CENTRO UNIVERSITÁRIO DE BRASÍLIA -UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

ANDRÉ LUIZ LOPES DE ALMEIDA

SISTEMA DE CONTROLE AUTOMATIZADO DE ANTENA DE TV

Orientadora: M.C. Maria Marony Sousa Farias

Brasília

Junho, 2011

ANDRÉ LUIZ LOPES DE ALMEIDA

SISTEMA DE CONTROLE AUTOMATIZADO DE ANTENA DE TV

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

Brasília
Junho, 2011

ANDRÉ LUIZ LOPES DE ALMEIDA

SISTEMA DE CONTROLE AUTOMATIZADO DE ANTENA DE TV

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientadora: M.C. 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. Abiezer Amarilia Fernandez
Coordenador do Curso

Banca Examinadora:

_____

Prof. Maria Marony, Mestre em Engenharia Elétrica
Orientadora

_____

Prof. Carlos Kleber da Silva Rodrigues, Doutor em Engenharia de Sistemas e Computação

_____

Prof. João Marcos Souza Costa, Especialista.

_____

Prof. Vera.

DEDICATÓRIA

Dedico esta monografia, a minha família, Jocélia Lopes de Almeida e Pedro Henrique Lopes de Almeida, pois foram os grandes propulsores do meu sucesso.

Dedico a todos que de alguma maneira, por mais simples que fosse, me ajudaram a superar e concluir mais esta fase de minha vida.

AGRADECIMENTOS

Primeiramente a Deus, por suas bênçãos que me fizeram dar os passos certos para chegar a mais esta vitória.

A toda minha família, minha mãe Jocélia, meu irmão Pedro Henrique, meus familiares e minha namorada Isabela Torres pelo apoio, amor, confiança, carinho, compreensão e atenção que eu poderia ter nesse decorrer do curso.

A todos meus amigos, pela ajuda que me deram para conclusão deste, e que ao longo desse período me proporcionaram conhecimentos e informações. E principalmente ao meu grande Amigo Carlos Epaminondas por ter me ajudado em momentos de grandes dificuldades.

Aos professores do UNICEUB, que me proporcionaram um grande conhecimento durante o curso.

E agradeço também, a minha orientadora e professora, Maria Marony, que com sua presteza e conhecimento me possibilitou a realizar um melhor trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS E TABELAS

RESUMO

Neste projeto, é apresentado o desenvolvimento de um protótipo para automatização do posicionamento de uma antena de TV, por meio de um controle sem fio utilizando a tecnologia ZigBee. O motivo da escolha dessa automação foi o controle do motor de passo por meio de um microcontrolador, que receberá as informações vindas da placa RCON-HOMBEE através de um software de gerenciamento de comando. O computador é o responsável pelo controle e monitoramento do sistema, a partir dele os comandos serão enviados do adaptador RCON-USBBEE para a RCON-HOMEBEE e assim definirá o sentido de rotação da antena de TV.

O diferencial do projeto está na intenção de proporcionar uma melhora na qualidade de imagem da TV, utilizando uma forma prática para o ajuste da antena, oferecendo mais segurança ao usuário e comodidade por não ter que obrigá-lo a subir em telhados.

Palavras-Chave: Motor de passo, microcontrolador, ZigBee, Antena de TV.

ABSTRACT

This project aims to develop a prototype for automation positioning a TV antenna, by using a wireless controller ZigBee technology. The reason for this choice was the automation control stepper motor through a microcontroller, which receives information from RCON HOMBEE-plate through a management software command. The computer is responsible for controlling and monitoring system, because from it are the commands sent from the RCON RCON-to-USBBEE HOMEBEE and well define the direction of rotation of the model antenna.

The difference in this project intended to provide an improvement the image quality of the TV, using a practical way to adjust the antenna, giving the user more security and convenience by not having to forcing him to climb on roofs.

Keywords: stepper motor, microcontroller, ZigBee, TV Antenna.

CAPÍTULO 1 - INTRODUÇÃO

Em uma residência, seja ela uma casa independente ou um apartamento, podem existir vários graus de automação. Mais especificamente, a Domótica é a disciplina que se ocupa em estudar e aplicar soluções tecnológicas para automatizar certas operações ou seqüências de ações executadas em um ambiente doméstico (SCHNEIDER ELECTRIC).

1.1 – Motivação

O que originou à motivação para realização deste projeto foi detectar a dificuldade que um morador tem em obter uma melhor qualidade do sinal de TV para canais abertos, devido ao fato de a antena estar mal posicionada, e também a dificuldade em fazer qualquer modificação na parte superior da casa para alterar a posição de uma antena qualquer. Essa alteração para alguns pode ser algo simples, porém para outros, se torna quase impossível, devido ao fato de alguns moradores serem portadores de dificuldades físicas. Portanto, a principal motivação para a criação do protótipo é a de apresentar uma solução para o morador, para que ele não tenha dificuldade em encontrar uma melhor posição na antena de TV, e também não corra o risco de sofrer um acidente ao subir na casa.

1.2 – Objetivos

O objetivo geral é elaborar uma solução para automatizar o controle de posicionamento da antena de TV, baseado na construção de um circuito controlador do motor de passo, utilizando a comunicação ZIGBEE.

Os objetivos específicos são:

- Desenvolvimento do circuito controlador da antena;
- Desenvolvimento do suporte de rotação da antena, junto com as engrenagens e motor de passo;
- Desenvolvimento do Software para o controle do circuito controlador da antena;
- Desenvolvimento da interface de controle do posicionamento da antena.

A figura 1.1 mostra a topologia do projeto, proporcionando ter uma visão geral do funcionamento do protótipo.

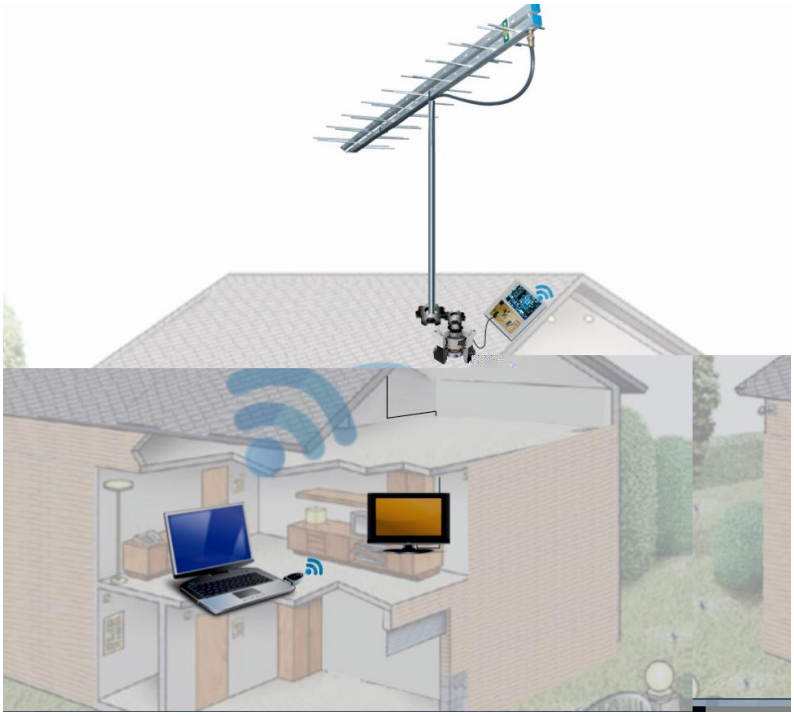


Figura 1.1 - Topologia do Projeto / Fonte: Autor.

1.3 – Estrutura do Trabalho

A monografia está distribuída da seguinte forma:

Capítulo 1: Introdução - Parte que trata da motivação do trabalho, objetivos gerais e específicos e como a monografia foi estruturada.

Capítulo 2: Referencial Teórico - Oferece o embasamento teórico da proposta de solução do problema. São abordados os temas Antenas, Motor de passo, Engrenagens, Comunicação ZIGBEE e Linguagens de Programação.

Capítulo 3: Apresentação do Problema - Explanação sobre o problema que o trabalho se propõe a tratar. Aborda assuntos como o mal posicionamento da antena de TV influencia na recepção de um bom sinal.

Capítulo 4: Desenvolvimento do Projeto - A parte de projeto de software e hardware é abordada nesse item. Todo o procedimento, metodologia, tecnologias e equipamentos são mostrados e detalhados. A estimativa de custos do projeto também está presente neste capítulo.

Capítulo 5: Testes e Resultados - Apresentação dos casos de testes efetuados para comprovar o funcionamento do dispositivo. É apresentado também a análise dos resultados.

Capítulo 6: Conclusão - Síntese conclusiva do projeto final e sugestões para trabalhos futuros.

1.4 - Identificação do Problema e Cenário Atual

No Brasil existem regiões que possuem transmissão de TV aberta. Tendo em vista que existem regiões onde não possuem esse tipo de transmissão, às vezes pelo fato de a antena estar muito distante, é necessário utilizar outros modelos de antena de TV como a antena parabólica.

Normalmente as casas que ficam longe das antenas de transmissão, necessitam de mais precisão no posicionamento da antena para a melhor captação do sinal. Existem casos que não só a distância prejudica a qualidade de recepção do sinal, mas também as barreiras como, por exemplo, prédios e árvores.

Um outro problema seria a má instalação da antena, pois existem casos em que antenistas as instalam conforme sua conveniência, pois o custo para ele vale conforme a instalação. Essa má instalação prejudica o melhor posicionamento da antena.

Figura 1.2 – Risco na mudança de posição da Antena / Fonte: Autor.

Mas já existem residentes que preferem não pagar para fazer a instalação da antena, e acabam se arriscando fazendo as instalações sozinhos como mostra a figura 1.2. Uma observação importante é que seria necessário no mínimo duas pessoas para definir uma melhor posição da antena, pois uma ficaria no alto da casa, e a outra ficaria próximo à televisão informando o estado da imagem, levando a consideração que isso não seria muito cômodo e muito menos seguro.

CAPÍTULO 2 - REFERENCIAL TEÓRICO

2.1 Antenas

O número de diferentes tipos de antenas que existem atualmente é consideravelmente grande, porém a maioria delas são projetadas para se adequar a uma aplicação predefinida.

Para discutir um dos principais tipos de antenas usadas hoje em dia, seguiremos com a descrição de operação da antena Yagi. Essa classe de antena é extremamente importante em uma variedade de aplicações UHF, VHF e de microondas em que se torna necessário reunir simultaneamente uma baixa resistência ao vento, um bom ganho elétrico e requisitos de diagrama de radiação polar. Tais antenas podem ser usadas em muitas aplicações nas quais é essencial que antena tenha um perfil baixo e seja aderente à estrutura de montagem conforme a construção.

A antena Yagi ou Yagi-Uda, denominada assim em homenagem a Hidet Sugu Yagi (1926), é um arranjo linear com apenas um elemento alimentado. Essa é uma estrutura de antena muito importante e usada em uma variedade de aplicações onde é necessário um transmissor de três elementos mostrados na figura 2.1 O elemento alimentado, elemento 2, é uma antena dipolo de meia onda. O elemento 1 é construído de forma a ter um comprimento um pouco maior que o elemento 2 para acomodar a reatância indutiva causada pelo acoplamento mútuo devido ao espaçamento de 0,25λ entre os elementos 1 e 3. Os outros elementos (a antena pode se constituída por mais de três elementos) são menores que o elemento alimentado visto que eles são espaçados por distância maior que 0,25λ, tipicamente 0,37λ. Isso faz com que esses elementos se comportem de forma capacitiva, funcionando como elementos diretores do sinal. (FUSCO, 2006)

Considere agora um arranjo Yagi simples como um elemento alimentado e um segundo elemento, não-alimentado, funcionando como um refletor ou um diretor; em geral, o elemento não-alimentado é denominado elemento parasita figura 2.1. Através dessa figura, podemos ver que a tensão induzida no elemento 2 é decorrência da presença de corrente no elemento ressonante, elemento 1.

$$V_2 = -I_1 Z_{12}$$

Em que Z12 é o acoplamento mutuo entre os elementos e o sinal negativo indica que a tensão induzida no elemento 2 está em oposição de fase em relação ao elemento1.

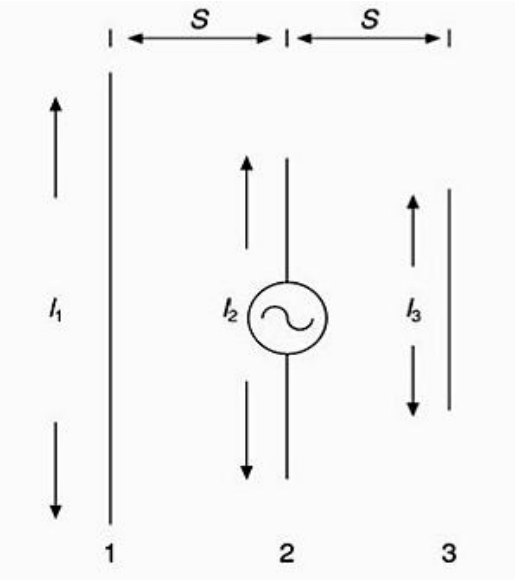


Figura 2.1 - Arranjo Yagi de três elementos / Fonte: FUSCO, 2006.

Se o elemento 2 for apropriadamente mais longo que o elemento 1, ou seja, indutivo a corrente nele (I_2) atrasa E_2 por um ângulo ϕ, portanto, ela atrasa I1 por 180° - ϕ, e assim operar de forma longitudinal.

Se o elemento 2 for apropriadamente menor que o elemento 1, ou seja, capacitivo, então I_2 adianta E_2 por um ângulo ϕ e, portanto, atrasa I_1 por um ângulo de 180° + ϕ, de forma que a direção longitudinal é invertida em relação ao caso anterior, ou seja, a máxima radiação é direcionada do elemento 1 para o elemento 2. Neste caso, o refletor parasita funciona como um diretor; no caso anterior ele funcionou como refletor. O espaçamento entre os elementos está geralmente entre 0,15 e 0,25ℓ.

A principal vantagem da configuração Yagi sobre o arranjo longitudinal é que a alimentação da configuração Yagi é muito simples. Se o elemento parasita for construído para ser reativo, então uma pequena potência é perdida nele, podendo ser fixado diretamente no suporte de metal do mastro sem a necessidade de isolação. Consequentemente, correntes muito pequenas serão induzidas ao mastro, o que simplifica a instalação necessária para esse tipo de antena.

Quanto mais elementos parasitas diretores forem introduzidos na antena, mais eles vão encurtando gradualmente à medida que se afasta do elemento alimentado, de forma que eles

tenham reatâncias maiores; portanto, a fase correta do componente parasita pode ser garantida de forma que a radiação longitudinal direta seja garantida.

Com referência na figura 2.2, se a impedância do elemento alimentador for

$$Z_d = \frac{V_1}{I_1}$$

Onde $V_1 = Z_{11}I_1 + Z_{12}I_2$, então

$$Z_d = \frac{V_1}{I_1} = Z_{11} + Z_{12}\frac{I_2}{I_1}$$

Então, como o elemento 2 é parasita,

$$V_2 = 0Z_{21}I_1 + Z_{22}I_2$$

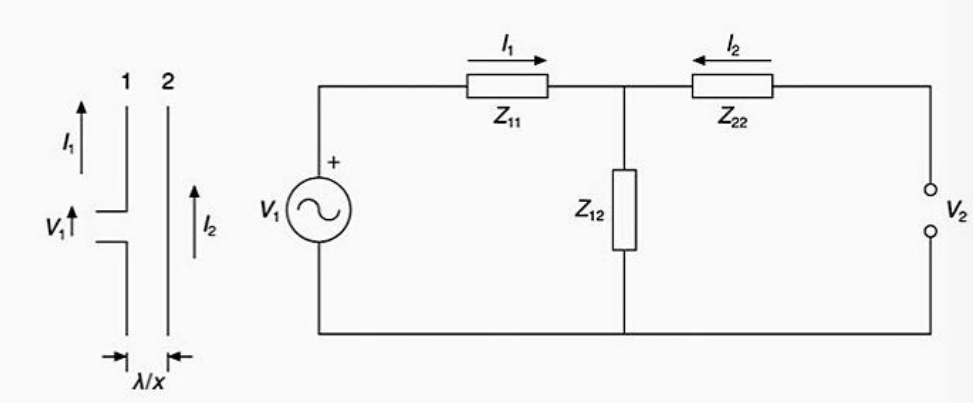


Figura 2.2 – Arranjo Yagi de dois elementos / Fonte: FUSCO, 2006.

Portanto

$$\frac{I_2}{I_1} = \frac{Z_{21}}{Z_{22}}$$

Assim,

$$\frac{V_1}{I_1} = Z_d = Z_{11} - \frac{Z_{12}Z_{21}}{Z_{22}}$$

Agora, para um circuito recíproco com impedâncias de terminação iguais

$$Z_{12} = Z_{21}$$

Portanto,

$$Z_d = Z_{11} - \frac{Z_{12}^2}{Z_{22}}$$

Esse resultado mostra que a impedância de entrada da antena Yagi é reduzida por um fator Z_{12}^2/Z_{22} relativo à impedância própria do elemento alimentado. Portanto, a resistência de radiação do elemento alimentado diminui e uma componente reativa é introduzida. A parte reativa é normalmente compensada fazendo com que o elemento alimentado seja um pouco mais longo, ou mais curto, que o comprimento ressonante, dependendo se impedância adicional é indutiva ou capacitiva.

Com uma Yagi de dois elementos, cerca de 3dB de ganho alem do dipolo de meia onda pode ser obtida com uma relação frente-costa de cerca de 12 dB. A introdução de elementos parasitas adicionais aumenta o ganho da antena Yagi (FUSCO).

A figura 2.3 ilustra um exemplo de uma antena YAGI onde é formada por um dipolo de meia onda como elemento excitador, um refletor e um ou mais diretores. Na transmissão, a interação eletromagnética entre os elementos produz múltiplas irradiações do sinal, na direção dos diretores, com significativo ganho do total irradiado. Na recepção, a malha formada pelos diretores e refletor reforça o sinal. Devido à simetria e igualdade de impedâncias, não há corrente entre elementos e um suporte condutor pode ser usado. Apenas o dipolo deve ser isolado.

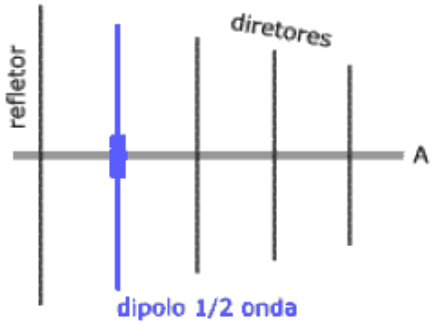


Figura 2.3 – Modelo de uma Antena com 5 elementos / Fonte: http://www.electronica-pt.com/index.php/content/view/231.

Dependendo do numero de diretores, o ganho pode ser alto. Valores típicos vão de 7 a 15 dB. Conforme já dito, é bastante direcional. A figura 2.4 ilustra uma curva aproximada da potência de irradiação. Apresenta uma largura de banda estreita o que pode ser vantajoso para algumas aplicações e limitante para outras.

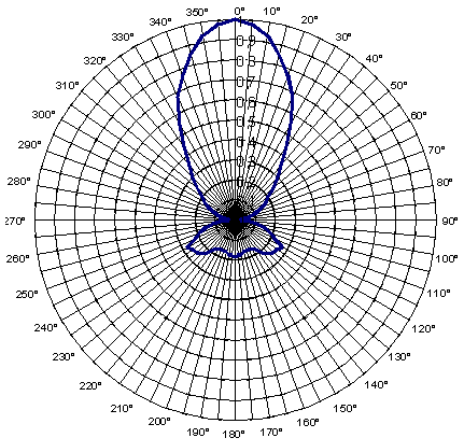


Figura 2.4 – Diagrama de Radiação da Antena Yagi / Fonte: http://www.idealantenas.com.br/ingles/produto/tv%20yagi%20UHFvhf/tvyagi.htm.

