

FACULDADE DE TECNOLOGIA DE SÃO PAULO
CURSO DE TECNOLOGIA EM ELETRÔNICA INDUSTRIAL

FÁBIO OLIVEIRA SILVA
VICTOR DE LIMA MILARE

BRAÇO ROBÓTICO RÁDIO CONTROLADO

SÃO PAULO, 2017.

Faculdade de Tecnologia de São Paulo

FÁBIO OLIVEIRA SILVA

VICTOR DE LIMA MILARE

BRAÇO ROBÓTICO RÁDIO CONTROLADO

Monografia apresentada à Faculdade de Tecnologia de São Paulo, como parte dos requisitos para obtenção do título de Tecnólogo em Eletrônica Industrial.

Orientador: Prof. Me. Maurício Antônio Deffert

FÁBIO OLIVEIRA SILVA
VICTOR DE LIMA MILARE

BRAÇO ROBÓTICO RÁDIO CONTROLADO

Monografia apresentada a Faculdade de
Tecnologia de São Paulo, como parte dos
requisitos para obtenção do título de Tecnólogo
em Eletrônica Industrial.

Orientador: Prof. Me. Maurício Antônio Deffert

Banca Examinadora

Professor:

Professor:

São Paulo, 12 de Dezembro de 2017.

DEDICATÓRIA

Ao professor Maurício pela orientação e confiança.

Agradecemos a todos os professores por proporcionar o conhecimento não apenas intelectual, mas também auxiliar no nosso crescimento seja ele profissional como também de caráter. Aos nossos pais, pelo amor, incentivo e apoio incondicional. A todos que direta ou indiretamente fizeram parte da nossa formação, o nosso muito obrigado.

A persistência é o caminho do êxito.

(Charles Chaplin)

RESUMO

Com o aumento da demanda industrial se faz necessário que os processos se tornem mais rápidos e precisos. Para isso existem tecnologias como robôs industriais. Uma variação significativa é o braço robótico que pode ter diversas funcionalidades na indústria, como por exemplo, retirar uma peça de uma esteira para uma caixa. Na indústria os braços robóticos são um tanto inflexíveis ao se tratar de programação, ou seja, uma vez programado executará sua rotina infinitamente, tendo dificuldades caso haja a necessidade de mudar sua trajetória. Analisando o funcionamento de algumas marcas e modelos de braços robóticos, notou-se que, sua interface de controle é complexa para quem não tem experiência, sendo assim, deve ser estudado e analisado cada função do braço robótico, demandando bastante tempo para o aprendizado. Neste projeto, utilizando um braço robótico será desenvolvido um controle remoto rádio controlado com funções simples que qualquer pessoa, mesmo com pouco ou nenhum conhecimento técnico, conseguirá realizar a manipulação e programação de seus movimentos. Com modelo anatômico, simplicidade nos menus de navegação e uma comunicação segura entre o controle remoto e o controlador (que pode ser utilizado em qualquer braço robótico no qual seus eixos sejam movimentados por servo motores), é garantido confiabilidade nos movimentos e facilidade na manipulação, tornando-se um diferencial na indústria. O sistema desenvolvido de controle, tornou a manipulação do braço robótico mais intuitivo e precisa, trazendo para o operador facilidades na hora de programação.

Palavras-chave: Braço Robótico Industrial; confiabilidade; manuseio; rádio controlado, anatômico.

ABSTRACT

With the increase of the industrial demand it becomes necessary that the processes become faster and precise. For this there are technologies such as industrial robots. A significant variation is the robotic arm that can have several functionalities in the industry, for example, to remove a piece from a mat to a box. In the industry the robotic arms are somewhat inflexible when it comes to programming, that is, once programmed, it will execute its routine infinitely, having difficulties if it is necessary to change its trajectory. Analyzing the operation of some brands and models of robotic arms, it was noticed that, its control interface is complex for those who do not have experience, therefore, each function of the robotic arm must be studied and analyzed, requiring a lot of time for learning. In this project, using a robotic arm will develop a radio controlled remote control with simple functions that anyone, even with little or no technical knowledge, will be able to perform the manipulation and programming of their movements. With anatomical model, simplicity in the navigation menus and a secure communication between the remote control and the controller (that can be used in any robotic arm in which its axes are moved by servo motors), is guaranteed reliability in the movements and ease in the manipulation, becoming a differential in the industry. The developed control system has made manipulation of the robotic arm more intuitive and precise, bringing to the operator facilities at the time of programming.

Key-words: Industrial Robotic Arm; reliability; handling; radio controlled, anatomical.

LISTA DE ILUSTRAÇÕES

Figura 1- Braço Robótico Mitsubishi.....	16
Figura 2 - Controle Remoto utilizado para movimentação do Robô	16
Figura 3 - Braço Robótico ABB	17
Figura 4 - Controle Remoto do Robô da ABB	18
Figura 5 - Divisão Básica do Projeto	19
Figura 6 - Pinagem do PIC18F4520.....	22
Figura 7 - Parte interna de um Servo Motor	23
Figura 8 - Largura de Sinais	24
Figura 9 - Exemplo de uma comunicação I ² C com um mestre e três escravos	26
Figura 10 - Exemplo de um barramento real.....	26
Figura 11 - Exemplo de Leitura e Escrita	27
Figura 12 - - Eixo X e Y de um joystick	28
Figura 13 - Exemplos de Antenas	30
Figura 14 - Bateria utilizada.....	31
Figura 15 – Arquitetura.....	31
Figura 16 - Fluxograma de funcionamento do projeto.....	32
Figura 17 - Protótipo do braço robótico e suas dimensões	33
Figura 18 – Padrão de menu principal	35
Figura 19 - Padrão de submenu.....	36
Figura 20 - Exemplo de função para leitura I ² C.....	37
Figura 21 - Exemplo de função para escrita I ² C.....	37
Figura 22 - Circuito básico do DS1307.....	38
Figura 23 - Configuração dos registradores ADCON0 e ADCON1	39
Figura 24 - Configuração de Registrador INTCON2.....	39
Figura 25 - Configuração de Registrador INTCON.....	40
Figura 26 - Configuração do Timer0 bit a bit	40
Figura 27 - Configuração do Timer1 bit a bit	41
Figura 28 - Configuração da EUSART bit a bit.....	41
Figura 29 - Exemplo do Protocolo de comunicação	42
Figura 30 - Layout controle remoto (Top e Bottom respectivamente)	45
Figura 31 - Visão 3D da placa do controle remoto	46
Figura 32 - Placa final com os componentes soldados	46
Figura 33 - Layout gabinete (Top e Bottom respectivamente).....	47
Figura 34 - Vista 3D da placa do gabinete	47
Figura 35 - Placa final com os componentes já soldados	48
Figura 36 - Protótipo final do controle remoto	49
Figura 37 - Comparação com a primeira versão	49
Figura 38 - Protótipo do gabinete	50
Figura 39 - Evolução dos gabinetes	50
Figura 40 - Forma de onda do pulso PWM - 2ms e 1ms respectivamente.....	51
Figura 41- Protótipo final	52
Figura 42 - Conexões com o Braço Robótico utilizando DB9.....	52

LISTA DE ABREVIATURAS E SIGLAS

Bit: menor unidade de informação que pode ser armazenada ou transmitida;

Byte: é um dos tipos de dados integrais em computação. É usado com frequência para especificar o tamanho ou quantidade da memória ou da capacidade de armazenamento de certo dispositivo;

CLP: controlador lógico programável;

CNC: comando numérico computadorizado é um sistema que permite o controle de máquinas, sendo utilizado principalmente em tornos e centros de usinagem;

Datasheet: ou folhas de especificações de componentes são os documentos de consulta com os dados de determinados componentes, de acordo com o fabricante;

Delay: é o termo técnico usado para designar o retardo de sinais em circuitos eletrônicos;

Duty Cycle: fração de tempo em que um sistema está em um estado “ativado”;

EEPROM: memória apenas de leitura apagado eletricamente;

Half-Duplex: comunicação em ambos os sentidos porém não simultaneamente;

IHM: interface homem máquina;

Ladder: linguagem de programação baseada em contatos elétricos;

LCD: display de cristal líquido;

Low-end: modelo mais simples;

Master: mestre;

Ms: milissegundo;

PIC: controlador de interrupção programável;

Pick and Place: termo que significa, ao pé da letra, “Pega e larga”;

Protoboard: matriz de contatos;

PWM: modulação por largura de pulso;

RF: rádio frequência;

Sumário

1. Introdução.....	12
1.1. Robótica	13
1.2. Definição de Robô	13
1.3. Necessidades e Soluções	15
1.4. Descrição do Projeto	18
1.5. Levantamento dos Componentes usados	19
2. Detalhamento dos Componentes utilizados.....	20
2.1. Controlador.....	20
2.2. Servos Motores	23
2.2.1. Controle do Servo motor	24
2.3. Comunicação I ² C e RTC DS1307	25
2.4. Módulo Joystick.....	28
2.5. Módulo de Rádio Frequência	29
2.6. Bateria	30
3. Arquitetura do Projeto.....	31
4. Fluxograma.....	32
5. Projeto Mecânico	33
Tabela 1 – Consumo de corrente do braço robótico	34
6. Controle Remoto.....	35
6.1. IHM.....	35
6.2. Circuito do Relógio	36
6.3. Bateria	38
6.4. Controlador e módulo RF	39
7. Gabinete	43
8. Circuitos Elétricos	44
9. Layouts e placas finais.....	45
10. Protótipos finais	48
11. Resultados e Discussões	51
12. Conclusão	54
13. Futuras Melhorias.....	55
14. Bibliografia.....	56
15. Apêndice A – Circuito elétrico do controle remoto.....	57

16.	Apêndice B – Circuito elétrico do gabinete	58
17.	APÊNDICE C – Código fonte do projeto	59

1. Introdução

Com a evolução do ser humano, o aumento do conhecimento e a chegada de novas tecnologias, a necessidade de obter ferramentas que auxiliem o homem está sendo cada vez mais almejada; muitos acreditam que as máquinas vieram para substituir o homem, crendo que, por exemplo, um robô traz praticidade, agilidade, visando sempre o aumento da produção.

A própria revolução industrial proporcionou isso, desenvolvimento de novas tecnologias para atender as demandas de produção. O robô industrial foi a tecnologia mais explorada para implementação em chãos de fábrica, visto que traz diversos benefícios como: precisão, maior rapidez de produção, melhor qualidade de finalização, baixos custos de mão de obra em longo prazo, produção ininterrupta, acessos a lugares não acessíveis ao ser humano, flexibilidades ao processo de fabricação, entre outras.

Porém, é uma área do mercado muito vasto, ou seja, possui muitas marcas e fabricantes. Isso resulta em muita tecnologia proprietária não disponibilizada, o que de certa forma pode gerar desconforto caso seja necessário mudanças de programação no sistema, tornando a empresa, conseqüentemente, uma "escrava" do fabricante.

Sendo assim, com base no exposto esse projeto tem como objetivo desenvolver um sistema de controle de um braço robótico que seja intuitivo e de fácil manipulação, e que sua programação seja muito simples, realizada até por pessoas com pouquíssimo conhecimento técnico na área.

O projeto está estruturado conforme segue: primeiro um estudo sobre robótica, a necessidade e logo em seguida tem-se o desenvolvimento propriamente dito, pesquisa de Hardware, esquemáticos, cronograma, e por fim os principais resultados e a conclusão.

1.1. Robótica

Ao se pensar em robótica, logo vem à mente o grande foco dessa área, os robôs. Contudo, a robótica abrange a arte, a base de conhecimentos e experiências de concepção, aplicação e uso de robôs em atividades humanas. Interagindo com inúmeras outras áreas diferentes que se caracterizam pela interdisciplinaridade entre engenharia mecânica, elétrica, eletrônica, da ciência da computação, das ciências cognitivas, da biologia e outras disciplinas [1].

O uso de dispositivos, periféricos e outras máquinas de fabricação integram os denominados sistemas robóticos, estes por sua vez não são formados apenas por robôs, mas também por outros dispositivos e sistemas utilizados em conjunto com os robôs. Integrando os robôs em um sistema que como um todo é projetado para executar uma tarefa ou realizar uma operação, observando-se que mesmo com as vantagens do uso de robôs estes ainda precisam ser programados e controlados [2].

A robótica, em diferentes formas, tem estado nas mentes dos seres humanos desde o momento em que conseguimos construir coisas, sendo que hoje os robôs são usados para diversas atividades e meios como manufatura industrial, exploração subaquática, exploração espacial, acessibilidade para deficientes, diversão, entretenimento e até a construção de outros robôs [3].

1.2. Definição de Robô

A palavra robô vem do Tcheco “*robota*” que possui como significado trabalho penoso ou operário automático. Com o passar dos anos foram criadas várias definições para robô [4].

Conforme a ISO 10218, um robô pode ser definido como uma máquina manipuladora, com vários graus de liberdade, controlada automaticamente, reprogramável multifuncional, que pode ter base fixa ou móvel para utilização em aplicações de automação industrial.

Toda máquina ferramenta poderá ser considerada um robô, porém nem todo robô é uma máquina ferramenta, pois existem exemplos de robôs que foram especialmente projetados para manipulação de peças e partes.

Devido a várias diferenças em função de características e propriedades, existem diversas classes de robôs que se diferenciam em suas aplicações e formas de trabalho.

Basicamente, são divididos nas respectivas classes:

- a) Robôs Inteligentes: são manipulados por sistemas multifuncionais controlados por computador, são capazes de interagir com seu ambiente através de sensores e de tomar decisões em tempo real. Atualmente dedicam-se grandes esforços no desenvolvimento desse tipo de robô.
- b) Robôs com controle por computador: são semelhantes aos robôs inteligentes, porém não tem a capacidade de interagir com o ambiente. Se estes robôs forem equipados com sensores e software adequado, se transformam em robôs inteligentes.
- c) Robôs de aprendizagem: se limitam a repetir uma sequência de movimentos, realizados com a intervenção de um operador ou memorizadas.
- d) Manipuladores: são sistemas mecânicos multifuncionais, cujo sensível sistema de controle permite governar o movimento de seus membros das seguintes formas:
 - Manual, quando o operador controla diretamente os movimentos;
 - Sequencial variável, quando é possível alterar algumas características do ciclo de trabalho.

