

PROJETOS IOT COM ARDUINO E RASPBERRY

Desenvolvendo um Robô

Publicação e Revisão 1.0

Autor:

Marcelo Maurin Martins

Sumário

Agradecimentos.....	16
Composição desta obra	17
Sugestões construtivas	18
O que há nos arquivos?	19
Eu precisarei ter todos os softwares para montar meu robô?	19
Atualizações.....	19
Erros e Inconformidades e Isenção de Garantia.....	20
Participação no Projeto	20
O que há de comum entre um robô e um dispositivo IOT.....	21
Entendendo o que é um Robô.....	22
Tipos de Ambientes	22
Construção genérica de um robô	23
Primeiros passos.....	24
1. Conceitos Básicos	25
1.1 Mecânica	25
SolidWorks	26
Gerando arquivos STL para impressão no SolidWorks 2014.....	27
Impressoras 3D	29
Como funciona uma impressora 3D	29
Software de Impressão	30
Montagem de peças utilizando o software XYZWARE	30
Cartucho de Filamento	31
Cálculos Mecânicos.....	32
Dimensionamento dos Motores	32
Cinemática - Estudo do movimento.....	33
Cálculo de Força Trativa	34

Torque.....	35
Cálculo da Força Trativa	36
Torque de Tração Máxima.....	37
Cálculo de força para aclive	38
Força de Atrito de Rolamento.....	39
Força de Arrasto Aerodinâmico	40
Cálculo de Esforço de Arrasto	41
Cálculo de Esforço dos Braços	42
Analizando problemas envolvidos	43
Cálculo de esforço.....	45
1.2 Básico sobre Eletrônica	46
Energia	46
Eletrônica de Potência	47
Dimensionamento da rede elétrica do robô.....	49
Cálculo de Tensão/Corrente e Resistência.....	49
Entendendo o Capacitor	51
Cálculo de Capacitor	52
Cálculo de Potência	53
Como estimar a potência de um circuito de forma prática e com poucos cálculos.....	53
Exemplo de Cálculo de Potência	54
Potência Aparente, Reativa e Ativa.....	54
Cálculo de Consumo	56
Ligações em Série ou em Paralelo de Bateria	58
Entenda porque funciona assim	59
Ferramentas básicas para eletrônica	60
Multímetro	60
Ferro de Solda.....	60
Estanho	60

Dry film	61
Película de Transparência jato de tinta para retroprojetor	61
Percloreto de Ferro	62
Placas de PCB.....	62
Eagle	63
1.3 Software	65
Tipos de Desenvolvimento	66
Ações voluntárias e involuntárias	67
Projetando o equipamento.....	67
Programação no Arduino.....	68
Download	69
Configuração do ambiente	70
Programação no Raspberry	73
Sistema Operacional	74
Download	74
Como instalar o Raspbian OS.....	74
Acessando.....	75
Instalando pacotes de aplicações	75
Ferramentas Auxiliares	76
Filezilla	76
mRemoteNG	77
Desenvolvimento em C para Raspberry.....	78
Makefile	79
Muitas bibliotecas, onde estão.....	81
PHP	82
Desenvolvimento em Windows e Linux X86.....	82
Lazarus	83
Bibliotecas.....	84

2. Construindo seu robô	85
2.1 Mecânica da Base do Robô.....	85
Base Inferior	85
Arquivos e Modelos	85
Base Superior.....	86
Arquivos e Modelos	87
Rodas	87
Motor com caixa de redução	88
Suporte para motor	88
Prolongadores da Base	91
Montagem mecânica	92
Procedimento de Montagem.....	92
2.2 Eletrônica.....	93
Planejamento da Fonte Externa	93
Projeto de Fonte AC/DC.....	94
Dimensionamento de um Capacitor	95
Projeto de um carregador de bateria	97
Entenda como funciona uma bateria:	97
Tipo de Metal.....	97
Valor de Tensão por Pilha	98
Capacidade	98
Carregador de Bateria.....	99
Esquema Elétrico de Recarga.....	100
Bateria Recarregável	101
Suporte de Porta Pilhas	102
Sensor de Tensão	103
Como é usado o sensor de tensão.....	105
Relê	106

Círculo do Carregador.....	108
Entrada de 12 V	108
Step Down	108
Bateria	108
Sensor de tensão	108
Relê	108
Saída	108
Placa do circuito do carregador	109
Calculando as especificações do step down	111
Como calibrar o step down.....	113
2.3 Software – Projeto do Software.....	114
Firmware.....	116
Bloco de Execução do Arduino	117
Bloco de funções de Setup do Arduino.....	118
Bloco de funções de Loop do Arduino	118
Linux	119
Inicialização do Robotinics.....	121
Robotinics.sh	122
VNC SERVER.....	122
Sshd	122
PHP/APACHE	122
Motion	122
MySQL.....	123
SrvMonitor.c	124
Sintetização de Voz	125
Instalação do eSpeak	126
Comando eSpeak.....	126
Configurando saída de som para o eSpeak.....	128