2.1.1 - Transmissão de RADIO/TV

Em nosso sistema de televisão, as informações visuais de uma cena real são convertidas em sinais elétricos que vão ao receptor.

Originalmente, a televisão foi concebida como uma outra forma de entretenimento e informação, transmitindo-se imagens por radiodifusão, assim como o rádio transmite o som. A transmissão por radiodifusão comercial ainda é o campo de aplicação mais amplo para a televisão. Entretanto, a capacidade de reproduzir figuras, textos, desenhos e informações visuais tornou-se tão importante que diversas aplicações tornaram-se comuns.

Na transmissão de sinais de RADIO/TV, o sinal em banda-base modula uma onda portadora de alta frequência para permitir a transmissão sem fio. No receptor, o detector de vídeo recupera o sinal de vídeo ou áudio original. As transmissões de vídeo e áudio são bastantes similares, exceto pelo fato da modulação de vídeo ser usada para formação de uma imagem. Todos estes sistemas requerem ondas de rádio eletromagnéticas para transmissão. Na transmissão de televisão, a modulação em amplitude (AM) é utilizada para o sinal de imagem enquanto a modulação em freqüência (FM) é usada para o sinal de som associado.

2.1.2 - Radiodifusão de Televisão

Entende-se por difusão o envio em todas as direções. A antena transmissora irradia ondas eletromagnéticas que podem ser captadas por uma antena receptora. A área coberta é de aproximadamente 150 km, em todas a direções à volta do transmissor.

A antena receptora capta tanto o sinal da portadora de som quanto imagem. Os sinais são amplificados e detectados para a recuperar-se a modulação original.

A faixa de freqüências para a transmissão dos sinais de vídeo e áudio é chamada de canal de televisão. No padrão de televisão utilizado no Brasil, a cada estação de TV corresponde uma faixa de 6 MHz de largura e uma freqüência portadora especificada pelo CCIR – Comitê Consultivo Internacional de Radiocomunicações. O quadro 2.5 mostra que todos os canais se encaixam em uma das três faixas seguintes (GROB):

1. Canais em faixa baixa de VHF.
2. Canais em faixa alta de VHF.
3. Canais de faixa ultra alta de UHF.

CANAL	LARGURA DE FAIXA	PORTADORA DE VIDEO	PORTADORA DE SOM	OXILADOR LOCAL	CANAL	LARGURA DE FAIXA	PORTADORA DE VIDEO	PORTADORA DE SOM	OXILADOR LOCAL
2	54-60	55.25	59.75	101	43	644-650	645.25	649.75	691
3	60-66	61.25	65.75	107	44	650-656	651.25	655.75	697
4	66-72	67.25	71.75	113	45	656-662	657.25	661.75	703
5	76-82	77.25	81.75	123	46	662-668	663.25	667.75	709
6	82-88	83.25	87.75	129	47	668-674	669.25	673.75	715
7	174-180	175.25	179.75	221	48	674-680	675.25	679.75	721
8	180-186	181.25	185.75	227	49	680-686	681.25	685.75	727
9	186-192	187.25	191.75	233	50	686-692	687.25	691.75	733
10	192-198	193.25	197.75	239	51	692-698	693.25	697.75	739
11	198-204	199.25	203.75	245	52	698-704	699.25	703.75	745
12	204-210	205.25	209.75	251	53	704-710	705.25	709.75	751
13	210-216	211.25	215.75	257	54	710-716	711.25	715.75	757
14	470-476	471.25	475.75	517	55	716-722	717.25	721.75	763
15	476-482	477.25	481.75	523	56	722-728	723.25	727.75	769
16	482-488	483.25	487.75	529	57	728-734	729.25	733.75	775
17	488-494	489.25	493.75	535	58	734-740	735.25	739.75	781
18	494-500	495.25	499.75	541	59	740-746	741.25	745.75	787
19	500-506	501.25	505.75	547	60	746-752	747.25	751.75	793
20	506-512	507.25	511.75	553	61	752-758	753.25	757.75	799
21	512-518	513.25	517.75	559	62	758-764	759.25	763.75	805
22	518-524	519.25	523.75	565	63	764-770	765.25	769.75	811
23	524-530	525.25	529.75	571	64	770-776	771.25	775.75	817
24	530-536	531.25	535.72	577	65	776-782	777.25	781.75	823
25	536-542	537.25	541.75	583	66	782-788	783.25	787.75	829
26	542-548	543.25	547.75	589	67	788-794	789.25	793.75	835
27	548-554	549.25	553.75	595	68	794-800	795.25	799.75	841
28	554-560	555.25	559.75	601	69	800-806	801.25	805.75	847
29	560-566	561.25	565.75	607	70	806-812	807.25	811.75	853
30	566-572	567.25	571.75	613	71	812-818	813.25	817.75	859
31	572-578	573.25	577.75	619	72	818-824	819.25	823.75	865
32	578-584	579.25	583.75	625	73	824-830	825.25	829.75	871
33	584-590	585.25	589.75	631	74	830-836	831.25	835.75	877
34	590-596	591.25	595.75	637	75	836-842	837.25	841.75	883
35	596-602	597.25	601.75	643	76	842-848	843.25	847.75	889
36	602-608	603.25	607.75	649	77	848-854	849.25	853.75	895
37	608-614	609.25	613.75	655	78	854-860	855.25	859.75	901
38	614-620	615.25	619.75	661	79	860-866	861.25	865.75	907
39	620-626	621.25	625.75	667	80	866-872	867.25	871.75	913
40	626-632	627.25	631.75	673	81	872-878	873.25	877.75	919
41	632-638	633.25	637.75	679	82	878-884	879.25	883.75	925
42	638-644	639.25	643.75	685	83	884-890	885.25	889.75	931

Quadro 2.5 – Freqüência Exata dos Canais de Televisão começando do canal 2 até 83 VHF – UHF / Fonte: http://www.teleondas.com.br/frequencias.html

Lembrando que a banda do VHF tem freqüência entre 30 e 300 MHz e a de UHF, de 300 a 3 mil MHz.

Em cada uma dessas bandas, cada canal de TV ocupa 6 MHz. As portadoras de vídeo e som são sempre separadas de 4,5 MHz em cada canal.

Nas faixas de VHF e UHF os sinais se propagam por transmissão direta entre a antena transmissora e a receptora. Os sinais irradiados não seguem a curvatura da Terra, nem existem reflexões na ionosfera, como ocorre em sinais de rádio em freqüências mais baixas. A transmissão em visão direta torna a altura da antena importante para obter uma boa recepção dos sinais de TV (GROB).

Neste protótipo será utilizado uma antena de pequeno porte, para apresentar determinados canais de televisão por um custo reduzido.

2.2 - Motor de Passo

Motores de passos são dispositivos mecânicos eletromagnéticos que podem ser controlados digitalmente através de um *hardware* específico ou através de softwares.

Motores de passos são encontrados em aparelhos onde a precisão é um fator muito importante. São usados em larga escala em impressoras, *plotters*, *scanners*, *drivers* de disquetes, discos rígidos e muitos outros aparelhos.

Existem vários modelos de motores de passos disponíveis no mercado que podem ser utilizados para diversos propósitos. Poderemos utilizá-los para mover robôs, câmeras de vídeo, brinquedos ou mesmo uma cortina (ROGERCOM).

A seguir é apresentado o funcionamento dos motores de passo nas figuras abaixo:

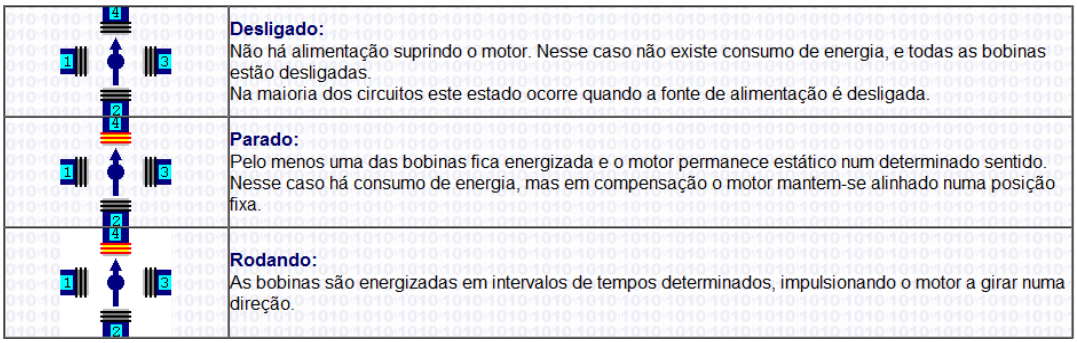


Figura 2.6 - Três estados de um motor de passo / Fonte: ROGERCOM,2008.

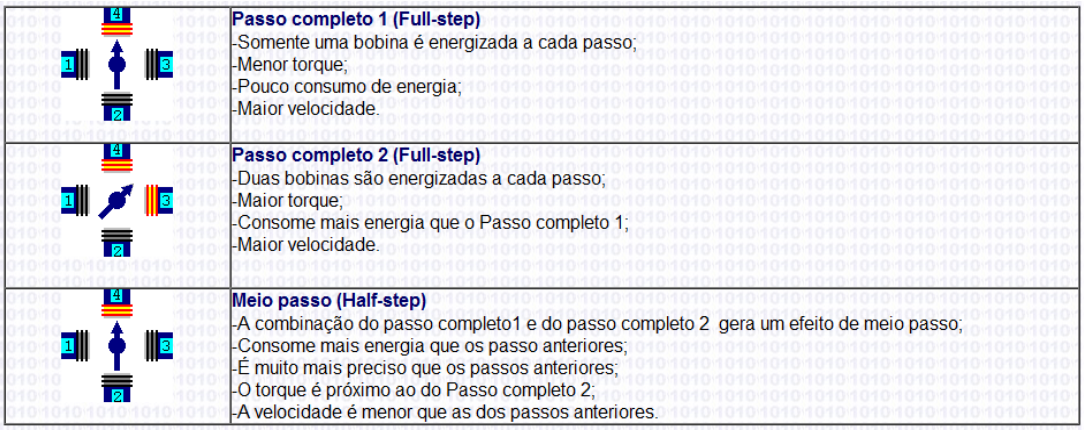


Figura 2.7 - Modos de operação de um motor de passo / Fonte: ROGERCOM, 2008.

A forma com que o motor irá operar dependerá bastante do que se deseja controlar. Há casos em que o torque é mais importante, outros a precisão ou mesmo a velocidade. Essas são características gerais dos motores de passos, a maioria deles permitem trabalhar dessa forma. Ao trabalhar com motores de passos, é necessário saber algumas características de funcionamento, tais como a tensão de alimentação, a máxima corrente elétrica suportada nas bobinas, o grau (precisão), o torque e muitos outros. As características importantes que

deveremos saber para poder controlar um motor de passo seriam a tensão de alimentação e a corrente elétrica que suas bobinas suportam (ROGERCOM).

Veja nos quadros abaixo, as seqüências corretas para se controlar um motor de passo:

Nº do passo	B3	B2	B1	B0	Decimal
1-->	1	0	0	0	8
2-->	0	1	0	0	4
3-->	0	0	1	0	2
4-->	0	0	0	1	1

Quadro 2.8 - Modos de Passo Completo(FULL-STEP) / Fonte: ROGERCOM, 2008.

Nº do passo	B3	B2	B1	B0	Decimal
1-->	1	1	0	0	12
2-->	0	1	1	0	6
3-->	0	0	1	1	3
4-->	1	0	0	1	9

Quadro 2.9 – Segundo modo de Passo Completo (FULL-STEP) / Fonte: ROGERCOM, 2008.

Nº do passo	B3	B2	B1	B0	Decimal
1-->	1	0	0	0	8
2-->	1	1	0	0	12
3-->	0	1	0	0	4
4-->	0	1	1	0	6
5-->	0	0	1	0	2
6-->	0	0	1	1	3
7-->	0	0	0	1	1
8-->	1	0	0	1	9

Quadro 2.10 – Modos de Meio Passo Completo (HALF-STEP)/ Fonte: ROGERCOM, 2008.

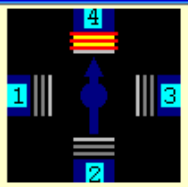
2.2.1 - Velocidade de um motor de passo

Para se controlar a velocidade de um motor de passo, envia-se uma seqüência de pulsos digitais num determinado intervalo. Quanto menor esse intervalo, maior será a velocidade em que o motor irá girar.

Não defina intervalo menor que 10ms entre cada passo, o motor perderá o torque e em vez de rodar, irá vibrar (ROGERCOM).

2.2.2 - A direção (esquerda / direita) de um motor de passo

Para mudar a direção de rotação do motor, simplesmente inverta a seqüência dos passos conforme os exemplos abaixo nos quadros 2.11 e 2.12:

Nº do passo	B3	B2	B1	B0	Decimal	Direita
1-->	1	0	0	0	8	
2-->	0	1	0	0	4	
3-->	0	0	1	0	2	
4-->	0	0	0	1	1	

Quadro 2.11 – Modos de Passo Completo à direita / Fonte: ROGERCOM, 2008.

Nº do passo	B3	B2	B1	B0	Decimal	Esquerda
1-->	0	0	0	1	1	
2-->	0	0	1	0	2	
3-->	0	1	0	0	4	
4-->	1	0	0	0	8	

Quadro 2.12 – Modos de Passo Completo à esquerda / Fonte: ROGERCOM, 2008.

2.2.3 - A precisão de um motor de passo:

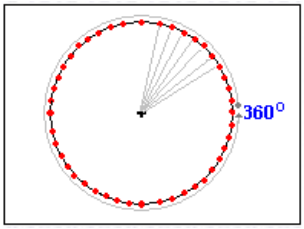


Figura 3.13 – Precisão de 7.5º / Fonte: ROGERCOM, 2008.

Na figura acima apresenta a distância entre um ponto vermelho e outro é de 7.5º.

Para sabermos quantos passos são necessários para que o motor dê um giro de 360º, faça os seguintes cálculos:

Passos por volta = 360º / 7.5º;

Passos por volta = 48.

Portanto, um motor com precisão de 7.5º, precisa dá 48 passos para completar uma volta. Um exemplo do acionamento das bobinas e ângulo é apresentado no quadro a seguir:

X	$\overline{X}$	Y	$\overline{Y}$	Step angle	X	$\overline{X}$	Y	$\overline{Y}$	Step angle
0	1	0	1	0.0°	0	1	0	1	180.0°
1	0	0	1	7.5°	1	0	0	1	187.5°
1	0	1	0	15.0°	1	0	1	0	195.0°
0	1	1	0	22.5°	0	1	1	0	202.5°
0	1	0	1	30.0°	0	1	0	1	210.0°
1	0	0	1	37.5°	1	0	0	1	217.5°
1	0	1	0	45.0°	1	0	1	0	225.0°
0	1	1	0	52.5°	0	1	1	0	232.5°
0	1	0	1	60.0°	0	1	0	1	240.0°
1	0	0	1	67.5°	1	0	0	1	247.5°
1	0	1	0	75.0°	1	0	1	0	255.0°
0	1	1	0	82.5°	0	1	1	0	262.5°
0	1	0	1	90.0°	0	1	0	1	270.0°
1	0	0	1	97.5°	1	0	0	1	277.5°
1	0	1	0	105.0°	1	0	1	0	285.0°
0	1	1	0	112.5°	0	1	1	0	292.5°
0	1	0	1	120.0°	0	1	0	1	300.0°
1	0	0	1	127.5°	1	0	0	1	307.5°
1	0	1	0	135.0°	1	0	1	0	315.0°
0	1	1	0	142.5°	0	1	1	0	322.5°
0	1	0	1	150.0°	0	1	0	1	330.0°
1	0	0	1	157.5°	1	0	0	1	337.5°
1	0	1	0	165.0°	1	0	1	0	345.0°
0	1	1	0	172.5°	0	1	1	0	352.5°

Quadro 2.14 – 48 passos de um motor de Passo / Fonte: Cleveland State University.

Lembrando que, para o acionamento do motor de passo, torna-se necessário um *hardware* específico, chamado driver. Isto pode ser feito através de driver usando transistores de potência. No entanto, a forma prática é a aquisição de componentes prontos, tais como ULN 2003 ou ULN 2803 mostrado na figura 2.15, que nada mais são que arranjos de

transistores Darlington que podem controlar correntes de até 500mA, estão em forma de circuitos integrados prontos para serem usados em interfaces que necessitem controlar motores de passos, solenóides, *relês*, motores DC e muitos outros dispositivo (ROGERCOM).

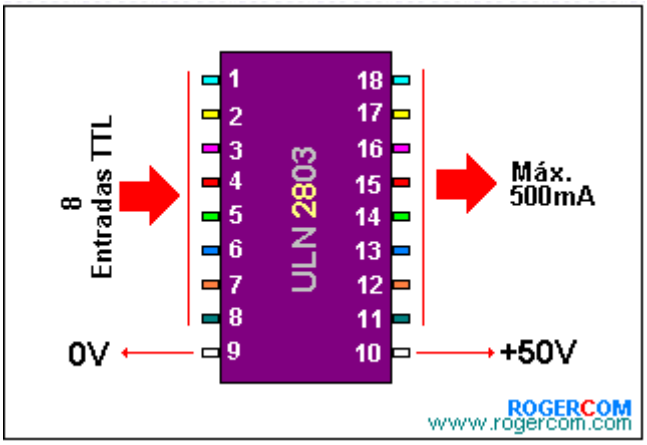


Figura 2.15 - Pinagens do ULN2803 / Fonte: ROGERCOM, 2008.

O CI ULN 2803 tem 8 entradas que podem controlar até 8 saídas. Com ele poderemos controlar até 2 motores de passo simultaneamente (ROGERCOM).

O motor de passo foi escolhido para este protótipo por ser um motor de movimentos precisos e por ser utilizado principalmente em aplicações de posicionamento. Tendo em vista que o protótipo utiliza uma antena de pequeno porte e componentes de plásticos como as engrenagens, não precisando de muito torque. O motor utilizado é um 55SPM25 fabricado pela Panasonic, e pode ser encontrado em impressoras.

2.3 - Engrenagens

Engrenagens são elementos rígidos utilizados na transmissão de movimentos rotativos entre eixos. Consistem basicamente de dois cilindros nos quais são fabricados dentes. A transmissão se dá através do contato entre os dentes. Como são elementos rígidos, a transmissão deve atender a algumas características especiais, sendo que a principal é que não haja qualquer diferença de velocidades entre pontos em contato quando da transmissão do movimento. Eventuais diferenças fariam com que houvesse perda do contato ou o travamento,

quando um dente da engrenagem motora tenta transmitir velocidade além da que outro dente da mesma engrenagem em contato transmite (SANTOS JÚNIOR).

A figura 2.16 ilustra o tipo mais comum de engrenagem, chamada de engrenagem cilíndrica de dentes retos, em inglês “spur gear”. O termo engrenagem, embora possa ser empregado para designar apenas um dos elementos, normalmente é empregado para designar a transmissão. Uma transmissão por engrenagens é composta de dois elementos ou mais.

Quando duas engrenagens estão em contato, chamamos de pinhão a menor delas e de coroa a maior. A denominação não tem relação com o fato de que um elemento é o motor e outro é o movido, mas somente com as dimensões (SANTOS JÚNIOR).