Além disso, pode ser extremamente interessante realizarmos a classificação dos robôs do ponto de vista do controle de seus movimentos, admitindo as seguintes configurações:

- a) Sem controle-servo: é o programa que controla o movimento dos diferentes componentes do robô, se realiza em um posicionamento “ponto a ponto” no espaço.
- b) Com controle-servo: este tipo de controle permite duas formas de trabalho:
 - Controle dos movimentos dos membros do robô em função de seus eixos. Os movimentos podem ser realizados ponto a ponto ou com trajetória contínua.
 - Os movimentos se estabelecem da respectiva posição de seus eixos de coordenada e da orientação da ferramenta do robô.

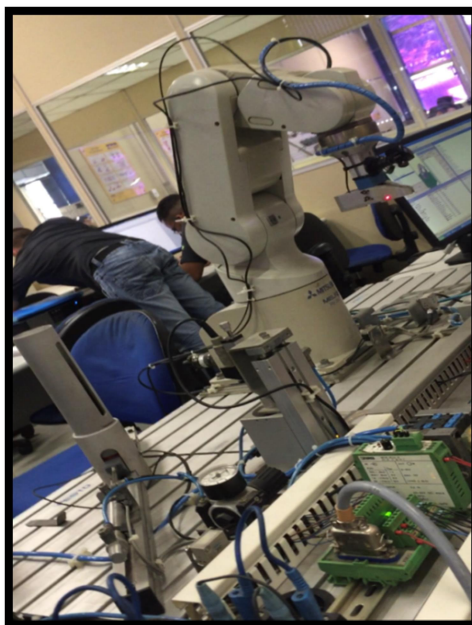
Na grande maioria, os robôs, são projetados e destinados a serem controlados por um computador ou dispositivos microprocessados, sob a supervisão do computador que está executando algum tipo de programa e define os seus movimentos. Assim quando sua programação é alterada, as ações do robô também se modificam. O objetivo dos estudos em robótica, atualmente, é conseguir desenvolver um dispositivo que pode realizar muitas tarefas diferentes e seja muito flexível no que pode fazer sem ter que ser reprojetado, haja vista que são projetados para serem capazes de executar várias tarefas com base no programa em execução simplesmente.

1.3. Necessidades e Soluções

É notável que os robôs industriais são pré-programados e relativamente inflexíveis, ou seja, nem todos podem fazer a sua programação. Como exemplo disso, estudou-se alguns modelos de robôs industriais reais usados no chão de fábrica, dando foco no modo de programação e também na manipulação utilizando o controle remoto.

O primeiro modelo estudado é o braço robótico da Mitsubishi de 5 graus de liberdade, muito utilizado para soldagem e *Pick and Place*. A seguir as Figuras 1 e 2 mostram, respectivamente, a imagem do robô e seu controle remoto.

Figura 1- Braço Robótico Mitsubishi



Fonte: Imagem cedida pelo SENAI Mariano Ferraz

Esse modelo utiliza um CLP como controlador e sua programação é totalmente textual (linguagem proprietária do fabricante) com comunicação serial.

Figura 2 - Controle Remoto utilizado para movimentação do Robô



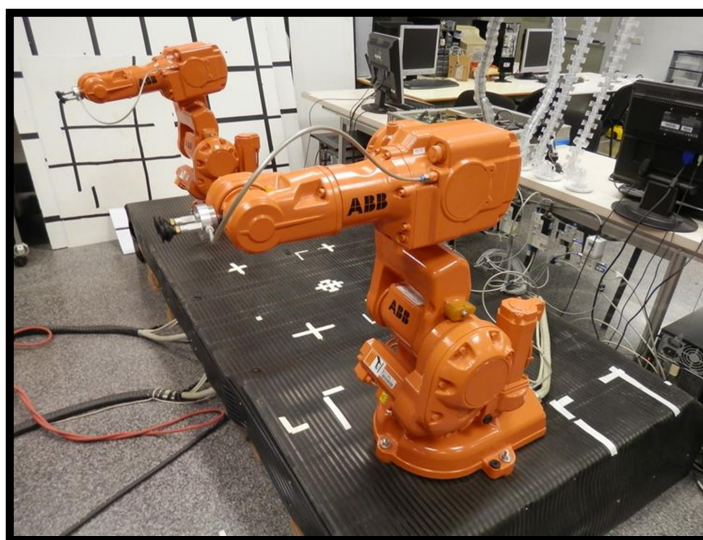
Fonte: Imagem cedida pelo SENAI Mariano Ferraz

Observou-se que a movimentação do robô usando o controle remoto foi bem complicada, visto que há muitos botões, além disso, por ser grande acaba

perdendo em ergonomia e o usuário precisa ficar deslocando a mão para alcançar todos os botões.

Já o segundo modelo é um braço robótico da ABB de 6 graus de liberdade, muito utilizado em conjunto com uma máquina CNC e uma esteira industrial. A seguir, as Figuras 3 e 4 ilustram o robô e seu controle remoto.

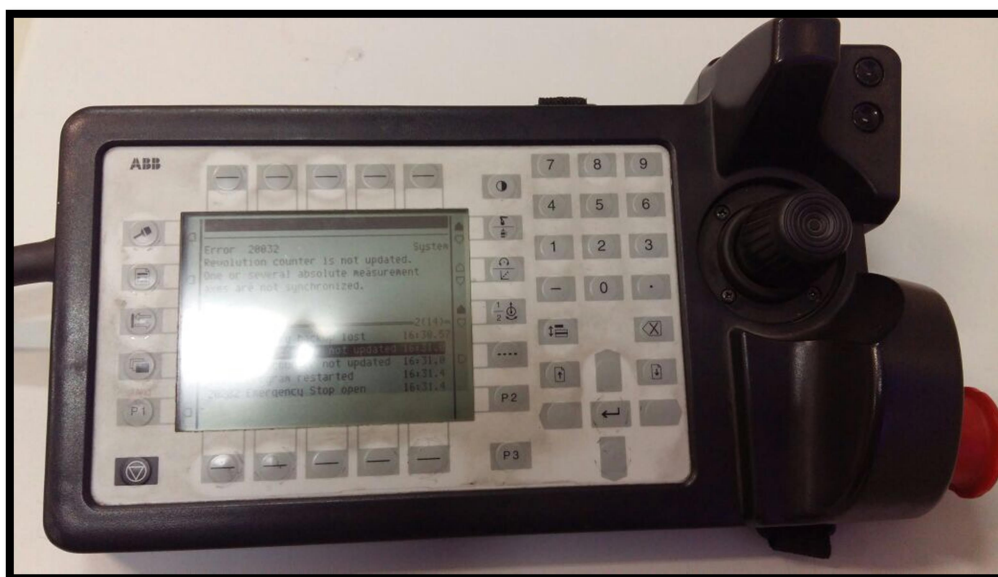
Figura 3 - Braço Robótico ABB



Fonte: Imagem cedida pelo SENAI Mariano Ferraz

Esse braço robótico utiliza um CLP como controlador, porém, o controlador é compartilhado com a máquina CNC e também com a esteira industrial. A linguagem de programação pode ser tanto textual quanto Ladder, ambos desenvolvidos pelo fabricante.

Figura 4 - Controle Remoto do Robô da ABB



Fonte: Imagem cedida pelo SENAI Mariano Ferraz

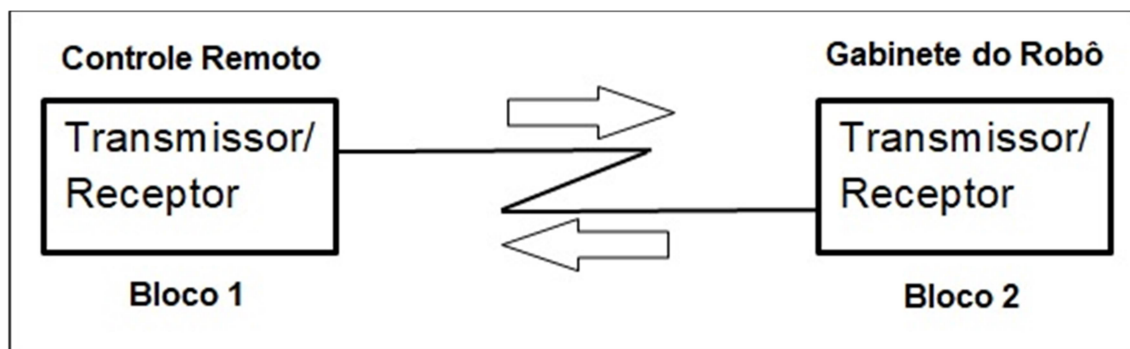
A experiência de uso desse controle remoto é muito pior que a do primeiro modelo; a primeira impressão é que o usuário se sente perdido ao manipular o robô, mesmo usando um joystick para facilitar a movimentação, as funções mostradas no *display* são bem complexas.

Pensando nisso, pretende-se desenvolver um sistema de controle que seja mais intuitivo, e não apenas na forma tradicional (através de linhas de código). O operário literalmente levará o braço robótico para as posições desejadas, salvando ao longo da movimentação manual e depois executar tudo automaticamente e na sequência desejada.

1.4. Descrição do Projeto

O projeto será dividido em dois grandes blocos principais, conforme a Figura 5:

Figura 5 - Divisão Básica do Projeto



Fonte: Criado pela Equipe

Controle Remoto: Esse primeiro bloco será responsável pelas entradas das informações através da interação do usuário, isto é, esse bloco fará a movimentação do braço robótico.

O controle remoto será rádio controlado, através de módulos de rádio frequência. O usuário fará as interações através de joysticks e botões para avançar ou recuar cada eixo do robô e mudar as funções do IHM. Esses dados serão tratados por um controlador central, através de um protocolo de comunicação desenvolvido especificamente para isso e transmitidos serialmente.

Além disso, no próprio controle remoto haverá uma IHM (*display* de LCD) que indicará todas as informações atualizadas de posições salvas, respectivos ângulos do robô, o modo (automático ou manual), nível de bateria e data e hora.

Gabinete do Robô: O segundo bloco será responsável pelo recebimento das informações (enviadas pelo bloco 1), o tratamento dessas informações de forma adequada (verificar os dados recebidos e devolver a informação confirmando o recebimento) e realizar a movimentação do braço robótico com base no que foi enviado pelo usuário.

1.5. Levantamento dos Componentes usados

Sabendo-se que ambos os blocos irão transmitir e receber dados, necessita-se de um módulo que faça uma comunicação *Half-Duplex* (enviar e receber dados “no mesmo barramento”, mas não simultaneamente com o objetivo de garantir que a informação enviada foi interpretada corretamente), por isso optou-

se pelo módulo Xbee séries 1, pois atende todas as exigências do projeto e um ótimo custo/benefício.

Após decidir os módulos de rádio frequência, necessita-se escolher um controlador para coletar as informações (bloco 1), receber as informações e tratar (bloco 2). Após testes, onde utilizou-se um PIC16F877A, percebeu-se que a memória de programa quase não foi suficiente para realizar as rotinas de LCD, o que provavelmente iria prejudicar o projeto. Portanto, decidiu-se por usar uma evolução de controladores da família 16F da Microchip: a família 18F, mais especificamente o microcontrolador 18F4520.

2. Detalhamento dos Componentes utilizados

Segundo o engenheiro Wagner Rambo (2014, p.10):

“É de extrema importância dar ênfase no estudo do Hardware, pois é através dele que tudo irá funcionar. Além disso, conhecendo bem os componentes sabe-se quais são os limites do projeto como um todo e possibilita trabalhar nas futuras melhorias.”

Por isso, serão detalhados todos os componentes utilizados, para facilitar o entendimento de como foram feito os protótipos finais.

2.1. Controlador

O controlador utilizado no projeto é um microcontrolador da Microchip. A linha de microcontroladores PIC, fabricados pela Microchip, é composta de microcontroladores de 8, 16 e 32 bits.

A linha de 8 bits compreende os microcontroladores “*low-end*” das séries PIC10 e PIC12 (compostas por chips de até oito pinos, com pequena capacidade de memória e periféricos simplificados), além dos microcontroladores das séries PIC16, PIC17 e PIC18.

Os microcontroladores PIC16 são baseados nas arquiteturas de 12 e de 14 bits da Microchip e constituem uma das linhas mais numerosas do fabricante. Os

chips com arquitetura de 12 bits utilizam instruções com essa largura e são oriundos do primeiro modelo (o 16V54), originalmente desenvolvido pela *General Instruments*.

Os chips de 14 bits utilizam instruções com largura de 14 bits, com mais memória de programa (até 8192 palavras) e mais memória de dados (até 368 bytes de RAM), além de diversos periféricos, tais como *timers*, interfaces seriais assíncronas, interfaces seriais síncronas (SPI e I²C), USB de baixa velocidade, conversores A/D, EEPROM interna, etc.

A linha PIC18 (objeto deste estudo) utiliza um conjunto de instruções bastante próximo ao da linha PIC17, com a adição de novas características importantes, como conjunto de instruções aumentado (de 58 instruções nos PIC17 para 75 instruções nos PIC18), capacidade de memórias aumentadas, pilhas de *hardware* mais ampla, etc.

O PIC18F4520 inclui uma CPU PIC18, além de 16384 (contra as 8192 palavras do PIC16F877A) palavras de memória flash (32768 bytes), 1536 bytes de memória RAM e 256 bytes de memória EEPROM [5].

Dentre os seus periféricos internos, podemos destacar:

- ✓ 3 pinos de interrupção externa e quatro pinos com detecção de mudança de estado;
- ✓ Um Timer de 8 bits (Timer 2) e três *Timers* de 16 bits (Timer 0, Time 1, Timer 3);
- ✓ Dois módulos de captura/comparação/PWM (CCP), sendo um deles com capacidades avançadas (ECCP) como *deadtime* programável, polaridade programável, até quatro saídas, além de *restart* e *shutdown automáticos*;
- ✓ Módulo de comunicação serial síncrona (MSSP) capaz de operar em modo SPI (mestre ou escravo, nos quatro modos de transmissão) e I²C (mestre ou escravo);
- ✓ Módulo de comunicação serial síncrona/assíncrona (EUSART) capaz de realizar comunicações seriais síncronas e assíncronas, com capacidade de endereçamento e outras funcionalidades avançadas;
- ✓ Conversor analógico/digital com resolução de 10 bits e 13 canais de entrada;
- ✓ Tensão de Operação entre 4,5 e 5,5V (18F4520) ou entre 2,0 e 5,5V (18LF4520);
- ✓ Frequência máxima de operação: 40MHz (quando operando em 5V);

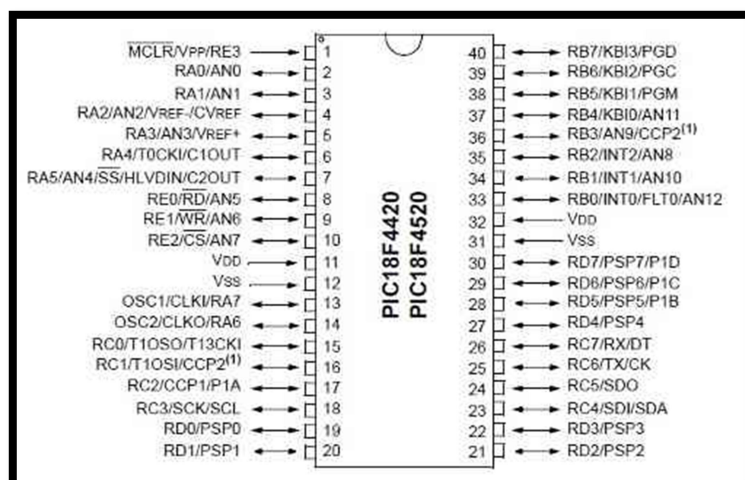
- ✓ Disponibilidade em encapsulamentos de 40 pinos DIP (utilizado nesse projeto), 44 pinos TQFP 44 pinos QFN .