Falar.sh.....	128
Ler.sh.....	129
Network.c	129
Integração do eSpeak com C.....	129
3 Base do Robô	133
3.1 Mecânica - Finalização da Etapa Mecânica da Base do Robô	133
Visão de Montagem.....	133
3.2 Eletrônica – Componentes Principais	135
Visão Geral da Alimentação do Robô	136
Alimentação Externa.....	137
Sensor Corrente	137
Regulador 5 V	139
Placa de Controle dos Motores DC	140
Tabela de Funcionamento dos pinos	144
Regulador de Tensão 5 V	145
Placa de Distribuição de 5 V ou 12 V	146
Shield Expansão Arduino – Sensor Shield.....	148
Sensor Ultrassom.....	150
Pinout de ligação do Arduino Shield.....	155
3.3 Software	157
Carregando o firmware no Arduino	157
Carregando o Fonte do Robô	158
Atenção.....	161
Gerenciando e dando os primeiros comandos.....	162
Lista de Comandos do Arduino Robotinics	163
MAN.....	163
VER.....	163
RE.....	163

PARA	163
FRENTE.....	163
GESQ.....	163
GDIR.....	163
GGARRADIR	163
GPUNHOESQ.....	164
CLS	164
MSG1	164
MSG2	164
SERIAL.....	164
LCDCLEAR.....	164
ACEL.....	164
ULTRA0.....	165
ULTRA1.....	165
GAS	165
CORR.....	165
TESTE	165
GBESQ.....	165
GBDir	165
GPUNHOESQ.....	166
GPUNHODIR.....	166
GCABECADIR	166
GCABECAESQ	166
GPGARRADIR	166
GPGARRAESQ.....	166
GPPUNHOESQ.....	167
GPPUNHODIR	167
SETMONITOR	167

Checklist de validação da Etapa	168
Teste1.....	168
Teste 2.....	169
Bluetooth.exe	170
4. Montagem do Corpo inferior.....	174
4.1 Mecânica.....	174
Criação das placas eletrônicas no SolidWorks	175
Base pequena	176
Arquivos e Modelos	177
Corpo Inferior	178
Arquivos e Modelos	179
Corpo Superior	180
Arquivos e Modelos	180
Montagem	181
4.2 Eletrônica.....	182
Bluetooth.....	182
LCD 16x2 I2C	184
Exemplo de I2C com Arduino.....	185
Servo motor.....	187
Como funciona um Servo motor	187
Servo Tower Pro Mg995 Digi Hi-Speed – 15 kg.....	189
Tower Pro MG996R Digital Metal Servo	191
Especificações técnicas	191
Placa de Controle dos Servos Motores	193
Placa controladora de LED	196
LCD Touch Screen	199
Elementos do Terminal	200
Placa Controladora do LCD	200

Integração Eagle SolidWorks.....	201
Etapa 1- Exportando placas no Eagle	202
Etapa 2 – Importação IDF no SolidWorks.....	207
Adicionando o CircuitWorks	207
Chamando o CircuitWorks	208
4.3 Software	214
Desenvolvimento com Raspberry PI.....	214
MySQL.....	214
Lendo dados no SQL	216
Programação Socket	218
srvMonitor	218
Fluxo de Execução.....	219
Funções.....	219
srvFala.c	220
Funções.....	220
Motion	221
O que é o motion.....	221
Como instalar.....	221
Motion.cfg	222
Parâmetros de configuração	222
Thread1-4.cfg.....	223
Script de Processamento da Imagem	223
Como funciona a visão do robô	223
5. Desenvolvimento das extremidades do robô	225
5.1 Mecânica	225
Braço Robótico - Extensão	226
Braço Robótico - Base	226
Montagem do Braço	228

Garra	229
Suporte U.....	231
Montagem do Ombro.....	234
Esquema Denavit-Hartenberg	235
Por que não será aplicado este cálculo?.....	236
5.2 Eletrônica.....	237
Componentes de Conexão.....	237
Cálculo de carga de fio elétrico em corrente contínua	237
Capacidade de Condução de Corrente (A).....	237
Fio paralelo 2,5 mm Vermelho e Preto	238
Fio Paralelo Preto e Vermelho 1,5 mm	239
Cabo Manga de Oito Vias	240
Conector RJ45.....	241
Cabeamento RJ45	241
Lista de Componentes	242
Conector RJ45 PCI Fêmea	242
Barra de Pinos Macho 180°	243
Borne	244
Resistores e LEDs	246
5.3 Software	249
Criação de Daemons.....	249
Bluetooth - Parte 2	250
Fala por TCP	251
Reconhecimento de Imagem.....	252
OPENCV	252
Pré-requisitos:.....	252
Instalação do Opencv.....	252
Compilação dos Exemplos apresentados.....	254

