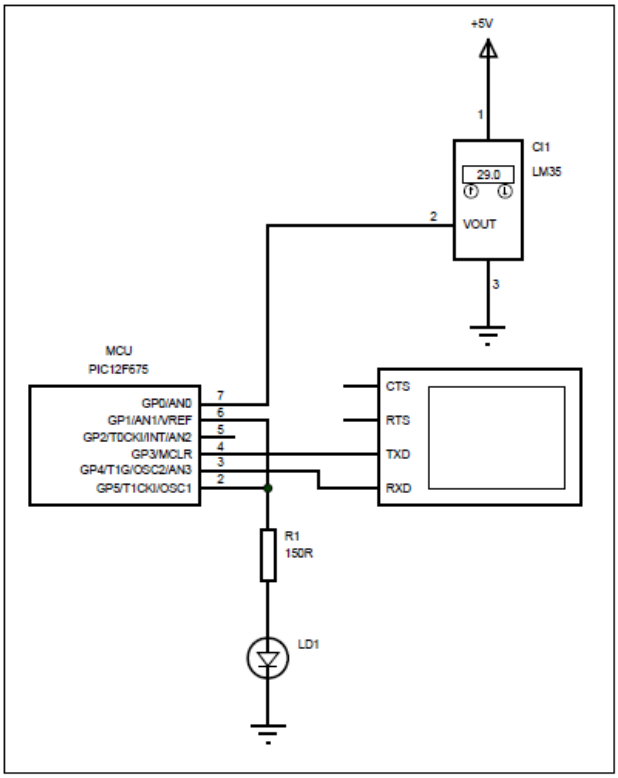


Figura 3.30 - Simulação do módulo sensor de temperatura. (Autor: José Carlos)

Na figura 3.31 é mostrado o trecho do programa para a leitura do sensor LM35 utilizando uma função (linha 46) para a conversão analógica para digital.

```
44 /************************************Função p/ leitura da temperatura*****************************/
45
46 void Termometro()
47 {
48  //printf("\f"); //Limpa o buffer do terminal serial.
49
50  //setup_adc_ports(ALL_ANALOG); //Configura o ADC p/ pinos analógicos.
51  setup_adc_ports(NO_ANALOGS|VSS_VDD); //Configura GP0/AN0 como pino de entrada analógico.
52  setup_adc(ADC_CLOCK_INTERNAL); //ADC com clock interno.
53
54  set_adc_channel(0); //Leitura da coleta de dados em GP0/AN0.
55  delay_ms(10); //Tempo necessário p/ estabilizar as configurações ADC.
56  valor = read_adc(); //Leitura do valor do ADC.
57  Celsius = valor * 100 * (5.0/1024); //Cálcula o valor da temperatura.
58  printf("%1.1f\n\r", Celsius);   //Mostra no terminal serial o valor da temperatura.
59
60  //printf("%1.1f\xDFC\n\r", Celsius);  //Mostra no terminal serial o valor da temperatura.
61  delay_ms(1000); //Tempo entre as leituras da coleta de dados.
62 }
63
```

Figura 3.31 – Função para leitura do sensor LM35. (Autor: José Carlos)

Na Figura 3.32 é mostrado o trecho do programa para o acionamento do sistema de ventilação/climatização.

```
 76  do
 77  {
 78   Tecla=getc();
 79   printf("\f");
 80   switch (Tecla)
 81   {
 82    case 't':
 83    printf("\ft");
 84    output_high(Enable); //Desabilita TX e habilita RX MAX485.
 85    delay_ms(100);
 86    Termometro();
 87    delay_ms(100);
 88    output_low(Enable); //Habilita TX e desabilita RX MAX485.
 89    break;
 90
 91    case '5':
 92    printf("\f5");
 93    output_high(Rele);
 94    delay_ms(100);
 95    output_high(Enable); //Desabilita TX e habilita RX MAX485.
 96    delay_ms(100);
 97    printf("LIGADO"); //Status do dispositivo = LIGADO.
 98    delay_ms(100);
 99    output_low(Enable); //Habilita TX e desabilita RX MAX485.
100    break;
101
102    case 'e':
103    printf("\fe");
104    output_low(Rele);
105    delay_ms(100);
106    output_high(Enable); //Desabilita TX e habilita RX MAX485.
107    delay_ms(100);
108    printf("DESLIGADO\r\n"); //Status do dispositivo = DESLIGADO.
109    delay_ms(100);
110    output_low(Enable); //Habilita TX e desabilita RX MAX485.
111    break;
112   }
113  }while(TRUE);
114 }
```

Figura 3.32 - Trecho do programa para o acionamento do sistema de ventilação/climatização. (Autor: José Carlos)

Conforme mostrado na Figura 3.32 ao enviar o comando com o caractere “5” (linha 91), o sistema de ventilação/climatização é ligado, ao enviar o caractere “e” (linha 102), o sistema de ventilação/climatização é desligado.

O módulo sensor de temperatura possui um relé auxiliar para acionar um contactor que liga ou desliga equipamentos eletroeletrônicos de maior potência, como é o caso dos sistemas de ar-condicionado.

3.5 - Desenvolvimento das Placas de Circuito Impresso no Proteus/ARES

Todas as placas de circuito impresso (módulos de acionamentos) foram desenvolvidas no ambiente Proteus/ARES com roteamento manual.

Na Figura 3.33 é mostrado o *layout* das trilhas do circuito do módulo *Bluetooth*-RS485. A placa de circuito impresso menor é a placa de *LEDs*.

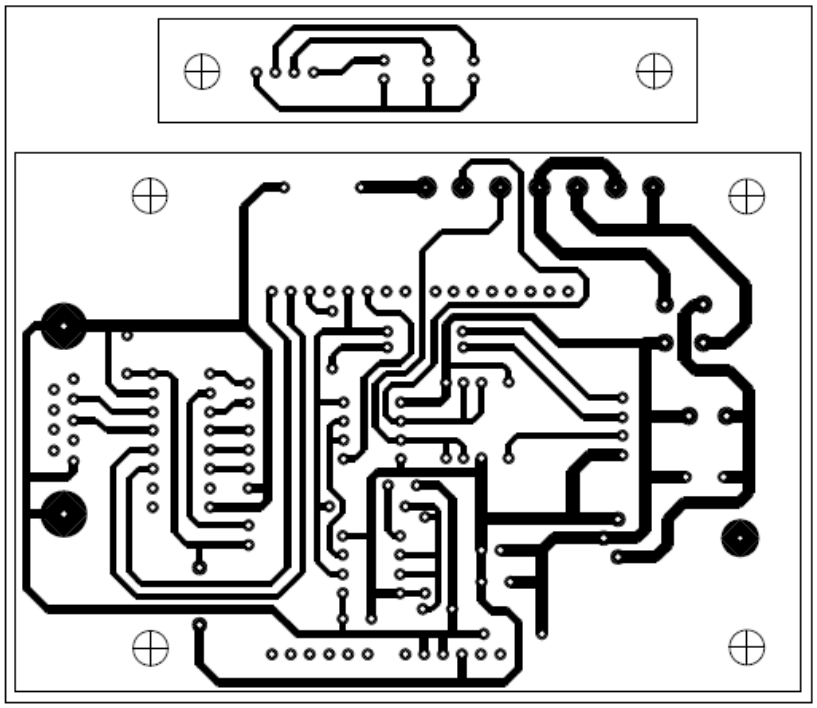


Figura 3.33 – Layout das trilhas do módulo Bluetooth-RS485. (Autor: José Carlos)

As dimensões da placa *Bluetooth*-RS485 é de 10,5 x 7,2 centímetros, a placa dos *LEDs* tem as dimensões de 7,2 x 1,4 centímetros.

O *layout* do circuito do módulo *Bluetooth*-RS485 pelo lado dos componentes é mostrado na Figura 3.34.

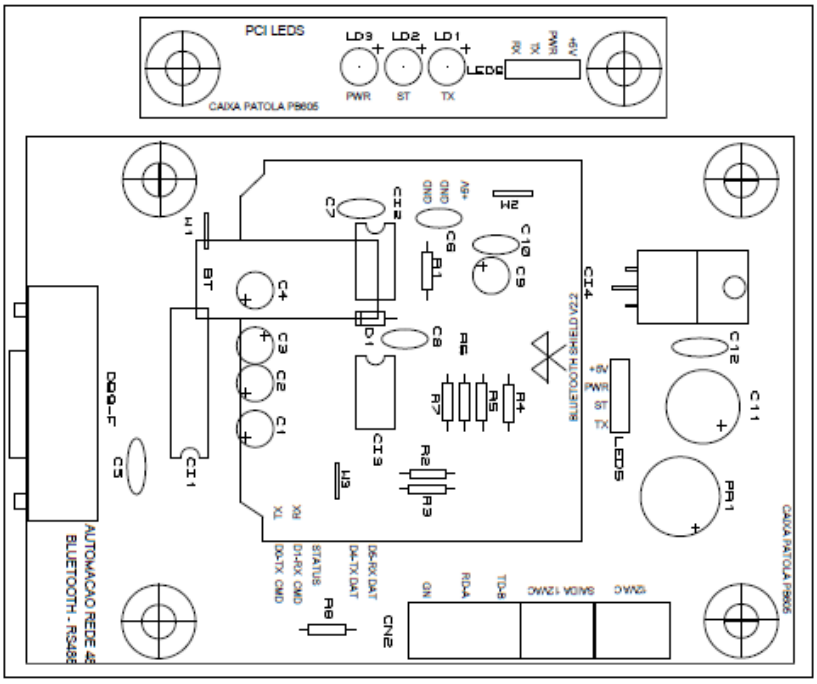


Figura 3.34 – *Layout* dos componentes do módulo *Bluetooth*-RS485. (Autor: José Carlos)

Conforme mostrado na Figura 3.34 o módulo *Bluetooth*-RS485 possui uma placa de circuito impresso com três *LEDs* de 3 mm, o *LED* vermelho indica alimentação ligada, o *LED* amarelo indica o *Status* da conexão *Bluetooth* e o *LED* verde indica a recepção do caractere ASCII enviado pelo *smartphone*.

Na Figura 3.35 é mostrado o *layout* das trilhas de circuito impresso dos módulos de acionamentos dos dispositivos eletroeletrônicos.

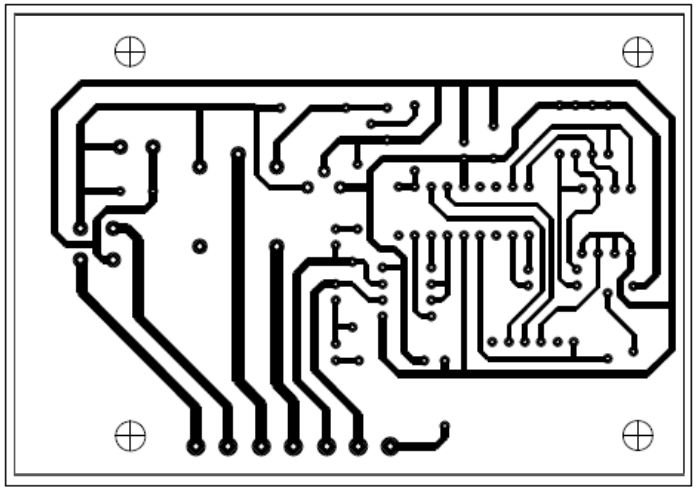


Figura 3.35 – *Layout* das trilhas do módulo de acionamento dos dispositivos eletroeletrônicos. (Autor: José Carlos)

Na Figura 3.36 é mostrado o *layout* do lado dos componentes da placa para os módulos de acionamentos dos dispositivos eletroeletrônicos.

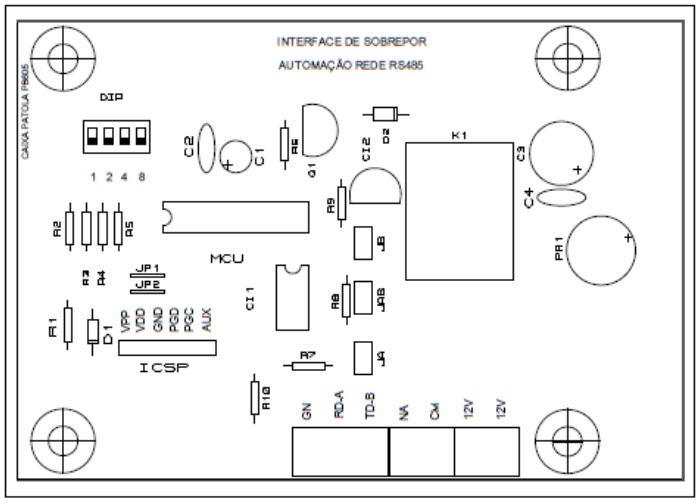


Figura 3.36 – *Layout* dos componentes do módulo de acionamento dos dispositivos eletroeletrônicos. (Autor: José Carlos)

Na Figura 3.37 é mostrado o *layout* das trilhas de circuito impresso da placa do módulo de acionamento manual.

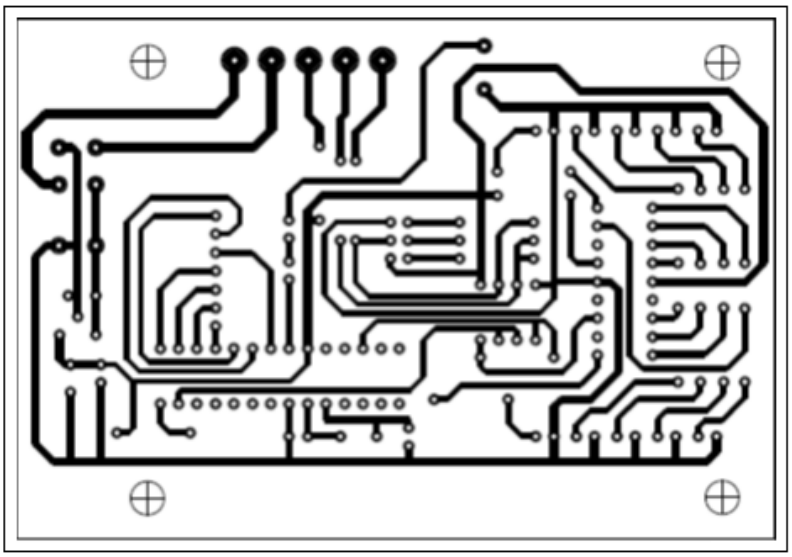


Figura 3.37 – *Layout* das trilhas do módulo de acionamento manual. (Autor: José Carlos)

Na Figura 3.38 é mostrado o *layout* pelo lado dos componentes da placa do módulo de acionamento manual.

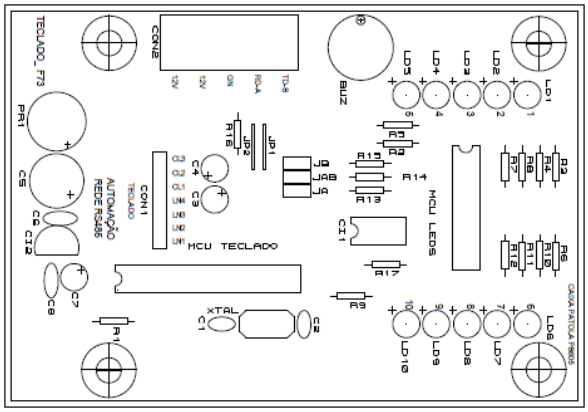


Figura 3.38 – *Layout* dos componentes do módulo de acionamento manual. (Autor: José Carlos)

Na Figura 3.39 é mostrado o *layout* das trilhas de circuito impresso da placa do módulo sensor de temperatura.

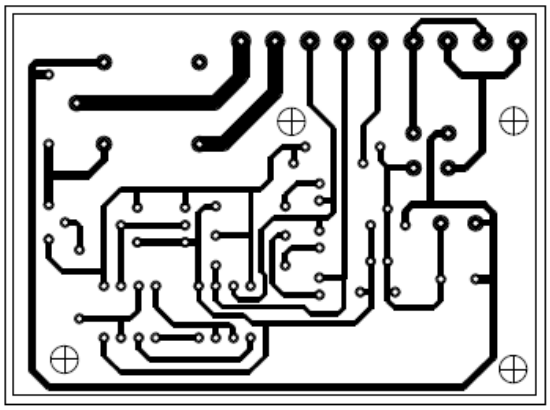


Figura 3.39 – *Layout* das trilhas do módulo sensor de temperatura. (Autor: José Carlos)

Na Figura 3.40 é mostrado o *layout* do lado dos componentes da placa do módulo sensor de temperatura.

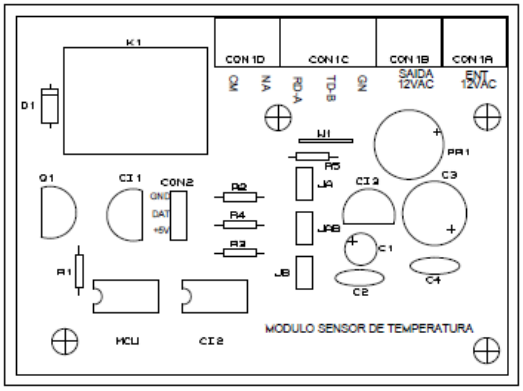


Figura 3.40 – *Layout* dos componentes do módulo sensor de temperatura. (Autor: José Carlos)

5.2 - Configuração do módulo *Bluetooth*-RS485

O módulo *Bluetooth* é configurado utilizando o cabo conversor USB RS-232. A configuração é feita com um programa terminal como o *HyperTerminal* do *Windows*. O ambiente de desenvolvimento do PIC C *Compiler* possui um programa terminal denominado *Serial Port Monitor*.