Interrupções: Uma interrupção é uma alteração no fluxo normal do programa provocada por um evento externo a ele. Um evento de interrupção pode ser gerado por uma mudança no estado de um pino de entrada, um estouro na contagem de um timer, a recepção de um caractere através de uma interface serial, etc. Podemos resumir interrupção como uma chamada de sub-rotina originada pelo hardware interno do microcontrolador.

O modo de tratamento das interrupções é umas das principais diferenças entre a família 16F e da família 18F. Enquanto que na família 16F existe apenas um desvio devido à interrupção (prioridade relativa), a família 18F tem duas grandes melhorias: Um mecanismo que permite atribuir diferentes prioridades (baixa ou alta) às interrupções, de forma que interrupções de alta prioridade têm primazia sobre as de baixa prioridade; e um salvamento rápido de contexto, que acaba diminuindo a latência das interrupções (tempo do evento de interrupção e a execução da primeira instrução útil da rotina de tratamento de interrupção).

É importante salientar, que a alta prioridade foi utilizada nesse projeto (no qual a prioridade alta ficou responsável pela leitura serial) e as de baixa prioridade (para contagem de tempo). A Figura 6 mostra os periféricos do microcontrolador [6].

Figura 6 - Pinagem do PIC18F4520



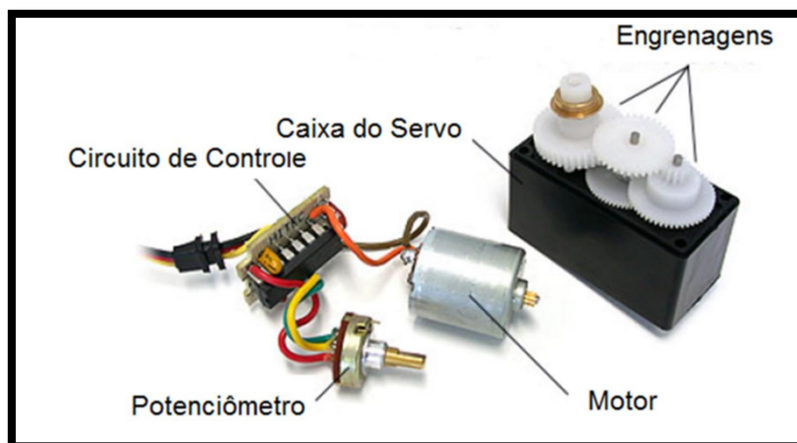
2.2. Servos Motores

Servo motores são dispositivos de malha fechada, ou seja, recebem um sinal de controle; verificam a posição atual; atuam no sistema indo para a posição desejada.

De um servo motor são exigidos, dinâmica, controle de rotação, torque constante e precisão de posição/posicionamento. As características mais desejadas nos servo motores são o torque constante em larga faixa de rotação (até 4.500 rpm), uma larga faixa de controle da rotação e variação (até 1:3000) e alta capacidade de sobrecarga.

Os servos motores contém um sistema de controle interno o qual verifica a posição de entrada com a posição de saída e um motor de corrente contínua. E apesar de terem precisão, os servos não têm muita força. Geralmente os servos motores encontrados giram apenas 180 graus. Mas pode-se modificá-los para girarem 360 graus. A Figura 7 mostra internamente como é o servo motor:

Figura 7 - Parte interna de um Servo Motor



Fonte: <http://goo.gl/KOc8uY>

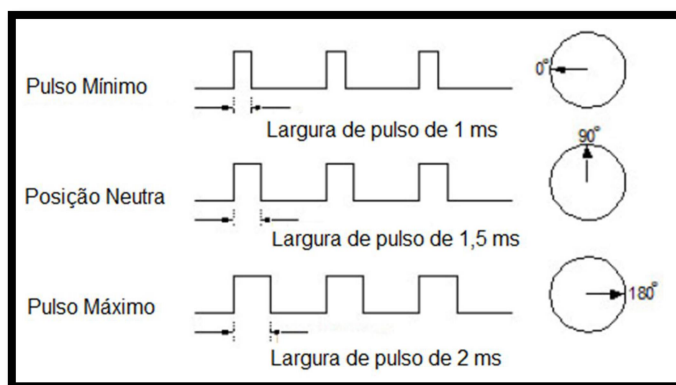
- Circuito de Controle: Responsável por receber os sinais e energia do receptor, monitorar a posição do potenciômetro e controlar o motor de acordo com o sinal do receptor e a posição do potenciômetro.
- Potenciômetro: Ligado ao eixo de saída do servo monitora a posição do mesmo.

- Motor: Movimenta as engrenagens e o eixo principal do servo.
- Engrenagens: Reduzem a rotação do motor, transferem mais força ao eixo principal de saída e movimentam o potenciômetro junto com o eixo.
- Caixa do servo – onde ficam alocados todos os componentes exceto as engrenagens.

2.2.1. Controle do Servo motor

O servo motor é alimentado com tensões de 5V e recebe um sinal no formato PWM (*Pulse Width Modulation*). O circuito de controle do servo fica monitorando este sinal em intervalos de 20ms. Se neste intervalo de tempo, o controle detecta uma alteração na largura do sinal, ele altera a posição do eixo para que a sua posição coincida com o sinal recebido, a Figura 8 mostra as diferentes posições para diferentes larguras de pulso.

Figura 8 - Largura de Sinais



Fonte: <http://goo.gl/KOc8uY>

- Um sinal com largura de pulso de 1 ms corresponde a posição do servo todo a esquerda ou 0 graus.
- Um sinal com largura de pulso de 1,5 ms corresponde a posição central do servo ou de 90 graus.
- Um sinal com largura de pulso de 2 ms corresponde a posição do servo todo a direita ou 180 graus.

Uma vez que o servo recebe um sinal de 1,5ms (por exemplo), ele verifica se o potenciômetro encontra-se na posição correspondente, se ele estiver nada é

feito. Se o potenciômetro não estiver na posição correspondente ao sinal recebido, o circuito de controle aciona o motor até que o potenciômetro esteja na posição correta. A direção de rotação do servo motor depende da posição do potenciômetro.

Ao se tentar alterar a posição do servo motor, verifica-se uma resistência feita pelo motor. Esta resistência é chamada de torque. O torque é uma das principais características do servo motor. Mede-se o torque em kg-cm (quilograma por centímetro) ou oz-in (onça por polegada). Uma alternativa para se obter maior robustez é utilizar engrenagens de metal [7].

2.3. Comunicação I²C e RTC DS1307

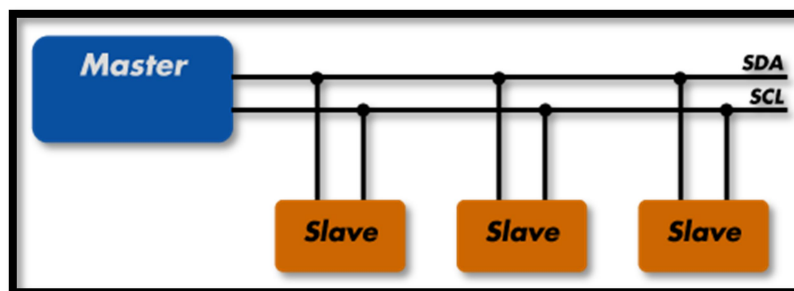
O meio de comunicação entre o circuito integrado DS1307 e o PIC é feito através do protocolo I²C, então é fundamental conhecer esse protocolo.

I²C é a sigla de *Inter-Integrated Circuit*, e basicamente é um protocolo de comunicação entre dispositivos que “falam” I²C. Esse protocolo trabalha no modelo *master-slave*, com pelo menos um dispositivo atuando como *master*, e os demais dispositivos atuando como *slave*. A função do *master* é coordenar a comunicação, sendo que ele é quem envia informações a determinado *slave* ou consulta informações. Pode-se ter mais de um *master* numa conexão I²C, e até 112 dispositivos *slaves*.

I²C é um protocolo de barramento (ou bus), ou seja, com os mesmos fios conectando todos os dispositivos do setup. Essa característica, barramento, é um dos grandes atrativos do I²C, pois reduzimos em muito a necessidade de pinos de conexão no Pic, pois usaremos sempre os mesmos fios para a conexão, não importa se tem 1 ou 127 dispositivos.

O Pic vem com pinos próprios para a conexão I²C, no caso do PIC18F4520 os pinos são 18 (SCL) e 23 (SDA), a Figura 9 mostra um exemplo de comunicação I²C entre mestre e escravos.

Figura 9 - Exemplo de uma comunicação I²C com um mestre e três escravos

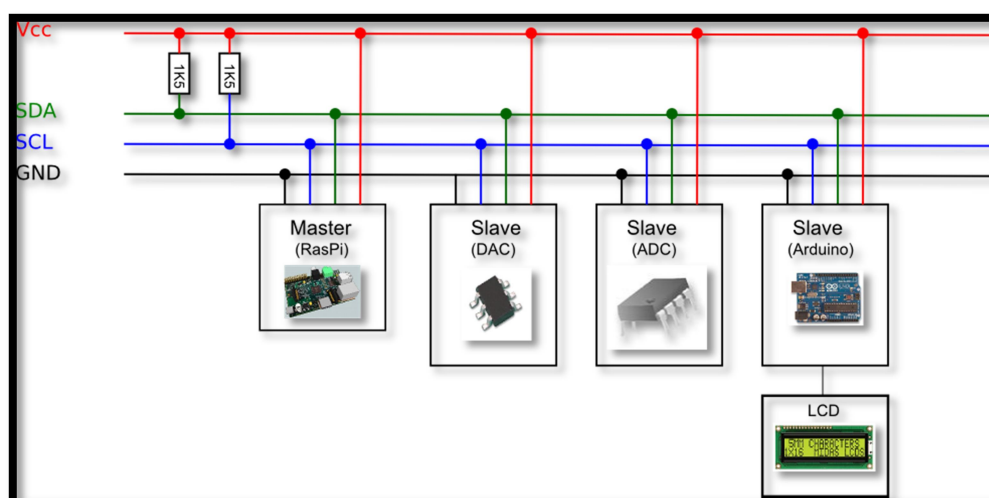


Fonte: <http://www.arduino.br.com/arduino/i2c-protocolo-de-comunicacao/>

SDA significa Serial Data e SCL significa Serial Clock. SDA é o pino que efetivamente transfere os dados, e SCL serve para temporização entre os dispositivos, de modo que a comunicação pela SDA possa ter confiabilidade. Como pode-se observar, tanto o envio quanto a recepção de dados são realizadas utilizando a linha SDA, ou seja, é uma linha bidirecional de comunicação, ora estamos enviando dados por este pino, ora estamos recebendo dados.

Existe uma variedade enorme de dispositivos que utilizam o protocolo I²C, como o próprio PIC, *Raspberry*, memórias externas (EEPROM), I/O expanders, RTC (*Real Time Clock*), Visor (LCD, TFT, etc), sensores diversos (temperatura, acelerômetro/giroscópio). Além do SDA e SCL, os dispositivos utilizam o terra (GND) e o Vcc para alimentação. A Figura 10 mostra um exemplo de ligação mais real [8].

Figura 10 - Exemplo de um barramento real



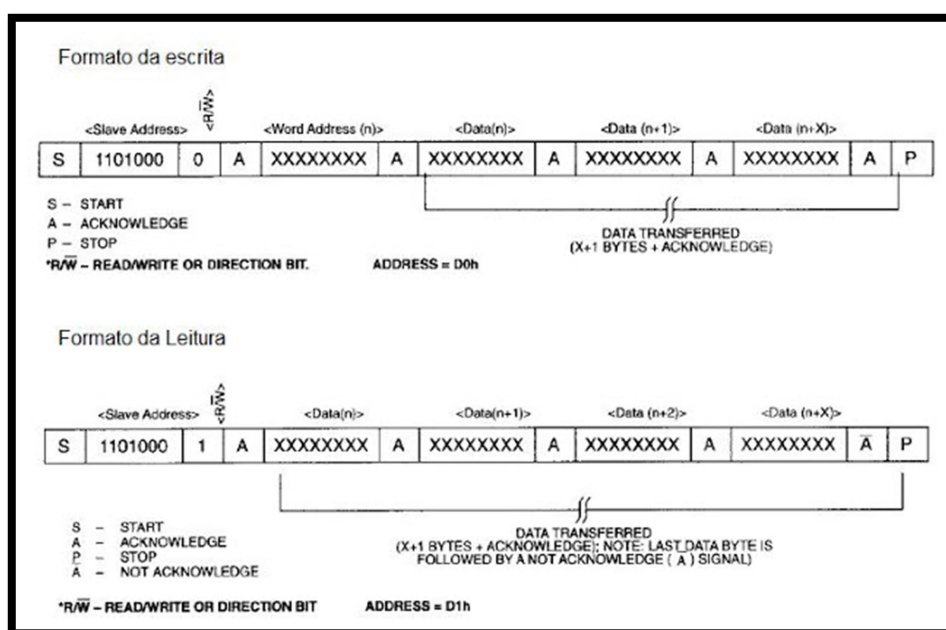
Fonte: <http://www.arduino.br.com/arduino/i2c-protocolo-de-comunicacao/>

Repare na imagem o uso de resistores *Pull-Up* para as linhas SDA e SCL. Estes resistores normalmente são utilizados quando há mais de um slave no barramento, e o valor do resistor é dependente dos dispositivos. Normalmente dispositivos I²C possuem um *Datasheet*, e nele podemos encontrar o valor adequado dos resistores *Pull-Up*. Se estiver utilizando apenas um dispositivo slave conectado ao PIC, não há necessidade de colocar visto que o Pic já possui resistores internos próprios para a comunicação I²C. Adotou-se os resistores de 10KΩ no projeto.