Figura 2.16 - Engrenagem Cilíndrica de Dentes Retos / Fonte: SANTOS JUNIOR, 2003.

A figura 2.17 ilustra uma transmissão por engrenagens cilíndricas de dentes retos. Trata-se apenas de um arranjo demonstrativo, mas serve para mostrar a forma como os dentes entram em contato. Quando as manivelas ao fundo giram, o elemento da direita transmite potência para o da esquerda.

Figura 2.17 – Transmissão por Engrenagens Cilíndricas de Dentes Retos / Fonte: SANTOS JUNIOR, 2003.

A expressão “transmite potência” é uma generalização para a lei de conservação de energia. Significa que um dos elementos executa trabalho sobre o outro, em uma determinada taxa. Aparentemente, toda a potência é transmitida, mas a realidade mostra que parte dela é perdida pelo deslizamento entre os dentes. Transmitir potência pode não descrever o objetivo de uma transmissão por engrenagens na maioria das aplicações de engenharia. O que se deseja é transmitir um determinado torque, ou seja, a capacidade de realizar um esforço na saída da transmissão (SANTOS JÚNIOR).

A velocidade dos motores é dada em RPM. Esta sigla quer dizer rotação por minuto. Como o nome já diz, a RPM é o número de voltas completas que um eixo, ou uma polia, ou uma engrenagem dá em um minuto.

O termo correto para indicar a grandeza medida em RPM é freqüência. Todavia, como a palavra velocidade é comumente empregada pelos profissionais da área de Mecânica, essa é a palavra que empregaremos.

A velocidade fornecida por um conjunto transmissor depende da relação entre os diâmetros das polias. Polias de diâmetros iguais transmitem para a máquina a mesma velocidade fornecida pelo motor, como ilustra figura 2.18.

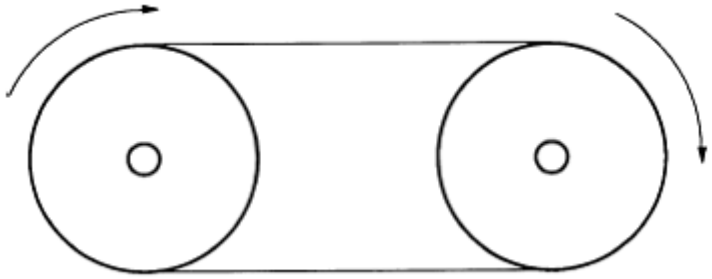


Figura 2.18 – Mesma RPM / Fonte: http://www.bibvirt.futuro.usp.br

Polias de tamanhos diferentes transmitem maior ou menor velocidade para a máquina. Se a polia motora que fornece o movimento, é maior que a movida, aquela que recebe o movimento, a velocidade transmitida para a máquina é maior. Se a polia movida é maior que a motora, a velocidade transmitida para a máquina é menor. A figura 2.19 ilustra a relação dessa diferença de velocidade.

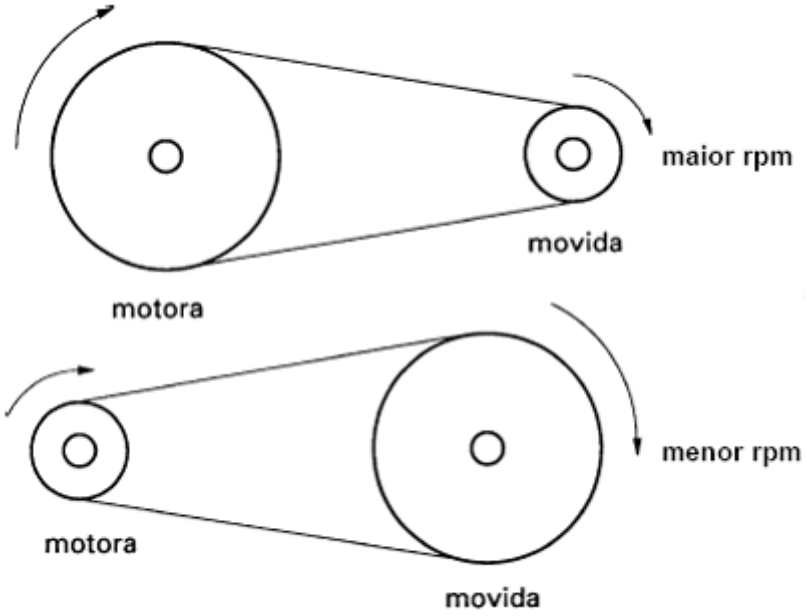


Figura 2.19 – Maior e Menor RPM / Fonte: http://www.bibvirt.futuro.usp.br

Existe uma relação matemática que expressa esse fenômeno:

$$\frac{n_1}{n_2} = \frac{D_2}{D_1}$$

Em que n1 e n2 são as rpm das polias motora e movida, respectivamente, e D2 e D1 são os diâmetros das polias movida e motora.

Da mesma forma, quando o conjunto transmissor de velocidade é composto por engrenagens, o que faz alterar a RPM é o número de dentes. É importante saber que, em engrenagens que trabalham juntas, a distância entre os dentes é sempre igual.

Desse modo, engrenagens com o mesmo número de dentes apresentam a mesma RPM, vide figura 2.20.

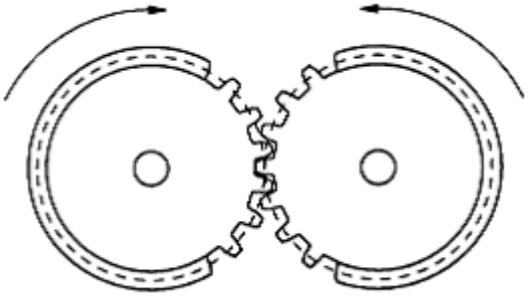


Figura 2.20 – Mesma RPM nas Engrenagens / Fonte: http://www.bibvirt.futuro.usp.br

Engrenagens com números diferentes de dentes apresentam mais ou menos RPM, dependendo da relação entre o menor ou o maior número de dentes das engrenagens motora e movida, vide figura 2.21.

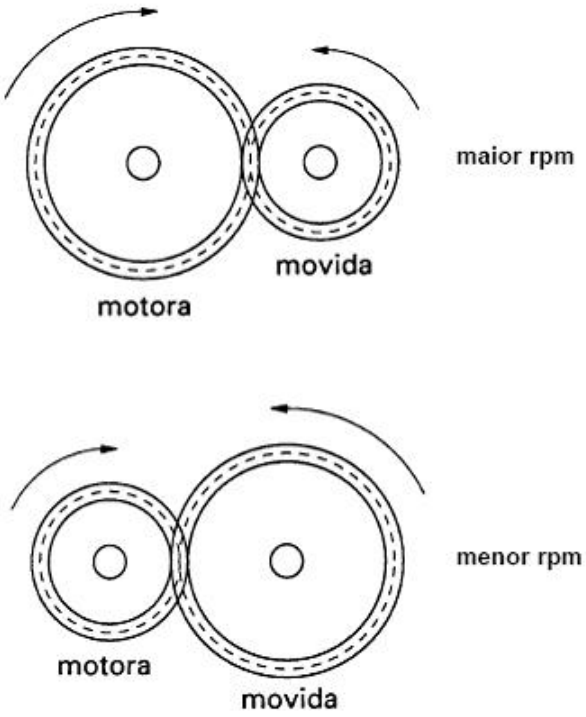


Figura 2.21 – Maior e Menor RPM de Engrenagem de Dentes Retos / Fonte: http://www.bibvirt.futuro.usp.br

Essa relação também pode ser expressa matematicamente:

$$\frac{n_1}{n_2} = \frac{Z_2}{Z_1}$$

Nessa relação, n1 e n2 são as rpm das engrenagens motora e movida, respectivamente. Z2 e Z1 são o número de dentes das engrenagens movida e motora, respectivamente.

2.4 - Comunicação Sem Fio

Atualmente o foco das redes *wireless* comerciais se encontra no contexto das redes locais WLAN's, tanto em soluções proprietárias como nos padrões desenvolvidos pelo IEEE,

por exemplo. Com a evolução natural das tecnologias das redes sem fio, estas passaram a atender não só as aplicações corporativas mais sofisticadas como também aquelas envolvendo pequenos volumes de dados que exigem baixas taxas de transmissão como, por exemplo, o controle de equipamentos eletroeletrônicos. Além disso, outras tecnologias sem fio têm sido utilizadas também com o objetivo de proporcionar a comunicação pessoal e o controle de dispositivos diversos, são as chamadas redes pessoais WPAN's (PINHEIRO).

Basicamente, essas tecnologias têm o propósito de permitir o controle remoto de equipamentos domésticos (TV's, videocassetes, geladeiras, etc) e periféricos (teclados, mouse, impressoras, etc), eliminando os cabos e tornando mais prática a operação desses equipamentos pelos usuários.

2.4.1 - ZIGBEE

Uma das tecnologias mais recentes de redes para aplicações pessoais e que permite o gerenciamento e controle desses dispositivos é o padrão ZigBee, também conhecido como HomeRF Lite e que corresponde ao IEEE 802.15.4, homologado em maio de 2003.

O padrão ZigBee foi desenvolvido para se tornar uma alternativa de comunicação em redes que não necessitem de soluções mais complexas para seu controle, barateando assim os custos com a aquisição, instalação de equipamentos, manutenção e mão de obra. Trata-se de uma tecnologia relativamente simples, que utiliza um protocolo de pacotes de dados com características específicas, sendo projetado para oferecer flexibilidade quanto aos tipos de dispositivos que pode controlar.

Os dispositivos baseados na tecnologia ZigBee operam na faixa ISM que não requer licença para funcionamento, incluindo as faixas de 2,4GHz (Global), 915Mhz (América) e 868Mhz (Europa) e com taxas de transferência de dados de 250kbps em 2,4GHz, 40kbps em 915Mhz e 20kbps em 868Mhz (PINHEIRO).

2.4.2 - Características do Padrão ZIGBEE

O padrão ZigBee (IEEE 802.15.4) foi projetado objetivando apresentar algumas características. O consumo de potência baixo e implementação simples, com interfaces de baixo custo. Possui dois estados principais de funcionamento: "active" para transmissão e

recepção e "sleep", quando não está transmitindo. A simplicidade de configuração e redundância de dispositivos (operação segura), também é uma característica do padrão. Assim como a densidade elevada dos nós por a rede. As camadas PHY e MAC permitem que as redes funcionem com grande número de dispositivos ativos. Este atributo é crítico para aplicações com sensores e redes de controle.

2.4.3 - Camadas de Protocolos

A publicação do padrão IEEE 802.15.4, definiu interfaces com baixas taxas de transmissão (menores que 250Kbps) e estabeleceu uma estrutura de rede que incorpora os conceitos de redes ad hoc, características de conexão em malha e em multi-hop (múltiplos saltos). Adicionalmente, novos algoritmos de segurança e perfis de aplicação foram definidos objetivando garantir a segurança e a perfeita interação entre os diversos equipamentos. Na Figura 2.22, temos as camadas de protocolos do Zigbee (PINHEIRO).

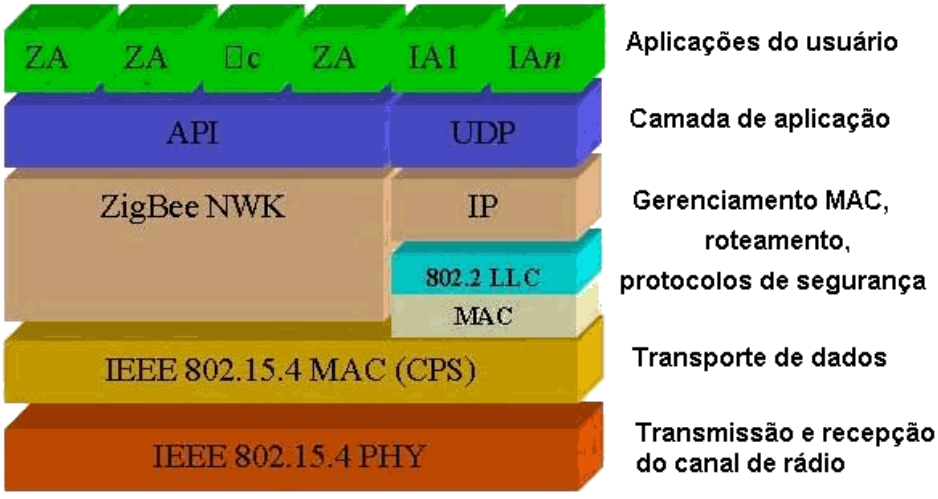


Figura 2.22 - Camadas de protocolos ZigBee / Fonte: PINHEIRO, 2006.

Neste projeto é utilizado a tecnologia ZigBee. Os dispositivos CON-USBBEE e o RCON-HOMEBEE são os principais componentes desse conjunto. Pois, a tecnologia ZigBee é utilizada para a comunicação entre essas duas placas, via rádio freqüência. O CON-USBBEE enviará através de um software de gerenciamento um comando para a placa RCON-

HOMEBE, para o acionamento dos reles, sendo que o computador é responsável pelo controle e do sistema.

2.5 – Linguagens de Programação

2.5.1 – Conceito

Uma linguagem de programação é uma ferramenta utilizada pelo profissional de computação para escrever programas, isto é, conjunto de instruções a serem seguidas pelo computador para realizar um determinado processo.

Em virtude das limitações físicas dos computadores e da pouca maturidade da ciência da computação na época de surgimento dos primeiros computadores, eles só podiam ser programados por meio de linguagens de programação muito simples. Tais linguagens disponibilizavam um pequeno conjunto de tipos de instruções capazes de realizar ações muito elementares e se caracterizam por serem de uso exclusivo de um computador especifico. Por conta disso, hoje elas são conhecidas como linguagens de maquina ou baixo nível.

À medida que a computação avançava e se vislumbrava o potencial dessa nova ferramenta, as aplicações iam se tornando cada vez mais complexas. Nessa época, foi constatado que o uso de linguagens tão simples e específicas reduzia significativamente a produtividade dos programadores e impedia a ampla disseminação dos computadores. Para contornar esse problema, surgiram as linguagens de programação de alto nível. Em contraste com as linguagens de maquina, essas linguagens se caracterizam por não serem especificas apenas um computador e por terem um conjunto mais amplo e expressivo de tipos de instrução (VAREJÂO).

2.5.2 - Linguagem Java

Quando a linguagem de programação Java foi lançada publicamente pela primeira vez, em novembro de 1995, era algo que as pessoas nunca tinham visto antes. Pois era uma linguagem que podia ser executada em página da World Wide Web, destacando entre suas correlatas para imagens gráficas, texto, áudio e outros sinais ainda em criação (LEMAY).

O Java é uma linguagem de programação muito conveniente para desenvolvimento de software que funcione em conjunto com a internet. Ela também é uma linguagem de programação orientada a objetos que utiliza metodologia que esta tornando cada vez mais útil no mundo do design de software. Além disso ela é uma linguagem multiplataforma, o que significa que seus programas podem ser criados para executar do mesmo modo no Microsoftware Windows, Apple Macintosh e na maioria das versões UNIX, incluindo a Solaris. A linguagem Java vai além da área de trabalho, sendo executada em dispositivos como televisões, relógios de pulso e telefones celulares. O Java-Station, o computador de rede Sun, executa o sistema operacional Java OS e é otimizado para a linguagem (LEMAY).

A linguagem Java está mais próxima das linguagens de programação populares, como C, C++, Visual Basic e Delphi, do que de uma linguagem de descrição de página, como HTML, ou de uma simples linguagem de script (LEMAY).

2.5.3 - Linguagem C

Em 1972 nasceu a linguagem C, criada por Dennis Ritchie da Bell Laboratories, e consiste, na realidade, em uma linguagem de nível intermediário entre o Assembly e as linguagens de alto nível (PEREIRA).

Até o desenvolvimento do C, não existiam linguagens de programação de alto nível adequadas à tarefa de criação de sistemas operacionais e outros softwares de baixo nível, restando aos desenvolvedores utilizar o Assembly para execução dessas tarefas.

Foi principalmente a partir das necessidades de reescrita do sistema operacional UNIX que surgiu a linguagem C (PEREIRA).

De fato, a implementação da linguagem é tão poderosa que C foi a escolhida para o desenvolvimento de outros sistemas operacionais além do UNIX, como WINDOWS e LINUX (PEREIRA).

2.5.4 - Programação de PIC em C

A utilização do C para a programação de microcontroladores PIC é uma escolha realmente natural.

Atualmente, a maioria dos microcontroladores disponíveis no mercado contam com compiladores de linguagem C para desenvolvimento de software.

O uso do C permite a construção de programas e aplicações muito mais complexas do que seria viável utilizando apenas o *Assembly*.

Além disso, o desenvolvimento em C permite uma grande velocidade na criação de novos projetos, devido a facilidade da programação oferecidas pela linguagem e também a sua portabilidade, o que permite adaptar programas de um sistema operacional para outro com um mínimo esforço.

Outro aspecto favorável da utilização da linguagem C é sua eficiência.

Eficiência no jargão dos compiladores é a medida de grau de inteligência com que o compilador traduz um programa em C para o código de maquina. Quanto menor e mais rápido o código gerado, maior será a eficiência da linguagem e do compilador.

Repare que o aspecto eficiência é muito importante quando tratamos de microcontroladores cujos recursos são tão limitados como nos PICs, afinal quando dispomos de apenas 512 palavra de memória de programa é 25 bytes de RAM, é imprescindível que se economize memória.

Alem disso, a utilização de uma linguagem de alto nível como C permite que o programador preocupe-se mais com a programação da aplicação em si, já que o compilador assume para si tarefas como o controle e localização das variáveis, operações matemáticas e lógicas, verificação de bancos de memória (PEREIRA).

No projeto proposto será utilizado duas linguagens de programação: a linguagem Java e a linguagem C. Essa utilização deve-se ao fato possui um melhor conhecimento das ferramentas, comparada a outras do mercado, e de ambas serem de fácil acesso.

CAPÍTULO 3 – AUTOMAÇÃO DE POSICIONAMENTO DA ANTENA

Este capítulo apresenta o desenvolvimento do projeto e sua implementação. O item 3.1 trata do desenvolvimento do projeto. O item 3.2 é referente as tecnologias utilizadas como o hardware e software. O item 3.3 representa o desenvolvimento da aplicação. E por último o item 3.4 onde será apresentado a estimativa dos custos.

3.1 – Desenvolvimento do Projeto

O projeto iniciou-se após a finalização das pesquisas de ferramentas necessárias e viáveis para o seu desenvolvimento. E por fim, foram definidas a utilização de uma antena de TV modelo YAGI de pequeno porte, um motor de passo modelo 55SPM25 da Panasonic por possuir facilidade de controle, um conjunto de engrenagens e o equipamento de comunicação sem fio ZIGBEE.

A automatização do sistema será através da integração da interface com a placa controladora do motor através do kit de comunicação sem fio ZIGBEE.

Esse kit realiza a ponte entre a interface e a placa de controle do motor, por possuir um meio de comunicação sem fio eficaz.

Os comandos determinados pela interface realizara a rotação da antena, de acordo com a escolha do usuário. Para uma melhor visualização do projeto, a Figura 3.1 ilustra o processo de todo o projeto.