Na Figura 3.41 é mostrado o ambiente PIC C *Compiler* com o ícone do *Serial Port Monitor*.

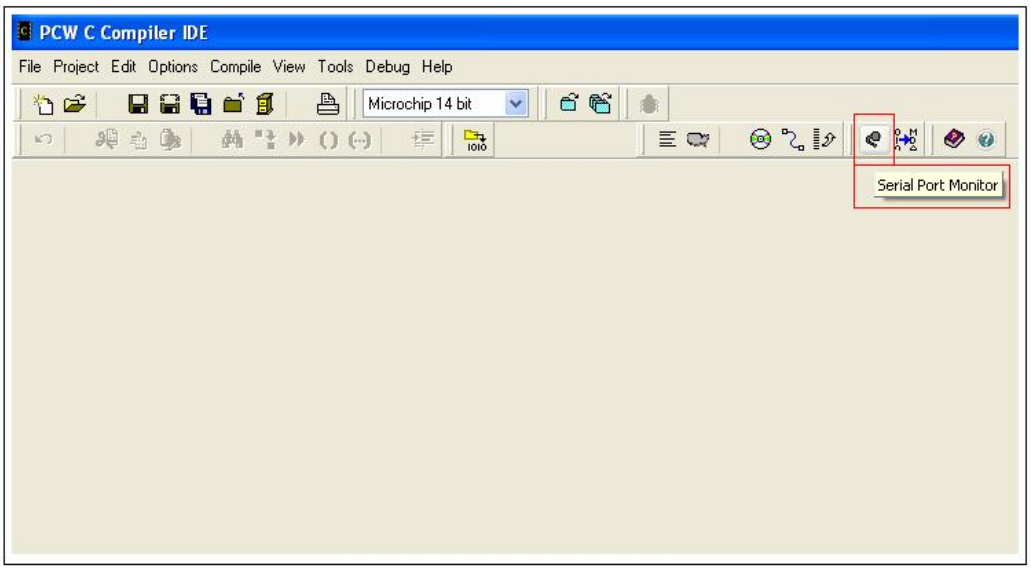


Figura 3.41 – Programa *Serial Port Monitor* no IDE PIC C *Compiler*. (Autor: José Carlos)

Clicando no ícone *Serial Port Monitor*, uma janela de opções é aberta para a configuração referente da porta de comunicação e o *Baud rates* de 38400 é selecionado para a comunicação com o módulo *Bluetooth*-RS485. Na Figura 3.42 é mostrada a janela de opções para a configuração da porta de comunicação serial.

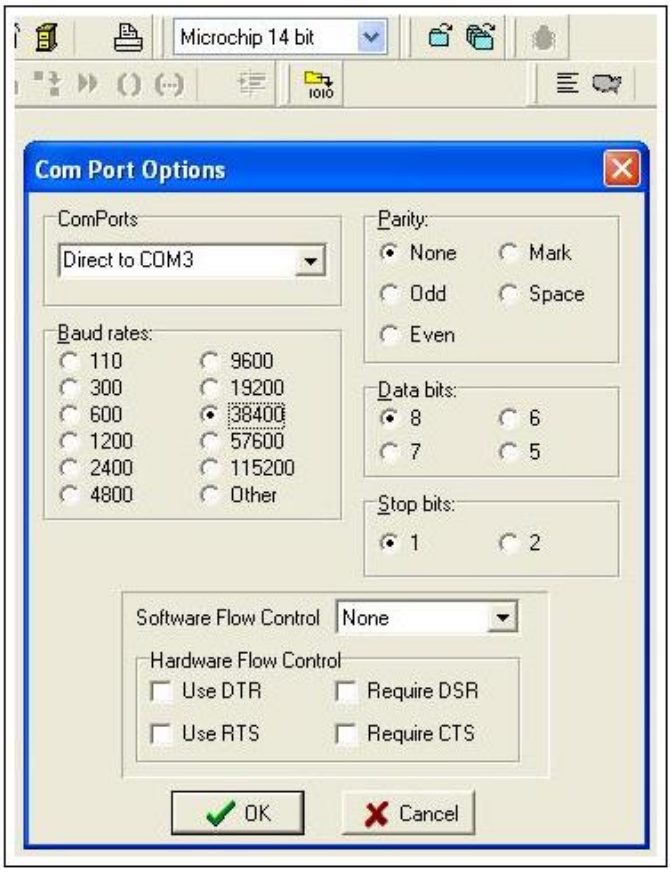


Figura 3.42 – Configuração da porta de comunicação serial. (Autor: José Carlos)

A taxa de transmissão de 38400 bps é o valor *default* para a comunicação com o programa terminal e o modulo *Bluetooth*-RS485.

Para configuração dos comandos AT, os *jumpers* TX-RX são configurados conforme mostrado na Figura 3.43.

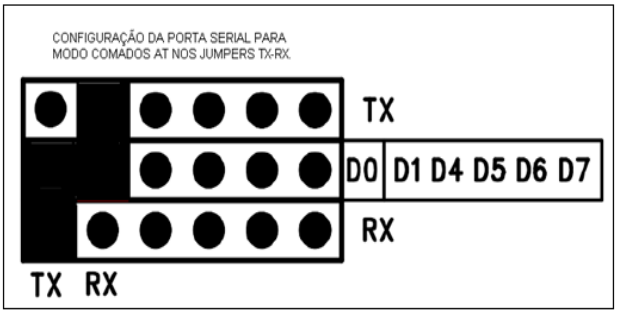


Figura 3.43 - Configuração dos *jumpers* para o modo comando AT. (Autor: José Carlos)

O interruptor de modo na configuração dos comandos AT no módulo *Bluetooth*-RS485 deve estar na posição CMD, conforme mostrado na Figura 3.44.

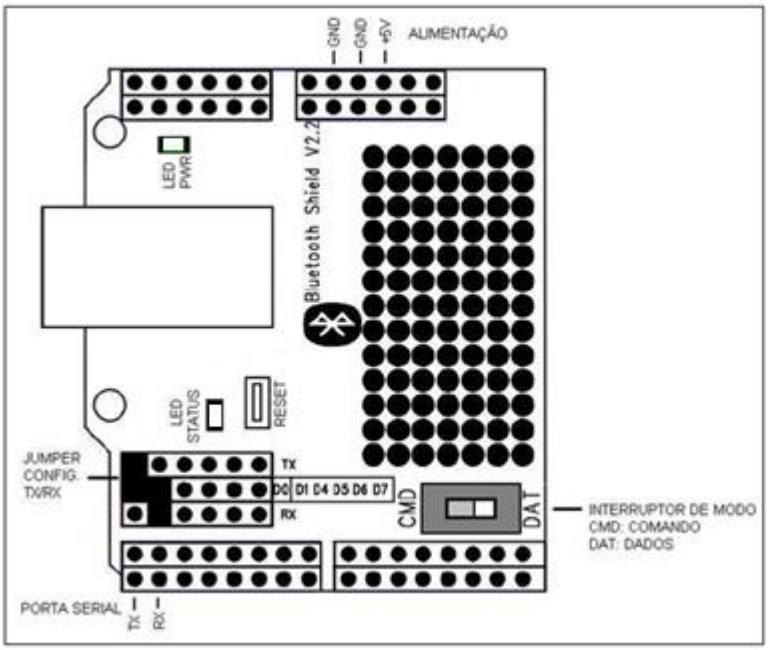


Figura 3.44 – Configuração do interruptor para modo CMD. (Autor: José Carlos)

Os comandos AT podem utilizar letras maiúsculas ou minúsculas e também terminar com \r\n.

Na configuração do módulo *Bluetooth*-RS485, utilizam-se as configurações:

- Código de pareamento:
  AT+PSWD:1234
- Parâmetros da serial: *baud rate* em 9600, 1 *stop bit*, nenhum *bit* de paridade.
  AT+UART:9600,0,0
- Modo do módulo: ROLE
  0 - *Slave* - ser ligado por outro dispositivo;
  1 - *Slave-loop* - ser conectado por outro dispositivo, receber e enviar de volta o que recebeu;
  2 - *Master* - ativamente sondar o dispositivo próximo e inicializar a ligação a outros dispositivos.
  AT+ROLE:0

- Nome do módulo: NAME
  AT+NAME:ROBOT EXPLORER-2
- Modo de conexão: CMODE
  0 – Endereço fixo na conexão;
  1 – Endereço para várias conexões;
  2 – *slave-loop* (ser conectado por outro dispositivo, receber e enviar de volta o que recebeu).
  AT+CMODE:0

No *Serial Port Monitor* é retornado o status das configurações dos comandos AT, um "*OK*" significa que a configuração foi executada com sucesso.

As ligações do módulo *Bluetooth*-RS485, módulos de acionamentos, módulo de acionamento manual e o módulo sensor de temperatura são conectados com o cabeamento UTP com comprimento de 10 metros entre cada módulo.

O interruptor de modo no módulo *Bluetooth*-RS485 depois de configurado pelos comandos AT deve estar na posição DAT.

3.7 – Gravação do *Firmware* dos microcontroladores PIC

Depois de compilar o programa dos microcontroladores dos módulos é feito a transferência do arquivo .hex para as memórias dos microcontroladores utilizando o programador PICkit2. Os módulos de acionamentos dos dispositivos eletroeletrônicos possuem um conector com seis pinos machos para a transferência do arquivo para o microcontrolador sem a necessidade de retirá-lo da placa de circuito impresso.

Na Figura 3.45 é mostrado o conector ICSP (*In Circuit Serial Programming*) nas placas dos módulos de acionamentos.

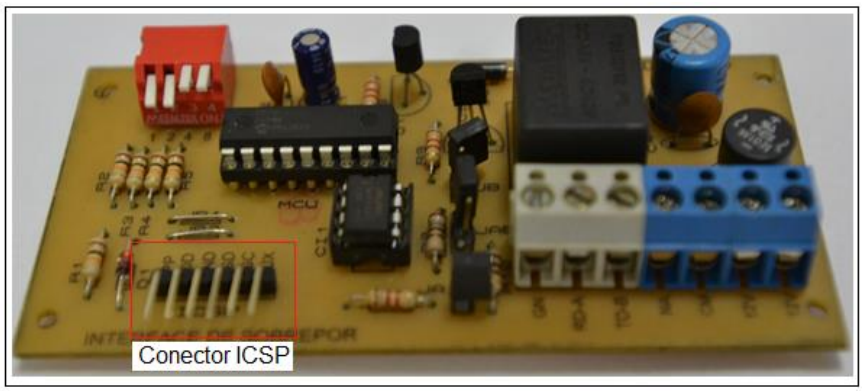


Figura 3.45 - Conector ICSP para gravação do microcontrolador. (Autor: José Carlos)

O módulo de acionamento manual e módulo sensor de temperatura não possui o conector ICSP na placa. Os microcontroladores desses módulos devem ser gravados utilizando o gravador PICkit2. Na Figura 3.46 é mostrado o gravador PICkit2 *Clone* utilizado para a gravação dos microcontroladores. Este gravado possui um soquete de 40 pinos ZIF (*Zero Insertion Force*).

Figura 3.46 - Gravador PICkit2 *Clone*. (Fonte: Robótica Simples, www.roboticasimples.com/catalog/)

3.8 – Configuração do *Dip Switch* dos Módulos de Acionamentos

A codificação dos módulos de acionamentos é feita pelo *Dip Switch* conforme mostrado na Figura 3.47.

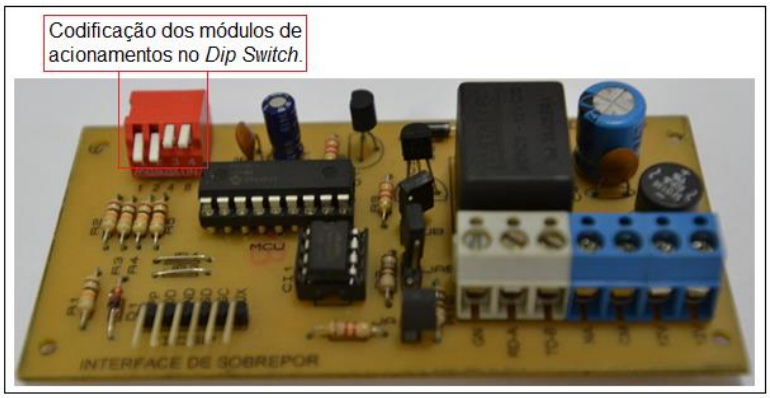


Figura 3.47 - Codificação dos módulos de acionamentos pelo *Dip Switch*. (Autor: José Carlos)

Cada módulo de acionamento foi codificado conforme mostrado no Quadro 4.

Quadro 4 – Codificação dos módulos de acionamentos. (Autor: José Carlos)

DIP SWITCH				
4	**3**	**2**	**1**	**MÓDULO**
0	0	0	1	MÓDULO DE ACIONAMENTO LUZ DA GARAGEM
0	0	1	0	MÓDULO DE ACIONAMENTO DO PORTÃO SOCIAL
0	0	1	1	MÓDULO DE ACIONAMENTO DO PORTÃO DA GARAGEM
0	1	0	0	MÓDULO DE ACIONAMENTO DA LUZ DA SALA
0	1	1	0	MÓDULO DE ACIONAMENTO DA PERSIANA DA SALA
0	1	1	1	MÓDULO DE ACIONAMENTO SOM/TV
1	0	0	0	MÓDULO DE ACIONAMENTO DA LUZ DO QUARTO

O módulo sensor de temperatura não possui *Dip Switch* para codificação, a este módulo não foi atribuído endereço e tão somente o comando com o caractere “5” para ligar o ar-condicionado, o caractere “e” para desligar o ar-condicionado e o caractere “t” para a leitura da temperatura no interior da residência.

3.9 – Configuração dos Resistores de Terminação

Os resistores de terminação proporciona o correto balanceamento de linha no conversor RS-485, conforme descrito na seção 3.3.5. Os resistores de terminação de todos os módulos que estão conectados ao cabeamento UTP devem ser configurados. As configurações são feitas por 3 *jumpers* que fazem o balanceamento da linha de transmissão/recepção no cabeamento UTP, e são designados como: “JA”, “JAB” e “JB”.

O módulo *Bluetooth*-RS485 é o módulo do início do cabeamento e os resistores de terminação já estão ligados ao circuito e não necessitam de configuração, somente o último módulo do cabeamento é configurado com o *jumper* fechado.

Na Figura 3.48 é mostrado como deve ser feito a configuração dos *jumpers* dos resistores terminadores nos módulos conectados ao cabeamento. Somente o *jumper* “JAB” do módulo que está no final do cabeamento UTP que é configurado com fechado.

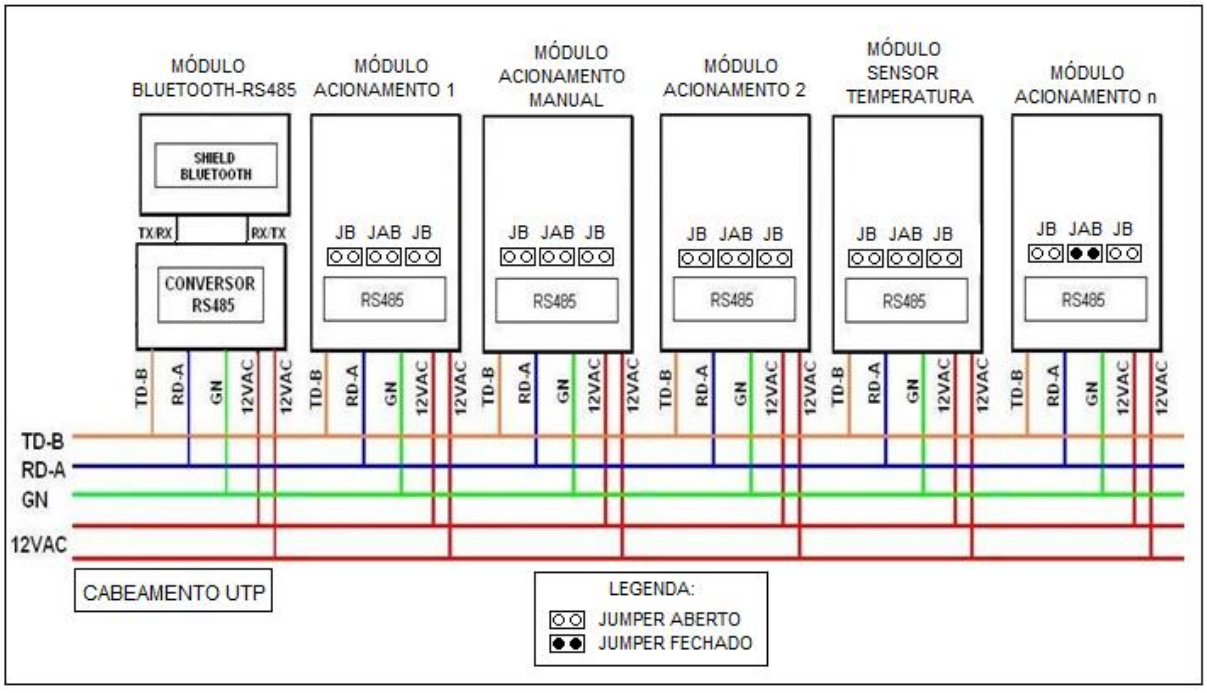


Figura 3.48 – Configuração dos *jumpers* dos resistores de terminação. (Autor: José Carlos)

CAPÍTULO 4 - DESENVOLVIMENTO DO APLICATIVO

Este capítulo mostra a descrição do desenvolvimento do aplicativo no *smartphone* para enviar comandos aos módulos de acionamentos. A plataforma de desenvolvimento é o MIT *App Inventor*.