O circuito integrado RTC (*Real Time Clock*), é um circuito capaz de fornecer informações completas de data, como dia da semana, dia do mês, ano e além de informações de horas, minutos e segundos, tanto no formato 12 ou 24 horas. Vale lembrar que meses com 31 dias e anos bissextos são ajustados automaticamente. É necessária uma bateria de Lítio para preservar as informações mesmo se o sistema for desligado.

Basicamente, o DS1307 possui 7 registradores internos destinados a contagem da data (dia da semana, dia do mês, mês e ano) e tempo (segundos, minutos e horas), então é necessário indicar qual informação se deseja na leitura do protocolo I²C, a Figura 11 exemplifica.

Figura 11 - Exemplo de Leitura e Escrita

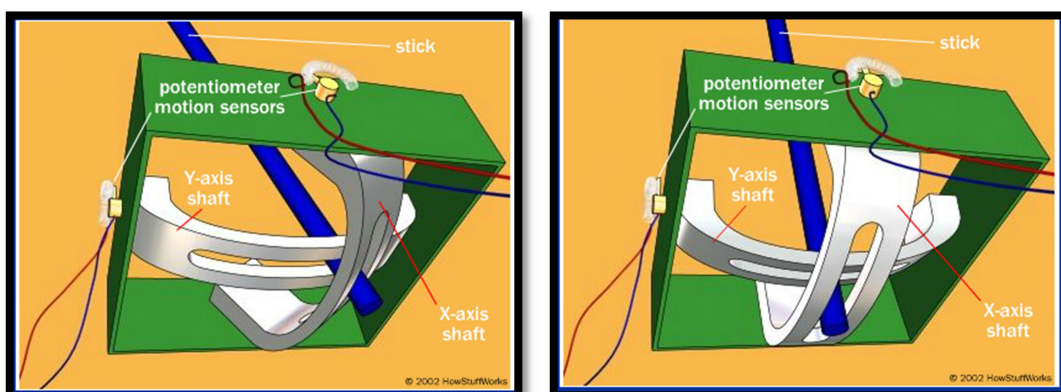


2.4. Módulo Joystick

Segundo a Nintendo (1997), “Joystick é um periférico de computador e videogame pessoal ou um dispositivo geral de controle que consiste em uma vara vertical na qual os pivôs se aproximam de uma extremidade e transmitem seu ângulo em duas ou três dimensões a um computador ou controlador”. A fim de movimentar o Braço Robótico do projeto, o *Joystick* precisa medir a posição do bastão nos eixos: o eixo X (esquerda ou direita) e o eixo Y (para cima e para baixo). Assim como na geometria básica, as coordenadas X-Y determinam exatamente a posição do bastão.

No design do *Joystick* padrão, o bastão move uma barra fina que se apoia em duas hastes com fenda que podem girar. Inclinar o bastão para frente e para trás faz a haste do eixo Y girar de um lado para outro. Incliná-lo para a esquerda ou direita faz a haste do eixo X girar sobre si mesmo. Quando você move a barra na diagonal, ambas as hastes giram. Várias molas centralizam o bastão quando ele é solto. Para determinar a localização do bastão, o sistema de controle do *Joystick* simplesmente monitora a posição de cada eixo. O design analógico convencional faz isso com dois potenciômetros, ou resistores variáveis. A Figura 12 mostra um arranjo típico.

Figura 12 - - Eixo X e Y de um joystick



Fonte: <https://student.dei.uc.pt/~rpcar/joysticks/>

Cada potenciômetro consiste de um resistor, na forma de uma trilha curva e de um braço de contato móvel. A alimentação elétrica do controlador conduz

eletricidade ao terminal de entrada, através do resistor curvo e do braço de contato, e de volta à porta do *Joystick* ao controlador.

Ao mover o braço de contato ao longo da trilha, aumenta-se ou diminui-se a resistência que atua sobre a corrente que flui através desse circuito. Se o braço de contato estiver na extremidade oposta da trilha em relação ao terminal da conexão de entrada, a eletricidade terá de fluir através de um comprimento maior do resistor e encontrará resistência máxima. Se o braço de contato estiver próximo do terminal de entrada, o potenciômetro terá resistência mínima.

Cada potenciômetro está conectado a uma das hastes do Joystick, de modo que girar a haste faz girar o braço de contato. Em outras palavras, se o bastão for empurrado para frente, ele irá girar o braço de contato do potenciômetro para uma extremidade da trilha; se puxá-lo para trás, ele irá girar o braço de contato na direção oposta.

Variar a resistência altera a corrente elétrica no circuito conectado. Desse modo, o potenciômetro traduz a posição física do bastão em um sinal elétrico que passa pela porta do Joystick rumo ao controlador.

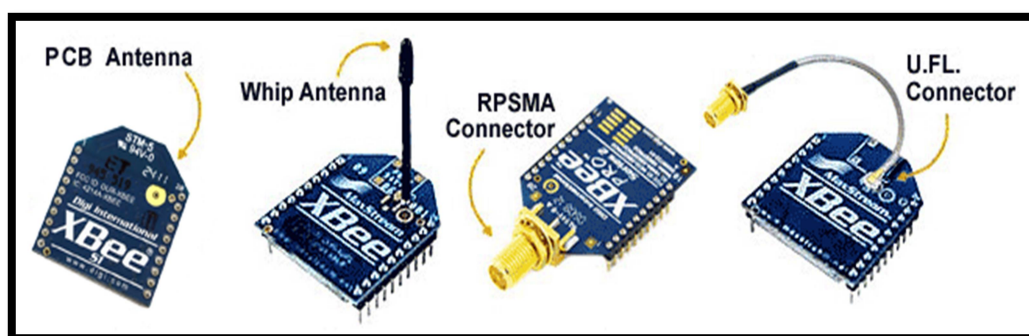
O Botão do *Joystick* funciona com um botão NA, no qual ao ser pressionado o contato é fechado (No projeto, foi necessária a utilização de um resistor de *Pull-Up* para eliminar ruídos ou estado flutuante).

2.5. Módulo de Rádio Frequência

Os dispositivos Xbee são módulos RF (Rádio Frequência) que fazem comunicações no padrão ZigBee IEE 802.15.4. O protocolo ZigBee permite comunicações robustas e opera na frequência ISM (*Industrial, Scientific Medical*), sendo aqui no Brasil 2,4 GHz (por isso não precisa de uma antena grande igual nos módulos 433MHz). As redes ZigBee oferecem uma excelente imunidade contra interferências com taxas de transferências de dados variando entre 20Kbps a 250Kbps. O Protocolo ZigBee é destinado a aplicações industriais, portanto, o fator velocidade não é crítico numa implementação ZigBee [9].

Além disso, os módulos RF padrão ZigBee foram criados para economizar o máximo de energia possível. Com isso, é possível criar aplicações de leituras de sensores remotamente apenas usando pilhas ou baterias comuns, isso porque quando os módulos não estão transmitindo/recebendo, entram num estado de dormência ou “sleep”, consumo mínimo de energia. A Figura 13 mostra exemplos de antenas RF [10].

Figura 13 - Exemplos de Antenas



Fonte: <https://goo.gl/aUDDSh>

2.6. Bateria

Dispositivos que trabalham com display LCD geralmente tem consumo relativamente alto de bateria, o que pode ser um calcanhar de Aquiles, principalmente em projetos que são portáteis. Por isso, optou-se pelas baterias de LiPo. As baterias LiPo (lítio-polímero) são geralmente mais seguras e mais amigas do meio ambiente do que outras baterias para Drones como NiCd e NiMH. As LiPo tornaram-se as baterias de alto desempenho mais comuns no mercado dos Drones e são utilizadas em veículos terrestres, barcos, aviões, helicópteros e muito mais. Essa bateria tem a capacidade de converter energia química em energia elétrica, e quando carregada converte energia elétrica em energia química. Basicamente faz-se esses processos sempre que a bateria é utilizada – carrega e descarrega.

A bateria utilizada possui 3 células ligadas em série, se totalmente carregadas a bateria fornece uma tensão de 12,6V, possuindo uma capacidade de 2200mAh, ou seja, se o circuito consumir uma corrente de 100mAh (no caso do controle desenvolvido nesse projeto), a bateria terá autonomia de fornecer carga

durante 22 horas continuamente, antes de precisar ser recarregada novamente. A Figura 14 mostra a bateria utilizada.

Figura 14 - Bateria utilizada

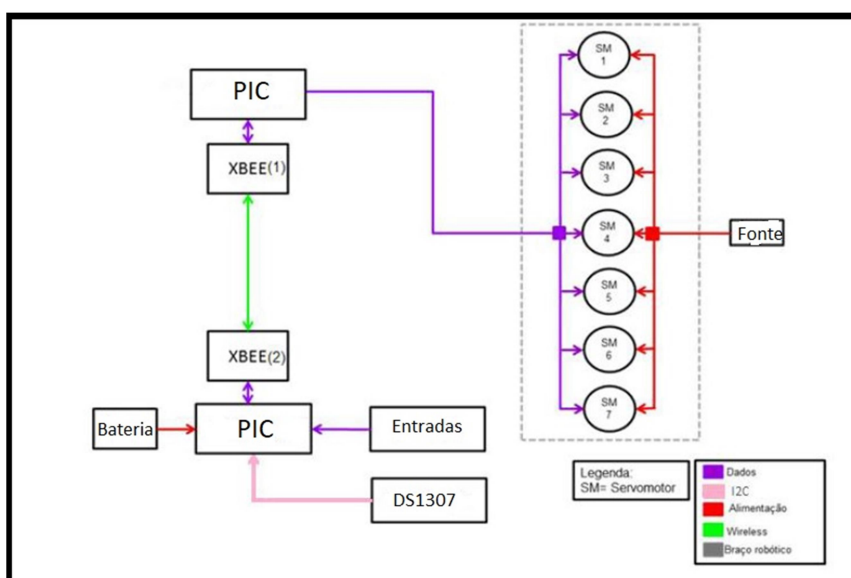


Fonte: Arquivo Pessoal

3. Arquitetura do Projeto

A Figura 15 é uma análise da arquitetura do projeto, onde se tem uma visão ampla de todo o esquemático.

Figura 15 – Arquitetura

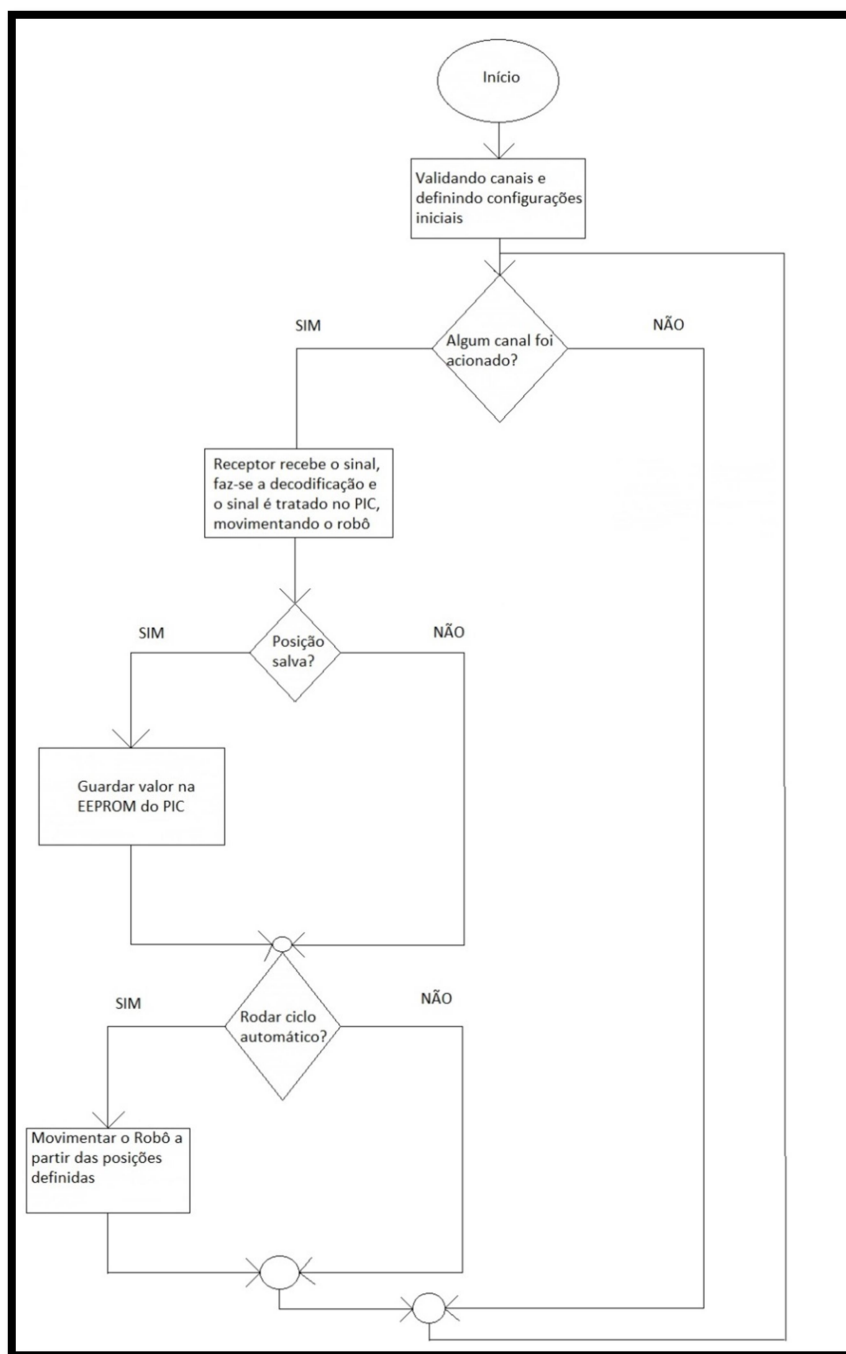


Fonte: Criado pela equipe

4. Fluxograma

Nessa parte, a Figura 16 ilustra como será desenvolvido o algoritmo de funcionamento do projeto, isto é, a lógica de funcionamento.