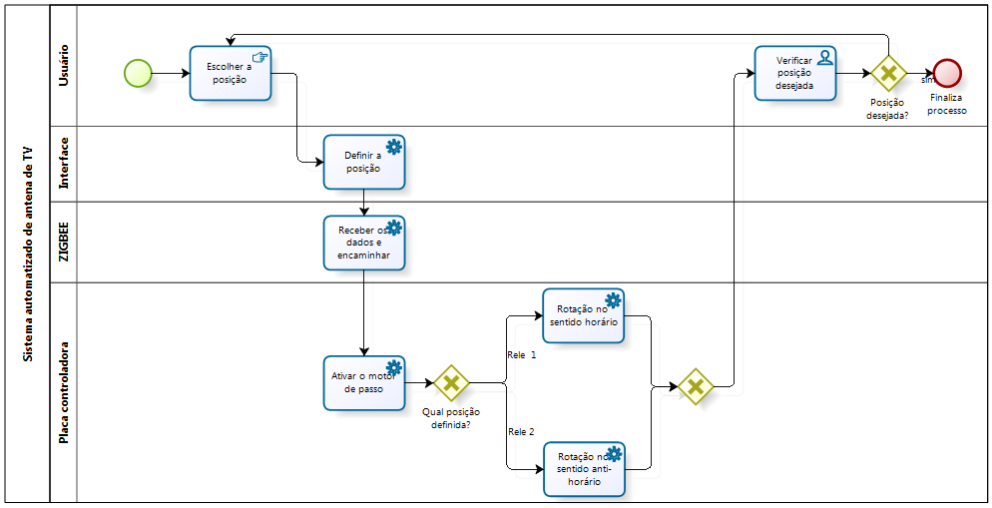


Figura 3.1 - Processo de controle da antena / Fonte: Autor

3.2 - Tecnologias Utilizadas

3.2.1 – Softwares

Os programas de computador necessários para o desenvolvimento e execução da aplicação são:

- Java SE *Runtime Environment* 6: Embora os recursos disponíveis na versão 6 do Java não sejam utilizados, optou-se por essa versão por ser a mais recente disponível no mercado. A versão 3 do Java atende perfeitamente a solução;
- IDE Eclipse: Ferramenta para desenvolvimento e compilação de aplicativos Java. Possui ferramentas e muitos recursos essenciais;
- Proteus: Ferramenta de simulação de circuitos eletrônicos. Possui a ISIS e ARES para a simulação de circuito e desenho da placa, assim economizando gastos.
- PIC C COMPILER: Do fabricante CSS onde faz a geração dos códigos em hexadecimal.
- Bizagi Process Modeler: Ferramenta de modelagem de processos.

3.2.2 - Hardware

As ferramentas de hardware necessárias para o desenvolvimento e execução da aplicação são:

3.2.2.1 - Placa CON-USBEE

A ROGERCOM desenvolveu a CON-USBBEE, com facilidade de conexão estilo PENDRIVE, para facilitar a conexão do módulo base XBEE/XBEE-PRO ao computador, seja para atualização de firmware ou mesmo para fazer coleta de dados ou controle. A Figura 3.2 ilustra a placa CON-USBBEE (ROGERCOM).

Figura 3.2 – Placa CON-USBBEE / Fonte: ROGERCOM, 2008.

A placa CON-USBEE aceita tanto o módulo XBee™ como o XBee-Pro™, como são totalmente compatíveis, Redes ZigBee podem ser construídas com ambos os módulos, simultaneamente (ROGERCOM).

A placa CON-USBBE usa um chip conversor USB/Serial; regulador de tensão LDO (baixa queda de tensão), comparador de tensão conectado aos LEDs (RSSI) que simulam a força do sinal de RF; LEDs indicadores de TX, RX, módulo ligado (ASS), e um micro-botão para "resetar" o módulo XBee/XBee-Pro™, como é mostrado na Figura 3.3 (ROGERCOM).

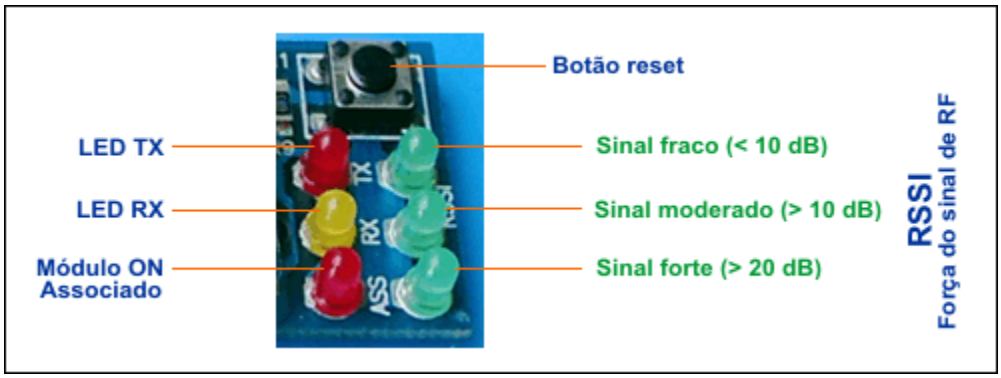


Figura 3.3 - Botão Reset e LEDs indicadores da placa CON-USBBEE / Fonte: ROGERCOM, 2008.

Característica

Abaixo seguem as principais características do CON-USBBEE:

A) Converte a interface Serial 3.3v do módulo XBee/XBee-Pro para USB;

B) Não precisa fonte de alimentação, a corrente é fornecida pelo próprio Bus USB;

C) Tem a mesma facilidade de conexão que um Pendrive;

D) LEDs indicadores de transmissão (TX), recepção (RX), Ligado e sinal RSSI;

E) Freqüência de operação: ISM 2.4 GHz;

F) Taxa de dados de RF: 250.000 bps;

G) Taxa de dados da Interface (Data Rate): 115.200 bps;

H) Alcance em áreas internas ou urbanas: 90m/100m;

I) Alcance em linha de visão (em campo aberto): 1,6Km;

J) Encriptação de 128-bit AES;

K) Comanda todos os módulos remotos XBee/XBee-Pro que estejam na Rede;

L) Troca de dados entre PCs e laptops;

M) Ideal para automação residencial, industrial, etc;

3.2.2.2 - Placa RCON-HOMEBEE

A função da placa HOMEBEE é a de automatizar determinados ambientes numa residência, escritório ou empresa. Ela pode trabalhar com ou sem fio para se comunicar com um computador ou outro dispositivo como CLP, microcontrolador, etc. Com fio opcionalmente a comunicação pode ser feita via RS232/TTL ou a partir de um conversor USB/Serial. Usando transmissão serial ou ZigBee no modo transparente, o controle da placa HOMEBEE é feita através do envio de 2 bytes, sendo o primeiro o identificador e o segundo o comando. Sem fio, a comunicação se faz através do protocolo ZigBee/IEEE 802.15.4, usando dois módulos transceivers Xbee ou Xbee-Pro. É possível usar encriptação AES de 128 bits, endereçamento de 16 bits, definir numero do canal e rede, entre outras possibilidades (ROGERCOM).

A Placa HOMEBEE possui duas saídas a relês, que podem ser usadas para ligar ou desligar dispositivos com tensão até 220v e corrente de 10A; duas saídas TTL 5v, duas entradas digitais para conectar interruptores ou sensores de contato seco (ROGERCOM).

No PC, através da criação de um software especifico pelo usuário, é possível gerenciar uma rede de placas HOMEBEE controladas por uma única placa CON-USBBEE (ROGERCOM).

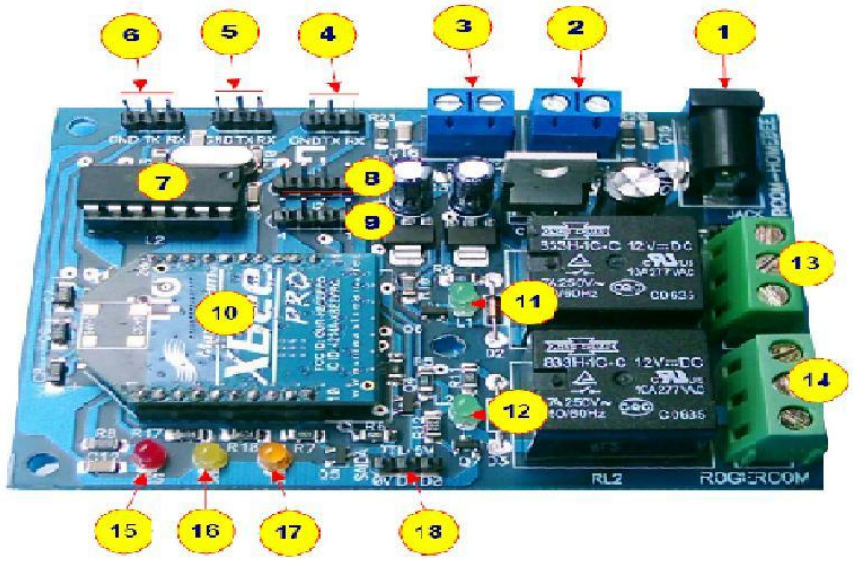


Figura 3.4 – Placa RCON-HOMEBEE / Fonte: ROGERCOM, 2008.

A Figura 3.4 ilustra a placa RCON-HOMEBEE com a identificação de seus componentes, que são:

1 – Alimentação:

A placa HOMEBEE deve ser alimentada por uma fonte externa capaz de fornecer entre 12-24v/600mA (ROGERCOM).

2 e 3 – Entradas Digitais E1 e E2:

As entradas Digitiais E1 e E2, como ilustradas na Figura 3.5, são entradas de contato seco. Um simples curto entre seus terminais, gera um pulso que é detectado pela placa, e enviado pela Serial ou via módulo XBee/XBee-Pro, conforme as configurações dos Jumps. Através de um software de configuração da placa é possível associar as entradas E1 e E2 aos relês R1 e R2 (ROGERCOM).

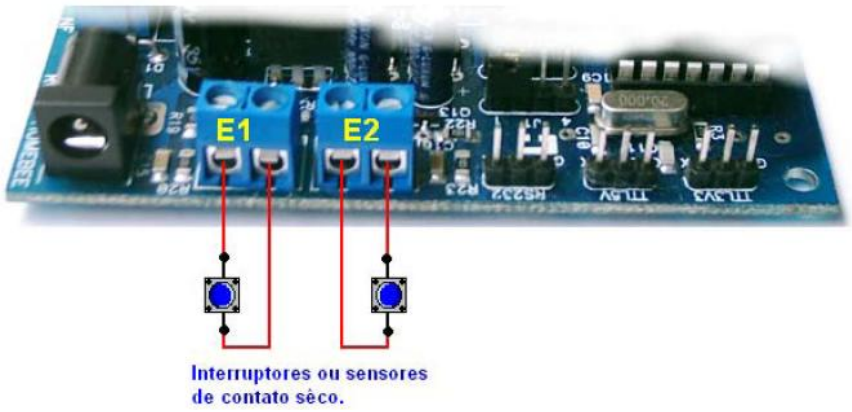


Figura 3.5 – Entradas E1 e E2 / Fonte: ROGERCOM, 2008.

4,5 e 6 – Interfaces Seriais RS232, TTL5v e TTL3v3:

A placa HOMEBEE dispõe de três opções para comunicação serial via cabo (RS232, TTL5v, TTL3v3), como pode-se ver na Figura 3.6 Após a escolha através dos jumps, somente um canal estará disponível (ROGERCOM).

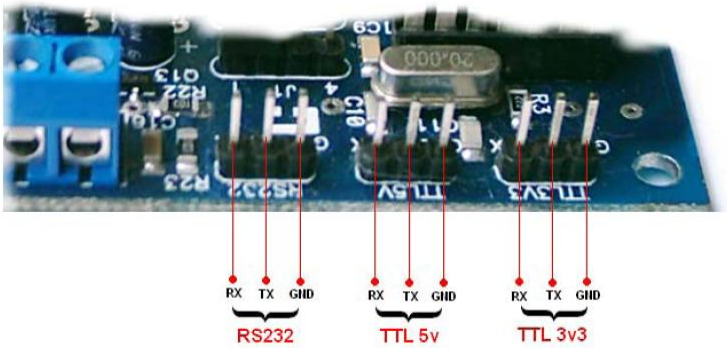


Figura 3.6 – Interfaces Seriais / Fonte: ROGERCOM, 2008.

7 – Microcontrolador;

Controla todas as funções da placa HOMEBEE. O microcontrolador usado é o PIC16F688 com tecnologia nanowatt alimentado com 3v (ROGERCOM).

8 e 9 – Configuração de Jumps:

A Figura 3.7 ilustra os Jumps, J1 e J2, que são configurados seguindo as informações citadas a seguir (ROGERCOM).

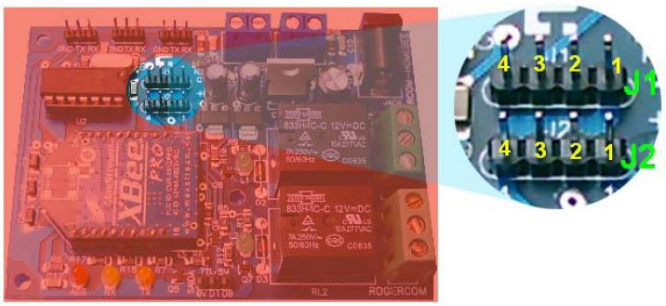


Figura 3.7 – Jumps, J1 e J2 / Fonte: ROGERCOM, 2008.

Configuração 1: XBee/XBee-Pro ←→ PIC

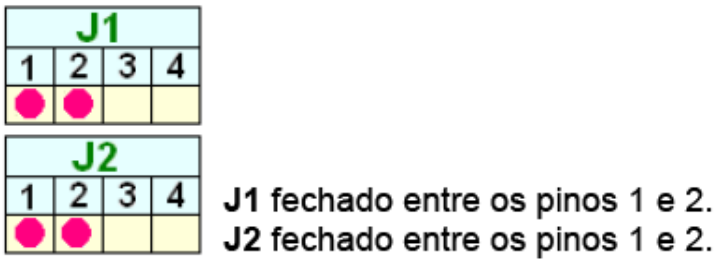


Figura 3.8 – Configuração de Jumps / Fonte: ROGERCOM, 2008.

Configuração 2: PIC ←→ RS232, TTL5v, TTL3v3

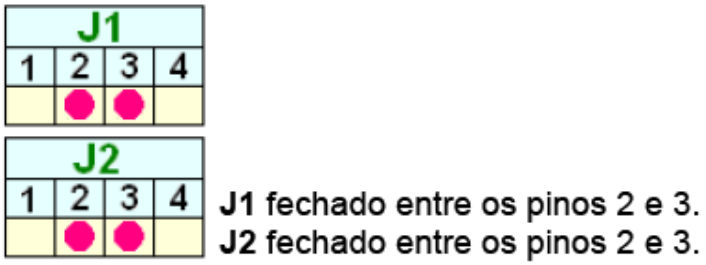


Figura 3.9 – Configuração de Jumps/ Fonte: ROGERCOM, 2008.

Configuração 3: XBee/XBee-Pro←→ RS232, TTL5v, TTL3v3

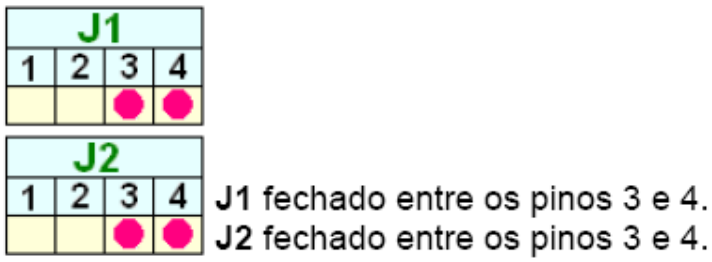


Figura 3.10 – Configuração de Jumps/ Fonte: ROGERCOM, 2008.

10 – Conector para modulo XBee/XBee-Pro:

Para que a placa HOMEBEE estabeleça comunicação sem fio, é necessário incluir um módulo Xbee ou XBee-Pro (ROGERCOM).

11 e 12 – LED's verdes L1 e L2:

Quando estão ligados indicam que o relê 1 e/ou 2 estão ligados. Quando apagados os relês 1 e/ou 2 estão desligados (ROGERCOM).

13 e 14 – Saidas a Relês (R1 e R2):

Através das saídas a relês, mostradas na Figura 3.11, é possível ligas/desligar dispositivos conectados a rede elétrica 110 ou 220v, ou mesmo aqueles alimentados com corrente continua (ROGERCOM).

NA – Interruptor **N**ormalmente **A**berto;
C – Comum
NF – Interruptor **N**ormalmente **F**echado.

Figura 15 – Conectores das saídas dos Relês

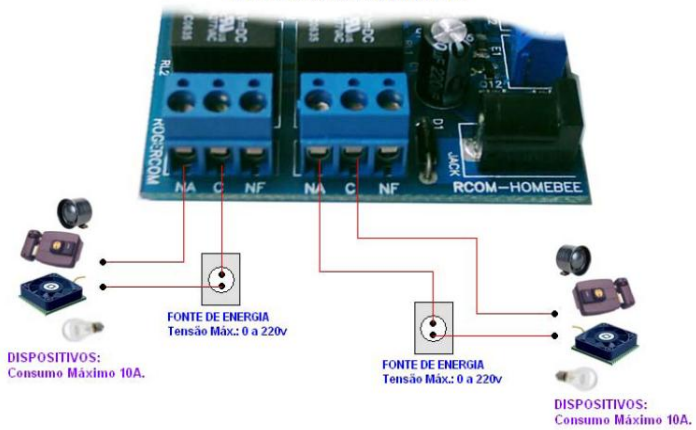


Figura 3.11 – Saída a Relês / Fonte: ROGERCOM, 2008.

15 – LED vermelho (Ass):

Quando aceso/piscando, indica que o modulo XBee/XBee-Pro da placa está ligado/operando (ROGERCOM).

Quando aceso sem piscar, indica que ele está aguardando.

16 – LED laranja (TX):

Quando aceso/piscando, indica que o modulo XBee/XBee-Pro da placa está transmitindo dados. (ROGERCOM)

17 – LED amarelo (RX):

Quando aceso/piscando, indica que o modulo XBee/XBee-Pro da placa está recebendo dados (ROGERCOM).

18 – Saidas TTL digitais (D0 e D1):

São saídas TTL 5v, que pode ser usadas para controlar um driver de relês externo, ou mesmo, enviar sinais para um microcontrolador. Veja a Figura 3.12 (ROGERCOM).

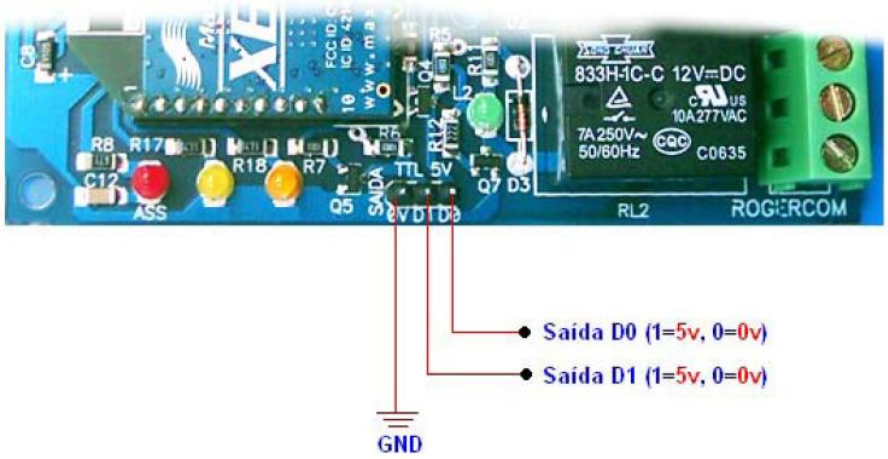


Figura 3.12 – Saídas TTL/ Fonte: ROGERCOM, 2008.