4.1 - *App Inventor*

O *App Inventor* para *Android* é uma plataforma desenvolvida originalmente e fornecida pelo *Google*. Foi descontinuada em agosto de 2011, sendo, agora mantida pelo MIT (*Massachusetts Institute of Technology*'s), sob o nome de MIT *App Inventor*.

O ambiente de desenvolvimento do *App Inventor* permite a criação de aplicativos, mesmo que o usuário tenha noções mínimas em programação de computadores. Fornece uma *interface* gráfica que permite os desenvolvedores a utilizarem a técnica *drag and drop* (arrastar e soltar) para os objetos visuais e criar aplicativos para serem utilizados em muitos dispositivos móveis.

Os aplicativos desenvolvidos no *App Inventor* para telefones com o sistema operacional *Android* utiliza um navegador da *web*, basta ter uma conta do *Google GMAIL* para ter o acesso à plataforma. Os projetos desenvolvidos no *App Inventor* pelos desenvolvedores são armazenados em servidores *web* do *App Inventor*.

As aplicações são construídas pelo *App Inventor Designer*, onde o usuário escolhe quais os componentes serão utilizados em sua aplicação. No *App Inventor Blocks Editor*, o usuário monta os blocos de programa, onde estes especificam como os componentes irão se comportar. Os blocos são montados encaixando as peças como se fosse um brinquedo de quebra-cabeça.

Quando o aplicativo (apk) estiver concluído, o desenvolvedor empacota-o, resultando numa aplicação pronta para ser instalada no *smartphone*. Na Figura 4.1 são mostradas as duas áreas onde o desenvolvedor projeta o seu aplicativo, emula e empacota o apk para o *smartphone*.

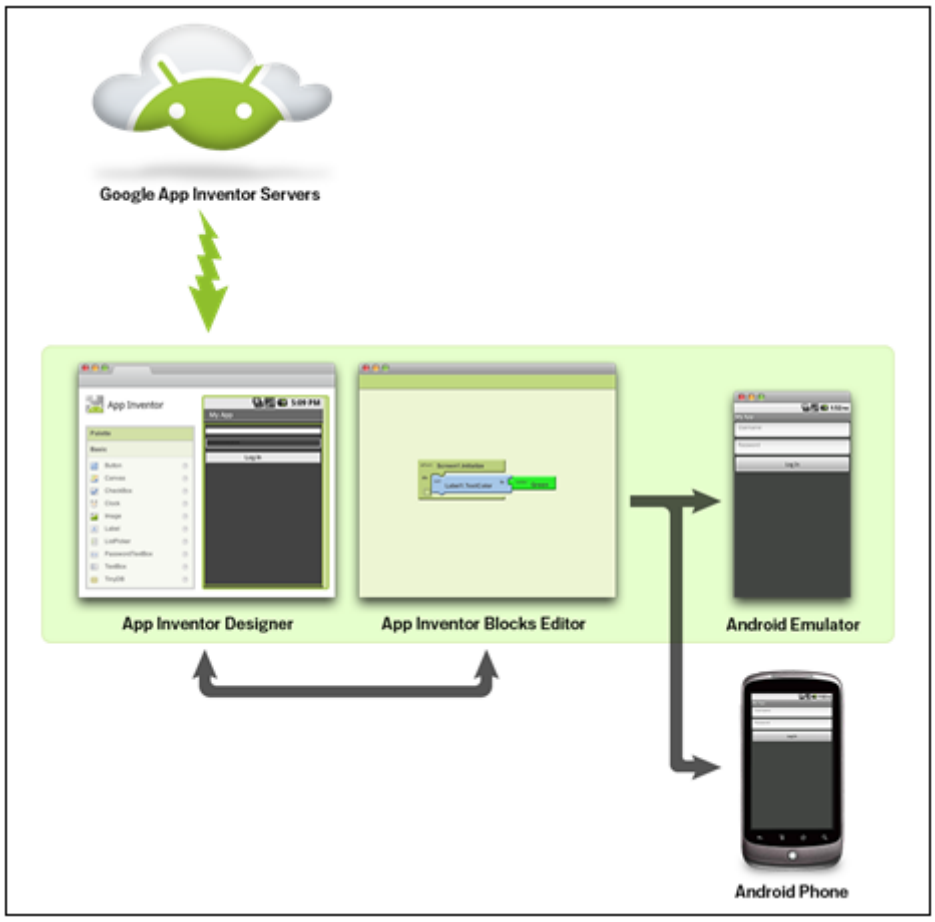


Figura 4.1 – Áreas de desenvolvimentos do *App Inventor* (Fonte: *App Inventor* MIT, http://appinventor.mit.edu/explore/content/what-app-inventor.html)

O ambiente de desenvolvimento do *App Inventor* roda em diversos sistemas operacionais como o *Mac* OSX, GNU/*Linux* e *Windows*, e uma enorme variedade de modelos de simulações para o *smartphone* contendo o Sistema Operacional *Android*.

4.2 - Aplicativo para automação residencial

A *interface* do aplicativo desenvolvido está baseada em botões de comandos associados a uma figura. Cada figura do botão de comando no aplicativo foi confeccionada no editor de desenho (*Paint*) do Sistema Operacional *Windows* 7. As figuras estão no formato *Bitmap* no padrão 24 bits.

Na Figura 4.2 são mostradas as figuras confeccionadas para o uso nos botões de comandos do aplicativo.

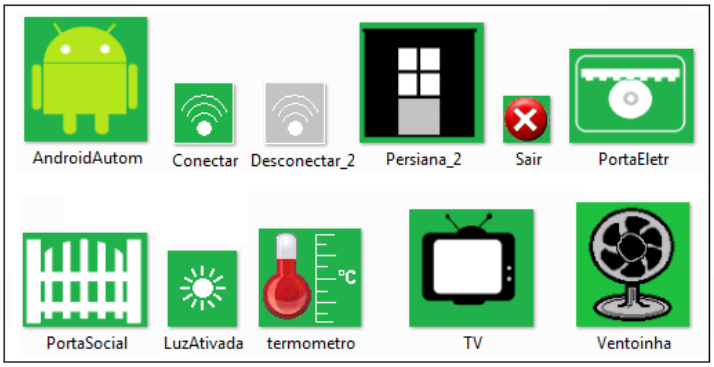


Figura 4.2 – Figuras de *Bitmap* dos botões de comandos do aplicativo. (Autor: José Carlos)

Na Figura 4.3 é mostrada a *interface* do aplicativo pronta com os botões de comandos e figuras de *Bitmap*.

Figura 4.3 – *Interface* do aplicativo com os botões de comando e figuras de *Bitmap*. (Autor: José Carlos)

Os oito botões de comandos para acionamento dos dispositivos possuem dois eventos, um clique curto ou breve no botão, em que o dispositivo eletroeletrônico é ligado e um clique longo, em que o dispositivo eletroeletrônico é desligado.

O botão Desconecta e Sair possui apenas um evento com um clique longo para desconectar o módulo *Bluetooth* e encerrar o aplicativo.

No botão de leitura de temperatura do ambiente (Temperatura) foi associado com um evento para um clique curto.

Em todos os botões é adicionado um *label*, que informa ao usuário o que cada botão realiza.

4.3 - Montando os blocos do aplicativo no Editor de Blocos

Na Figura 4.4 são mostrados os blocos montados com o Editor de Blocos com dois eventos associados para um clique curto e um clique longo em dois botões de comandos da Luz da Garagem e Luz da Sala.

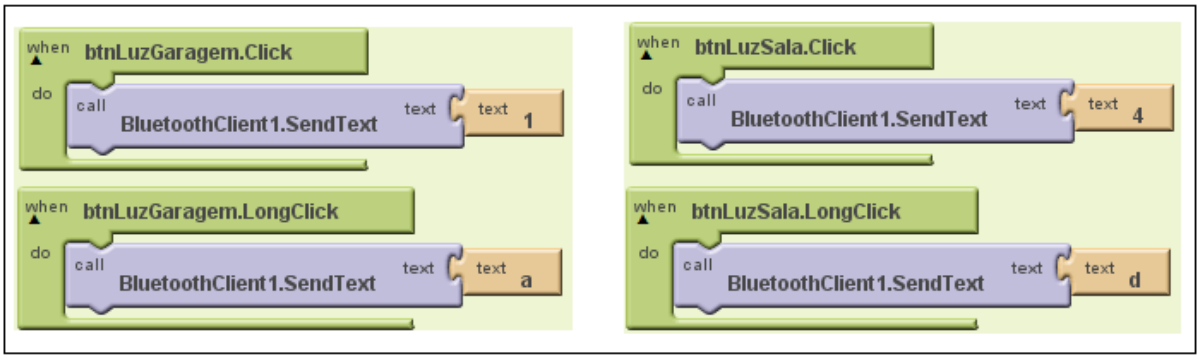


Figura 4.4 – Montagem dos blocos dos botões de comandos. (Autor: José Carlos)

Conforme mostrado na Figura 4.4 o botão btnLuzGaragem.*Click* quando acionado por um clique curto, envia a codificação do número“1” em ASCII por intermédio da chamada à função *BluetoothClient1.SendText*, fazendo com que a lâmpada da luz da garagem acenda por meio do módulo de acionamento codificado para o endereço 1 no *Dip Switch*.

Outro evento foi associado ao botão de comando da luz da garagem btnLuzGaragem.*LongClick*, quando um clique longo for estabelecido a luz é desligada enviando a codificação da letra "a" com a chamada à função *BluetoothClient1*.*SendText*.

Com um único botão de comando com dois eventos, evita-se que sejam colocados dois botões na tela, um para ligar e outro para desligar a lâmpada, este procedimento visa à funcionalidade e economia de espaços na tela do *smartphone*.

O botão de comando para a luz da sala utiliza o mesmo conceito conforme mostrado na Figura 4.4. Um clique curto no botão é enviado o código do número "4" para acender a lâmpada e um clique longo no botão é enviado a codificação para a letra "d" para desligar a lâmpada.

A montagem dos blocos dos botões: Luz do Quarto, Portão Social, Portão da Garagem, Ar Cond Sala, Cortina da Sala e Som/TV, são idênticas aos blocos dos botões de comandos da Luz da Garagem e Luz da Sala. Na Figura 4.5 são mostradas as montagens desses botões.

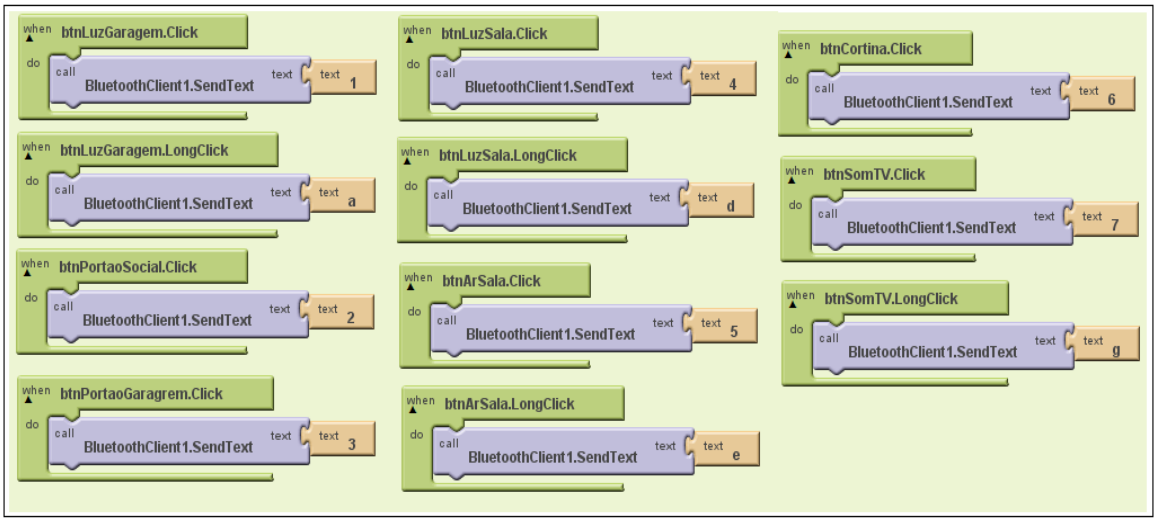


Figura 4.5 – Montagem dos blocos dos demais botões de comandos. (Autor: José Carlos)

O botão "Conecta..." está associado a uma lista de seleção contendo os dispositivos *Bluetooh* que foi pareado com o *smartphone*. Este botão ao ser clicado, o componente *ListPicker* seleciona todos os dispositivos *Bluetooth* para o usuário escolher um a partir de uma lista.

O componente *ListPicker1.BeforePicking* antes de conectar o aplicativo seleciona o evento *BluetoothClient1.AdressesAndNames* que abre uma caixa de seleção para o usuário visualizar qual dispositivo *Bluetooth* pareado utilizar. Após a seleção o evento *ListPicker1.AfterPicking* conecta o *Bluetooth* do *smartphone* com o módulo *Bluetooth*-RS485.

Na Figura 4.6 são mostrados os eventos para a conexão do *Bluetooth*, o *ListPicker1.BeforePicking* e o *ListPicker1.AfterPicking* disparados pelo botão "Conecta...".

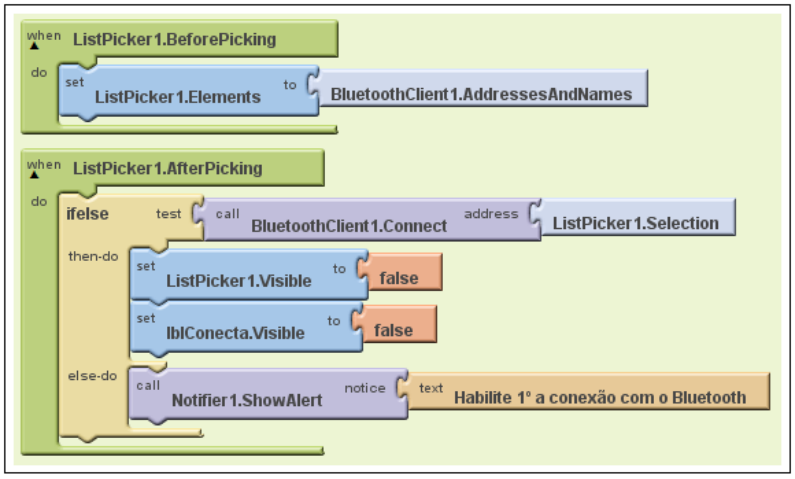


Figura 4.6 – Montagem dos blocos do botão "Conecta...". (Autor: José Carlos)

O botão Desconecta com o evento btnDesconecta.*Click* desativa as conexões entre os dois *Bluetooth*, chamando à função *BluetoothClient1.Disconnect* que deixa visível novamente a lista de dispositivos pareados no momento que o usuário efetuar a conexão.

O botão Sair com o evento btnSair.*LongClick* desconecta a conexão entre os *Bluetooth* e encerra o aplicativo.

Na Figura 4.7 são mostrados os eventos associados aos botões Desconecta e Sair com a montagem dos blocos.

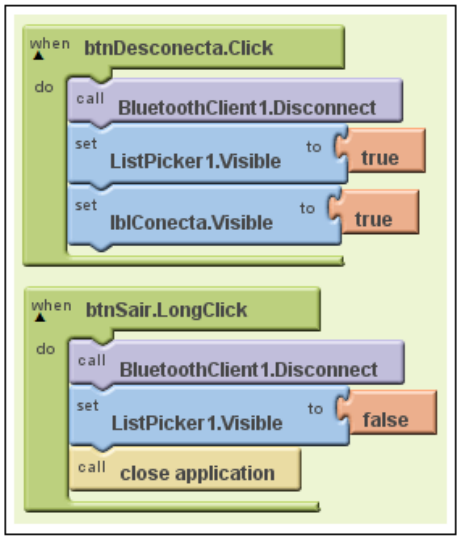


Figura 4.7 – Montagem dos blocos dos botões Desconecta e Sair. (Autor: José Carlos)

O botão temperatura ao ser clicado colhe as informações de temperatura no interior da residência. Os dois temporizadores colocado no aplicativo ativa as caixas de texto em intervalos de tempo para o recolhimento das informações (*Logger*) de temperatura.

O evento btnTemperatura.*Click* realiza uma chamada à função *BluetoothClient1*.*SendText* que configura a função CaixaTexto.*Text* com o *Clock1* em 10 ms para receber o valor da temperatura no interior da residência.

O temporizador *Clock2* mantém as informações do valor de temperatura visível na tela do *smartphone* pelo tempo de 3 segundos.

Na Figura 4.8 são mostrados os blocos responsáveis pelo *Logger* de temperatura na tela do *smartphone*.