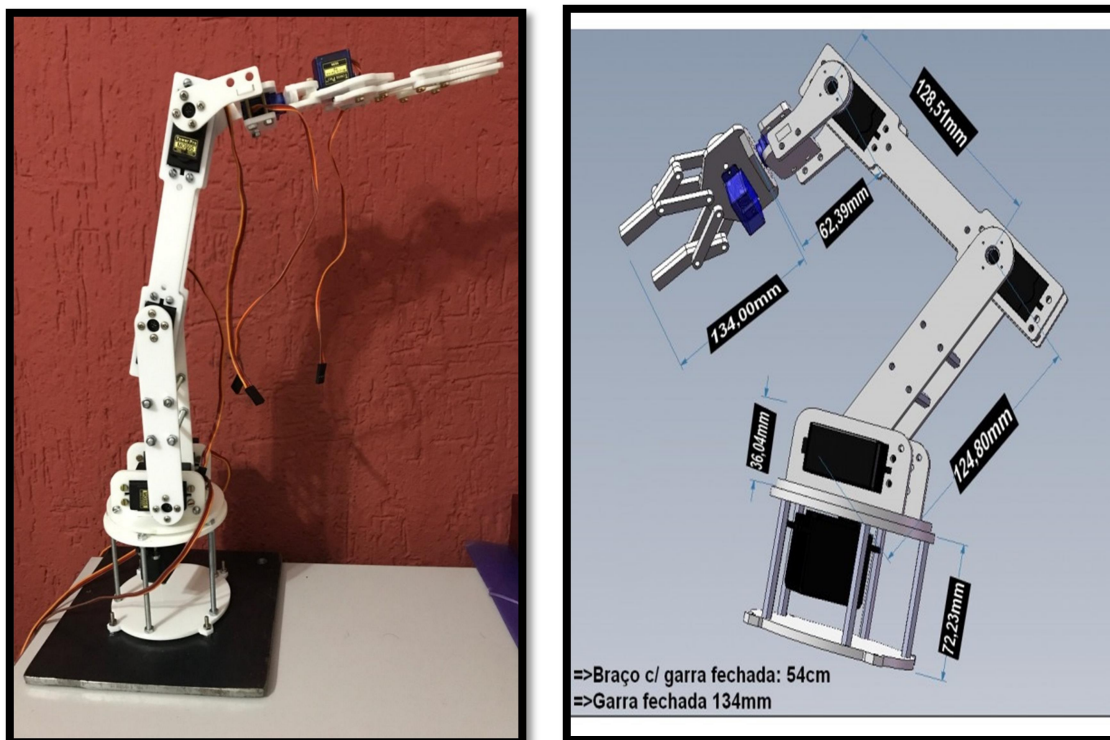
Figura 16 - Fluxograma de funcionamento do projeto



Fonte: Criado pela equipe.

5. Projeto Mecânico

Figura 17 - Protótipo do braço robótico e suas dimensões



Fonte: Arquivo Pessoal

O Protótipo mecânico (Figura 17 mostra o protótipo real e suas respectivas dimensões) consiste em um braço robótico de 6 graus de liberdades (Garra, Giro Garra, Antebraço, Cotovelo, Ombro e Base). Possui peso de aproximadamente 1,1 Kg e feito em acrílico.

Foi necessária uma modificação na base, onde utilizou-se de uma chapa metálica (20cm por 20cm, com uma espessura de 1,5cm) para fixar o braço robótico, para que não tombe quando estiver em movimentação.

Foram usados 5 Servos Motores do tipo TowerPro MG995 e 2 Servos Motores do tipo Micro Servo SG90.

É importante, também, saber o consumo de corrente de cada servo motor, para dimensionar a fonte de alimentação. Os valores de corrente que serão indicados na tabela foram dados obtidos através de um multímetro em duas situações de operação. A primeira com o Servo em repouso e a segunda com o Servo em movimentação. A Tabela 1 apresenta o consumo de corrente.

Tabela 1 – Consumo de corrente do braço robótico

	Parado	Em movimentação (Pequenas variações de graus)	Em movimentação (Grandes variações em graus)	Picos de Corrente
Eixo				
Servo motor da Base	180mA à 200mA	380mA	230mA	400mA à 500mA
Servo motor do Ombro*	500mA	700mA à 1A	500mA à 700mA	1,1A
Servo motor do Cotovelo	400mA	500mA à 600mA	350mA à 400mA	800mA
Servo motor do Antebraço	150mA	380mA	230mA	400mA à 500mA
Servo motor do giro da garra*	30mA	180mA	80mA	100mA à 200mA
Servo motor da Garra	30mA	180mA	80mA	100mA à 200mA

*O eixo do Ombro do Braço Robótico possui dois servos motores em paralelo, por isso seu consumo é praticamente.

Observação 1: Observando a tabela, pode-se concluir que para movimentação de graus próximos, isto é, de 10° para 20°, por exemplo, o consumo de corrente é maior do que para uma movimentação de 0° à 180°. Ou seja, para se obter uma precisão maior o Servo Motor acaba consumindo mais corrente para ajustes finos.

Observação 2: Observou-se que eventualmente na movimentação dos eixos do braço robótico há picos de corrente, no qual, ao longo do tempo esta corrente vai caindo até estabilizar.

Observação 3: Essa ressalva vale para todos os eixos, se os servos motores sofrerem esforço, seja por algum objeto que está movimentando ou pelo próprio chassi (um ângulo que esforce mais devido sobrepeso) há um grande aumento de corrente. Para evitar este efeito será limitado os ângulos de cada eixo.

Observação 4: Corrente total 2,7A.

6. Controle Remoto

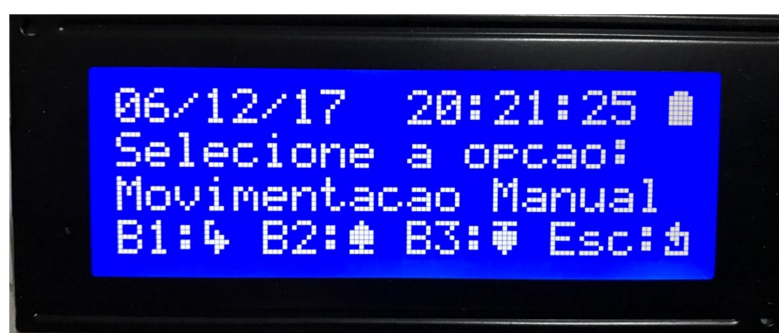
O controle remoto do braço robótico será o meio no qual o usuário fará as interações com o objetivo de movimentar o robô, inicialmente na forma manual e posteriormente na forma automática.

Existem 4 grandes circuitos no controle remoto que serão detalhados daqui em diante: o primeiro é a IHM (Todos os Menus e as interações através dos botões); O segundo é o circuito do relógio; O terceiro é a bateria e o quarto é controlador com o módulo de comunicação sem fio.

6.1. IHM

Para a IHM será usado um display LCD 20x4 (4 linhas por 20 colunas). Serão 3 menus principais, onde o primeiro é o menu de movimentação manual (com 6 telas de submenus), o segundo é o menu de movimentação automática (com 3 telas de submenus) e o terceiro o ajuste de data e hora (com apenas 1 submenu). Para alternar entre os menus, o usuário tem a disposição 4 botões (*Enter*, *Up*, *Down* e *Esc*).

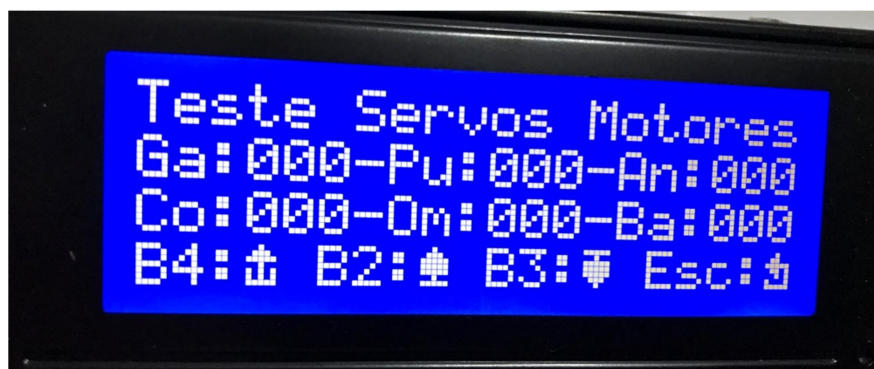
Figura 18 – Padrão de menu principal



Fonte: Arquivo Pessoal

Os menus principais seguem esse padrão de layout (conforme Figura 18), onde na primeira linha estão as informações de data e hora; na segunda linha a opção a ser selecionada (com a terceira linha com a respectiva função) e a última linha indicando quais os botões serão usados para a navegação e interação com o sistema.

Figura 19 - Padrão de submenu



Fonte: Arquivo Pessoal

Já os submenus possuem esse layout (conforme Figura 19), onde as 3 primeiras linhas são dedicadas as informações e a última usada para indicar quais os botões serão usados para navegação e interação com o sistema.

6.2. Circuito do Relógio

A comunicação entre o RTC (*Real time clock*) e o PIC é feita através do protocolo I²C, então serão usados dois pinos o SCL e o SDA.

Basicamente, serão feitos processos de leitura e escrita de acordo com o objetivo da informação, ou seja, se for desejado à leitura de data e hora será feito um processo de leitura I²C e se for desejado o ajuste de data e hora será feito um processo de escrita I²C.

Em relação à leitura I²C, de acordo com o *Datasheet*, o endereço a ser escrito em hexadecimal é "0xD1", então escreve-se o primeiro endereço (endereço do segundo) e termina o comando com o valor "1", indicando que será continuado a leitura, após chegar no último endereço (endereço 7), finaliza-se a leitura [11].

Figura 20 - Exemplo de função para leitura I²C

```
// =====
// ----- Função para Ler no DS1307 (Leitura no protocolo I2C) -----
void Leitura_I2C(){
    I2Cl_Start();           //Envia um sinal de START
    I2Cl_Wr(Escriver);      //Endereço no DS1307 para fazer Escrita
    I2Cl_Wr(0x00);          //Endereço Inicial
    I2Cl_Repeated_Start();  //Restarta o barramento
    I2Cl_Wr(Ler);           //Endereço no DS1307 para ler
    segundos = I2Cl_Rd(1);  //Lê o 1° endereço e informa que irá continuar leitura
    minutos = I2Cl_Rd(1);  //Lê o 2° endereço e informa que irá continuar leitura
    horas = I2Cl_Rd(1);    //Lê o 3° endereço e informa que irá continuar leitura
    semana = I2Cl_Rd(1);   //Lê o 4° endereço e informa que irá continuar leitura
    dia = I2Cl_Rd(1);      //Lê o 5° endereço e informa que irá continuar leitura
    mes = I2Cl_Rd(1);      //Lê o 6° endereço e informa que irá continuar leitura
    ano = I2Cl_Rd(0);      //Lê o 7° endereço e informa que não irá continuar a leitura
    I2Cl_Stop();           //Envia um sinal de Stop
} //end Leitura I2C
```

Fonte: Criado pela equipe

Toda atualização de data e hora é necessário uma chamada dessa função (Figura 20 exemplifica a função). Lembrando que as informações estão no formato BCD, então antes de atualizar o display é necessário converter o formato de BCD para String. Além disso, a função é do tipo void, ou seja, ela não retorna nenhuma valor e não recebe nenhum valor, por isso todos os endereços são lidos.

Em relação à escrita I²C, toda vez que é necessário fazer um ajuste é necessário chamar a função escrita_I²C. O endereço de escrita em hexadecimal é “0xD0”.

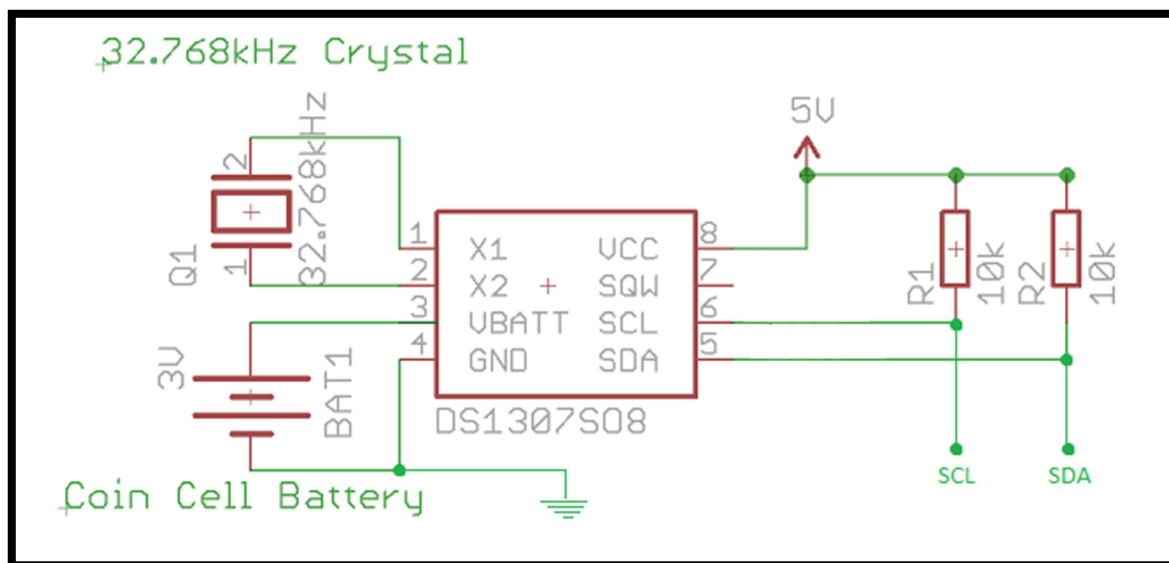
Figura 21 - Exemplo de função para escrita I²C

```
// =====
// ----- Função para Escrever no DS1307 (Escrever no protocolo I2C) -----
void Escrita_I2C(unsigned short int registro, unsigned short int valor){
    I2Cl_Stop();           //Caso barramento estiver ocupado envia um sinal de STOP
    I2Cl_Start();          //Envia um sinal de START
    I2Cl_Wr(Escriver);      //Endereço no DS1307 para fazer Escrita
    I2Cl_Wr(registro);      //Endereço do Registro que vai escrever
    I2Cl_Wr(valor);         //Valor que se quer escrever
    I2Cl_Stop();           //Envia um sinal de Stop
} //end Escrita I2C
```

Fonte: Criado pela equipe

Já nesse caso, a função recebe parâmetros do campo que será alterado e o valor que será escrito, com isso pode-se alterar apenas um campo específico (seja segundo ou mesmo apenas o ano), a Figura 21 exemplifica a função.