Características da placa RCON-HOMEBEE:

- Compatível com módulos XBee e XBee-Pro ZB ou IEEE 802.15.4. Placa para ligar/desligar lâmpadas, aparelhos eletro-eletrônicos, fechaduras elétricas, irrigação de jardins, abrir/fechar portas, portões, cancelas, etc;
- Dimensões: 9,0 cm x 6,5 cm.
- Características:

  • Segurança com encriptação de 128-bit AES (nos XBee/XBee-Pro ZB ou IEEE 802.15.4);

  • 2 Saídas tipo contato seco a Relês 110/220v / 10A;

  • 2 Entradas digitais contato seco;

  • 2 Saídas TTL 5v;

  •Fonte de alimentação 12v/500mA (não inclusa);

  • Interface serial opcional:

  - **RS232** (TX, RX, GND);

  - **TTL 5v** (TX, RX, GND);

  - **3,3v** (TX, RX, GND);

  (ROGERCOM).

3.3 – Desenvolvimento da Aplicação

3.3.1 – Software de Controle da Antena

Para a comunicação entre os dispositivos foi criado um *software* de controle. Onde toda informação enviada ou recebida pela placa é em forma de pacotes de bits.

O software foi desenvolvido na linguagem Java, cuja plataforma de desenvolvimento utilizada foi o IDE Eclipse. Mas para que o desenvolvimento do software funcione perfeitamente, são necessárias algumas configurações de ambiente, como fazer o download da API RxTxCOMM, e seguir o padrão de configuração das propriedades da API no sistema operacional, no qual possui todas as classes necessárias para que a comunicação serial ocorra.

A tela que foi desenvolvida é muito fácil de ser utilizada. Para isso, ela possuirá algumas funções como escolher em qual porta deseja conectar, e também abrir uma conexão ou fechar uma conexão, rotações nos sentidos horário e anti-horário, desligar qualquer rotação e limpar o campo de status. A figura 3.13 ilustra o software de controle da antena.

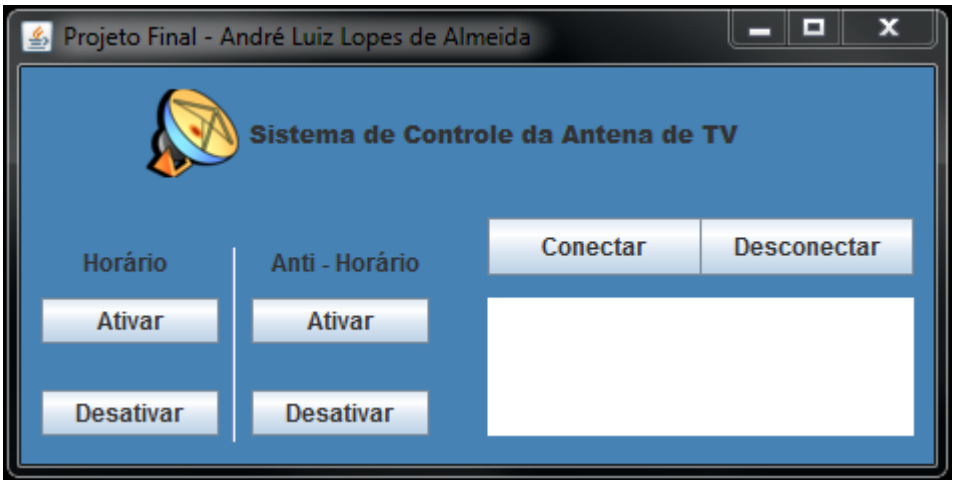


Figura 3.13 – Software de Controle da Antena / Fonte: Autor

Para reconhecer e conectar a porta, serão utilizadas as seguintes linhas de código exibidas na Figura 3.14.

```
public void ObterIdDaPorta() {
    try {
        cp = CommPortIdentifier.getPortIdentifier(Porta);
        if (cp == null) {
            System.out.println("Erro na porta");
            IDPortaOK = false;
            System.exit(1);
        }
        IDPortaOK = true;
    } catch (Exception e) {
        System.out.println("Erro obtendo ID da porta: " + e);
        IDPortaOK = false;
        System.exit(1);
    }
}

public void AbrirPorta() {
    try {
        porta = (SerialPort) cp.open("SerialComLeitura", timeout);
        PortaOK = true;
        // configurar parÃ¢metros
        porta.setSerialPortParams(baudrate, porta.DATABITS_8,
                porta.STOPBITS_1, porta.PARITY_NONE);
        porta.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);
    } catch (Exception e) {
        PortaOK = false;
        System.out.println("Erro abrindo comunicação: " + e);
        System.exit(1);

    }

}
```

Figura 3.14 – Conexão com Porta COMx / Fonte: Autor

Para ativar as rotações tanto no sentido horário como no anti-horário, separadamente, e depois desligá-los um por vez, foram definidas utilizando as linhas de código exibidas nas Figuras 3.15 e 3.16. No comando foi utilizada a função “Write”.

```
public void OnRele1() {

    byte[] a = new byte[11];
     a[0] = 0x7E;
     a[1] = 0x00;
     a[2] = 0x07;
     a[3] = 0x01;
     a[4] = 0x01;
     a[5] = 0x50;
     a[6] = 0x01;
     a[7] = 0x00;
     a[8] = 0x7B;
     a[9] = 0x01;
     a[10] = 0x30;

    if (Escrita == true) {
        try {
            saida = porta.getOutputStream();
            System.out.println("FLUXO OK!");
        } catch (Exception e) {
            System.out.println("Erro.STATUS: " + e);
        }
        try {
            System.out.println("Enviando um byte para " + Porta);
            System.out.println("Enviando : " + a);
            saida.write(a);
            Thread.sleep(10);
            saida.flush();
        } catch (Exception e) {
            System.out.println("Houve um erro durante o envio. ");
            System.out.println("STATUS: " + e);
            System.exit(1);
        }
    } else {
        System.exit(1);
    }

}
```

Figuras 3.15 – Mostra o método que aciona a rotação do sentido horário / Fonte: Autor

```
public void OffRele1() {

     byte[] b = new byte[11];
      b[0] = 0x7E;
      b[1] = 0x00;
      b[2] = 0x07;
      b[3] = 0x01;
      b[4] = 0x01;
      b[5] = 0x50;
      b[6] = 0x01;
      b[7] = 0x00;
      b[8] = 0x7B;
      b[9] = 0x00;
      b[10] = 0x30;


     if (Escrita == true) {
         try {
             saida = porta.getOutputStream();
             System.out.println("FLUXO OK!");
         } catch (Exception e) {
             System.out.println("Erro.STATUS: " + e);
         }
         try {
             System.out.println("Enviando um byte para " + Porta);
             System.out.println("Enviando : " + b);
             saida.write(b);
             Thread.sleep(10);
             saida.flush();
         } catch (Exception e) {
             System.out.println("Houve um erro durante o envio. ");
             System.out.println("STATUS: " + e);
             System.exit(1);
         }
     } else {
         System.exit(1);
     }

 }
```

Figuras 3.16 – Mostra o método que desativa a rotação do sentido horário / Fonte: Autor

Esses métodos apresentados acima,explica como a interface se comunica com a placa RCON-HOMEBEE para acionar e desligar a rotação no sentido horário. Para acionar o sentido anti-horário os métodos são parecidos mudando apenas o final do conjunto de bytes, por esta razão não foi apresentado.

3.3.2 – Kit ZiGBEE

Para que a comunicação sem fio do projeto ocorra, será utilizado o padrão ZigBee, onde será utilizado dois módulos XBee-Pro. Sendo que um será para placa CON-USBBEE e outro para a placa RCON-HOMEBEE.

A placa CON-USBBEE tem a função de comunicar um dos módulos XBee-Pro com o computador através de uma porta USB onde a mesma possui um chip conversor USB/Serial. Através dessa placa que serão enviados os comandos para os relês.

A placa RCON-HOMEBEE receberá os comandos, enviados pelo software de gerenciamento, através do módulo XBee-Pro. Esses comandos acionaram os Relês 1 e/ou 2 para ligar/desligar. E assim aplicará o sentido da rotação da antena de TV.

Toda informação enviada ou recebida pela placa RCON-HOMEBEE é em forma de pacotes. A Figura 3.17 ilustra o mapa de bits do byte de controle de escrita na placa.

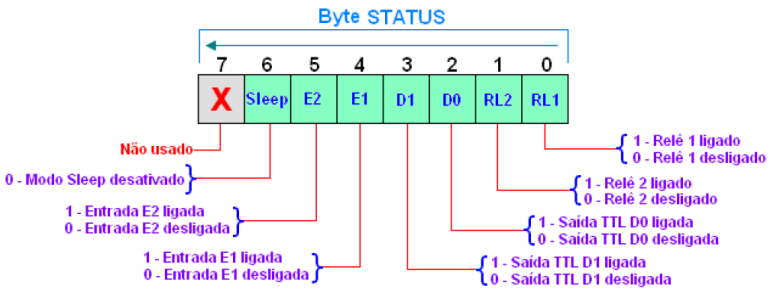


Figura 3.17 – Mapa de bits / ROGERCOM, 2008.

O Quadro 3.18 mostra o formato de um pacote XBee-pro para enviar dados de controle para a placa HOMEBEE.

Bytes	Descrição	
1	Delimitador Inicial.	7E
2	Tamanho dos bytes.	00 07
1	Identificador da API.	01
1	API Frame ID.	01
2	Parte baixa do endereço destino (DL).	50 01
1	Byte de opção.	7B
2	Pacote de dados.	01
1	Checksum.	30

Quadro 3.18 – Pacote de dados para envio de comando / ROGERCOM, 2008.

Exemplo de um pacote para ligar o relê 1;

7E 00 07 01 01 50 01 00 7B 01 30

3.3.3 – Placa Controladora

A placa controladora é uma peça fundamental no funcionamento do protótipo, pois a mesma possui um microcontrolador que receberá informações em sua porta através da saída a relê da placa RCON-HOMEBEE, onde irá definir o sentido da rotação do motor de passo. Isso quer dizer que, quando o primeiro *relê* for acionado a placa controladora enviará comandos ao circuito integrado que acionará a rotação do motor de passo no sentido horário. Ou quando o segundo *relê* for acionado, a placa controladora também enviará comandos ao circuito integrado que acionará a rotação do motor de passo no sentido anti-horário. Até que ocorra a desativação dos relês, isso fará que o motor desative sua rotação.

Para o desenvolvimento do circuito da placa controladora, foi utilizado o software Proteus 7.2. Esta ferramenta robusta e confiável permite simular vários tipos de circuitos eletrônicos através do computador. Com isso, economiza-se tempo e dinheiro, já que com os circuitos testados no software não há a possibilidade de queimar componentes e é construído com mais praticidade. Com este software é possível que sejam simulados os microcontroladores, ou seja, após a compilação do algoritmo é possível implementar o

arquivo “.HEX” no microcontrolador e testar as funcionalidades do circuito por completo. A figura 3.20 ilustra como o proteus pode facilitar o desenho do circuito.

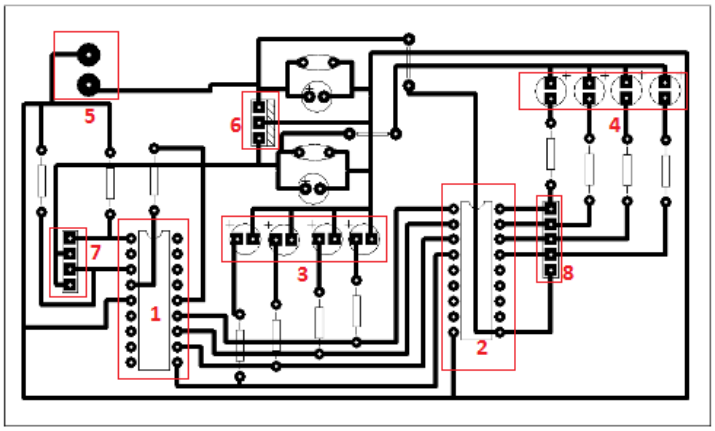


Figura 3.19 – Desenho do circuito da placa controladora produzida no ARES - Proteus/ Fonte: Autor

Na Figura 3.19 acima demonstra a placa controladora com a identificação de seus componentes, que poderão facilitar seu entendimento:

1 – Microcontrolador:

Controla todas as funções da placa controladora. O microcontrolador usado é o PIC16F628 alimentado com 5v.

2 - Driver ULN2803:

O Driver ULN2803 é um circuito integrado que é composto por oito transistores Darlington em um único chip de 20 pinos. Isto torna a montagem bem compacta.

3 – Primeiro Conjunto de Leds:

Esse conjunto é utilizado para sinalizar a saída do microcontrolador, e também por possuir características de durabilidade e eficiência.

4 – Segundo Conjunto de leds, sinalizando a saída do driver para o motor de passo:

Esse conjunto é utilizado para sinalizar a saída do driver ULN2803 para o motor de passo, e também por possuir características de durabilidade eficiência.

5 – Fonte de alimentação da placa controladora 12v:

A placa controladora deve ser alimentada por uma fonte externa capaz de fornecer entre 12v/850mA.

6 – Regulador de tensão:

Usado para alterar a tensão de 12v para 5v e alimentar o microcontrolador;

7 – Entrada do sinal de controle para o microcontrolador:

Essa entrada utiliza 4 pinos onde recebe informações da placa HOMEBEE, vindas através da saída dos reles para definição das posições;

8 – Saída do sinal de controle do motor de passo:

Possui 5 pinos, sendo que um seria entrada de alimentação de 12v do motor de passo e os outros quatros para o controle das bobinas;

A figura 3.20 ilustra uma imagem da placa controladora concluída.

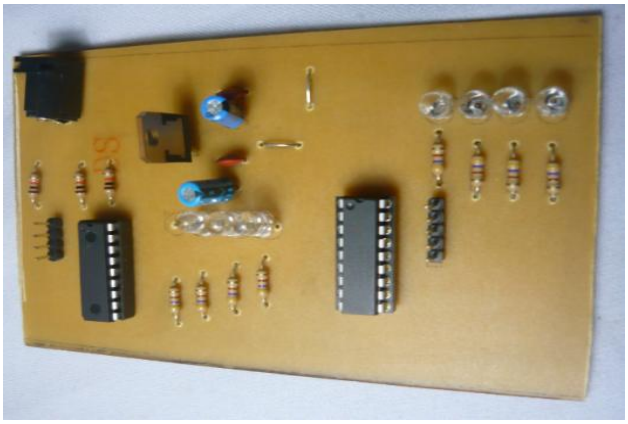


Figura 3.20 – Placa controladora / Fonte: Autor

3.3.4 – Motor de Passo e o Conjunto de Engrenagens

Os motores de passo trabalham através de sinais digitais, que indicam quando uma das bobinas está recebendo energia ou não. A ordem em que as bobinas são energizadas indica se o motor esta em movimento e como ele esta se movendo.

No projeto as engrenagens atuam em conjunto com o motor de passo. As ranhuras das engrenagens estão sempre em contato com a pequena engrenagem do motor de passo. Essa

engrenagem do motor é pequena propositalmente para dar um torque maior nas engrenagens maiores. A figura 3.21 ilustra um como as engrenagens estão atuando no protótipo.

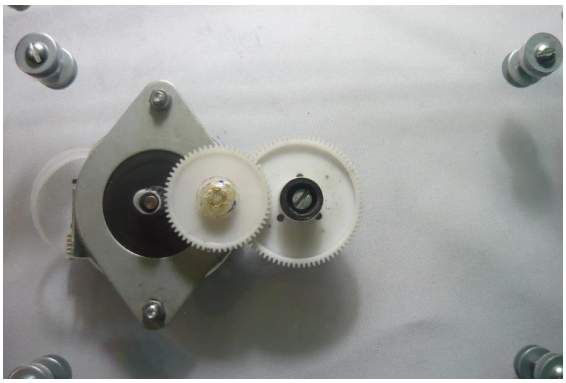


Figura 3.21 – Motor de passo em conjunto com as engrenagens / Fonte: Autor

Segue abaixo parte da programação utilizada no sistema para girar o motor de passo de acordo com a energização das bobinas, porque modo de energização dessas bobinas irá definir o sentido na rotação.

```
while(true){
   if (input(BOTAO1)) ---- Sentido horário
         {
      output_high(PIN_B4);
      delay_ms(tempo);
      output_low(PIN_B4);
      delay_ms(tempo);
      //Ativa uma bobina
      output_high(PIN_B5);
      delay_ms(tempo);
      output_low(PIN_B5);
      delay_ms(tempo);
      //ativa 1 bobina
      output_high(PIN_B6);
      delay_ms(tempo);
      output_low(PIN_B6);
```

```
    delay_ms(tempo);
    //ativa 1 bobina
    //ativa 2 bobina
    output_high(PIN_B7);
    delay_ms(tempo);
    output_low(PIN_B7);
    delay_ms(tempo);
    //ativa 1 bobina
  }
if (input(BOTAO2))   ----- Sentido Anti-horário
        {
    output_high(PIN_B7);
    delay_ms(tempo);
    output_low(PIN_B7);
    delay_ms(tempo);
    //Ativa outra bobina
    output_high(PIN_B6);
    delay_ms(tempo);
    output_low(PIN_B6);
    delay_ms(tempo);
    //Ativa outra bobina
    output_high(PIN_B5);
    delay_ms(tempo);
    output_low(PIN_B5);
    delay_ms(tempo);
    //ativa outra bobina
    output_high(PIN_B4);
    delay_ms(tempo);
    output_low(PIN_B4);
    delay_ms(tempo);
```

3.3.5 – Tubo de Suporte da Antena

O tubo de suporte que é citado neste protótipo, é um tubo em PVC. Esse tubo faz duas ligações. A primeira ligação fica na parte superior do tubo com a antena e a segunda na parte inferior juntamente com um conjunto de engrenagens que contêm uma adaptação da boca de uma garrafa pet e um conector de rosca. Esse suporte de rosca possui na sua parte interna fios condutores de cobre que enviarão as informações a outro conector externo na placa de acrílico. A figura 3.22 ilustra como funciona a conexão na parte superior do tubo.

Figura 3.22 – Conexão do tubo e a Antena / Fonte: Autor

A figura 3.23 ilustra como funciona a conexão na parte inferior do tubo. E também apresenta o conector externo do fio da antena.

Figura 3.23 – Conexão inferior do tubo / Fonte: Autor

3.3.6 – Modelo de Estrutura da Antena

Como as antenas convencionais não apresentavam uma fixação adequada ao protótipo, foi necessário o desenvolvimento de outra. O modelo que definido foi a Yagi, que possui facilidade na construção e por apresentarem uma grande recepção no sinal. No entanto neste tópico será apresentado os detalhes da antena que foi desenvolvida.

A antena foi feita em PVC, porque possui um peso considerável para o protótipo e pela facilidade na mão de obra.

A antena foi construída com a idéia de facilitar sua montagem, desmontagem e também sua locomoção, caso o residente queira alterar a posição da mesma na casa. A figura 3.24 ilustra uma idéia de como ela apresenta desmontada.

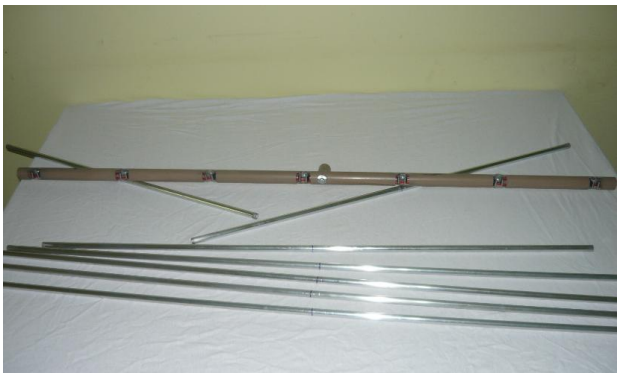


Figura 3.24 – Antena desmontada / Fonte: Autor

A Figura 3.25 ilustra o modelo do encaixe dos elementos da antena e também como é a ligação entre esses sete elementos.