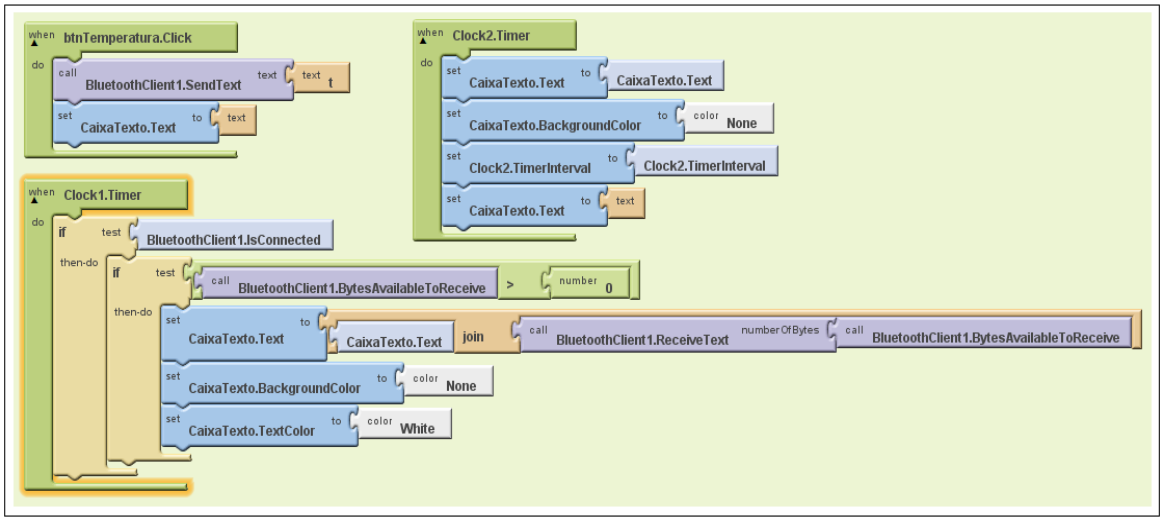


Figura 4.8 – Montagem dos blocos do *Logger* de temperatura. (Autor: José Carlos)

CAPÍTULO 5 - TESTES E RESULTADOS

Este capítulo apresenta uma descrição dos testes realizados no aplicativo de envio de comandos e no protótipo para o acionamento dos dispositivos eletroeletrônicos descrito nos capítulos 3 e 4.

5.2 – Testes do Aplicativo no *Smartphone*

A instalação do aplicativo foi feito em um *smartphone* marca *Samsung* modelo *Galaxy*-Y *Young* GT-S5360B com versão do sistema operacional *Android* 2.3.5 e a versão *kernel* 2.6.35.7 se.infra@SEP-55#1.

Com o aplicativo de automação residencial instalado no *smartphone* foi feito testes utizando em primeiro momento somente o módulo *Bluetooth*-RS485. Neste teste verifica-se o pareamento entre o módulo *Bluetooth* do *smartphone* e do módulo *Bluetooth*-RS485. O pareamento do módulo *Bluetooth* com o *smartphone* é feito normalmente como em qualquer outro dispositivo para pareamento.

Após a etapa de pareamento, foi feito o acesso abrindo o aplicativo no *smartphone*. Na Figura 5.1 é mostrada a tela do *smartphone* com o aplicativo Automação Residencial aberto.

Figura 5.1 - Tela do *smartphone* com o aplicativo aberto. (Autor: José Carlos)

Clicando no botão “Conecta...” uma caixa de seleção de dispositivos *Bluetooth* é aberta e a escolha é no dispositivo *ROBOT EXPLORER-2*. Na Figura 5.2 é mostrada a caixa de seleção dos dispositivos *Bluetooth* pareados com o *smartphone*.

Figura 5.2 - Caixa de seleção dos dispositivos *Bluetooth* pareados. (Autor: José Carlos)

Quando o módulo *Bluetooth*-RS485 e o *smartphone* são conectados, o botão “Conecta...” fica invisível e o usuário tem acesso aos botões de comandos para o acionamento dos dispositivos eletroeletrônicos.

Nesta etapa, os testes realizados são nos botões “Conectar...”, “Desconectar” e “Sair” do aplicativo do *smartphone*. Foram efetuadas dez baterias de testes para verificar se o aplicativo não apresenta *bugs*, como travar ou encerrar a aplicação de maneira inesperada.

5.2 - Testes nos Módulos de Acionamentos

Foram efetuadas as ligações através do cabeamento UTP do módulo *Bluetooth*-RS485 aos demais módulos (módulos de acionamentos, módulo de acionamento manual e módulo sensor de temperatura). Na Figura 5.3 são mostradas as ligações dos módulos.

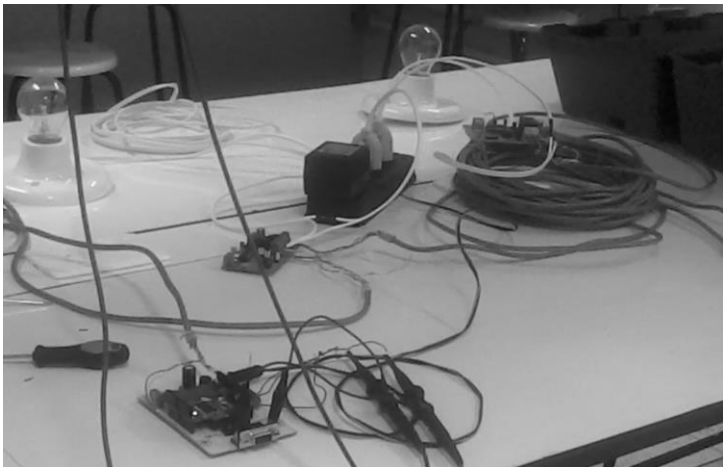


Figura 5.3 – Ligações dos módulos ao cabeamento UTP. (Autor: José Carlos)

Nesta etapa, foram efetuados dez acionamentos nos botões de comandos do *smartphone* para ligar e desligar os dispositivos eletroeletrônicos, acionando a luz da garagem, luz da sala e acionamento do sistema de climatização/ventilação. Os testes realizados nesta etapa foram feitos utilizando lâmpadas eletrônicas de 15 W.

No teste do fecho elétrico foi utilizada também uma lâmpada eletrônica de 15 W na simulação, ao pressionar o botão de comando no *smartphone* para abrir o portão de entrada social, a lâmpada é ligada e desligada no mesmo instante. Dez acionamentos no botão de comando foram efetuados neste teste. Na Figura 5.4 é mostrado o acionamento das luzes da sala, garagem e a simulação com a lâmpada para o fecho elétrico e para o sistema de climatização/ventilação.

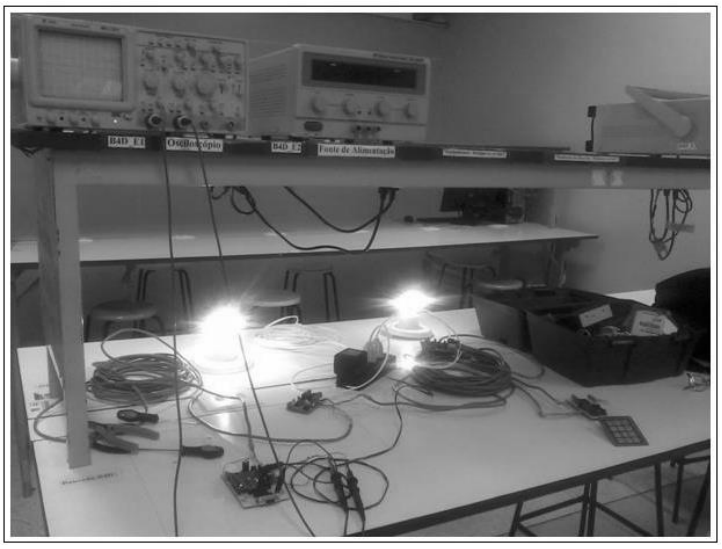


Figura 5.4 – Teste de acionamento dos módulos. (Autor: José Carlos)

Para o teste da leitura de temperatura do módulo sensor de temperatura, foi utilizado o ar-condicionado do laboratório 5000.

Pressionando o botão de leitura no *smartphone*, aparece o valor de temperatura na parte inferior da tela do *smartphone*, na visualização aparece o valor inteiro com duas casas decimais uma casa para a parte fracionária separada por um ponto decimal.

Na Figura 5.5 é mostrada a área da tela do *smartphone* utilizada para a leitura da temperatura.

Figura 5.5 - Área da tela do *smartphone* utilizada para a leitura da temperatura. (Autor: José Carlos)

O teste do módulo de acionamento manual foi efetuado pressionando as teclas no teclado matricial. A tecla com número "1" aciona a luz da garagem, a tecla "2" o portão de entrada social, a tecla "3" o portão da garagem, a tecla "4" a luz da sala, a tecla "5" o acionamento do sistema de climatização/ventilação.

Pressionando com um breve toque nas teclas descritas acima, o acionamento do dispositivo eletroeletrônico é efetuado e o som de tecla acionada é percebido por um som audível no *buzzer*.

A tecla "2" e a tecla "3" faz o acionamento momentâneo do fecho elétrico do portão de entrada social e do portão da garagem.

Para desligar os dispositivos eletroeletrônicos, o usuário pressiona a tecla correspondente com um toque longo, o som audível no *buzzer* é notado novamente.

A tecla sustenido (#) é utilizada para o acionamento de todos os dispositivos eletroeletrônicos da residência, exceto o acionamento do fecho elétrico do portão de entrada social e do portão da garagem.

A tecla asterisco (*) desliga todos os de todos os dispositivos eletroeletrônicos da residência, exceto fecho elétrico do portão de entrada social e do portão da garagem.

Nesta etapa foram efetuados vários pressionamentos no teclado do módulo de acionamento manual. Foram realizados vinte acionamentos com toque breve e longo no teclado para ligar e desligar os dispositivos eletroeletrônicos.

Na Figura 5.6 é mostrada as teclas do teclado matricial 4 x 3 do módulo de acionamento manual.

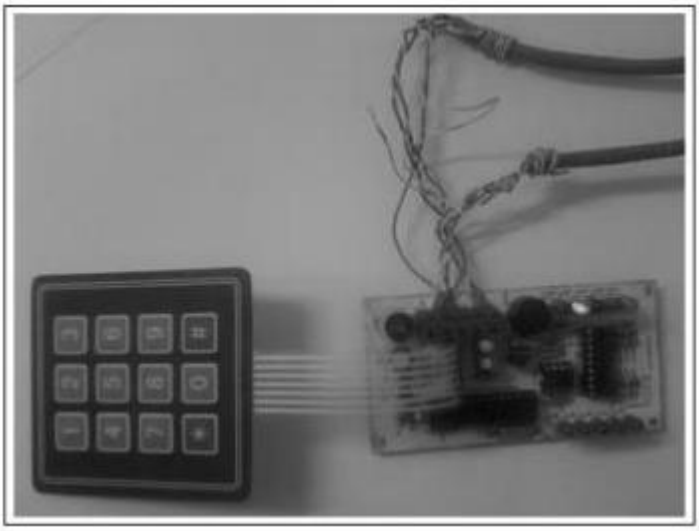


Figura 5.6 – Teclas do módulo de acionamento manual. (Autor: José Carlos)

5.3 - Protótipo Obtido

O desenvolvimento deste trabalho proporcionou um protótipo de automação residencial utilizando *smartphone Android*, *Bluetooth* e conversor RS-485 com acionamento de dispositivos eletroeletrônicos por meio de sete módulos conectados a um cabeamento UTP.

Vários componentes eletroeletrônicos foram utilizados neste projeto. No Quadro 5 é mostrado os materiais utilizados para a montagem do protótipo e seus custos de obtenção no comércio local de componentes eletroeletrônicos.

Quadro 5 - Descrição dos componentes e custo do projeto. (Autor: José Carlos)

DESCRIÇÃO	QUANT.	VALOR	TOTAL (R$)
BARRA DE PINOS FÊMEA 1 X 6 X 11,2 – 180 GRAUS	02	0,25	0,50
BARRA DE PINOS FÊMEA 1 X 8 X 11,2 – 180 GRAUS	02	0,31	0,62
BARRA DE PINOS MACHO 1 X 40 X 11,2 – 90 GRAUS	02	0,55	1,10
BARRA DE PINOS MACHO 1 X 40 X 11,2 – 180 GRAUS	02	0,55	1,10
BORNE KRE – 2 TERMINAIS PASSO 5mm – AZUL	12	0,50	6,00
BORNE KRE- 3 TERMINAIS PASSO 5mm – AZUL	6	0,60	3,60
CABO UTP CAT 5	30 m	1,50	45,00
CAIXA PLÁSTICA PATOLA PB 605	06	8,00	48,00
CAPACITOR CERAMICO 22 pF/50V	02	0,04	0,08
CAPACITOR CERAMICO 100 nF/50V	13	0,05	0,65
CAPACITOR ELETROLITICO 10 uF/50V	10	0,05	0,50
CAPACITOR ELETROLITICO 1000 uF/25V	06	0,40	2,40
CIRCUITO INTEGRADO MAX232	01	1,65	1,65
CIRCUITO INTEGRADO MAX485	06	3,85	23,10
CIRCUITO INTEGRADO 555	01	0,50	0,50
CHAVE *DIP SWITCH* 4 VIAS	04	1,10	4,40
CONTACTOR WEG TRIPOLAR CW 07 BOBINA 220 V	01	40,00	40,00
CONECTOR DB9 FEMEA 90 GRAUS (PCB)	01	1,65	1,65
CONECTOR *JACK* RJ45 YH55-05 8P8C 90 GRAUS	15	0,79	11,88
CONECTOR JUMPER COM ALÇA	20	0,07	1,40
CONECTOR *PLUG* P/ RJ45 8P8C CRISTAL	20	0,29	5,80
CRISTAL QUARTZO 10 MHz	01	0,77	0,77
FECHO ELÉTRICO FEC91 HDL COM ESPELHO FIXO	01	50,90	50,90

FIO FLEXÍVEL PARALELO 1,5mm	5 m	0,80	4,50
LÂMPADA ELETRÔNICA 15 WATTS 220 VOLTS	02	6,90	13,80
LED DIFUSO AMARELO 3mm	01	0,11	0,11
LED DIFUSO VERDE 3mm	01	0,11	0,11
LED DIFUSO VERMELHO 3mm	01	0,12	0,12
LED DIFUSO VERMELHO 5mm	10	0,14	1,40
MICROCONTROLADOR PIC12F675	01	3,72	3,72
MICROCONTROLADOR PIC16F628A	06	6,19	37,14
MICROCONTROLADOR PIC16F73	01	5,60	5,60
PAFLON PARA LUMINÁRIA COM SOQUETE	02	2,90	5,80
PLACA FENOLITE VIRGEM FACE SIMPLES 20 x 20 cm	04	8,45	33,80
PLUG FÊMEA 2 PINOS 220 VOLTS 10A	04	1,10	4,40
RESISTOR DE FILME DE CARBONO 180R - 1/4W	06	0,04	0,24
RESISTOR DE FILME DE CARBONO 470R - 1/4W	02		
RESISTOR DE FILME DE CARBONO 390R - 1/4W	10	0,04	0,40
RESISTOR DE FILME DE CARBONO 1k2 - 1/4W	12	0,04	0,48
RESISTOR DE FILME DE CARBONO 2k2 - 1/4W	06	0,04	0,24
RESISTOR DE FILME DE CARBONO 4k7 - 1/4W	01		
RESISTOR DE FILME DE CARBONO 10k - 1/4W	30	0,04	1,20
REGULADOR DE TENSÃO LM7805	01	0,85	0,58
REGULADOR DE TENSÃO LM78L05	05	1,76	8,80
SENSOR DE TEMPERATURA LM35DZ	01	3,50	3,50
SOQUETE CI 08 PINOS ESTAMPADO ESTREITO	07	0,12	8,40
SOQUETE CI 16 PINOS ESTAMPADO ESTREITO	01	0,16	0,16
SOQUETE CI 18 PINOS (ESTAMPADO ESTREITO)	05	0,18	0,90
SOQUETE CI 28 PINOS (ESTAMPADO ESTREITO)	01	0,31	0,31
TECLADO MATRICIAL DE MEMBRANA 4 X 3	01	6,00	6,00
TOMADA MACHO 2 PINOS 220 VOLTS 10A	02	1,10	2,20
TRANSFORMADOR 6 + 6 VOLTS 1A	02	17,90	35,80

TRANSISTOR BC337 (NPN)	05	0,11	0,55
RELÉ 12V - 30mA 1 CONTATO REVERSÍVEL 7A 250V (A1RC2 – METALTEX)	05	1,89	9,45
SHIELD COM MÓDULO *BLUETOOTH* V2.2 *MASTER/SLAVE*	01	66,00	66,00
SMARTPHONE GALAXY-Y YOUNG GT-S5360B	01	329,00	329,00
CUSTO TOTAL DO PROJETO			**R$ 836,43**

Conforme mostrado no Quadro 5 os componentes e o custo do projeto é para a montagem de sete módulos (01 módulo *Bluetooth*-RS485, 04 módulos de acionamentos, 01 módulo de acionamento manual e 01 módulo sensor de temperatura).

CAPÍTULO 6 - CONCLUSÕES

6.1 - Conclusões

A automação residencial facilita o dia-a-dia das pessoas, proporcionando uma qualidade de vida melhor, os equipamentos eletroeletrônicos passam a interagir de forma automática com seus usuários, visando à segurança, o conforto, a praticidade e economia de recursos hídricos e elétricos.

Neste trabalho foi desenvolvida uma solução para o acionamento de dispositivos eletroeletrônicos em uma residência, mediante a utilização de um módulo *Bluetooth* externo conectado com o *Bluetooth* de um *smartphone* com o *Android*, gerenciado por um aplicativo desenvolvido na plataforma MIT *App Inventor* e foram atingidos todos os objetivos dentro do que foi proposto.