Figura 22 - Circuito básico do DS1307



Fonte: <https://goo.gl/eZmpN1>

A Figura 22 mostra o circuito básico de montagem do DS1307. É necessário um cristal de 32.768KHz para obter a base de 1 segundo, uma bateria de 3V de lítio para manter a contagem do tempo mesmo com o circuito do controle desligado e dois resistores de *Pull-Up* nas linhas de comunicação SCL e SDA.

6.3. Bateria

A bateria de LiPo possui uma saída de 12,6V, por isso é necessário um regulador (7805) para alimentação do PIC e dos outros periféricos (DS1307, módulo Xbee).

Existem duas funções básicas implementadas no controle remoto para economia de bateria: a primeira é a leitura do nível de bateria (através de um divisor de tensão direto para um pino analógico do PIC) e a segunda é apagar o Backlight do display por inatividade.

O divisor de tensão tem uma proporção no qual para 12,6V na entrada, a saída tem uma equivalência para 4,8V, dessa forma, o microcontrolador é capaz de fazer a leitura analógica e controlar o nível de bateria do controle remoto e informar ao usuário.

Já a temporização do Backlight é feita através do Timer1, onde após o estouro da contagem uma variável é incrementada. Se o controle remoto ficar 2 minutos sem uso, o Backlight é desligado através do chaveamento de um transistor, para voltar com o Backlight é só pressionar qualquer botão. O consumo de corrente diminui com essa função (de 100mAh cai para 89mAh) consequentemente aumenta a autonomia da bateria.

6.4. Controlador e módulo RF

Algumas configurações foram feitas no PIC18F4520. Primeiramente, a configuração do canal analógico do utilizando 5 pinos como entrada analógica (4 para leitura dos Joysticks e 1 para o divisor de tensão da bateria). Os registradores responsáveis pela configuração do canal analógico são: ADCON0 e o ADCON1. A Figura a seguir informa como realizar as configurações dos registradores.

Figura 23 - Configuração dos registradores ADCON0 e ADCON1

```
ADCON0 = 0x13; //Seleciona Canal AN0 até Canal AN4 e liga Conversor
ADCON1 = 0x0A; //Configura AN0 a AN4 como canal analógico
```

Fonte: Criado pela equipe

Configurando (de acordo com a Figura 23) um valor “0x13” para o registrador ADCON0 (em binário o valor é equivalente a 0001 0011), liga-se o conversor analógico, coloca-se a conversão em andamento e seleciona apenas do canal AN0 até o canal AN4 para ser lido.

Configurando um valor “0x0A” para o registrador ADCON1 (em binário o valor é equivalente a 0000 1010), define-se apenas os pinos 2,3,4,5 e 7 como analógicos.

Outra sacada interessante foi utilizar os resistores de Pull-Up interno do PORTB do PIC, ou seja, não foram necessários componentes externos junto com os botões. O registrador utilizado para setar os Pull-Ups internos é o INTCON2.

Figura 24 - Configuração de Registrador INTCON2

```
INTCON2 = 0x7F; //Habilita resistores de Pull-Up (Registrador RBPU_bit)
```

Fonte: Criado pela equipe

Configurando (de acordo com a Figura 24) um valor de “0x7F”, habilita-se os resistores de Pull-Up interno.

No projeto utilizou-se 3 interrupções: Timer0 (Leitura dos botões), Timer1 (temporização do Backlight) e Interrupção da comunicação EUSART (Comunicação Serial).

Figura 25 - Configuração de Registrador INTCON

```
//Habilita interrupção Global
INTCON = 0xE0;           //Interrupção Global, Periféricos e TIMER0
```

Fonte: Criado pela equipe

O primeiro passo é habilitar a interrupção global, a dos periféricos e o Timer0, para isso é preciso setar “0xE0” (1110 0000 em binário) no registrador INTCON, conforme mostra a Figura 25.

A primeira interrupção a ser declarada é a do Timer0. Essa interrupção pode ser configurada com 8 ou 16 bits.

Figura 26 - Configuração do Timer0 bit a bit

```
//Inicialização da Interrupção do Timer0
TMR0ON_bit = 0x01;           //Habilita o Timer0
T08BIT_bit = 0x01;           //Defino o Timer para 8 bits
T0CS_bit = 0x00;             //Define como incremento interno
T0SE_bit = 0x00;             //Transição de Low to HIGH
PSA_bit = 0x00;              //Ativa o Prescaler
//Prescaler de 1:256
TOPS2_bit = 0x01;
TOPS1_bit = 0x01;
TOPS0_bit = 0x01;
```

Fonte: Criado pela equipe

Configurou-se a interrupção do Timer0 para 8 bits. A prioridade da interrupção é baixa e tem por objetivo realizar a leitura dos botões do controle remoto, conforme mostra a Figura 26.

A segunda interrupção a ser configurada é a do Timer1. Essa interrupção é necessariamente de 16 bits.

Figura 27 - Configuração do Timer1 bit a bit

```
//Inicialização da Interrupção do Timer1
TMR1ON_bit = 0x01;           //Habilita o Timer1
RD16_bit = 0x01;             //Define o Timer para 16 bits
T1RUN_bit = 0x00;            //Oscilador do Timer não é fonte do oscilador do sistema
//Prescaler de 1:8
T1CKPS1_bit = 0x01;
T1CKPS0_bit = 0x01;
T1OSCEN_bit = 0x00;          //Oscilador externo desativado
T1SYNC_bit = 0x00;          //Clock sincronizado com o do sistema
TMR1CS_bit = 0x00;          //Clock interno
```

Fonte: Criado pela equipe

O objetivo da interrupção do Timer1 é o controle temporizado do Backlight e também possui uma prioridade baixa, conforme mostra a Figura 27.

A terceira interrupção é da comunicação serial. Essa interrupção possui prioridade alta e é de extrema importância para a comunicação sem fio do projeto. A configuração é setada bit a bit, definindo Baud Rate, configurando os pinos TX (RC6) como saída e RX (RC7) como entrada.

Figura 28 - Configuração da EUSART bit a bit

```
//Configuração da EUSART (Comunicação Serial)
SPBRG = 0x81;               //Baud Rate de 9600 (Cristal de 20Mhz)

//Registrador TXSTA
TXEN_bit = 0x01;           //Habilita Transmissão
BRGH_bit = 0x01;           //Habilita Baud Rate em alta velocidade
SYNC_bit = 0x00;           //Habilita modo assíncrono

//Registrador RCSTA
SPEN_bit = 0x01;           //Habilitação da EUSART
CREN_bit = 0x01;           //Habilita Interrupção contínua

//Configuração da Interrupção EUSART

//Registrador PIR1
RCIF_bit = 0x00;           //Limpa flag de sinalização de recepção de Caractere

//Registrador PIE1
RCIE_bit = 0x01;           //Habilita interrupção da EUSART
```

Fonte: Criado pela equipe

Configurações da comunicação Serial, conforme mostra a Figura 28.

Desenvolveu-se um protocolo de comunicação para o projeto com o objetivo de reduzir os ruídos intrínsecos do ambiente, dessa forma, qualquer tipo de caractere indesejável não afeta a comunicação RF entre o controle remoto e o gabinete do Robô.

Basicamente, o protocolo consiste em enviar os comando entre “[]”, além de um bit de verificação “V”, dessa forma, qualquer caractere fora disso será descartado.

Por exemplo, caso queira-se enviar um comando para a movimentação da garra do Braço Robótico: envia-se o início de protocolo “[“, após isso indica qual o eixo será movimentando, no caso “GAR”, em seguida o valor em relação ao Duty Cicle do PWM emulado (será abordado mais para frente), um bit de verificação “V” e posteriormente o fim de protocolo “]”. Tudo isso fará parte de uma String “[GARXXV]”, no qual será interpretada apenas a informação entre colchetes, exemplificando isso a Figura 29 mostra a lógico do algoritmo.

Figura 29 - Exemplo do Protocolo de comunicação

```

aux_recebimento = RCREG;

if( aux_recebimento == Init_Protocol){
    InitProtocol = TRUE;
    cntBuffer = 0;
} //end if

else{
    if( aux_recebimento == End_Protocol){
        InitProtocol = FALSE;
        for(cnt = 0; cnt < cntBuffer; cnt++){
            TXREG = Buffer[cnt];
            Testbuff();
        } //end for
        TXREG = 0x0D;          //CAR Return
        Testbuff();
        TXREG = 0x0A;          //Line feed
        Testbuff();

        //Limpa o Buffer após os testes
        for(cnt = 0; cnt < cntBuffer; cnt++){
            Buffer[cnt] = 0;
        } //end for
    } //end if

    else if(InitProtocol){
        Buffer[cntBuffer] = aux_recebimento;
        cntBuffer++;
        aux_recebimento = 0;
    } //end if
} //end else
} //end Interrupção EUSART

```

Fonte: Criado pela equipe

O código fonte apresentado mostra como foi feita a lógica do protocolo de comunicação simples. Basicamente, é verificado o caractere que chegou à serial, se é um início de protocolo ou fim de protocolo. Caso seja um início de protocolo, começa a encher um Buffer com as informações posteriores até chegar um fim de protocolo. Quando encerrada a transmissão, é produzido um Echo da informação (Enviar o que foi recebido), há uma rotina de testes e após isso limpa-se o Buffer para a próxima informação ser executada.

7. Gabinete

A comunicação serial é configurada da mesma maneira do controle remoto, e ao tempo todo é feita uma comunicação bidirecional, ou seja, todo comando enviado pelo controle remoto o gabinete envia uma confirmação de volta.

Para a movimentação do braço robótico será utilizada a interrupção do Timer0 configurado para 8 bits, da mesma forma que foi configurado no controle remoto.

Os servos motores, como já foi dito, trabalham com uma leitura de 20 milissegundos a cada período, e o que vai determinar a sua posição é um pulso nesse período. O pulso tem que ser entre 1 milissegundo (0° grau) a 2 milissegundos (180° graus), e posições intermediárias precisam estar entre esse intervalo. Uma forma simples de controlar esse pulso seria por PWM (Modulação por Largura de Pulso), onde é possível alterar a largura através do *duty cycle* (termo utilizado para descrever a fração de tempo em que um sistema está num estado ativo) ou mesmo usar funções de “Delays” (Que são funções para ocupar tempo, em Assembly funções NOP – No Operation).

Porém, existem inconvenientes em usar essas opções, primeiro que no PWM, o PIC18F4520 possui apenas 2 pinos com essa função, ou seja, 2 pinos para controlar 6 eixos (6 Servos Motores), não sendo possível controlar a movimentação completa do Braço Robótico. Já as funções “Delays” possuem uma grande desvantagem que são fazer com que o Microcontrolador fique pausado na sua leitura de rotinas, ou seja, enquanto o tempo não acabar o código não será

executado, e caso seja necessário testar um botão de emergência onde sua leitura está no começo do código é preciso esperar o tempo se encerrar, o que é totalmente inviável.

A solução encontrada foi utilizando interrupção. A interrupção é uma chamada de procedimento, isto é, o PIC pausa a execução do código principal, executa toda a rotina de interrupção, depois volta para o código principal e continua a leitura. No caso, configurou-se para que a interrupção do TIMER0 estourasse a cada 20 milissegundos, e o pulso seria feito colocando o pino de saída do Servo Motor para nível alto e baixo, dependendo da posição desejada. E para alternar entre os eixos, usou-se uma função de Switch – Case, através de uma variável de controle. Essa solução utilizando interrupção acaba emulando um PWM.

8. Circuitos Elétricos

No circuito elétrico do controle remoto (apêndice A) é mostrado o resultado final com todas as melhorias propostas adicionadas. Display IHM integrado, facilitando ao operador identificar a posição de cada eixo conforme a movimentação, circuito integrado DS1307 no qual traz um preciso relógio, com possibilidades através de botões o ajuste de data e hora e um medidor de nível de bateria no qual usa o próprio microcontrolador, que realiza a leitura analógica da tensão da bateria e converte essa informação através de um símbolo.

O circuito elétrico do gabinete (apêndice B) pode ser dividido em dois blocos principais: alimentação e gerenciamento de dados. O primeiro devido cada eixo do braço possuir um servo motor com um consumo de corrente alto, se fez necessário separar e isolar sua alimentação. Um conector no qual é alimentado com 12V alimenta cinco reguladores de tensão que diminui essa tensão para 5Vdc e a garra, pulso, antebraço, cotovelo e base são alimentados isoladamente. Os servos motores do ombro, são alimentados por uma fonte externa de 5V já que a de 12V não possuía corrente o suficiente para fornecer a todos os eixos.

O bloco de gerenciamento de dados recebe as informações do transmissor e através de um protocolo de comunicação, devolve a confirmação do recebimento de dados, realiza o tratamento e executa as ações comandadas. Além

de salvar as posições na EEPROM do microcontrolador para caso haja falta de alimentação.

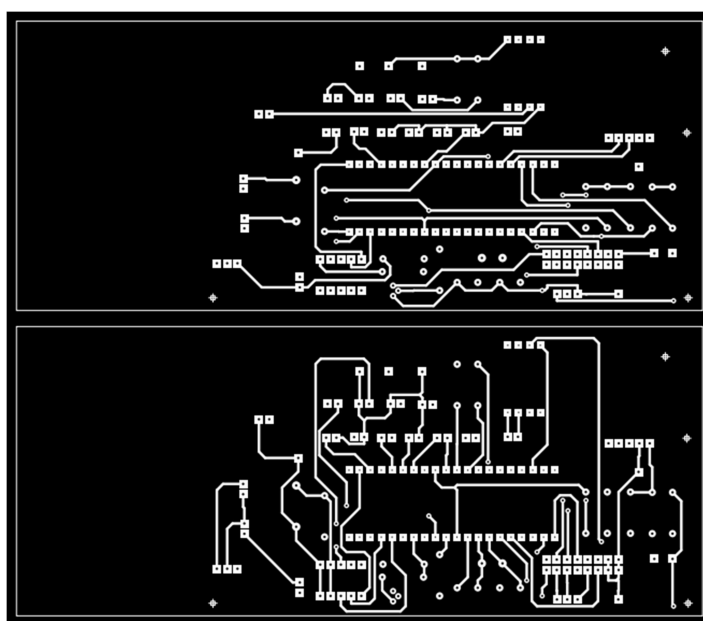
Para simular a comunicação serial no Proteus foi utilizado um terminal serial e para simular o PWM emulado foi utilizado um osciloscópio. Além disso, o *Backlight* temporizado foi representado por um Led, pois o *Display* disponível no Proteus não tem os pinos referentes à alimentação do Backlight.