Figura 3.25 – Encaixe dos elementos e o fio condutor / Fonte: Autor

E por último o modelo final da antena com todos os elementos mostrado na figura 3.26.

Figura 3.26 – Antena montada / Fonte: Autor

3.4 – Estimativa de Custo

Tabela 3.27 - Tabela de Custos do Projeto / Fonte: Autor

Equipamentos	Valor
MultiPROG	R$ 180,00
PIC16F628	R$ 20,00
Acrílico	R$ 50,00
Leds	R$ 20,00
Drivers ULN2803	R$ 15,00
Percloreto	R$ 7,00
Placa de cobre	R$ 20,00
Fonte de alimentação 12v	R$ 35,00

CAPÍTULO 4 – TESTES E RESULTADOS

4.1 - Casos de Testes

4.1.1 – Teste da Placa HOMEBEE

Para testar a placa HOMEBEE serão executados alguns testes com o software RS232HomeBee. Selecionamos a porta onde está conectada a CON-USBBEE, escolhemos a velocidade, Bit de dados, a paridade e o bit de parada. A Figura 4.1 ilustra a interface do software e quais as configurações para o teste.

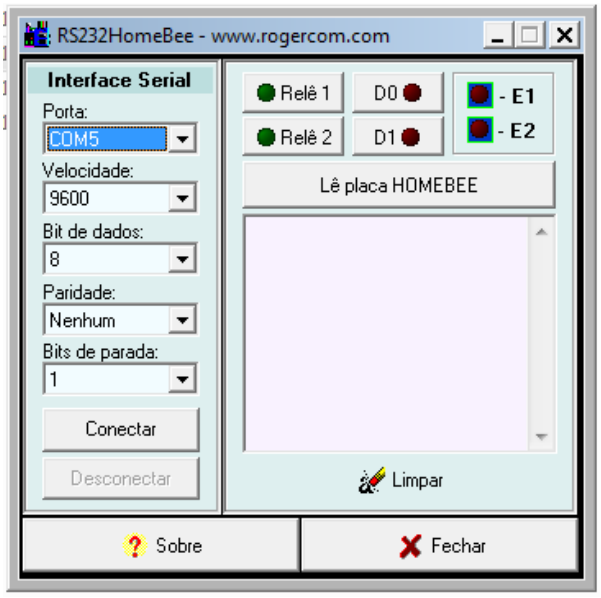


Figura 4.1 – Tela de teste / Fonte: Autor

Para testar a HOMEBEE são executadas algumas ações, que são passadas a seguir. Conecte a porta e selecione qual relê quer testar.

Figura 4.2 – Selecionando relê 1 / Fonte: Autor

Feito isso, o LED correspondente ao relê 1 será aceso. E assim acenderá o led do primeiro relê.

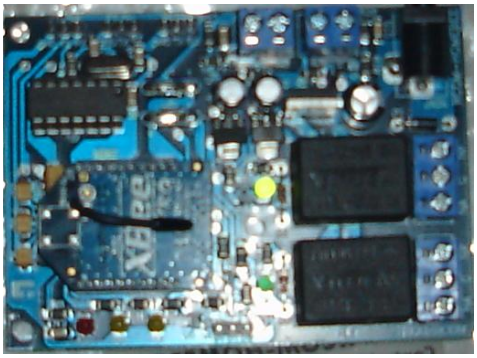


Figura 4.3 – Led 1 aceso / Fonte: Autor

Faça o mesmo para o relê 2.

Figura 4.4 – selecionando relê 2 / Fonte: Autor

Figura 4.5 – LED 2 aceso / Fonte: Autor

Pronto, a placa HOMEBEE está funcionando.

4.1.2 – Teste da Placa Controladora

Concluído os teste da placa HOMEBEE, agora foram feitos os testes com a placa controladora. Onde sãoo testados os conjunto de leds, para descobrir se estão acionando de acordo com a comunicação com a placa HOMEBEE, através dos relês.

O primeiro conjunto de leds apresentado representa o sucesso na comunicação entre o microcontrolador e o circuito integrado.

O segundo conjunto de leds que esta próximo aos fios azuis, representa o segundo sucesso da comunicação entre o circuito integrado e o motor de passo.

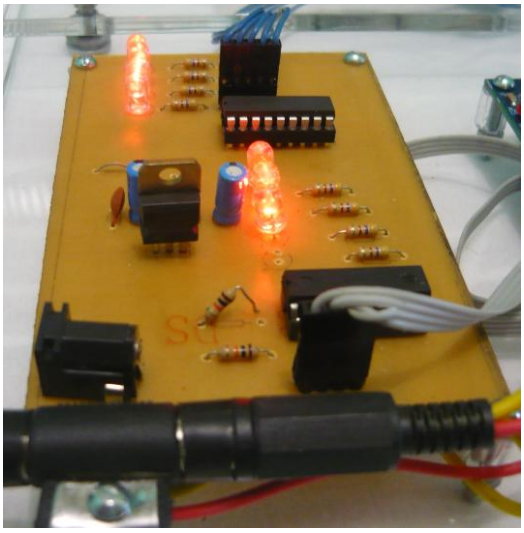


Figura 4.6 – LEDs acesos nas saídas do pic e driver / Fonte: Autor

5.1.3 – Teste do Funcionamento do Protótipo

O teste do protótipo tem o intuito de melhorar a imagem da televisão, de acordo com os comandos enviados através da interface do protótipo, as figuras 4.7 e 4.8 apresentam uma péssima imagem da televisão e a posição inicial da antena, que a princípio esta mal posicionada.

Figura 4.7 – Péssima Imagem da Televisão / Fonte: Autor

Figura 4.8 – Posição Inicial da Antena / Fonte: Autor

O primeiro passo para encontrar a solução de melhora da imagem é abrir o programa, fazer a conexão e aguardar a mensagem de confirmação, como ilustrado a figura 4.9.

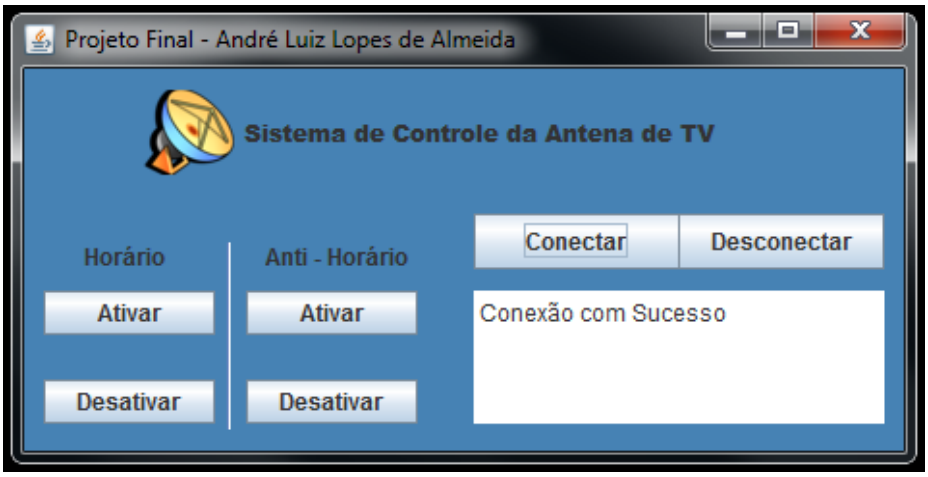


Figura 4.9 – Conexão do Software com a Placa RCON-HOMEBEE / Fonte: Autor

O segundo passo depende de qual sentido o usuário deseja aplicar a rotação da antena, caso o mesmo escolha o sentido horário, ele poderá pressionar o botão ativar que fica no espaço de sentido horário da tela. A figura 4.10 ilustra a mensagem de confirmação na ativação de rotação da antena.

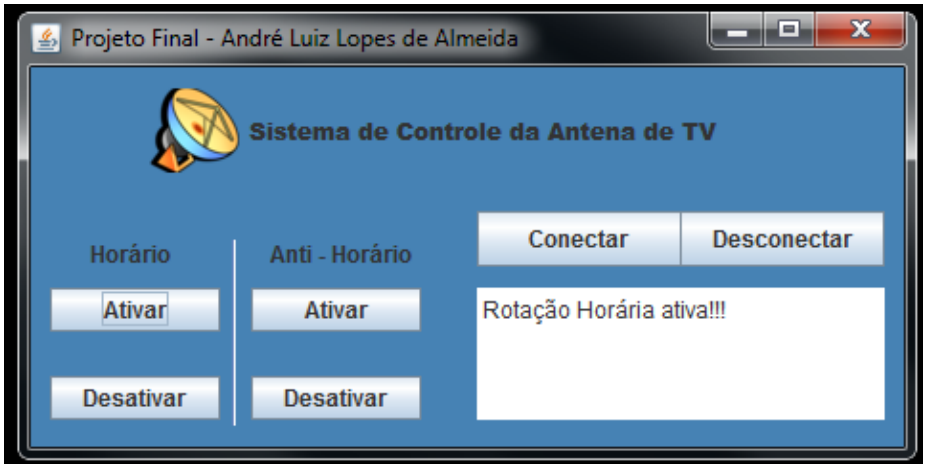


Figura 4.10 – Ativação da Rotação Horária da Antena / Fonte: Autor

Após possivelmente encontrar a posição é necessário desativar a rotação, para isto o usuário deverá pressionar o botão desativar que fica no espaço de sentido horário da tela. As figura 4.11 e 4.12 respectivamente, são apresentadas a confirmação de desativação da rotação e a posição final da antena.

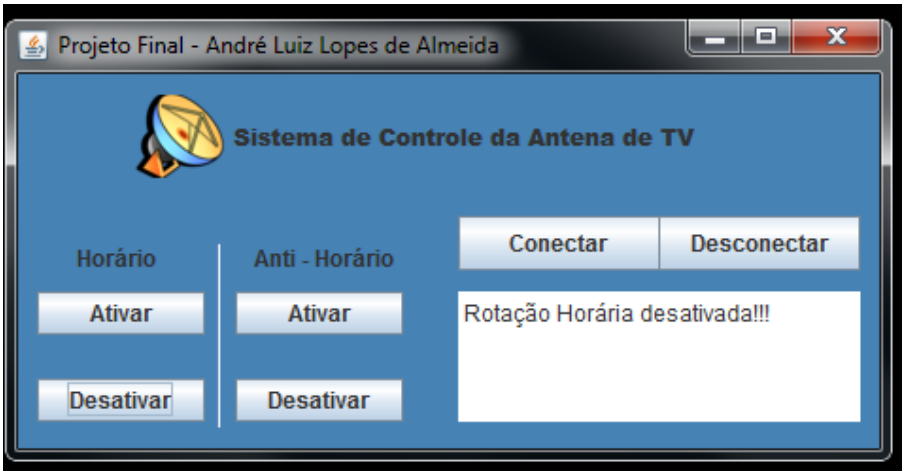


Figura 4.11 – Desativação da Rotação Horária da Antena / Fonte: Autor

Figura 4.12 – Posição Final da Antena / Fonte: Autor

A figura 4.13 ilustra uma melhora considerável na imagem da televisão após mudar a posição da antena de TV.

Figura 4.13 – Melhora na Imagem da Televisão / Fonte: Autor

4.2 – Análise de Resultados

4.2.1 – Pontos Positivos

Os resultados dos testes representam o sucesso da aplicação desenvolvida. A simulação de controle através da interface de gerenciamento de comandos foi realizada na parte inferior da casa, juntamente com todo o protótipo inclusive a antena. Portanto, acredita-se que quando a antena estiver posicionada na parte superior da casa a imagem irá ter uma melhora satisfatória na imagem da televisão.

4.2.2 – Dificuldades

Algumas dificuldades encontradas durante o desenvolvimento do projeto:

- Encontrar o motor ideal para fazer a rotação da antena;
- Aplicar os controles do motor de passo através do circuito integrado;
- Construir a caixa de engrenagens com o motor de passo;
- Construir a antena e a associação da antena com o conjunto de engrenagens;
- Desenvolver o sistema de controle da antena, pois não tinha o conhecimento do pacote de comunicação serial RXTX do Java;

CAPÍTULO 5 - CONCLUSÃO

5.1 - Conclusões do Projeto

Atualmente a automação residencial é muito aplicada para melhorar a qualidade nos processos considerados repetitivos, estando presentes no dia-a-dia das residências. Para efetivar projetos nesta área exige-se uma quantidade de conhecimento considerável, que por esta razão existiu uma influência na escolha do desenvolvimento desse projeto, onde existiu uma identificação da necessidade de um controle automático da antena de TV perante os comandos de um microcomputador.

Para implementação do projeto, foram utilizados canos PVC e mini-tubos de alumínio para o desenvolvimento da antena, placa de acrílico que atua no suporte da antena, motor de passo para movimentação da antena, o microcontrolador PIC responsável por realizar o controle de ajuste da antena, o circuito integrado ULN 2803, que possibilita que o motor seja capaz de girar tanto no sentido horário quanto no anti-horário e por último o kit ZigBee que realiza toda versatilidade no projeto com sua comunicação sem fio.

O desenvolvimento do projeto resultou em um protótipo onde permite ajuste do posicionamento de uma antena de TV utilizando um controle sem fio, ou seja, o residente poderá controlar a movimentação da antena de TV da sua casa, através de qualquer microcomputador próximo a 40 metros da antena.

O sistema de forma geral se mostrou bastante útil, pois os resultados obtidos com o protótipo foram satisfatórios para os objetivos iniciais. A principal vantagem deste projeto, foi apresentar uma melhora na qualidade da imagem de TV e trazer comodidade a usuário.

5.2 - Sugestões para Trabalhos Futuros

Com o aprendizado que conquistei neste projeto identifiquei algumas sugestões futuras, como a alteração da posição da antena sem joystick, sendo que a mesma deverá ser controlada automaticamente, por existir seu próprio analisador de ganho do sinal. Tendo em vista que ele mesmo fará o tratamento do sinal dos canais de TV e definirá a posição da antena, sem ter que em momento algum enviar qualquer comando a ele.

REFERÊNCIAS BIBLIOGRÁFICAS

BASTOS, Wagner de Sousa. Sistema de Controle para Posicionamento Automatizado de Antenas Parabólicas. Monografia de Graduação do Curso de Engenharia de Computação. Brasília: UniCEUB, 1º semestre de 2007.

ECLIPSE, Eclipse IDE for Java Developers. Disponível em: <http://www.eclipse.org/downloads/> Acesso: Acesso em: 11 de fevereiro de 2011.

FUSCO, Vicent F. Teoria e técnicas de antenas: Princípios e praticas. 1º Edição. Editora Bookman, 2006.

GROB, Bernard. Televisão e Sistemas de Vídeo. 5ª Edição. Rio de Janeiro: Editora Guanabara, 1989.

LEMAY, Laura. Aprenda em 21 dias Java 2. 3º Edição. Editora Campus, 1999.

PEREIRA, Fábio. Microcontroladores PIC - Programação em C. 4º Edição. Editora Érica, 2005.

PINHEIRO, José Mauricio Santos. As Redes com ZigBee. Disponível em: <http://www.projetoderedes.com.br/artigos/artigo_zigbee.php> (acesso em 10/04/2011).

ROGERCOM, Home Page. Disponível em <http://www.rogercom.com/> Acesso em 10 de fevereiro de 2011.

SANTOS JÚNIOR, Auteliano Antunes. Engrenagens Cilíndricas de Dentes Retos. Apostila de Sistemas Mecânicos. Campinas: Faculdade de Engenharia Mecânica da UNICAMP, fevereiro de 2003.

SCHNEIDER ELECTRIC, Empresa. Introdução a Automação Disponível em <http://www.schneider-electric.com.br/sites/brasil/pt/produtos-servicos/treinamento/e-learnings.page> Acesso em: 10 de abril de 2011.

VAREJÃO, Flávio Miguel. Linguagem de Programação: Conceitos e Técnicas. 1º Edição. Rio de Janeiro: Editora Campus, 2004.

APÊNDICE A – Código Armazenado no Microcontrolador

```
#include<16f628a.h>
#use delay(clock=4000000)
#fuses intrc_io,nowdt,put,brownout,nolvp,nomclr

//Entrada
#define BOTAO1 PIN_A2
#define BOTAO2 PIN_A3

 void main(){

   int tempo=100;
   while(true){
   if (input(BOTAO1))
            {
       output_high(PIN_B4);
       delay_ms(tempo);
       output_low(PIN_B4);
       delay_ms(tempo);
       //Ativa uma bobina
       output_high(PIN_B5);
       delay_ms(tempo);
       output_low(PIN_B5);
       delay_ms(tempo);
       //ativa 1 bobina
       output_high(PIN_B6);
       delay_ms(tempo);
       output_low(PIN_B6);
       delay_ms(tempo);
       //ativa 1 bobina
       output_high(PIN_B7);
       delay_ms(tempo);
```

```
        output_low(PIN_B7);
        delay_ms(tempo);
        //ativa 1 bobina
      }
    if (input(BOTAO2))
          {
        output_high(PIN_B7);
        delay_ms(tempo);
        output_low(PIN_B7);
        //delay_ms(tempo);
        output_high(PIN_B6);
        delay_ms(tempo);
        output_low(PIN_B6);
        //delay_ms(tempo);
        output_high(PIN_B5);
        delay_ms(tempo);
        output_low(PIN_B5);
        //delay_ms(tempo);
        output_high(PIN_B4);
        delay_ms(tempo);
        output_low(PIN_B4);
        //delay_ms(tempo);
      }

    }


}
```

APÊNDICE B – Código da interface de Gerenciamento dos Comandos

Main do Projeto:

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JTextPane;

import br.serial.SerialComLeitura;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JLabel;
import java.awt.Font;
import java.awt.Color;
import javax.swing.ImageIcon;
import javax.swing.JSeparator;
import javax.swing.SwingConstants;
import javax.swing.JTable;
import javax.swing.BoxLayout;
import javax.swing.JSplitPane;
import javax.swing.JDesktopPane;
import javax.swing.JInternalFrame;
import java.awt.ScrollPane;
import java.awt.Point;
import java.awt.Canvas;
import javax.swing.JEditorPane;


public class ProjetoZigBee {

	private JFrame frame;
	private JTextPane textPane = new JTextPane();

	SerialComLeitura serialEscrita = new SerialComLeitura("COM3", 9600,
0);

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					ProjetoZigBee window = new ProjetoZigBee();
					window.frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}
```

```
/**
 * Create the application.
 */
public ProjetoZigBee() {
	initialize();
}

/**
 * Initialize the contents of the frame.
 */
private void initialize() {
	frame = new JFrame();
	frame.setBounds(100, 100, 472, 236);
	frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

	JPanel panel = new JPanel();
	panel.setBackground(new Color(70, 130, 180));
	frame.getContentPane().add(panel, BorderLayout.CENTER);
	panel.setLayout(null);

	JButton btnNewButton = new JButton("Conectar");
	btnNewButton.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent arg0) {
			executaComando(arg0);
		}

		private void executaComando(ActionEvent arg0) {
			// TODO Auto-generated method stub

				serialEscrita.HabilitarEscrita();
				serialEscrita.ObterIdDaPorta();
				serialEscrita.AbrirPorta();
				textPane.setText("Conexão com Sucesso");

		}
	});


	btnNewButton.setBounds(233, 75, 107, 29);
	panel.add(btnNewButton);

	JButton btnNewButton_1 = new JButton("Desconectar");
	btnNewButton_1.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent arg0) {
			desconectar();
		}

		private void desconectar() {
			serialEscrita.FecharCom();
			textPane.setText("Conexão encerrada");