As seguintes dificuldades encontradas foram:

- Na aquisição da *Shield* do módulo *Bluetooth* v2.2 mestre/escravo, foi necessário efetuar a compra desse módulo quatro vezes para que um módulo funcionasse com o *smartphone* descrito na seção 5.2;
- O tamanho da tela do *smartphone*, por ser pequena compromete a quantidade de botões de comandos para o acionamento dos dispositivos eletroeletrônicos;
- O tamanho do cabeamento UTP envolvido nas ligações dos módulos pode aumentar consideravelmente com o tamanho da residência.
- À medida que se expande a quantidade de módulos, a quantidade de nós e conexões no cabeamento UTP aumentam. Para tornar mais rápida as conexões dos módulos, foi projetada outra placa com conectores RJ45 e a crimpagem do cabeamento através de conectores. Em vez de utilizar os bornes KRE para a conexão do cabo, foram utilizados conectores RJ45. Na Figura 6.1 é mostrada a PCI para o conector RJ 45 do módulo *Bluetooth*-RS485 (PCI RJ45 com *LEDs*) e demais módulos.

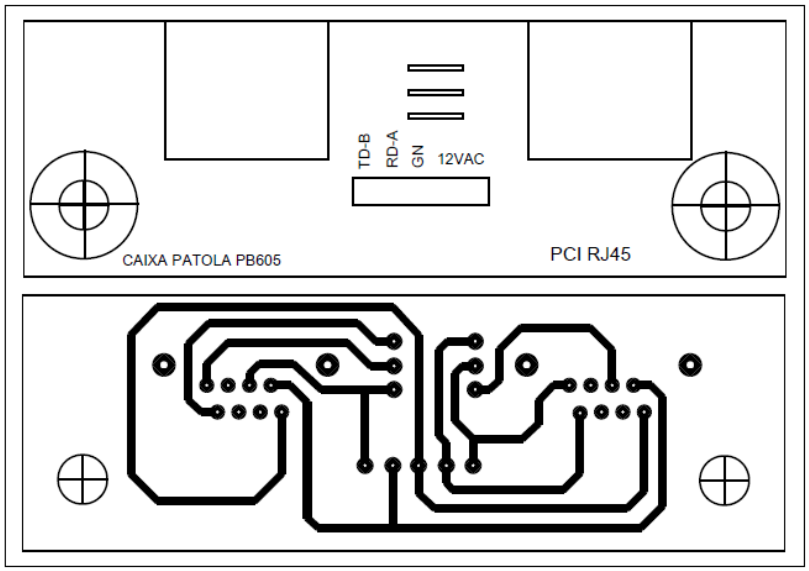


Figura 6.1 - PCI RJ45 para conexão dos módulos. (Autor: José Carlos)

6.2 - Proposta para Futuros Projetos

O trabalho proposto pode ser agregado a outros dispositivos eletroeletrônicos de automação residencial que utilize um *smartphone*, *tablet* ou a *web* como forma de envio de sinais.

Pode ser modificado para um sistema de controle central com apenas um PIC16F877 que possui maior quantidade de pinos para o acionamento de vários dispositivos eletroeletrônicos diminuindo a quantidade de nós no cabeamento UTP.

Outra proposta é a inclusão de um módulo para o reconhecimento de comandos pela voz, como a *Shield EasyVR* para o acionamento dos dispositivos eletroeletrônicos com comandos em Inglês, Italiano, Japonês, Alemão, Espanhol e Francês já instalados em biblioteca própria.

REFERÊNCIAS

Axelson, Jan, Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems, 2ª ed. Madison: Lakeview Research LLC, 2007.

Forouzan, Behrouz A, Comunicação de Dados e Redes de Computadores, 4ª ed. São Paulo: McGrawHill, 2008.

Passos, Bruno Pereira. Sistema *Bluetooth* para controle de acessórios veiculares utilizando smartphone com Android. Monografia de conclusão do curso de engenharia de computação, UniCEUB. Brasília, 2011.

Pereira, Fabio. Microcontroladores PIC: Programação em C, 7ª ed. São Paulo: Érica, 2007.

Tanenbaum, Andrew S, Redes de Computadores, 4ª ed. São Paulo: Pearson, 2003.

Tanenbaum, Andrew S, Redes de Computadores, 5ª ed. São Paulo: Pearson, 2011.

Torres, Gabriel, Hardware Curso Completo, 4ª ed. Rio de Janeiro: Axcel Books do Brasil Editora, 2001.

Oliveira, André Schneider de, Sistemas Embarcados: Hardware e o Firmware na Prática, 1ªed. São Paulo: Editora Érica, 2006.

www.aureside.org.br, ASSOCIAÇÃO BRASILEIRA DE AUTOMAÇÃO RESIDENCIAL. Publicações. Artigos, acessado em 06 de agosto de 2013.

Deal Extreme. Disponível em: www.dx.com/pt/p/3x4-matrix-12-key-membrane-switch-keypad-keyboard-117718, acessado em 06 de agosto de 2013.

Itead Studio. Disponível em**:** imall.iteadstudio.com/IM120417010_BT_Shield_v2.2/DS_BluetoothHC05.pdf, acessado em 10 de agosto de 2013.

Leroy Merlim. Disponível em: www.leroymerlin.com.br, acessado em 05 de outubro de 2013.

Loja do Circuito elétrico. Disponível em: www.circuitoeletrico.com.br**,** acessado em 05 de outubro de 2013.

Microchip Documentation and Application Notes. Disponível em:

http://ww1.microchip.com/downloads/en/appnotes/00236a.pdf, acessado em 06 de agosto de 2013.

Microchip Technology Inc. PIC16F7XData Sheet28/40-pin, 8-bit CMOS FLASH Microcontrollers. Disponível em:

http://ww1.microchip.com/downloads/en/DeviceDoc/30325b.pdf, acessado em 06 de agosto de 2013.

Microchip Technology Inc. Flash Microcontrollers PIC16F627A/628A/648A Data Sheet Flash-Based, 8-Bit CMOS Microcontrollers with nanoWatt Technology. Disponível em:

http://ww1.microchip.com/downloads/en/DeviceDoc/40044G.pdf, acessado em 06 de agosto de 2013.

Microchip Technology Inc. PIC12F629/675 Data Sheet 8-Pin FLASH-Based 8-Bit CMOS Microcontrollers. Disponível em:

http://ww1.microchip.com/downloads/en/DeviceDoc/41190G.pdf, acessado em 06 de agosto de 2013.

How Stuff Works. Disponível em: www.informatica.hsw.uol.com.br/portas-seriais2.htm, acessado em 18 de agosto de 2013.

Hu Infinito. Disponível em: www.huinfinito.com.br, acessado em 18 de agosto de 2013.

Maxim Integrated. Disponível em:

www.datasheets.maximintegrated.com/en/ds/MAX220-MAX249.pdf, acessado em 12 de setembro de 2013.

Analog Devices. Disponível em: www.analog.com, acessado em 12 de setembro de 2013.

Maxim Integrated. Disponível em: www.maximintegrated.com, acessado em 12 de setembro de 2013.

Texas Instruments. Disponível em: www.ti.com, acessado em 12 de setembro de 2013.

Texas Instruments. Disponível em: http://www.ti.com/lit/ds/symlink/lm35.pdf, acessado em 23 de outubro de 2013.

Robótica Simples. Disponível em: www.roboticasimples.com/catalog/, acessado em 28 de novembro de 2013.

APÊNDICE A - CÓDIGO FONTE DO MÓDULO DE ACIONAMENTO

/**********RS485 RXF628**********

Nome do Arquivo: rs485_rxf628.c

Versão: 1.0

Descrição: Automação residencial pelo cabeamento UTP com conversor RS485. Os módulos de acionamentos são controlados pelo Bluetooth do smartphone Android. É utilizado um Dip Switch com 4 chaves para codificação dos módulos com valores em binário de 0000 a 1111 (16 placas). Para tornar a automação mais inteligente os módulos enviam o status pelo cabeamento para a interface de usuário, informando se o dispositivo está ligado ou desligado. O cabeamento é composto por 5 fios: 1 fio para transmissão, 1 fio para recepção dos, 1 fio comum (GND) e 2 fios para alimentação 12Vac.

OBS: O acionamento fecho eletromagnético do portão de entrada social, acionamento do motor do portão de entrada de veículos e acionamento da cortina da janela da sala, o pulso de acionamento dos módulos devem ser momentâneos.

Módulo de acionamento 2: Fechadura magnética do portão social.

Módulo de acionamento 3: Portão de entrada de veículos.

Módulo de acionamento 6: Cortina da janela da sala.

Autor: José Carlos da S. Santa Cruz.

Compilador: PIC COMPILER Versão 3.43.

Ambiente de simulação: Proteus 7.7 SP2 toolsuíte ISIS Professional.

Microcontrolador utilizado: PIC16F628A.

Data: agosto de 2011.

UpDate: agosto de 2013.

**********Definições do microcontrolador**********/

```
#include <16f628A.h> //Utiliza a biblioteca do PIC16F628A.
#fuses INTRC_IO,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT
#use delay(clock=4000000) //Oscilador interno com 4MHz.
#include <Envia_Comando.h> //Função p/ envio dos caracteres pela serial do PIC.
```

/**********Atribuição de valores das variáveis na RAM**********/

```
#byte port_A = 5        //Configura as variáveis p/ mapeamento da memória.
#byte port_B = 6

/**********Rotina de Interrupção da recepção serial**********/
void rda_isr()  //Chama interrupção RS232.
{
 Tecla_Pressionada=0x00; //Tecla acionada envia 1 decimal pela serial.
 if(kbhit()) //Função p/ verificar a presença de caracteres recebido na porta serial.
```

```
 {
  Tecla_Pressionada=getc(); //Aguarda a chegada dos caracteres pela serial e retorna seu valor.
 }
}
/*********************************Função principal*************************/
void main()
{
 //set_tris_A(0b00000000);    //Configura os pinos de I/O no Port B.
 output_low(pin_B0); //Habilita TX e desabilita RX MAX485.
 enable_interrupts(global);     //Interrupção global habilitada.

  do
  {
    Endereco = PORT_A & 0b00001111;    //Máscara nos pinos de entrada do Port A RA<3:0>.
 //Cada módulo de acionamento tem endereço configurado de 0000 a 1111 no Dip Switch.
    if(Tecla_Pressionada!=0x00)       //Verifica se alguma tecla foi pressionada.
   /*****************************Módulo de Acionamento 1**********************/
          switch(Endereco)
          {
            case 0b00000001: //Módulo de Acionamento 1.
            {
             Envia_Comando(); //Função de envio de caracteres pela serial.
             } //Fim do case endereço.
           Tecla_Pressionada=0x00;
          } //Fim switch endereço do Módulo de Acionamento 1.
/*************************Módulo de Acionamento 2**************************/
          switch(Endereco)
          {
                case 0b00000010: //Módulo de Acionamento 2.
                {
                 Envia_Comando(); //Função de envio de caracteres pela serial.
                } //Fim do case endereço.
          Tecla_Pressionada=0x00;
          } //Fim switch endereço do Módulo de Acionamento 2.
```

```
/*****************************Módulo de Acionamento 3************************/
        switch(Endereco)
        {
              case 0b00000011: //Módulo de Acionamento 3.
             {
               Envia_Comando(); //Função de envio de caracteres pela serial.
             } //Fim do case endereço.
         Tecla_Pressionada=0x00;
        } //Fim switch endereço do Módulo de Acionamento 3.
/*****************************Módulo de Acionamento 4************************/
        switch(Endereco)
        {
              case 0b00000100: //Módulo de Acionamento 4.
              {
               Envia_Comando(); //Função de envio de caracteres pela serial.
              } //Fim do case endereço.
        Tecla_Pressionada=0x00;
        } //Fim switch endereço do Módulo de Acionamento 4.
/*****************************Módulo de Acionamento 5************************/
    switch(Endereco)
        {
              case 0b00000101: //Módulo de Acionamento 5.
              {
               Envia_Comando(); //Função de envio de caracteres pela serial.
              } //Fim do case endereço.
        Tecla_Pressionada=0x00;
        } //Fim switch endereço do Módulo de Acionamento 5.
/*****************************Módulo de Acionamento 6************************/
    switch(Endereco)
        {
              case 0b00000110: //Módulo de Acionamento 6.
              {
               Envia_Comando(); //Função de envio de caracteres pela serial.
              } //Fim do case endereço.
        Tecla_Pressionada=0x00;
```

```
        } //Fim switch endereço do Módulo de Acionamento 6.
/*************************Módulo de Acionamento 7*************************/
    switch(Endereco)
        {
            case 0b00000111: //Módulo de Acionamento 7.
            {
             Envia_Comando(); //Função de envio de caracteres pela serial.
            } //Fim do case endereço.
        Tecla_Pressionada=0x00;
        } //Fim switch endereço do Módulo de Acionamento 7.
/*************************Módulo de Acionamento 8*************************/
    switch(Endereco)
        {
            case 0b00001000: //Módulo de Acionamento 8.
            {
             Envia_Comando(); //Função de envio de caracteres pela serial.
            } //Fim do case endereço.
        Tecla_Pressionada=0x00;
        } //Fim switch endereço do Módulo de Acionamento 8.
       } //Fim do if.
  } while(TRUE);
} //Fim do void main.
/*****************************************************************************/
```