Feito todos os testes dos circuitos montados em *Protoboards*, realizou-se a confecção das placas de circuito impresso. Para diminuir as dimensões das placas de circuito impresso optou-se por usar um layout com duas faces, isto é, há trilhas na vista superior e inferior da placa. Utilizou-se o laboratório da Fatec-SP de montagem eletrônica para a confecção.

9. Layouts e placas finais

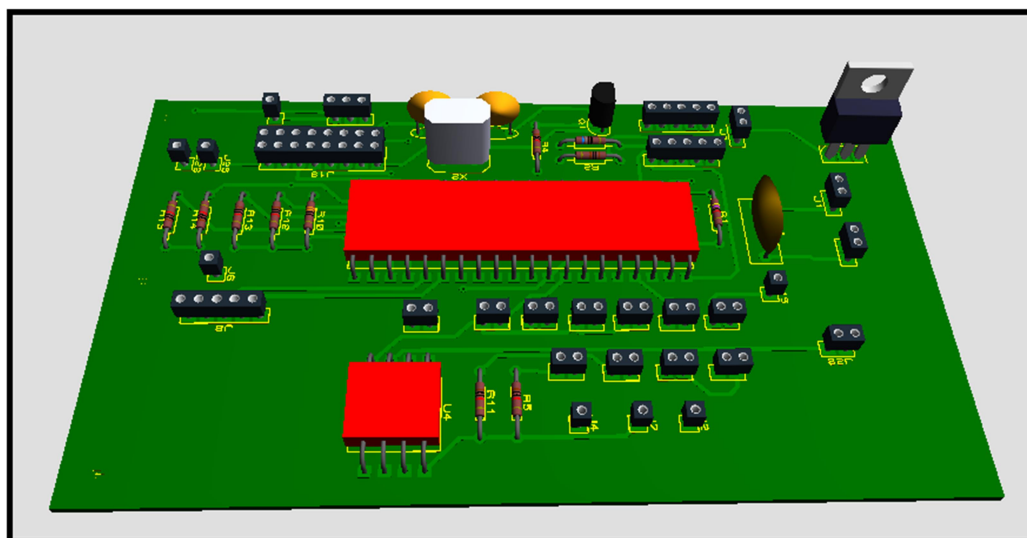
O layout (Figura 30) e a vista 3D (Figura 31) foram feitos no software Proteus. A vista 3D é uma ferramenta interessante, pois além de verificar o dimensionamento dos componentes ao longo da placa, fica mais fácil de ajustar na caixa (colocar os conectores do lado que facilite a conexão dos cabos flat).

Figura 30 - Layout controle remoto (Top e Bottom respectivamente)



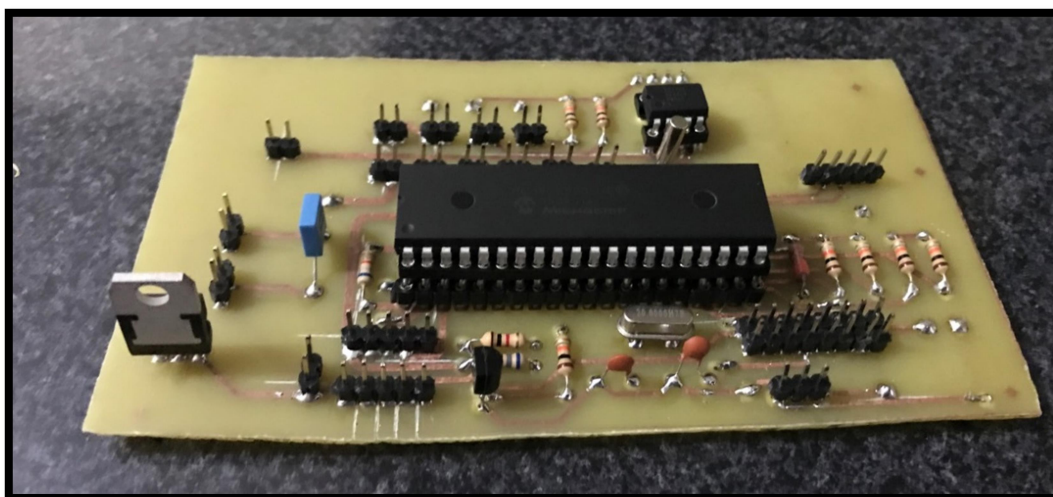
Fonte: Criado pela equipe

Figura 31 - Visão 3D da placa do controle remoto



Fonte: Criado pela equipe

Figura 32 - Placa final com os componentes soldados

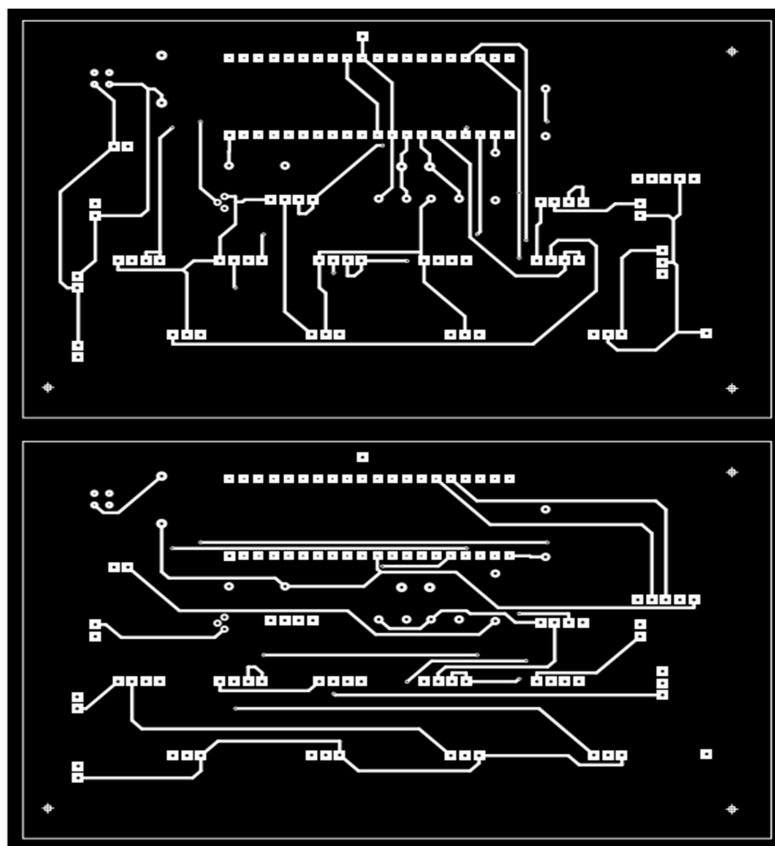


Fonte: Criado pela equipe

Pode-se observar que a representação 3D sugeriu um roteamento de componentes bem fiel à placa final (Figura 32). Basicamente, temos o controlador no meio, diversas barras de pino para conexão dos periféricos que vão à caixa, o circuito integrado do relógio com seu cristal e o regulador de tensão 7805.

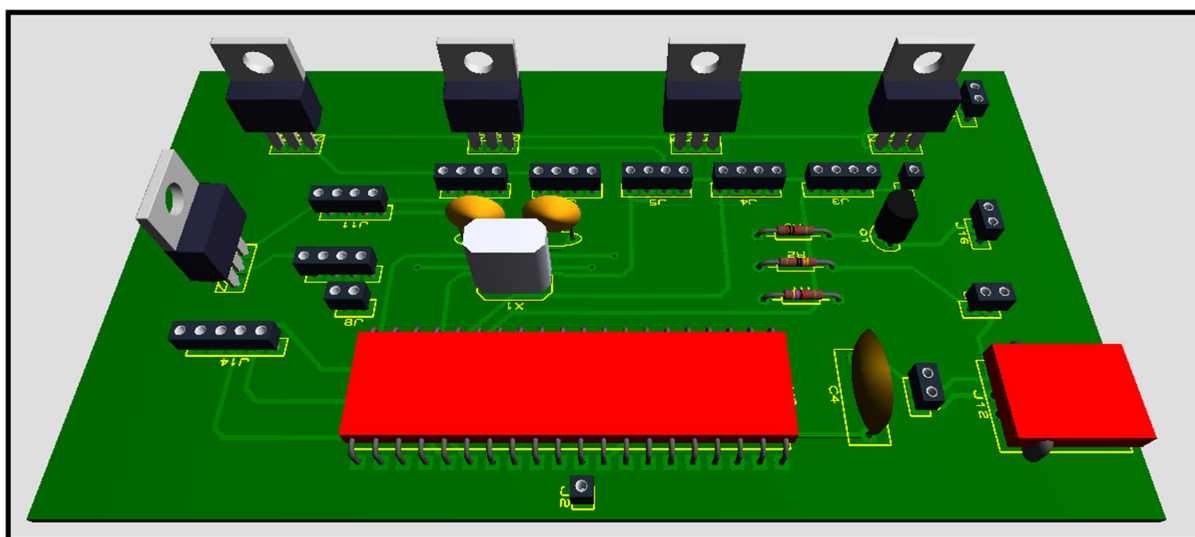
As Figuras 33, 34 e 35 mostram, respectivamente, o layout, a vista 3D, e a placa finalizada do circuito do gabinete.

Figura 33 - Layout gabinete (Top e Bottom respectivamente)



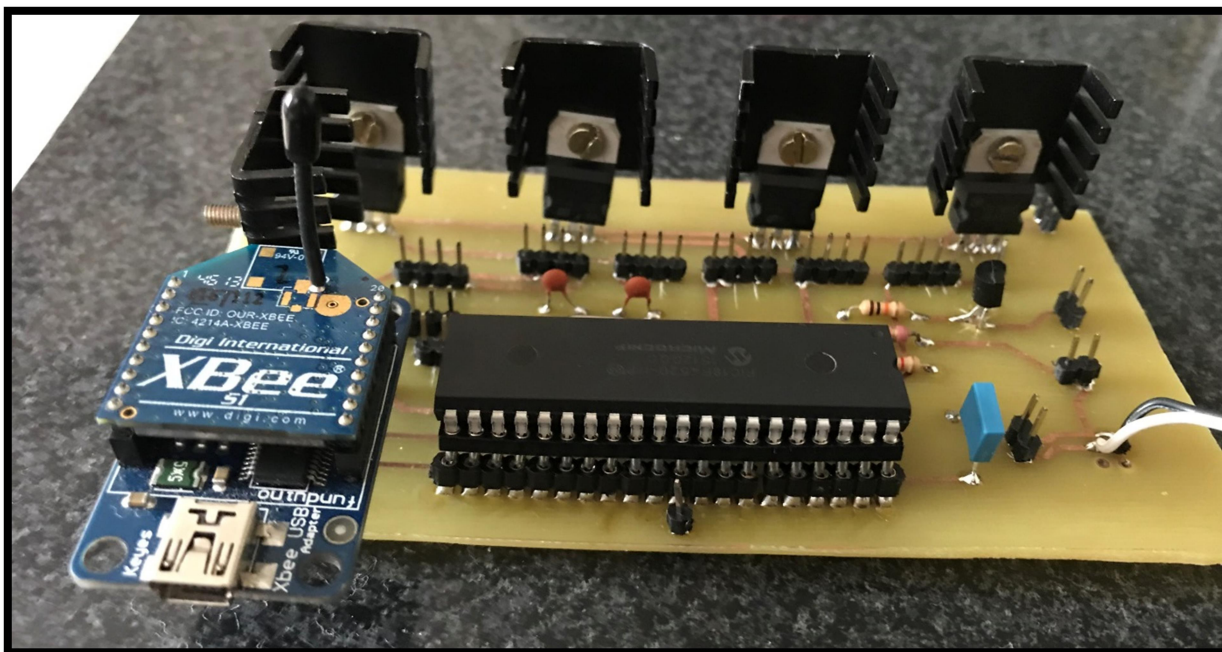
Fonte: Criado pela equipe

Figura 34 - Vista 3D da placa do gabinete



Fonte: Criado pela equipe

Figura 35 - Placa final com os componentes já soldados



Fonte: Criado pela equipe

10. Protótipos finais

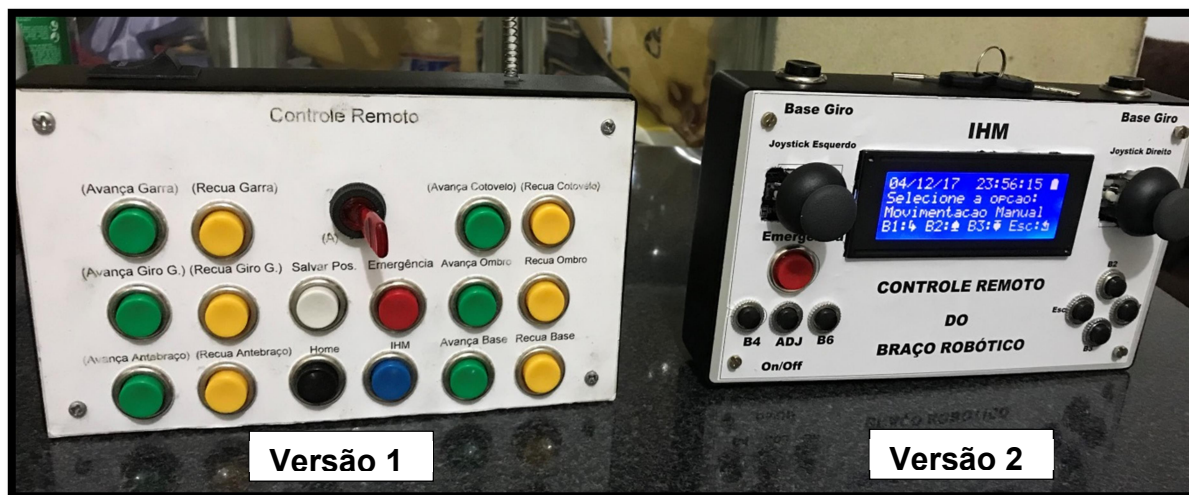
Os botões do protótipo do controle remoto foram distribuídos de forma que o usuário consiga manipular sem retirar a mão do controle (que consiga alcançar todos os botões). Ou seja, o controle remoto (Figura 36) foi desenvolvido pensando na ergonomia do usuário, além de ficar mais intuitivo que os exemplos apresentados na necessidade do projeto (Figuras 2 e 4).