		}
	});
	btnNewButton_1.setBounds(339, 75, 107, 29);
	panel.add(btnNewButton_1);

	JButton btnNewButton_3 = new JButton("Desativar");
	btnNewButton_3.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent arg0) {
			desativarele1();
		}

		private void desativarele1() {
			// TODO Auto-generated method stub
			serialEscrita.OffRele1();
```

```
                        textPane.setText("Rotação Horária desativada!!!");

                  }
            });
            btnNewButton_3.setBounds(10, 161, 85, 23);
            panel.add(btnNewButton_3);


            textPane.setBounds(233, 115, 213, 69);
            panel.add(textPane);

            JLabel lblNewLabel = new JLabel("Sistema de Controle da Antena
de TV");
            lblNewLabel.setIcon(new
ImageIcon(ProjetoZigBee.class.getResource("/Imagens/1307726844_antenna.png"
)));
            lblNewLabel.setFont(new Font("Arial Black", Font.PLAIN, 12));
            lblNewLabel.setBounds(62, 11, 300, 44);
            panel.add(lblNewLabel);

            JButton btnNewButton_2 = new JButton("Ativar");
            btnNewButton_2.setBounds(10, 115, 85, 23);
            panel.add(btnNewButton_2);

            JLabel lblHorrio = new JLabel("Hor\u00E1rio");
            lblHorrio.setBounds(31, 90, 46, 14);
            panel.add(lblHorrio);

            JButton btnNewButton_4 = new JButton("Desativar");
            btnNewButton_4.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent arg0) {
                        desativarele2();
                  }

                  private void desativarele2() {
                        serialEscrita.OffRele2();
                        textPane.setText("Rotação Anti-Horária
desativada!!!");

                  }
            });
            btnNewButton_4.setBounds(115, 161, 89, 23);
            panel.add(btnNewButton_4);

            JButton btnNewButton_5 = new JButton("Ativar");
            btnNewButton_5.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent arg0) {
                        acionarele2();
                  }

                  private void acionarele2() {
                        serialEscrita.OnRele2();
                        textPane.setText("Rotação Anti-Horária ativa!!!");

                  }
            });
            btnNewButton_5.setBounds(115, 115, 89, 23);
            panel.add(btnNewButton_5);

            JLabel lblAntiHorrio = new JLabel("Anti - Hor\u00E1rio");
            lblAntiHorrio.setBounds(125, 90, 79, 14);
            panel.add(lblAntiHorrio);

            JSeparator separator = new JSeparator();
            separator.setOrientation(SwingConstants.VERTICAL);
```

```
            separator.setBounds(105, 90, 8, 97);
            panel.add(separator);
            btnNewButton_2.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent arg0) {
                        acionarele1();
                  }

                  private void acionarele1() {
                        serialEscrita.OnRele1();
                        textPane.setText("Rotação Horária ativa!!!");

                  }
            });
      }
}
```

Serial Com:

```
package br.serial;

import gnu.io.CommPortIdentifier;
import java.util.Enumeration;

public class SerialCom {

      protected String[] portas;
      protected Enumeration listaDePortas;

      public SerialCom() {
            listaDePortas = CommPortIdentifier.getPortIdentifiers();
      }

      public String[] ObterPortas() {
            return portas;
      }

      protected void ListarPortas() {
            int i = 0;
            portas = new String[10];
            while (listaDePortas.hasMoreElements()) {
                  CommPortIdentifier ips = (CommPortIdentifier)
listaDePortas
                              .nextElement();
                  portas[i] = ips.getName();
                  i++;
            }

      }

      public boolean PortaExiste(String COMp) {
            String temp;
            boolean e = false;
            while (listaDePortas.hasMoreElements()) {
                  CommPortIdentifier ips = (CommPortIdentifier)
listaDePortas
                              .nextElement();
                  temp = ips.getName();
                  if (temp.equals(COMp) == true) {
                        e = true;
                  }
            }
```

```
            return e;

      }

}
```

Serial ComLeitura:

```
package br.serial;

import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class SerialComLeitura implements Runnable, SerialPortEventListener
{

      //public String Dadoslidos;
      public byte dadoslidos;
      public int nodeBytes;
      private int baudrate;
      private int timeout;
      private CommPortIdentifier cp;
      private SerialPort porta;
      private OutputStream saida;
      private InputStream entrada;
      private Thread threadLeitura;
      private boolean IDPortaOK;
      private boolean PortaOK;
      private boolean Leitura;
      private boolean Escrita;
      private String Porta;
      protected String peso;


      public void setPeso(String peso) {
            this.peso = peso;
      }

      public String getPeso() {
            return peso;
      }

      public SerialComLeitura(String p, int b, int t) {
            this.Porta = p;
            this.baudrate = b;
            this.timeout = t;
```

```
	}

	public void HabilitarEscrita() {
		Escrita = true;
		Leitura = false;

	}

	public void HabilitarLeitura() {
		Escrita = false;
		Leitura = true;
	}

	public void ObterIdDaPorta() {
		try {
			cp = CommPortIdentifier.getPortIdentifier(Porta);
			if (cp == null) {
				System.out.println("Erro na porta");
				IDPortaOK = false;
				System.exit(1);
			}
			IDPortaOK = true;
		} catch (Exception e) {
			System.out.println("Erro obtendo ID da porta: " + e);
			IDPortaOK = false;
			System.exit(1);
		}
	}

	public void AbrirPorta() {
		try {
			porta = (SerialPort) cp.open("SerialComLeitura",
timeout);
			PortaOK = true;
			// configurar parÃ¢metros
			porta.setSerialPortParams(baudrate, porta.DATABITS_8,
					porta.STOPBITS_1, porta.PARITY_NONE);
			porta.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);
		} catch (Exception e) {
			PortaOK = false;
			System.out.println("Erro abrindo comunicação: " + e);
			System.exit(1);

		}

	}

	public void LerDados() {
		byte[] buffer = new byte[11];
		if (Escrita == false) {
			try {
				entrada = porta.getInputStream();
			} catch (Exception e) {
				System.out.println("Erro de stream: " + e);
				System.exit(1);
			}
			try {
				dadoslidos = (byte) entrada.read(buffer);
				System.out.println("Dados:"+dadoslidos);
				porta.addEventListener(this);
			} catch (Exception e) {
				System.out.println("Erro de listener: " + e);
				System.exit(1);
```

```
                }
                porta.notifyOnDataAvailable(true);

                try {
                      threadLeitura = new Thread(this);
                      threadLeitura.start();
                      run();
                } catch (Exception e) {
                      System.out.println("Erro de Thred: " + e);
                }
          }
    }


    public void OnRele1() {


          byte[] a = new byte[11];
           a[0] = 0x7E;
        a[1] = 0x00;
        a[2] = 0x07;
        a[3] = 0x01;
        a[4] = 0x01;
        a[5] = 0x50;
        a[6] = 0x01;
        a[7] = 0x00;
        a[8] = 0x7B;
        a[9] = 0x01;
        a[10] = 0x30;

          if (Escrita == true) {
                try {
                      saida = porta.getOutputStream();
                      System.out.println("FLUXO OK!");
                } catch (Exception e) {
                      System.out.println("Erro.STATUS: " + e);
                }
                try {
                      System.out.println("Enviando um byte para " +
Porta);
                      System.out.println("Enviando : " + a);
                      saida.write(a);
                      Thread.sleep(10);
                      saida.flush();
                } catch (Exception e) {
                      System.out.println("Houve um erro durante o envio.
");
                      System.out.println("STATUS: " + e);
                      System.exit(1);
                }
          } else {
                System.exit(1);
          }

    }

public void OffRele1() {


          byte[] b = new byte[11];
           b[0] = 0x7E;
        b[1] = 0x00;
        b[2] = 0x07;
        b[3] = 0x01;
        b[4] = 0x01;
```

```
            b[5] = 0x50;
            b[6] = 0x01;
            b[7] = 0x00;
            b[8] = 0x7B;
            b[9] = 0x00;
            b[10] = 0x30;


               if (Escrita == true) {
                     try {
                           saida = porta.getOutputStream();
                           System.out.println("FLUXO OK!");
                     } catch (Exception e) {
                           System.out.println("Erro.STATUS: " + e);
                     }
                     try {
                           System.out.println("Enviando um byte para " +
Porta);
                           System.out.println("Enviando : " + b);
                           saida.write(b);
                           Thread.sleep(10);
                           saida.flush();
                     } catch (Exception e) {
                           System.out.println("Houve um erro durante o envio.
");
                           System.out.println("STATUS: " + e);
                           System.exit(1);
                     }
               } else {
                     System.exit(1);
               }

         }

public void OnRele2() {


         byte[] c = new byte[11];
          c[0] = 0x7B;
        c[1] = 0x00;
        c[2] = 0x07;
        c[3] = 0x01;
        c[4] = 0x01;
        c[5] = 0x50;
        c[6] = 0x01;
        c[7] = 0x00;
        c[8] = 0x7B;
        c[9] = 0x02;
        c[10] = 0x1f;

         if (Escrita == true) {
               try {
                     saida = porta.getOutputStream();
                     System.out.println("FLUXO OK!");
               } catch (Exception e) {
                     System.out.println("Erro.STATUS: " + e);
               }
               try {
                     System.out.println("Enviando um byte para " + Porta);
                     System.out.println("Enviando : " + c);
                     saida.write(c);
                     Thread.sleep(10);
                     saida.flush();
               } catch (Exception e) {
                     System.out.println("Houve um erro durante o envio. ");
```

```
                    System.out.println("STATUS: " + e);
                    System.exit(1);
              }
        } else {
              System.exit(1);
        }

}

public void OffRele2() {


        byte[] d = new byte[11];
         d[0] = 0x7B;
       d[1] = 0x00;
       d[2] = 0x07;
       d[3] = 0x01;
       d[4] = 0x01;
       d[5] = 0x50;
       d[6] = 0x01;
       d[7] = 0x00;
       d[8] = 0x7B;
       d[9] = 0x00;
       d[10] = 0x1f;

        if (Escrita == true) {
              try {
                    saida = porta.getOutputStream();
                    System.out.println("FLUXO OK!");
              } catch (Exception e) {
                    System.out.println("Erro.STATUS: " + e);
              }
              try {
                    System.out.println("Enviando um byte para " + Porta);
                    System.out.println("Enviando : " + d);
                    saida.write(d);
                    Thread.sleep(10);
                    saida.flush();
              } catch (Exception e) {
                    System.out.println("Houve um erro durante o envio. ");
                    System.out.println("STATUS: " + e);
                    System.exit(1);
              }
        } else {
              System.exit(1);
        }

}

        public void run() {
              try {
                    Thread.sleep(5);
              } catch (Exception e) {
                    System.out.println("Erro de Thred: " + e);
              }
        }

        public void serialEvent(SerialPortEvent ev) {
              StringBuffer bufferLeitura = new StringBuffer();
              int novoDado = 0;
              switch (ev.getEventType()) {
              case SerialPortEvent.BI:
              case SerialPortEvent.OE:
              case SerialPortEvent.FE:
```

```
                case SerialPortEvent.PE:
                case SerialPortEvent.CD:
                case SerialPortEvent.CTS:
                case SerialPortEvent.DSR:
                case SerialPortEvent.RI:
                case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
                      break;
                case SerialPortEvent.DATA_AVAILABLE:
                      // Novo algoritmo de leitura.
                      while (novoDado != -1) {
                            try {
                                  novoDado = entrada.read();
                                  if (novoDado == -1) {
                                        break;
                                  }
                                  if ('\r' == (char) novoDado) {
                                        bufferLeitura.append('\n');
                                  } else {
                                        bufferLeitura.append((char) novoDado);
                                  }
                            } catch (IOException ioe) {
                                  System.out.println("Erro de leitura serial: "
+ ioe);
                            }
                      }
                      setPeso(new String(bufferLeitura));
                      System.out.println(getPeso());
                      break;
                }

          }


          public void FecharCom() {
                try {
                      porta.close();
                      System.out.println("Conexão encerrada");
                } catch (Exception e) {
                      System.out.println("Erro fechando porta: " + e);
                      System.exit(0);
                }
          }