APÊNDICE B - FUNÇÃO ENVIA_COMANDO

```
/*************************************************************************
Nome do Arquivo: Envia_Comando.h
Versão: 1.0
Descrição: Função para envio de caracteres utilizando microcontrolador PIC e conversor
RS485 para o acionamento de dispositivos eletroeletrônico na automação residencial.
OBS: O acionamento do fecho eletromagnético do portão de entrada social, acionamento do
motor do portão de entrada de veículos e acionamento da cortina da janela da sala, o pulso de
acionamento dos módulos devem ser momentâneos.
Módulo de acionamento 2: Fechadura magnética do portão social.
Módulo de acionamento 3: Portão de entrada de veículos.
Módulo de acionamento 6: Cortina da janela da sala.
Autor: José Carlos da S. Santa Cruz.
Compilador: PIC COMPILER Versão 3.43.
Ambiente de simulação: Proteus 7.7 SP2 toolsuíte ISIS Professional.
Data: agosto de 2011.
UpDate: agosto de 2013.
**********Função de envio de caracteres pela serial**********/
char Tecla_Pressionada =' ';
char Endereco;
void Envia_Comando()
{
/**********Codificação do Módulo de Acionamento 1**********/
//Cada módulo de acionamento tem endereço configurado de 0000 a 1111 no Dip Switch.
 switch(Endereco)
 {
  case 0b00000001:   //Endereço do Módulo de Acionamento 1.
  {
   switch(Tecla_Pressionada)          //Cada tecla acionada controla um dispositivo.
   {
    case '1':      //Valor do número "1" em ASCII (hex) = 31 convertido p/ decimal = 49.
          printf("\f1"); //Envia p/ serial o número "1".
          output_high(pin_B5);
          delay_ms(100);
          output_high(pin_B0);          //Desabilita TX e habilita RX MAX485.
```

```
            printf("\fLIGADO");
            delay_ms(250);
            output_low(pin_B0);          //Habilita TX e desabilita RX MAX485.
     break;
     case 'a':      //Valor de "a" em ASCII (hex) = 61 convertido p/ decimal = 97.
                 printf("\fa");
                 output_low(pin_B5);
                delay_ms(100);
                output_high(pin_B0); //Desabilita TX e habilita RX MAX485.
                printf("\fDESLIGADO");
                delay_ms(250);
                output_low(pin_B0); //Habilita TX e desabilita RX MAX485.
    break;
    } //Fim switch tecla pressionada.
  } //Switch case do Módulo de Acionamento 1.
}  //Switch endereço do Módulo de Acionamento 1.
/**********************Codificação do Módulo de Acionamento 2********************/
switch(Endereco)
{
 case 0b00000010:     //Endereço do Módulo de Acionamento 2.
 {
  switch(Tecla_Pressionada)  //Cada tecla acionada controla um dispositivo.
  {
    case '2': //Valor do número "2" em ASCII = 50 em decimal.
          printf("\f2");    //Envia p/ serial o número "2".
          output_high(pin_B5);
          delay_ms(250);
          output_low(pin_B5);
          delay_ms(100);
          output_high(pin_B0); //Desabilita TX e habilita RX MAX485.
          printf("\fACIONADO");
          delay_ms(250);
          output_low(pin_B0);  //Habilita TX e desabilita RX MAX485.
   break;
   } //Fim switch tecla pressionada.
```

```
  } //Switch case do Módulo de Acionamento 2.
} //Switch do Módulo de Acionamento 2.
/**********************Codificação do Módulo de Acionamento 3*******************/
  switch(Endereco)
  {
    case 0b00000011:  //Endereço do Módulo de Acionamento 3.
    {
     switch(Tecla_Pressionada)          //Cada tecla acionada controla um dispositivo.
     {
       case '3':    //Valor do número "3" em ASCII = 51 em decimal.
                 printf("\f3"); //Envia p/ serial o número "3".
                 output_high(pin_B5);
                delay_ms(250);
                 output_low(pin_B5);
                delay_ms(100);
                output_high(pin_B0);        //Desabilita TX e habilita RX MAX485.
                printf("\fACIONADO");
                delay_ms(250);
                output_low(pin_B0);         //Habilita TX e desabilita RX MAX485.
       break;
     } //Fim switch tecla pressionada.
   } //Switch case do Módulo de Acionamento 3.
}  //Switch endereço do Módulo de Acionamento 3.
/**********************Codificação do Módulo de Acionamento 4*******************/
  switch(Endereco)
  {
    case 0b00000100: //Endereço do Módulo de Acionamento 4.
   {
     switch(Tecla_Pressionada) //Cada tecla acionada controla um dispositivo.
     {
      case '4': //Valor do número "4" em ASCII = 52 em decimal.
                printf("\f4"); //Envia p/ serial o número "4".
                output_high(pin_B5);
                delay_ms(100);
                output_high(pin_B0);        //Desabilita TX e habilita RX MAX485.
```

```
            printf("\fLIGADO");
            delay_ms(250);
            output_low(pin_B0); //Habilita TX e desabilita RX MAX485.
    break;

    case 'd': //Valor de "d" em ASCII = 100 em decimal = 100.
            printf("\fd");
            output_low(pin_B5);
            delay_ms(100);
            output_high(pin_B0); //Desabilita TX e habilita RX MAX485.
            printf("\fDESLIGADO");
            delay_ms(250);
            output_low(pin_B0); //Habilita TX e desabilita RX MAX485.
    break;
   } //Fim switch tecla pressionada.
  } //Switch case do Módulo de Acionamento 4.
} //Switch endereço do Módulo de Acionamento 4.
/**********************Codificação do Módulo de Acionamento 5********************/
  switch(Endereco)
  {
   case 0b00000101: //Endereço do Módulo de Acionamento 5.
   {
    switch(Tecla_Pressionada) //Cada tecla acionada controla um dispositivo.
    {
    case '5': //Valor do número "5" em ASCII = 53 em decimal.
            printf("\f5"); //Envia p/ serial o número "5".
            output_high(pin_B5);
            delay_ms(100);
            output_high(pin_B0);          //Desabilita TX e habilita RX MAX485.
            printf("\fLIGADO");
            delay_ms(250);
            output_low(pin_B0); //Habilita TX e desabilita RX MAX485.
    break;
    case 'e': //Valor de "e" em ASCII = 101 em decimal.
            printf("\fe");
```

```
            output_low(pin_B5);
            delay_ms(100);
            output_high(pin_B0); //Desabilita TX e habilita RX MAX485.
            printf("\fDESLIGADO");
            delay_ms(250);
            output_low(pin_B0);] //Habilita TX e desabilita RX MAX485.
     break;
    } //Fim switch tecla pressionada.
  } //Switch case do Módulo de Acionamento 5.
} //Switch endereço do Módulo de Acionamento 5.
/*********************Codificação do Módulo de Acionamento 6*********************/
  switch(Endereco)
  {
   case 0b00000110: //Endereço do Módulo de Acionamento 6.
    {
   switch(Tecla_Pressionada) //Cada tecla acionada controla um dispositivo.
   {
     case '6': //Valor do número "6" em ASCII = 54 em decimal.
              printf("\f6"); //Envia p/ serial o número "6".
              output_high(pin_B5);
              delay_ms(250);
              output_low(pin_B5);
              delay_ms(150);
              output_high(pin_B0);  //Desabilita TX e habilita RX MAX485.
              printf("\fACIONADO");
             delay_ms(250);
            output_low(pin_B0); //Habilita TX e desabilita RX MAX485.
     break;
    } //Fim switch tecla pressionada.
  } //Switch case do Módulo de Acionamento 6.
} //Switch endereço do Módulo de Acionamento 6.
/*********************Codificação do Módulo de Acionamento 7*********************/
  switch(Endereco)
  {
    case 0b00000111: //Endereço do Módulo de Acionamento 7.
```

```
   {
    switch(Tecla_Pressionada) //Cada tecla acionada controla um dispositivo.
     {
      case '7': //Valor do número "7" em ASCII (hex) = 37 convertido p/ decimal = 55.
            printf("\f7"); //Envia p/ serial o número "7".
            output_high(pin_B5);
            delay_ms(100);
            output_high(pin_B0);          //Desabilita TX e habilita RX MAX485.
            printf("\fLIGADO");
            delay_ms(250);
            output_low(pin_B0); //Habilita TX e desabilita RX MAX485.
      break;
      case 'g': //Valor de "g" em ASCII (hex) = 67 convertido p/ decimal = 103.
            printf("\fg");
            output_low(pin_B5);
            delay_ms(100);
            output_high(pin_B0);          //Desabilita TX e habilita RX MAX485.
            printf("\fDESLIGADO");
            delay_ms(250);
            output_low(pin_B0);          //Habilita TX e desabilita RX MAX485.
      break;
     } //Fim switch tecla pressionada.
   } //Switch case do Módulo de Acionamento 7.
} //Switch endereço do Módulo de Acionamento 7.
/*********************Codificação do Módulo de Acionamento 8*******************/
  switch(Endereco)
  {
   case 0b00001000: //Endereço do Módulo de Acionamento 8.
    {
     switch(Tecla_Pressionada) //Cada tecla acionada controla um dispositivo.
     {
      case '8': //Valor do número "8" em ASCII (hex) = 38 convertido p/ decimal = 56.
            printf("\f7");
            output_high(pin_B5);
            delay_ms(100);
```

```
            output_high(pin_B0);          //Desabilita TX e habilita RX MAX485.
            printf("\fLIGADO");
            delay_ms(250);
            output_low(pin_B0); //Habilita TX e desabilita RX MAX485.
        break;
        case 'h': //Valor de "h" em ASCII (hex) = 68 convertido p/ decimal = 104.
            printf("\fh");
            output_low(pin_B5);
            delay_ms(100);
            output_high(pin_B0);          //Desabilita TX e habilita RX MAX485.
            printf("\fDESLIGADO");
            delay_ms(250);
            output_low(pin_B0);           //Habilita TX e desabilita RX MAX485.
         break;
        } //Fim switch tecla pressionada.
      } //Switch case do Módulo de Acionamento 8.
    } //Switch endereço do Módulo de Acionamento 8.
} //Fim da função Envia_Comando.
/**************************************************************************
/
```

APÊNDICE C – CÓDIGO DO MÓDULO DE ACIONAMENTO MANUAL

```
/*****************************************************************************/
Descrição: Acionamento manual utilizando um teclado matricial de membranas 4x3.
Nome do arquivo: teclado_f73.c
Autor: José Carlos S.S. Cruz
Micro utilizado: PIC16F73;
Compilador: IDE PCW C Compiler versão 3.43 CCS;
Ambiente de simulação: ISIS Proteus 7.8SP2;
Data de criação: Agosto 2013-Versão: 1.0;
UpDate:
/*****************************Configurações do PIC****************************/
#include <16F73.h>
//#device adc=8
#fuses HS, NOWDT, NOPROTECT, PUT, NOBROWNOUT
#use delay(clock=10000000)
#include <Tecla_628.h> //Função p/ o teclado numérico.
/*************************Definições dos pinos do PIC************************/
//Os defines de linhas e colunas estão no arquivo Tecla_628.h
#define L4  pin_B7 //Linha 4 conectada a RB7.
#define L3  pin_B6 //Linha 4 conectada a RB6.
#define L2  pin_B5 //Linha 4 conectada a RB5.
#define L1  pin_B4 //Linha 4 conectada a RB4.
#define C3  pin_B3 //Linha 4 conectada a RB3.
#define C2  pin_B2 //Linha 4 conectada a RB2.
#define C1  pin_B1 //Linha 4 conectada a RB1.
/************************Definições na memória de programa********************/
byte Valor = 0;
#BYTE PORT_A = 5 //Configura as variáveis p/ mapeamento da memória.
#BYTE PORT_B = 6
#BYTE PORT_C = 7
#use fast_io(a) //A MCU terá controle sobre o hardware.
#use fast_io(b)
#use fast_io(c)
/*********************************Função principal****************************/
```

```
void main()
{
 set_tris_A(0b00000000); //Configura os pinos de I/O no Port A.
 set_tris_B(0b00000000); //Configura os pinos de I/O no Port B.
 set_tris_C(0b10000000); //Configura os pinos de I/O no Port C.
  output_A(0x00);  //Limpa o Port A.
 setup_adc_ports(NO_ANALOGS);
 setup_adc(ADC_OFF);
 setup_spi(FALSE);
 setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
 setup_timer_1(T1_DISABLED);
 setup_timer_2(T2_DISABLED,0,1);
 output_low(pin_A0);//Pino RA0 habilita RX e desabilita TX do MAX485.
 do
 {
  while(Tecla_1()) //Tecla 1.
  {
   Gera_Ton(1000,100); //Função gera som (1kHz duração=100ms), caso uma tecla seja pressionada.
   output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
   printf("\f1"); //Liga carga 1.
   delay_ms(150);
   output_low(pin_A0);          //Habilita RX e desabilita TX MAX485.
    while(Tecla_1()) //Tecla 1.
    {
     delay_ms(50);
     output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
     printf("\fa"); //Desliga carga 1.
            delay_ms(50);
     output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
    }
   }
    while(Tecla_2()) //Tecla 2.
    {
     Gera_Ton(1000,100); //Função gera som (1kHz duração=100ms), caso uma tecla seja pressionada.
```

```
   output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
   printf("\f2"); //Liga carga 2.
   delay_ms(50);
   output_low(pin_A0); //Habilita RX e desabilita TX MAX485.

  while(Tecla_2()) //Tecla 2.
  {
   delay_ms(50);
   output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
   printf("\fb"); //Desliga carga 2.
   delay_ms(50);
   output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
  }
  }
    while(Tecla_3()) //Tecla 3.
    {
     Gera_Ton(1000,100); //Função gera som (1kHz duração=100ms), caso uma tecla seja pressionada.
     output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
     printf("\f3"); //Liga carga 3.
    delay_ms(50);
    output_low(pin_A0);       //Habilita RX e desabilita TX MAX485.

     while(Tecla_3()) //Tecla 3.
     {
      delay_ms(50);
      output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
      printf("\fc"); //Desliga carga 3.
      delay_ms(50);
      output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
      }
     }

    while(Tecla_4()) //Tecla 4.
    {
```

```
      Gera_Ton(1000,100); //Função gera som (1kHz duração=100ms), caso uma tecla seja pressionada.
      printf("\f4"); //Liga carga 4.
      output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
      delay_ms(50);
      output_low(pin_A0); //Habilita RX e desabilita TX MAX485.

      while(Tecla_4()) //Tecla 4.
      {
       delay_ms(50);
       output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
       printf("\fd"); //Desliga carga 4.
       delay_ms(50);
       output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
      }
    }

     while(Tecla_5()) //Tecla 5.
     {
      Gera_Ton(1000,100); //Função gera som (1kHz duração=100ms), caso uma tecla seja pressionada.
      printf("\f5"); //Liga carga 5.
      output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
      delay_ms(50);
      output_low(pin_A0);      //Habilita RX e desabilita TX MAX485.

      while(Tecla_5()) //Tecla 4.
      {
       delay_ms(50);
       output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
       printf("\fe"); //Desliga carga 5.
       delay_ms(50);
       output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
      }
     }
```

```
while(Tecla_6()) //Tecla 6.
{
 Gera_Ton(1000,100); //Função gera som (1kHz duração=100ms), caso uma tecla seja pressionada.
 printf("\f6"); //Liga carga 6.
 output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
 delay_ms(50);
 output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
 while(Tecla_6()) //Tecla 6.
 {
  delay_ms(50);
  output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
  printf("\ff"); //Desliga carga 6.
  delay_ms(50);
  output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
 }
}

while(Tecla_7()) //Tecla 7.
{
 Gera_Ton(1000,100); //Função gera som (1kHz duração=100ms), caso uma tecla seja pressionada.
 printf("\f7"); //Liga carga 7.
 output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
 delay_ms(50);
 output_low(pin_A0);      //Habilita RX e desabilita TX MAX485.

 while(Tecla_7()) //Tecla 7.
 {
  delay_ms(50);
  output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
  printf("\fg"); //Desliga carga 7.
  delay_ms(50);
  output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
 }
}
```

```
while(Tecla_8()) //Tecla 8.
{
 Gera_Ton(1000,100); //Função gera som (1kHz duração=100ms), caso uma tecla seja pressionada.
 printf("\f8"); //Liga carga 8.
 output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
 delay_ms(50);
output_low(pin_A0);       //Habilita RX e desabilita TX MAX485.

 while(Tecla_8()) //Tecla 8.
 {
  delay_ms(50);
  output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
  printf("\fh"); //Desliga carga 8.
  delay_ms(50);
  output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
 }
}

 while(Tecla_9()) //Tecla 9.
{
 Gera_Ton(1000,100); //Função gera som (1kHz duração=100ms), caso uma tecla seja pressionada.
 printf("\f9"); //Liga carga 9.
 output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
 delay_ms(50);
output_low(pin_A0);       //Habilita RX e desabilita TX MAX485.

while(Tecla_9()) //Tecla 9.
 {
  delay_ms(50);
  output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
  printf("\fi"); //Desliga carga 9.
        delay_ms(50);
  output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
 }
```

```
    }
    while(Tecla_0()) //Tecla 0.
    {
     Gera_Ton(1000,100); //Função gera som (1kHz duração=100ms), caso uma tecla seja pressionada.
     printf("\f0"); //Liga carga 10.
     output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
     delay_ms(50);
    output_low(pin_A0);       //Habilita RX e desabilita TX MAX485.

     while(Tecla_0()) //Tecla 0.
     {
      delay_ms(50);
      output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
      printf("\fj"); //Desliga carga 9.
      delay_ms(50);
      output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
     }
    }

  while(Tecla_12()) //Tecla # desliga todas as cargas acionadas.
  {
   Gera_Ton(1000,500); //Função gera som (1kHz duração=0,5s), caso uma tecla seja pressionada.
   output_high(pin_A0); //Desabilita RX e habilita TX MAX485.
   delay_ms(50);
   printf("\fa");
   printf("\fb");
   printf("\fc");
   printf("\fd");
   printf("\fe");
   printf("\ff");
   printf("\fg");
   printf("\fh");
   printf("\fi");
   printf("\fj");
```

```
   delay_ms(50);
   output_low(pin_A0); //Habilita RX e desabilita TX MAX485.
  }
 delay_ms(100);
 } while(true);
}
/*****************************************************************************/
```