Figura 36 - Protótipo final do controle remoto



Fonte: Arquivo Pessoal

Figura 37 - Comparação com a primeira versão



Fonte: Arquivo Pessoal

Observe a evolução dos Controles Remotos da primeira versão para a segunda versão (Figura 37). A segunda versão além de ser mais “clean”, é também mais agradável visualmente e mais intuitiva que a primeira. Ademais, a segunda versão possui uma chave para que apenas pessoas autorizadas manipulem o braço robótico.

Figura 38 - Protótipo do gabinete



Fonte: Arquivo Pessoal

Figura 39 - Evolução dos gabinetes



Fonte: Arquivo Pessoal

O tamanho do gabinete foi a maior diferença entre uma versão e outra. As Figuras 38 e 39 mostram, respectivamente, o gabinete final e a comparação entre as versões dos gabinetes.

11. Resultados e Discussões

Verificou-se o pulso do PWM emulado no TIMER0 com o osciloscópio. A seguir, a Figura 40 mostra o que foi obtido como resultado.

Figura 40 - Forma de onda do pulso PWM - 2ms e 1ms respectivamente



Fonte: Arquivo Pessoal

Pode-se observar que o PWM emulado se comporta muito bem para gerar os pulsos necessários para a movimentação dos Servos Motores. Essas ondulações na forma de onda podem ser facilmente eliminadas utilizando capacitores de desacoplamento (Poliéster – 100nF).

Testou-se diversas vezes a movimentação do Braço Robótico, a gravação das respectivas posições na EEPROM, a comunicação sem fio, e o resultado foi bastante satisfatório, o sistema funcionou perfeitamente, com o Robô executando todas as posições desejadas inúmeras vezes (entretanto, por ser um protótipo o Braço Robótico tem uma taxa de repetibilidade mediana) e após longos períodos.

Para as movimentações automáticas utilizou-se de laços “for” para diminuir a velocidade dos Servos e, conseqüentemente, não forçar os motores. A grande vantagem é que o torque se mantém. E por fim, para sair do modo automático, basta o botão de emergência ser acionado.

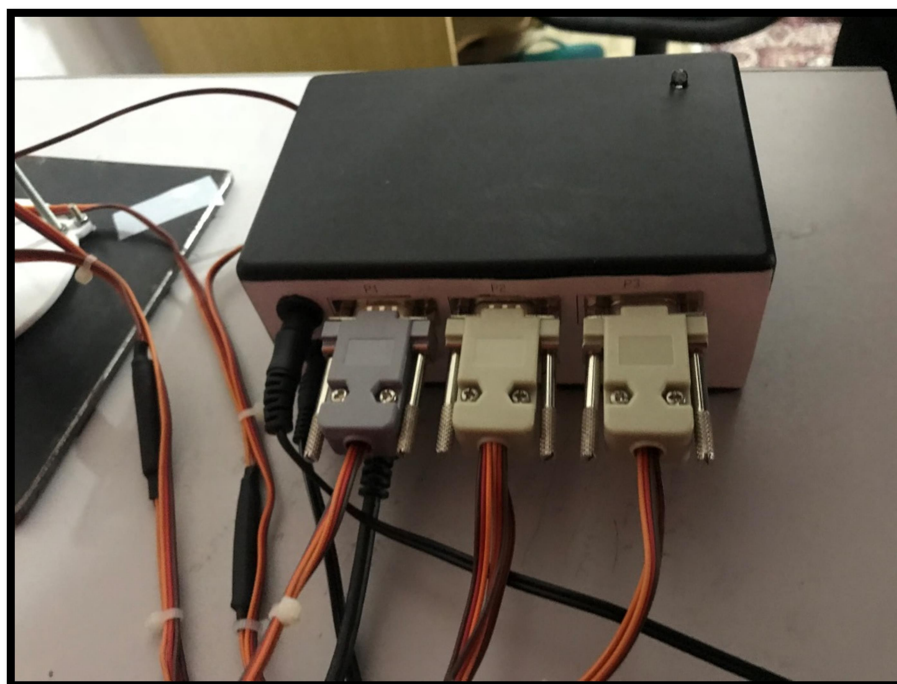
A seguir, a Figura 41 ilustra o protótipo final deste trabalho.

Figura 41- Protótipo final



Fonte: Arquivo Pessoal

Figura 42 - Conexões com o Braço Robótico utilizando DB9



Fonte: Arquivo Pessoal

Todos os protótipos foram pensados numa conexão “*Plug and Play*” (Figura 42), para facilitar a montagem em qualquer ambiente e com máxima organização possível do cabeamento. Além disso, todas as conexões possuem uma legenda para facilitar a o encaixe correto.

Com o sistema finalizado, é possível salvar até 5 posições para movimentação automática e o alcance da comunicação sem fio é de até 35 metros sem obstáculos e 10 metros com obstáculos. O funcionamento ficou dentro do esperado e ficou bem otimizado para o usuário final.

12. Conclusão

O desenvolvimento deste projeto teve por objetivo melhorar um sistema já existente na indústria, que é o seguimento dos braços robóticos. Trazendo facilidade para a programação e versatilidade já que sua rotina de trabalho pode ser modificada facilmente.

Com análises baseadas em modelos já existentes de braços robóticos (Figuras 1 a 4), foi desenvolvido no controle remoto tudo que se julgou necessário para fazer a movimentação segura e precisa do braço robótico, além de informações que caso não houvesse poderia interferir na funcionalidade, como nível de bateria, já que se acabasse no momento da programação interromperia o processo trazendo transtornos para o controlador.

Os resultados obtidos foram frutos de pesquisas e testes de como funcionaria um braço robótico na prática, apresentando nesse trabalho possíveis melhorias que poderiam ser aplicadas no setor industrial, trazendo padronização para a “programação”.

Claro que sempre há algo melhorar e implementar, devido a falta de tempo, não foi possível implementar por exemplo um controle remoto inteiramente touch-screen, que traria mais telas com informações e um visual mais limpo para o controlador.

Por fim, neste trabalho acadêmico apesar das adversidades encontradas ao longo do caminho, o resultado final como um todo foi de satisfatória gratificação, já que, foram realizados em cima dos estudos necessários, saídas nas quais através destas, possibilitou à criação de um protótipo do braço robótico radio controlado totalmente funcional.

13. Futuras Melhorias

O setor industrial têm diversos seguimentos e para cada setor poderia ser listado um tipo de melhoria a ser proposta, como por exemplo, no protótipo mecânico ao invés de garra utilizar ventosa, ou mesmo um modelo diferente de garra.

Mas algumas serviriam para um contexto geral como, a integração da IHM aos botões e *joystick*, isto é, desenvolver uma tela *touch screen*, de forma a aproveitar melhor o espaço e tornar o controle remoto mais moderno. Também há a possibilidade de desenvolver um sistema supervisor para controlar o processo remotamente e integram serviços de nuvem.

Todas essas possíveis melhorias ficam em aberto para os próximos projetos, ou até mesmo projetos baseados neste que foi apresentado. Certo é que a indústria precisa de constante evolução e com isso as tecnologias criadas devem responder a essa necessidade.

14. Bibliografia

[1], [2], [3], [4] NIKU, Saeed B., 2013, Ed. LTC.

[5] PEREIRA, Fábio. Microcontrolador PIC18 detalhado: Hardware e Software. 1ª Edição, Ed. Érica. São Paulo, 2010.

[6] Microchip Technology Inc., “PIC18F2420/2520/4420/4520 Data Sheet” – 2008 - DS39582B.

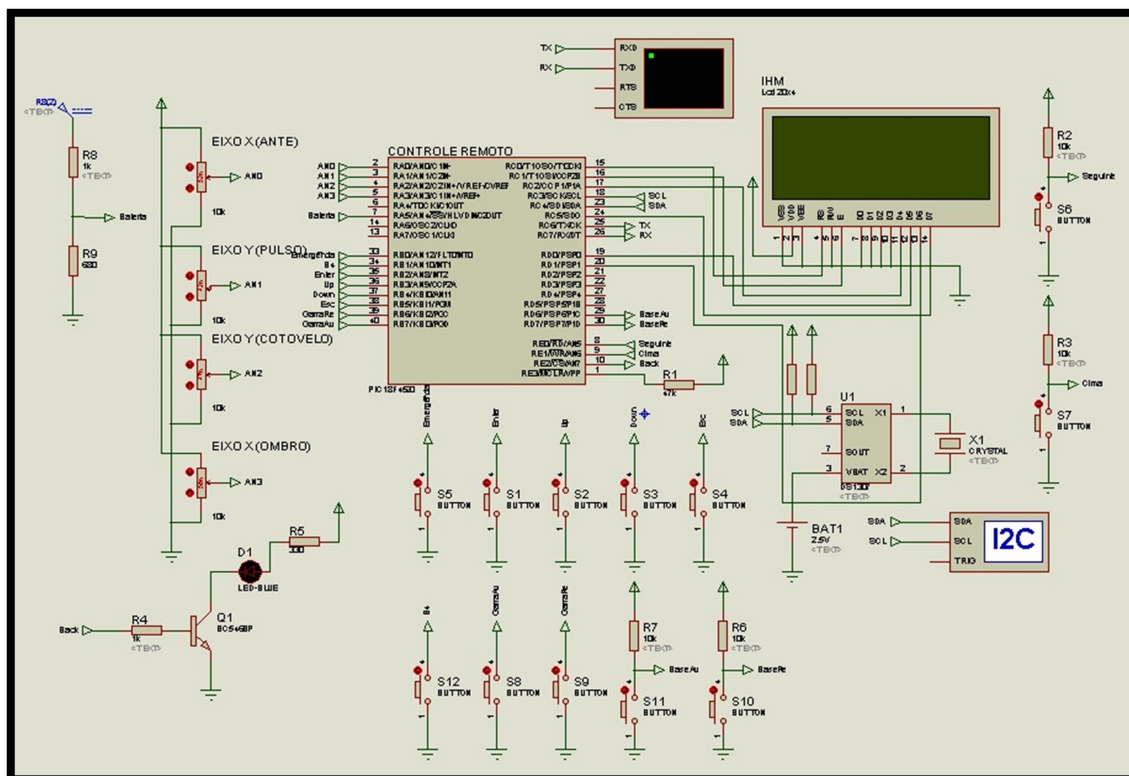
[7] SERVO MOTOR. Disponível em: <<https://www.citisystems.com.br/servo-motor/>>. Acesso em: 06 abr. 2017.

[8] PROTOCOLO DE COMUNICAÇÃO. Disponível em: <<http://www.arduino.br.com/arduino/i2c-protocolo-de-comunicacao/>>. Acesso em: 07 ago.2017.

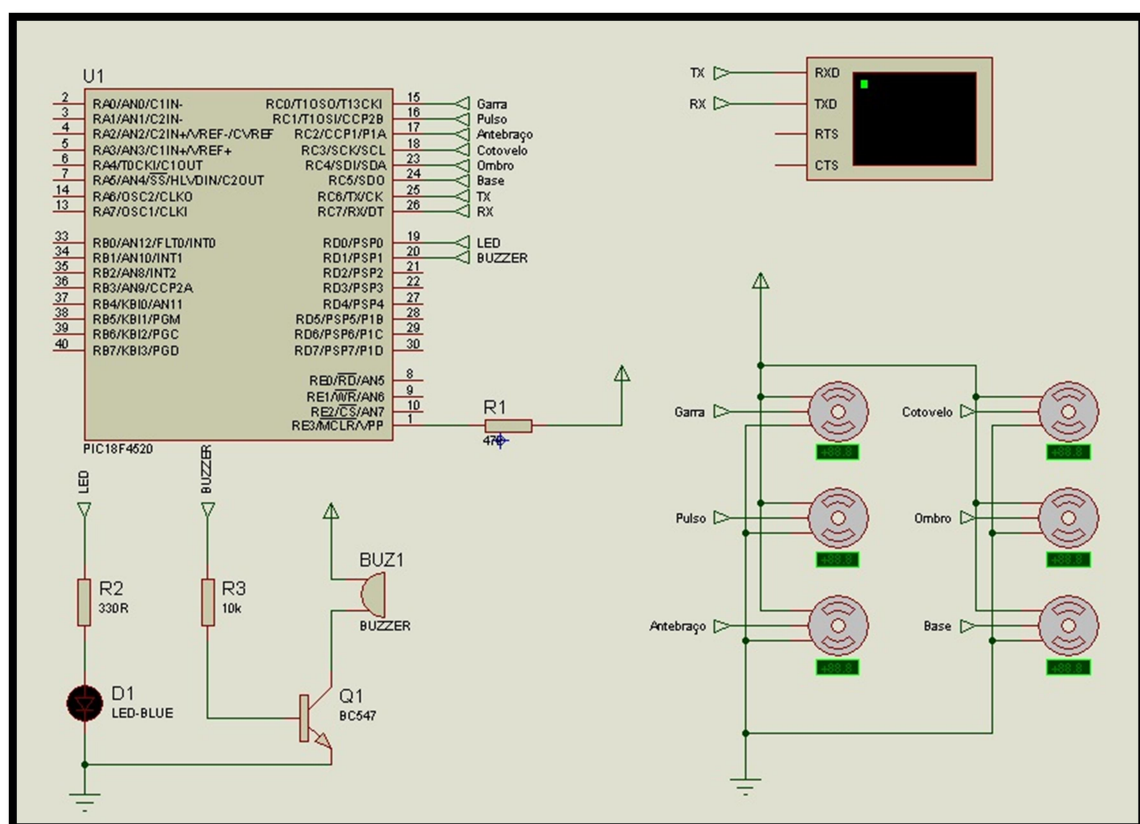
[9], [10], LABORATÓRIO DE GARAGEM. Disponível em: <<http://labdegaragem.com/profiles/blogs/tutorial-como-utilizar-o-xbee>>. Acesso em: 20 set.2017.

[11] Canal Wrkits Chanell, engenheiro Wagner Rambo. 2017. Disponível em: <<https://www.youtube.com/channel/UCazAvTtoRIOrFDWDJDB2DKQ>>. Acesso em: 07 nov. 2017.

15. Apêndice A – Circuito elétrico do controle remoto



16. Apêndice B – Circuito elétrico do gabinete



17. APÊNDICE C – Código fonte do projeto

Vide CD.