          public String obterPorta() {
                return Porta;
          }

          public int obterBaudrate() {
                return baudrate;
          }


}
```

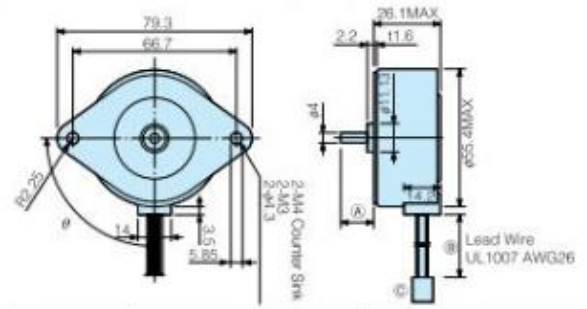
ANEXOS

Panasonic

Stepping Motor PM TYPE

55SPM25

Standard Dimensions(mm)



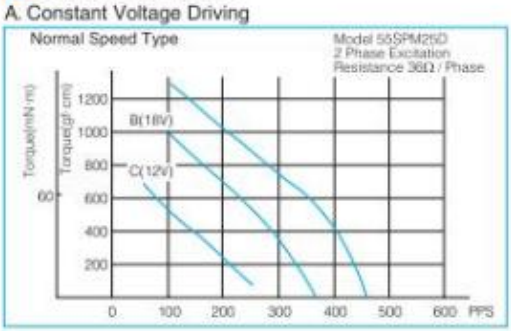
Shaft Length(mm)	Ⓐ	11	15	18
Lead Wire Length(mm)	Ⓑ	150±10	250±10	350±10
Connector	Ⓒ	Connecter type available		

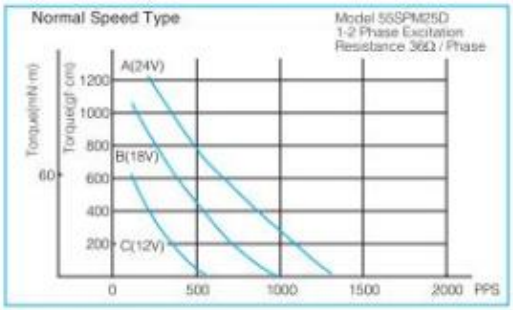
Typical Data

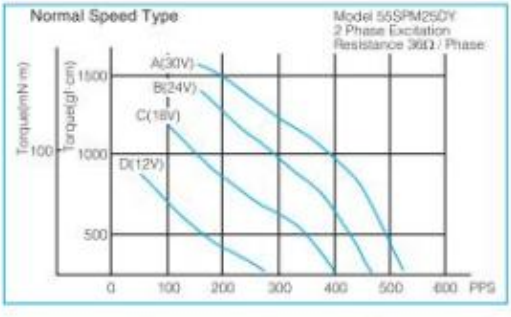
Model	Rated Input (W)	Resistance (Ω/Phase)	mH	Step Angle (DEG)	Roter Inertia (g-cm²)		Weight (g)	Magnet Type
55SPM25D5	4	6.5	6.3	7.5	Y	62	240	Blanked (Isotropic ferrite)
55SPM25D7		12	11					Y (Radial anisotropic ferrite)
55SPM25DA		36	32		N	41		N (Nd-Fe-B)
55SPM25DC		50	50					

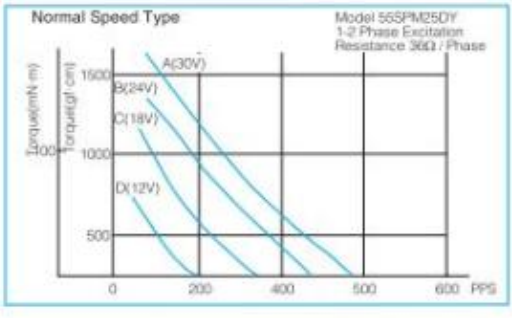
Characteristics

A. Constant Voltage Driving









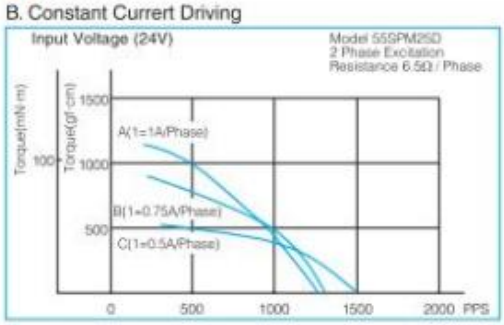
Panasonic

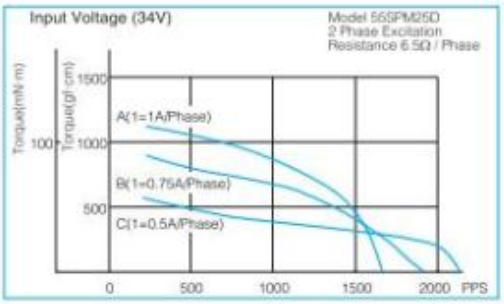
Stepping Motor PM TYPE

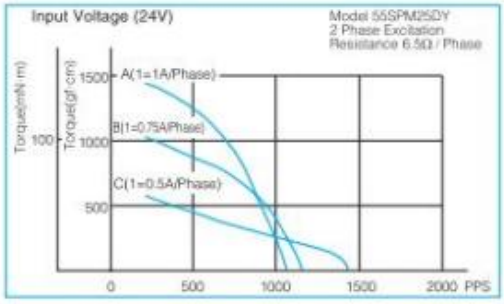
55SPM25

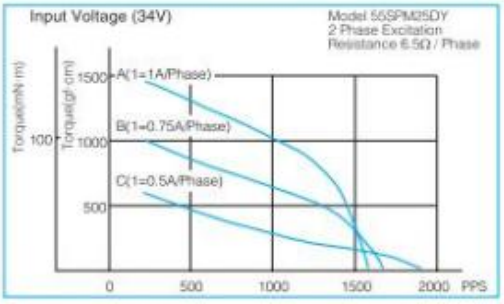
Characteristics

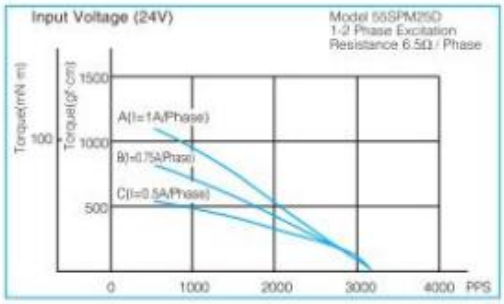
B. Constant Currert Driving

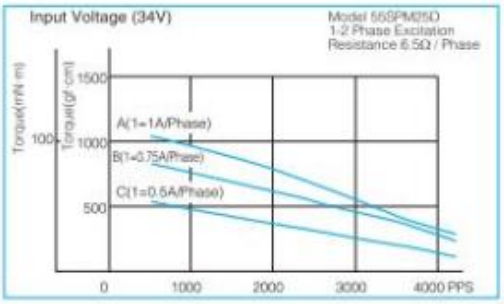


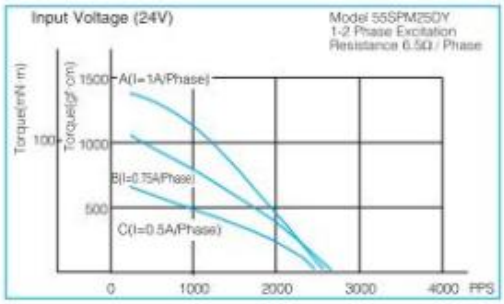


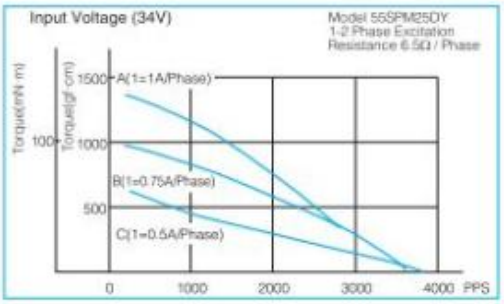










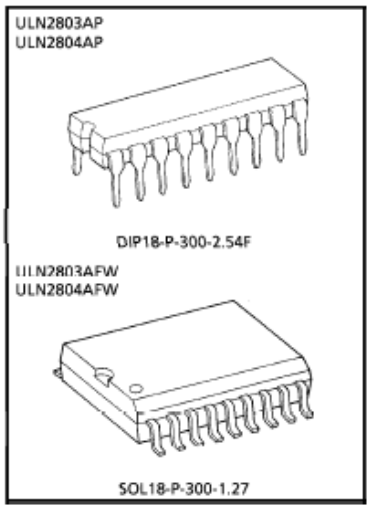


TOSHIBA BIPOLAR DIGITAL INTEGRATED CIRCUIT SILICON MONOLITHIC

ULN2803AP, ULN2803AFW, ULN2804AP, ULN2804AFW

8CH DARLINGTON SINK DRIVER

The ULN2803AP / AFW Series are high-voltage, high-current darlington drivers comprised of eight NPN darlington pairs.
All units feature integral clamp diodes for switching inductive loads.
Applications include relay, hammer, lamp and display (LED) drivers.



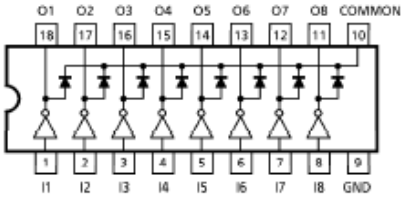
Weight
DIP18-P-300-2.54F : 1.478g (Typ.)
SOL18-P-300-1.27 : 0.48g (Typ.)

FEATURES

- Output current (single output)
  500mA (Max.) (ULN2803AP / AFW series)
- High sustaining voltage output
  50V (Min.) (ULN2803AP / AFW series)
- Output clamp diodes
- Inputs compatible with various types of logic.
- Package type-AP : DIP-18pin
- Package type-AFW : SOL-18pin

TYPE	INPUT BASE RESISTOR	DESIGNATION
ULN2803AP / AFW	2.7kΩ	TTL, 5V CMOS
ULN2804AP / AFW	10.5kΩ	6~15V PMOS, CMOS

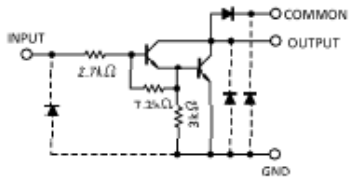
PIN CONNECTION (TOP VIEW)



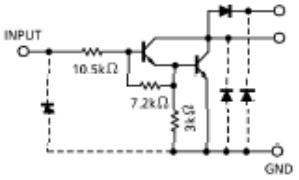
961001EBA1

SCHEMATICS (EACH DRIVER)

ULN2803AP / AFW



ULN2804AP / AFW



(Note) The input and output parasitic diodes cannot be used as clamp diodes.

MAXIMUM RATINGS (Ta = 25°C)

CHARACTERISTIC		SYMBOL	RATING	UNIT
Output Sustaining Voltage		$V_{CE\ (SUS)}$	−0.5~50	V
Output Current		I_{OUT}	500	mA / ch
Input Voltage		V_{IN}	−0.5~30	V
Clamp Diode Reverse Voltage		V_R	50	V
Clamp Diode Forward Current		I_F	500	mA
Power Dissipation	AP	P_D	1.47	W
	AFW		0.92 / 1.31 (Note)	
Operating Temperature		T_{opr}	−40~85	°C
Storage Temperature		T_{stg}	−55~150	°C

(Note) On Glass Epoxy PCB (75 × 114 × 1.6mm Cu 20%)

RECOMMENDED OPERATING CONDITIONS (Ta = −40~85°C)

CHARACTERISTIC		SYMBOL	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
Output Sustaining Voltage		$V_{CE\ (SUS)}$		0	—	50	V
Output Current	AP	I_{OUT}	T_{pw} = 25ms, Duty = 10%, 8 Circuits	0	—	347	mA / ch
			T_{pw} = 25ms, Duty = 50%, 8 Circuits	0	—	123	
	AFW		T_{pw} = 25ms, Duty = 10%, 8 Circuits	0	—	268	
			T_{pw} = 25ms, Duty = 50%, 8 Circuits	0	—	90	
Input Voltage		V_{IN}		0	—	30	V
Input Voltage (Output On)	ULN2803AP / AFW	$V_{IN\ (ON)}$		3.5	—	30	V
	ULN2804AP / AFW			8	—	30	
Clamp Diode Reverse Voltage		V_R		—	—	50	V
Clamp Diode Forward Current		I_F		—	—	400	mA
Power Dissipation	AP	P_D	Ta = 85°C	—	—	0.76	W
	AFW		Ta = 85°C (Note)	—	—	0.48	

(Note) On Glass Epoxy PCB (75 × 114 × 1.6mm Cu 20%)

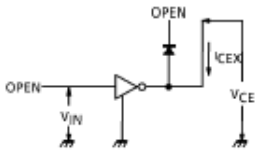
ELECTRICAL CHARACTERISTICS (Ta = 25°C)

CHARACTERISTIC		SYMBOL	TEST CIR-CUIT	TEST CONDITION		MIN.	TYP.	MAX.	UNIT
Output Leakage Current		I_{CEX}	1	V_{CE} = 50V	Ta = 25°C	—	—	50	μA
	ULN2804AP / AFW			V_{CE} = 50V	Ta = 85°C	—	—	100	
				V_{CE} = 50V	V_{IN} = 1V	—	—	500	
Collector-Emitter Saturation Voltage		$V_{CE\ (sat)}$	2	I_{OUT} = 350mA, I_{IN} = 500μA		—	1.3	1.6	V
				I_{OUT} = 200mA, I_{IN} = 350μA		—	1.1	1.3	
				I_{OUT} = 100mA, I_{IN} = 250μA		—	0.9	1.1	
Input Current	ULN2803AP / AFW	$I_{IN\ (ON)}$	2	V_{IN} = 3.85V		—	0.93	1.35	mA
	ULN2804AP / AFW			V_{IN} = 5V		—	0.35	0.5	
				V_{IN} = 12V		—	1.0	1.45	
		$I_{IN\ (OFF)}$	4	I_{OUT} = 500μA, Ta = 85°C		50	65	—	μA
Input Voltage (Output On)	ULN2803AP / AFW	$V_{IN\ (ON)}$	5	V_{CE} = 2V, I_{OUT} = 200mA		—	—	2.4	V
				V_{CE} = 2V, I_{OUT} = 250mA		—	—	2.7	
				V_{CE} = 2V, I_{OUT} = 300mA		—	—	3.0	
	ULN2804AP / AFW			V_{CE} = 2V, I_{OUT} = 125mA		—	—	5.0	
				V_{CE} = 2V, I_{OUT} = 200mA		—	—	6.0	
				V_{CE} = 2V, I_{OUT} = 275mA		—	—	7.0	
				V_{CE} = 2V, I_{OUT} = 350mA		—	—	8.0	
DC Current Transfer Ratio		h_{FE}	2	V_{CE} = 2V, I_{OUT} = 350mA		1000	—	—	
Clamp Diode Reverse Current		I_R	6	Ta = 25°C (Note)		—	—	50	μA
				Ta = 85°C (Note)		—	—	100	
Clamp Diode Forward Voltage		V_F	7	I_F = 350mA		—	—	2.0	V
Input Capacitance		C_{IN}	—			—	15	—	pF
Turn-On Delay		t_{ON}	8	R_L = 125Ω, V_{OUT} = 50V		—	0.1	—	μs
Turn-Off Delay		t_{OFF}		R_L = 125Ω, V_{OUT} = 50V		—	0.2	—	

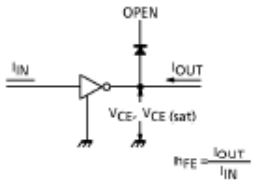
(Note) $V_R = V_R$ MAX.

TEST CIRCUIT

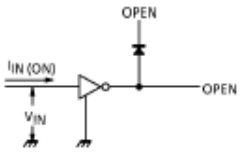
1. I_{CEX}



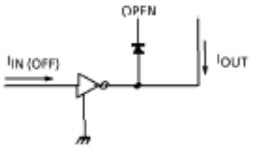
2. $V_{CE\ (sat)}$, h_{FE}



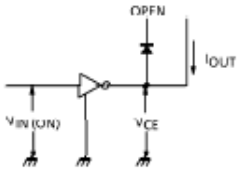
3. $I_{IN\ (ON)}$



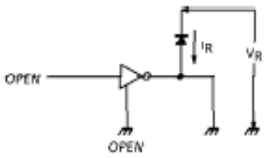
4. $I_{IN\ (OFF)}$



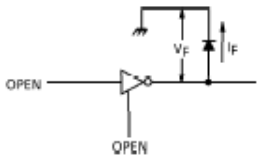
5. $V_{IN\ (ON)}$



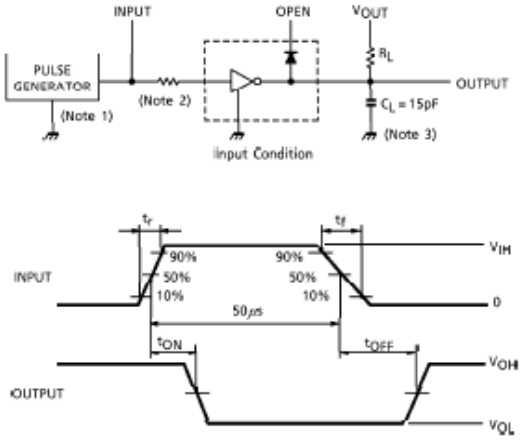
6. I_R



7. V_F



8. t_{ON}, t_{OFF}



(Note 1) Pulse Width 50μs, Duty Cycle 10%
Output Impedance 50Ω, $t_r \leqq$ 5ns, $t_f \leqq$ 10ns

(Note 2) See below.

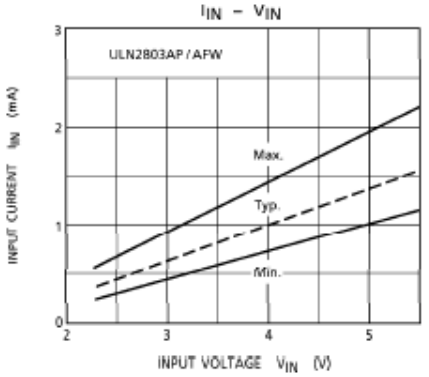
INPUT CONDITION

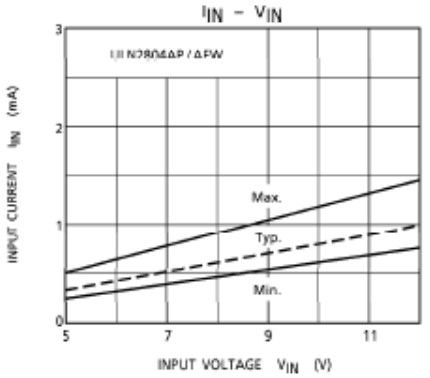
TYPE NUMBER	R1	V_{IH}
ULN2803AP / AFW	0Ω	3V
ULN2804AP / AFW	0Ω	8V

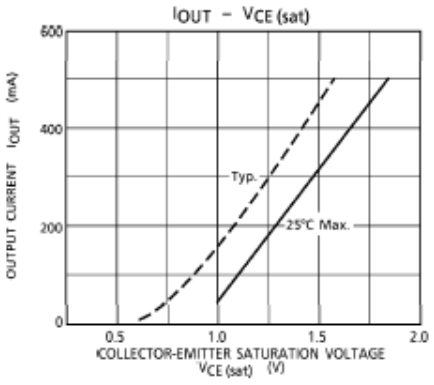
(Note 3) C_L includes probe and jig capacitance

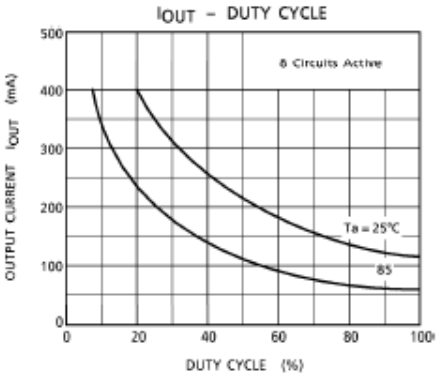
PRECAUTIONS for USING

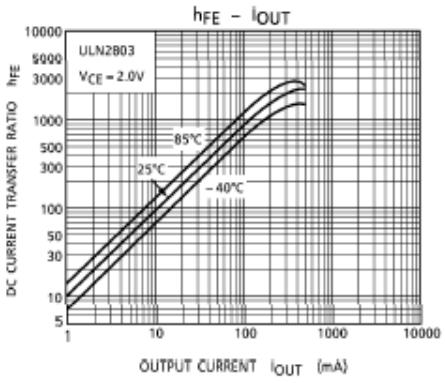
Utmost care is necessary in the design of the output line, COMMON and GND line since IC may be destroyed due to short-circuit between outputs, air contamination fault, or fault by improper grounding.

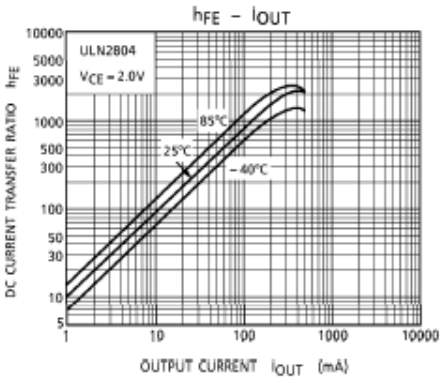

I_{IN} – V_{IN}
ULN2803AP / AFW
Max.
Typ.
Min.
INPUT CURRENT I_{IN} (mA)
INPUT VOLTAGE V_{IN} (V)

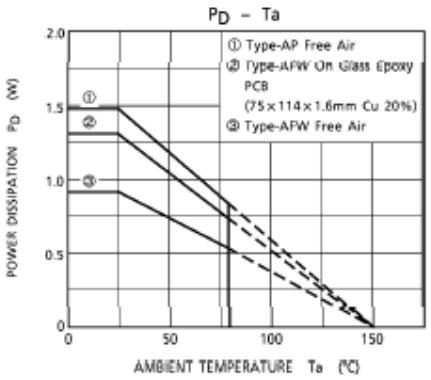

I_{IN} – V_{IN}
Max.
Typ.
Min.
INPUT CURRENT I_{IN} (mA)
INPUT VOLTAGE V_{IN} (V)


I_{OUT} – $V_{CE (sat)}$
Typ.
25°C Max.
OUTPUT CURRENT I_{OUT} (mA)
COLLECTOR-EMITTER SATURATION VOLTAGE
$V_{CE (sat)}$ (V)


I_{OUT} – DUTY CYCLE
8 Circuits Active
Ta = 25°C
85
OUTPUT CURRENT I_{OUT} (mA)
DUTY CYCLE (%)


h_{FE} – I_{OUT}
ULN2803
V_{CE} = 2.0V
85°C
25°C
– 40°C
DC CURRENT TRANSFER RATIO h_{FE}
OUTPUT CURRENT I_{OUT} (mA)

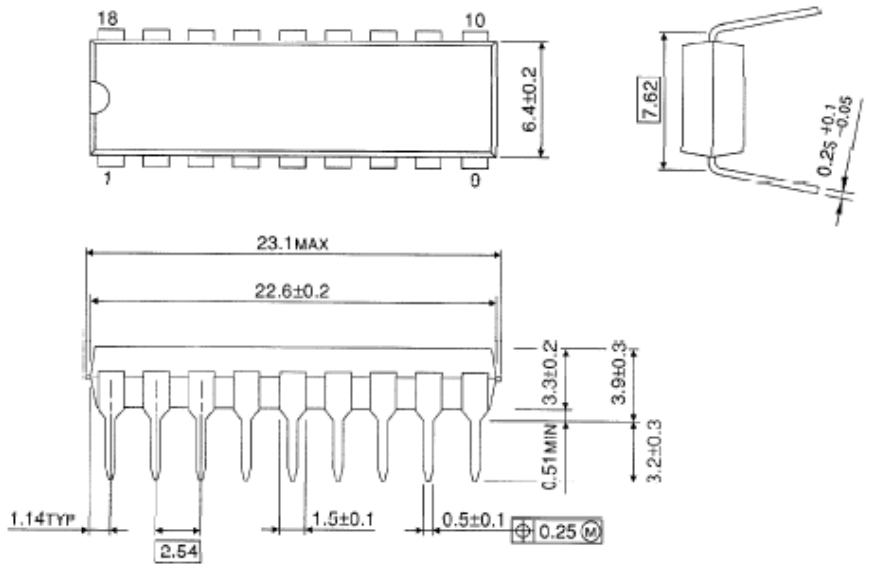

h_{FE} – I_{OUT}
ULN2804
V_{CE} = 2.0V
85°C
25°C
– 40°C
DC CURRENT TRANSFER RATIO h_{FE}
OUTPUT CURRENT I_{OUT} (mA)


P_D – Ta
① Type-AP Free Air
② Type-AFW On Glass Epoxy PCB
(75×114×1.6mm Cu 20%)
③ Type-AFW Free Air
POWER DISSIPATION P_D (W)
2.0
1.5
1.0
0.5
0
0
50
100
150
AMBIENT TEMPERATURE Ta (°C)

OUTLINE DRAWING

DIP18-P-300-2.54F

Unit : mm

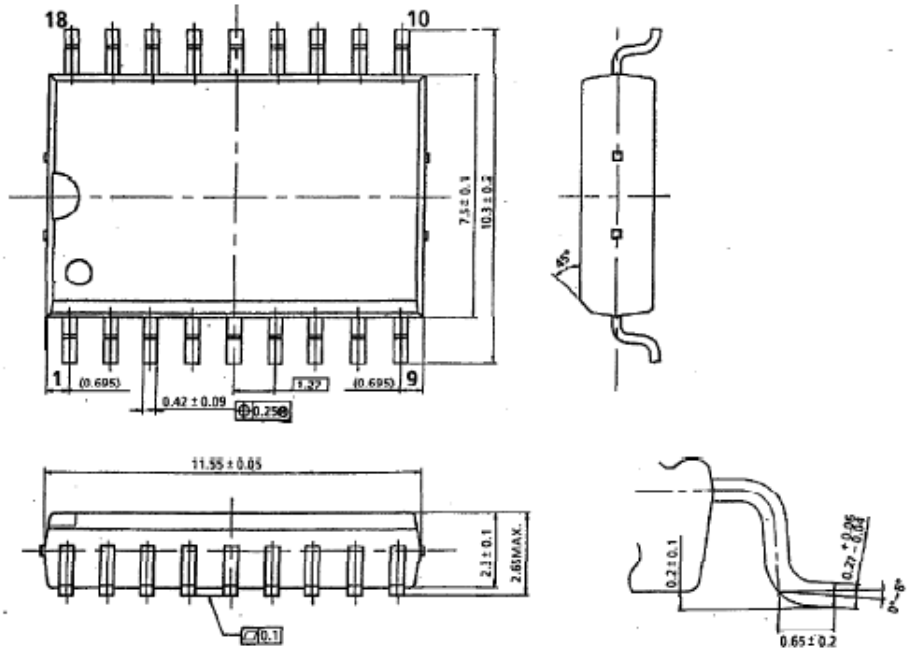


Weight : 1.478g (Typ.)

OUTLINE DRAWING

SOL18-P-300-1.27

Unit : mm



Weight : 0.48g (Typ.)