APÊNDICE D - FUNÇÃO TECLA_628

```
/*************************Função p/ Teclado Matricial 4 x 3**********************
Descrição: Função para o acionamento das linhas e colunas do teclado 4x3.
Nome do arquivo: Tecla_628.h
Versão:
Autor: José Carlos S. S. Cruz
Compilador: IDE PCW C Compiler versão 3.43 CCS
Ambiente de simulação: ISIS Proteus 7.8SP2
Data de criação: julho de 2013.
UpDate:
*********************************Definições*********************************/
#define L4  pin_B7 //Linha 4 conectada a RB7.
#define L3  pin_B6 //Linha 4 conectada a RB6.
#define L2  pin_B5 //Linha 4 conectada a RB5.
#define L1  pin_B4 //Linha 4 conectada a RB4.
#define C3  pin_B3 //Linha 4 conectada a RB3.
#define C2  pin_B2 //Linha 4 conectada a RB2.
#define C1  pin_B1 //Linha 4 conectada a RB1.
#include <tones-v628.h>       //Biblioteca da CCS alterada p/ ligação do Buzzer no pino RB0 do PIC16F628.
/*****************************Função para tecla #*****************************/
int Tecla_12() //Tecla #.
{
 output_B(0b11111000); //RB<7:1> em 1.
 output_low(L4); //L4 desativada conectada ao pino RB7.
 input(C3); //C3 entrada conectado ao pino RB3.
 delay_ms(10); //Tempo de 10ms.
   if(input(C3)) //Testa entrada de C3...
   return(0); //Retorna 0 se RB3 = 1.
   return(1); //Retorna 1, RB3 = 0 indicando tecla pressionada.
}
/************************Função para tecla* *********************************/
int Tecla_11() //Tecla *.
{
```

```
 output_B(0b11111000); //RB<7:1> em 1.
 output_low(L4); //L4 desativada conectada ao pino RB7.
 input(C1); //C1 entrada conectado ao pino RB1.
 delay_ms(10); //Tempo de 10ms.
   if(input(C1)) //Testa entrada de C1...
   return(0); //Retorna 0 se RB1 = 1.
   return(1); //Retorna 1, RB1 = 0 indicando tecla pressionada.
}
/*******************************Função para tecla 0*****************************/
int Tecla_0() //Tecla 0.
{
 output_B(0b11111000); //RB<7:1> em 1.
 output_low(L4); //L4 desativada conectada ao pino RB7.
 input(C2); //C2 entrada conectado ao pino RB2.
 delay_ms(10); //Tempo de 10ms.
   if(input(C2)) //Testa entrada de C2...
   return(0); //Retorna 0 se RB2 = 1.
   return(1); //Retorna 1, RB2 = 0 indicando tecla pressionada.
 }
/*******************************Função para tecla 9*****************************/
int Tecla_9() //Tecla 9.
{
 output_B(0b11111000); //RB<7:1> em 1.
 output_low(L3); //L3 desativada conectada ao pino RB6.
 input(C3); //C3 entrada conectado ao pino RB3.
 delay_ms(10); //Tempo de 10ms.
   if(input(C3)) //Testa entrada de C3...
   return(0); //Retorna 0 se RB3 = 1.
   return(1); //Retorna 1, RB3 = 0 indicando tecla pressionada.
}
/*******************************Função para tecla 8*****************************/
int Tecla_8() //Tecla 8.
{
 output_B(0b11111000); //RB<7:1> em 1.
 output_low(L3); //L3 desativada conectada ao pino RB6.
```

```
 input(C2); //C2 entrada conectado ao pino RB2.
 delay_ms(10); //Tempo de 10ms.
   if(input(C2)) //Testa entrada de C2...
   return(0); //Retorna 0 se RB2 = 1.
   return(1); //Retorna 1, RB2 = 0 indicando tecla pressionada.
}
/*****************************Função para tecla 7*****************************/
int Tecla_7() //Tecla 7.
{
 output_B(0b11111000); //RB<7:1> em 1.
 output_low(L3); //L3 desativada conectada ao pino RB6.
 input(C1); //C1 entrada conectado ao pino RB1.
 delay_ms(10); //Tempo de 10ms.
   if(input(C1)) //Testa entrada de C1...
   return(0); //Retorna 0 se RB1 = 1.
   return(1); //Retorna 1, RB1 = 0 indicando tecla pressionada.
}
/*****************************Função para tecla 6*****************************/
int Tecla_6() //Tecla 6.
{
 output_B(0b11111000); //RB<7:1> em 1.
 output_low(L2); //L2 desativada conectada ao pino RB5.
 input(C3); //C3 entrada conectado ao pino RB3.
 delay_ms(10); //Tempo de 10ms.
   if(input(C3)) //Testa entrada de C3...
   return(0); //Retorna 0 se RB3 = 1.
   return(1); //Retorna 1, RB3 = 0 indicando tecla pressionada.
}
/*****************************Função para tecla 5*****************************/
int Tecla_5() //Tecla 5.
{
 output_B(0b11111000); //RB<7:1> em 1.
 output_low(L2); //L2 desativada conectada ao pino RB5.
 input(C2); //C2 entrada conectado ao pino RB2.
 delay_ms(10); //Tempo de 10ms.
```

```
  if(input(C2)) //Testa entrada de C2...
  return(0); //Retorna 0 se RB2 = 1.
  return(1); //Retorna 1, RB2 = 0 indicando tecla pressionada.
}
/*****************************Função para tecla 4*****************************/
int Tecla_4() //Tecla 4.
{
 output_B(0b11111000); //RB<7:1> em 1.
 output_low(L2); //L2 desativada conectada ao pino RB5.
 input(C1); //C1 entrada conectado ao pino RB1.
 delay_ms(10); //Tempo de 10ms.
  if(input(C1)) //Testa entrada de C1...
  return(0); //Retorna 0 se RB1 = 1.
  return(1); //Retorna 1, RB1 = 0 indicando tecla pressionada.
}
/*****************************Função para tecla 3*****************************/
int Tecla_3() //Tecla 3.
{
 output_B(0b11111000); //RB<7:1> em 1.
 output_low(L1); //L1 desativada conectada ao pino RB4.
 input(C3); //C3 entrada conectado ao pino RB3.
 delay_ms(10); //Tempo de 10ms.
  if(input(C3)) //Testa entrada de C3...
  return(0); //Retorna 0 se RB3 = 1.
  return(1); //Retorna 1, RB3 = 0 indicando tecla pressionada.
}
/*****************************Função para tecla 2*****************************/
int Tecla_2() //Tecla 2.
{
 output_B(0b11111000); //RB<7:1> em 1.
 output_low(L1); //L1 desativada conectada ao pino RB4.
 input(C2); //C2 entrada conectado ao pino RB2.
 delay_ms(10); //Tempo de 10ms.
  if(input(C2)) //Testa entrada de C2...
  return(0); //Retorna 0 se RB2 = 1.
```

```
  return(1); //Retorna 1, RB2 = 0 indicando tecla pressionada.
}
/*******************************Função para tecla 1*************************/
int Tecla_1() //Tecla 1.
{
 output_B(0b11111000); //RB<7:1> em 1.
 output_low(L1); //L1 desativada conectada ao pino RB4.
 input(C1); //C1 entrada conectado ao pino RB1.
 delay_ms(10); //Tempo de 10ms.
  if(input(C1)) //Testa entrada de C1...
  return(0); //Retorna 0 se RB1 = 1.
  return(1); //Retorna 1, RB1 = 0 indicando tecla pressionada.
}
/*********************************************************************/
```

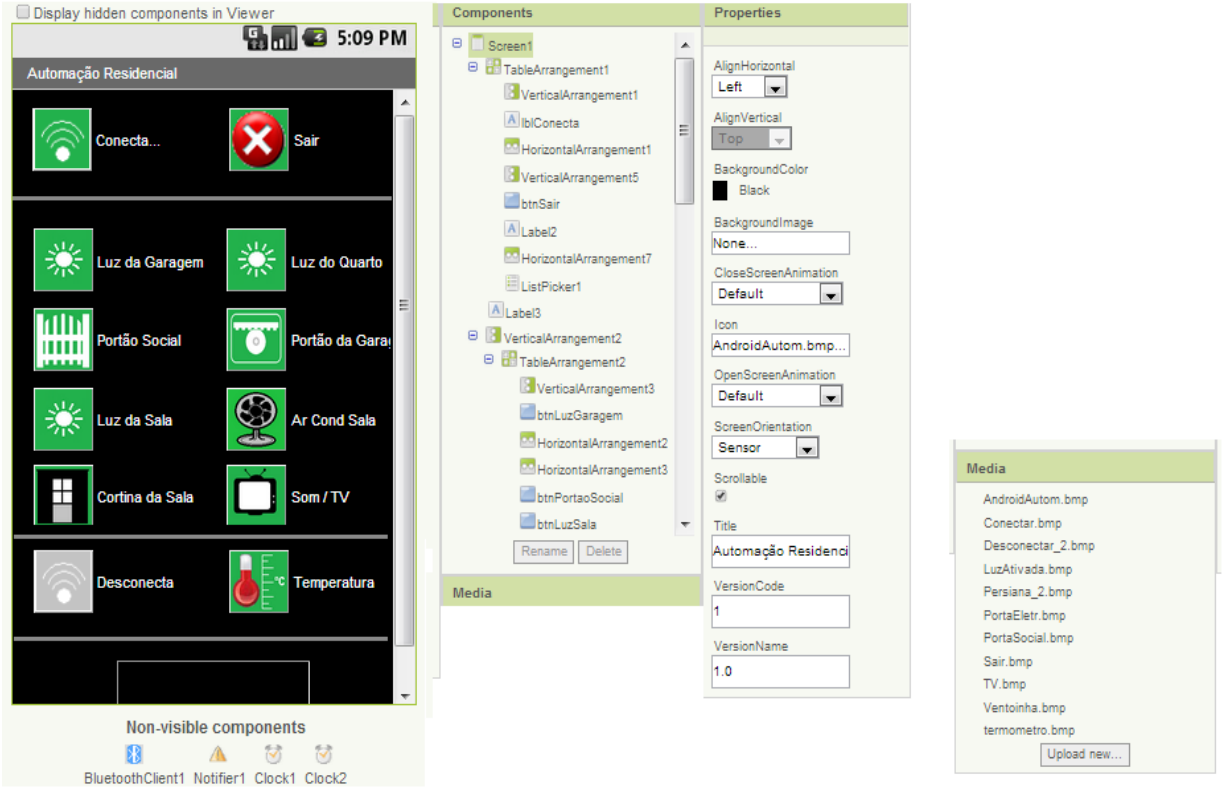
APÊNDICE E – CÓDIGO DO MÓDULO SENSOR DE TEMPERATURA

```
/*****************************TEMPERATAURA PIC12F675*********************
Nome do Arquivo: temperatura_f675.c
Versão: 1.0
Descrição:
Autor: José Carlos da S. Santa Cruz.
Compilador: PIC COMPILER Versão 3.43.
Ambiente de simulação: Proteus 7.7 SP2 toolsuíte ISIS Professional.
Microcontrolador utilizado: PIC12F675.
Data: outubro de 2013.
*************************Definições do microcontrolador**********************/
#include <12F675.h>
#device adc=10
#use delay(clock=4000000)
#fuses NOWDT,INTRC_IO, NOCPD, NOPROTECT, NOMCLR, PUT, BROWNOUT
//#use standard_io(a)
/*************************Definições dos pinos do microcontrolador***************/
#use fast_io (a) //Inicialização rápida dos pinos de I/O.
#byte port_a = 0x05
#define Rele      Pin_A5
#define Enable    Pin_A2
#define Entrada   Pin_A1
/*************************Definições das constantes e variáveis******************/
unsigned long valor = 0;
float Celsius = 0;
char Tecla;
/*****************************Função p/ leitura da temperatura*******************/
void Termometro()
{
 //printf("\f\r\n");
 setup_adc_ports(ALL_ANALOG); //Configura o ADC p/ pinos analógicos.
 setup_adc(ADC_CLOCK_INTERNAL); //ADC com clock interno.
 set_adc_channel(0);  //Leitura da coleta de dados em RA1.
 delay_ms(10); //Tempo necessário p/ estabilizar as configurações ADC.
```

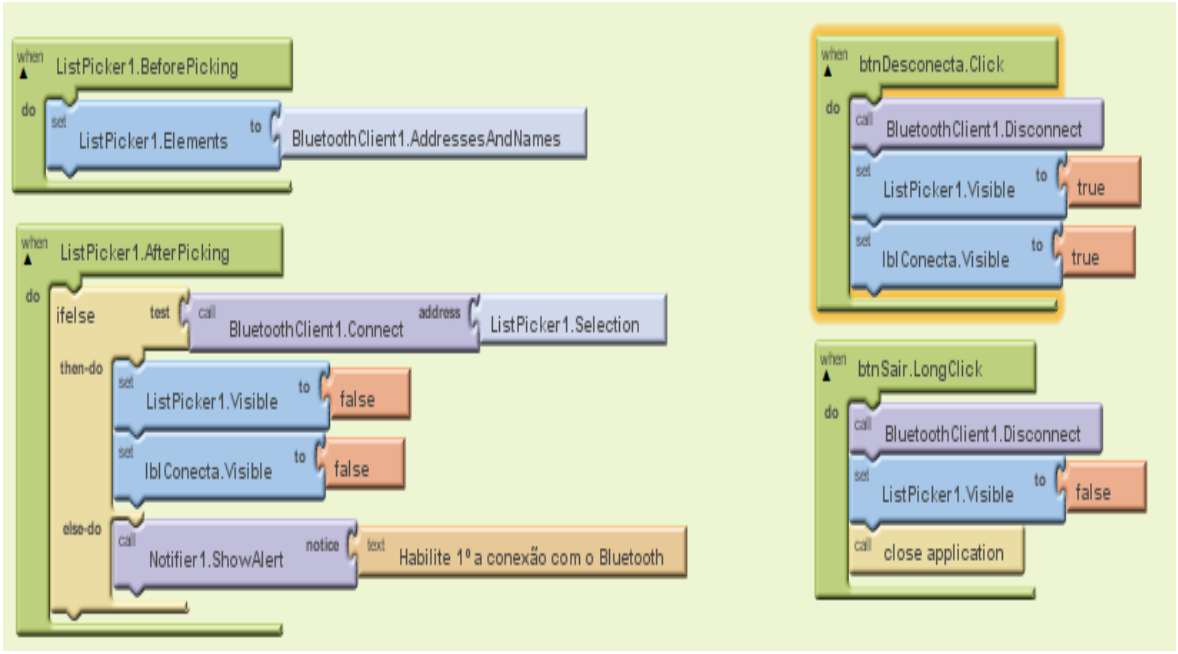
```
 valor = read_adc();
 Celsius = valor * 100 * (5.0/1024); //Atribui à variável a leitura do canal ADC 1.
 printf("Temperatura: %1.1f\xDFC\n\r", Celsius);    //Mostra no terminal serial.
 delay_ms(500); //Tempo entre as leituras da coleta de dados.
}
/*****************************Função principal*****************************/
void main()
{
 setup_comparator(NC_NC_NC_NC);
 setup_vref(FALSE);
 set_tris_A(0b0001011); //Configura GPIO como entradas e saídas.
 //output_A(0x00);  //Limpa o Port A.
 output_low(Enable); //Habilita TX e desabilita RX MAX485.
 do
 {
  Tecla=getc();
  //printf("\f");
  switch (Tecla)
  {
   printf("\f");
     case 't':
                output_high(Enable); //Desabilita TX e habilita RX MAX485.
                delay_ms(50);
                printf("\f");
               delay_ms(100);
               Termometro();
               output_low(Enable); //Habilita TX e desabilita RX MAX485.
     break;
     case 'u':
                output_high(Enable); //Desabilita TX e habilita RX MAX485.
               delay_ms(50);
               output_high(Rele);
               printf("\f");
               delay_ms(100);
               printf("LIGADO\r\n"); //Status do dispositivo = LIGADO.
```

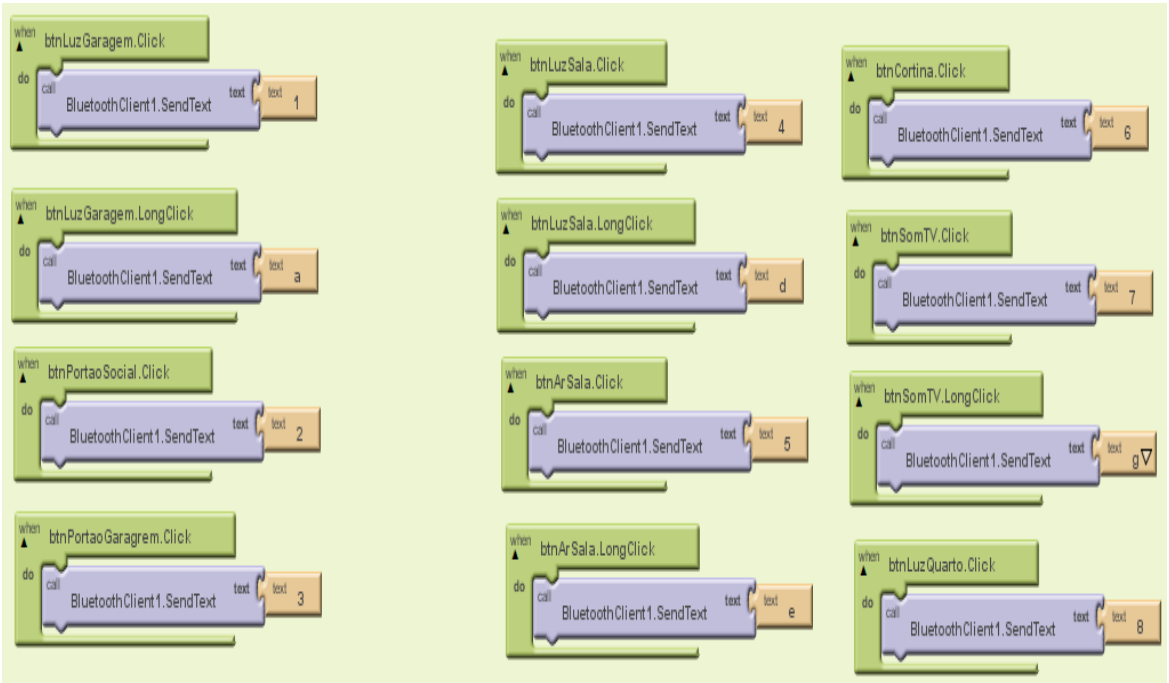
```
            delay_ms(250);
            printf("\fu");
           delay_ms(100);
           printf("\f");
           output_low(Enable); //Habilita TX e desabilita RX MAX485.
    break;
   }
  }while(TRUE);
}
```

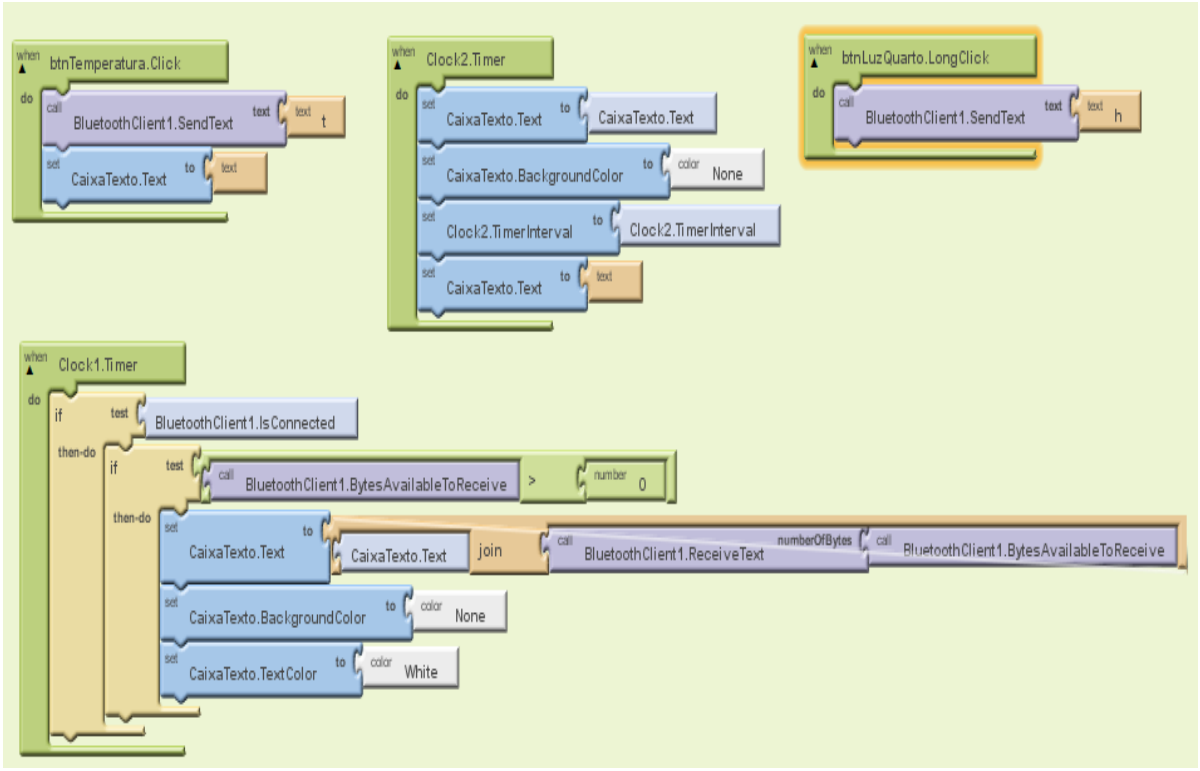
APÊNDICE F - *LAYOUT* DO APLICATIVO PARA O *SMARTPHONE*



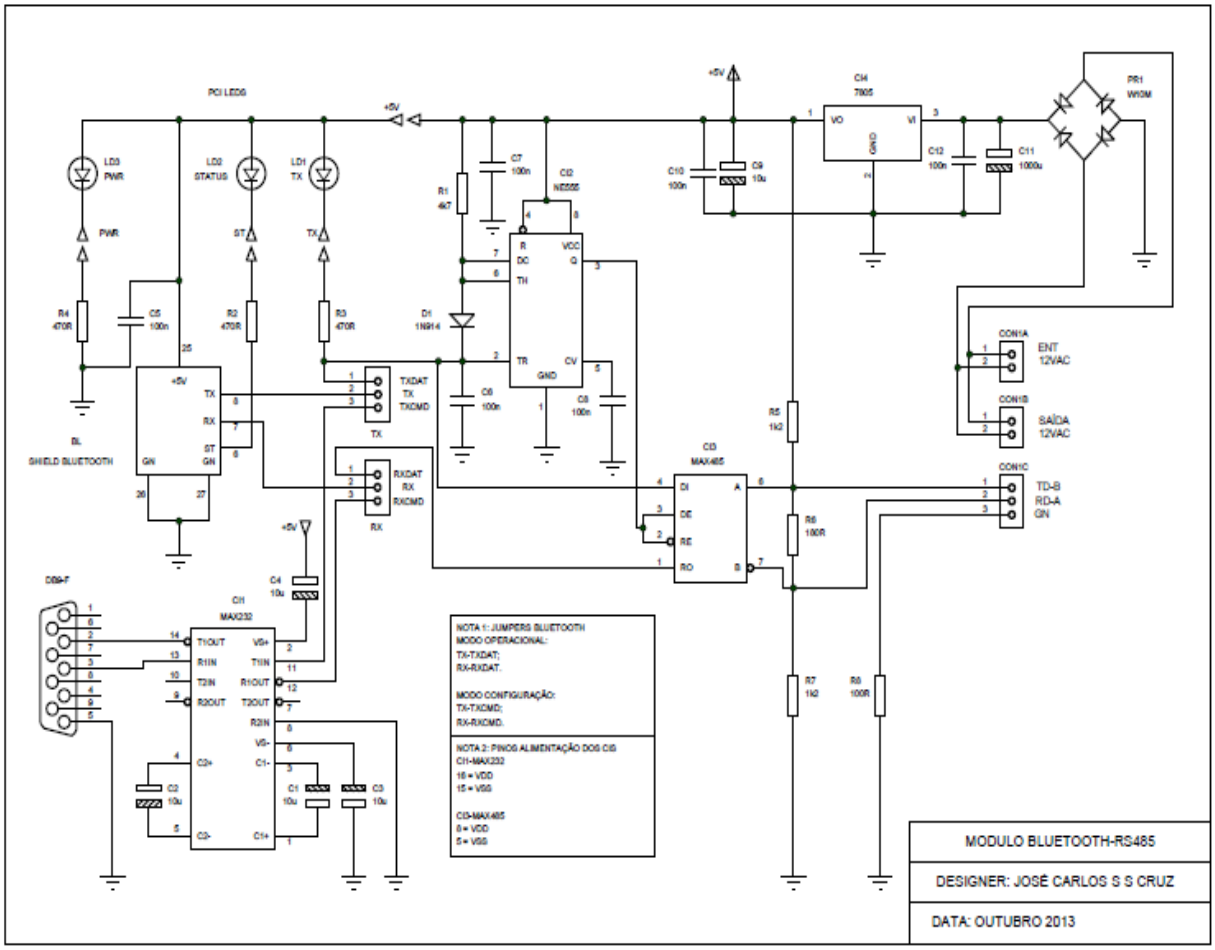
APÊNDICE G - BLOCOS DO CÓDIGO DO APLICATIVO



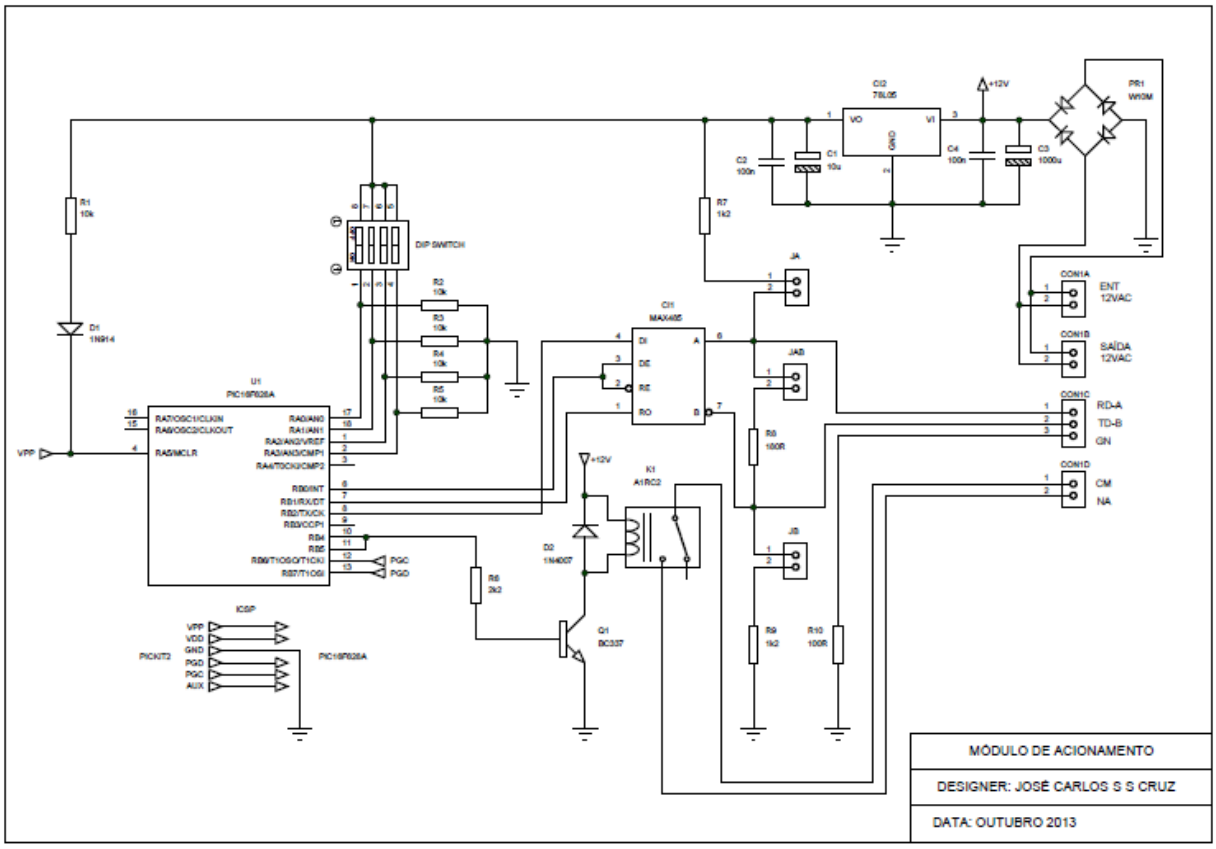



when btnTemperatura.Click
do
call BluetoothClient1.SendText
text
text t
set CaixaTexto.Text
to
text
when Clock2.Timer
do
set CaixaTexto.Text to CaixaTexto.Text
set CaixaTexto.BackgroundColor to color None
set Clock2.TimerInterval to Clock2.TimerInterval
set CaixaTexto.Text to text
when btnLuzQuarto.LongClick
do
call BluetoothClient1.SendText
text
text h
when Clock1.Timer
do
if
test
BluetoothClient1.IsConnected
then-do
if
test
call BluetoothClient1.BytesAvailableToReceive
>
number 0
then-do
set CaixaTexto.Text to
CaixaTexto.Text
join
call BluetoothClient1.ReceiveText
numberOfBytes
call BluetoothClient1.BytesAvailableToReceive
set CaixaTexto.BackgroundColor to color None
set CaixaTexto.TextColor to color White

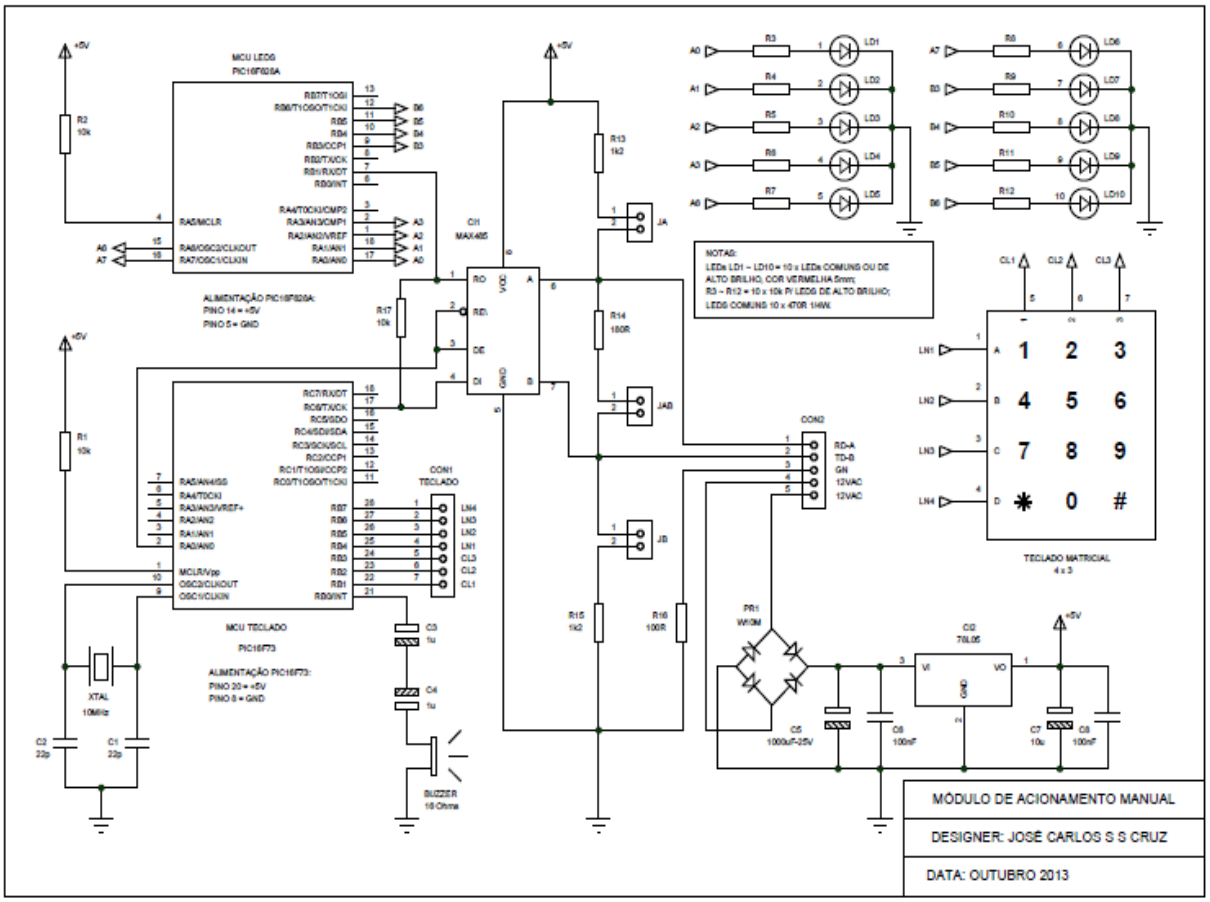
APÊNDICE H - DIAGRAMA ESQUEMÁTICO DO MÓDULO *BLUETOOTH*-RS485



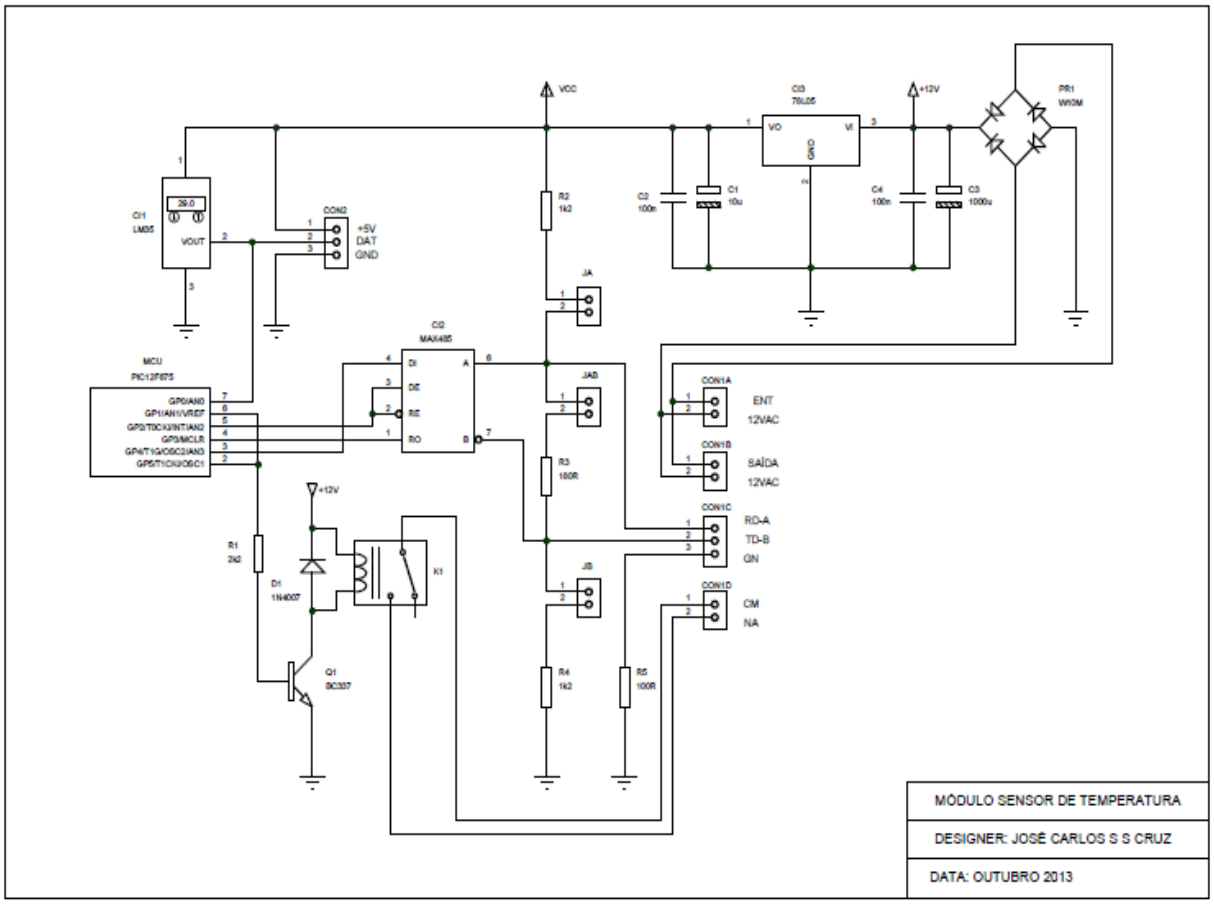
APÊNDICE I - DIAGRAMA ESQUEMÁTICO DO MÓDULO DE ACIONAMENTO



APÊNDICE J - DIAGRAMA ESQUEMÁTICO DO MÓDULO DE ACIONAMENTO MANUAL



APÊNDICE K - DIAGRAMA ESQUEMÁTICO DO MÓDULO SENSOR DE TEMPERATURA



ANEXO A – ALTERAÇÃO DA BIBLIOTECA GERA_TON DO PIC C *COMPILER*

//// (C) Copyright 1996, 2003 Custom Computer Services ////

/*

This source code may only be used by licensed users of the CCS C compiler. This source code may only be distributed to other licensed users of the CCS C compiler. No other use, reproduction or distribution is permitted without written permission. Derivative programs created using this software in object code form are not restricted in any way.

*/

```
//#define TONE_PIN  PIN_B0
//#define BUZZER    PIN_B2 //Modificado p/ buzzer no pino RB2 do PIC16F877A.
#define BUZZER      PIN_B0 //Modificado p/ buzzer no pino RB0 do PIC16F628.
```