Criação de Aplicação Exemplo	255
Reconhecimento de Voz no Raspberry	256
CMU SPHINX.....	256
O que é o Sphinx.....	256
Aplicação prática do CMU Sphinx	257
Pré requisitos	257
Instalação.....	258
Instalação Manual.....	259
Procedimento Manual da Base.....	260
Procedimento Manual do Pocketsphinx	261
Um pouco mais de pocketsphinx_continuous	262
Desenvolvimento e Integração com Aplicações em C	264
Modelo de linguagem.....	266
Incluindo novo modelo de linguagem no cmu-sphinx.....	267
Redes Neurais	268
Instalação do FANN	269
Projeto FANN	271
Modelo de Fonte de Treino de Rede Neural.....	271
Modelo de Análise de entrada em Rede Neural	273
Compilando Exemplos de Rede Neural.....	274
Como compilar um novo projeto.....	275
6. Finalizando o Projeto	276
6.1 Mecânica – Cabeça do Robô.....	276
Montagem da Cabeça.....	277
Iniciando a montagem da cabeça	278
6.3 Software	280
PHPMYADMIN.....	280
Instalação do phpmyadmin.....	280

Acessar o phpmyadmin	281
Entendendo o Projeto Web	282
Instalando a interface web	283
Implantando os scripts	284
Criação e Configuração do Banco Web	285
Terminal remoto via web.....	287
Servidor de Data e Hora.....	288
Para não dizer que não falei de GPIO	289
Instalação.....	289
Primeiro Exemplo	291
Compilação	292
Visão Computacional	293
Configurando o motion como serviço de restart.....	293
Configurando motion para captura de imagem.....	294
Configurando acesso ao Banco de Dados pelo Motion	295
Registrando Thread de Reconhecimento de Padrões	295
Criando a Tabela no MySQL necessária	295
Registrando o banco MySQL no motion	297
Finalização do projeto	300
Apêndices	301
Baixando o projeto Robotinics.....	301
Simulação de Software	302
Gerando um espelho do SD do Robotinics para equipamento real.....	307
Bibliografia.....	308
Índice de Ilustrações	309

Em Memória

Professor Danilo

Agradecimentos

A DEUS, que nos deu a oportunidade de existir, liberando nossos sonhos.

Sonhos como este que vocês estão lendo.

Desta forma, um agradecimento tão necessário quanto qualquer outro é destinado ao meu amigo Carlos, que tenho em alta estima.

Meus agradecimentos às minhas filhas Marcella e Camille, bem como minha querida mãe, que me ajudou a superar grandes desafios até hoje.

Gostaria de agradecer também a todos os mestres do Centro Paula Souza, tanto ETEC quanto Fatec, pois participei de ambas e sei da qualidade de seu ensino e do valor de seus mestres.

Em especial, Professor Renato, Ricardo, Tatiane, Fernando e tantos outros, pois sinto haver poucas páginas para prosseguir.

Agradeço ao Prof. Francisco Fambrini por revisar meu trabalho na área de elétrica, contribuindo com conselhos preciosos sobre a área.

Ao Prof. Hugo Custódio da Silva da ETEC Ribeirão Preto, por revisar meu trabalho na área de Mecânica, contribuindo também com ótimas sugestões.

Também agradeço ao Prof. Milton Silva pelo trabalho de revisão e pela imensa ajuda.

Agradecimento ao Amigo Marcio Teixeira pelo trabalho de revisão.

Pela minha noiva e amiga de todos os dias, Fernanda Justino, pela paciência e incentivo, que vão além do amor.

Ao fechar este agradecimento, finalizando em especial às duas unidades ETEC Ribeirão Preto, antigo Industrial e Fatec Taquaritinga, como reconhecimento de tudo que me foi ensinado. Sem estas instituições, jamais chegaria aqui.

Composição desta obra

Este livro é resultante de um esforço de quatro anos no desenvolvimento e estudo na área de robótica.

Ele contém todas as informações necessárias para que qualquer um, com ou sem experiência em eletrônica ou mecânica, consiga desenvolver seus próprios projetos de robótica ou de IOT (Internet das Coisas).

O desenvolvimento do robô é feito passo a passo, ilustrando cada etapa do mesmo, bem como seguindo uma sequência cronológica para seu desenvolvimento.

O robô desenvolvido nesta obra não é o único objeto que pode ser criado a partir deste livro, pois o desenvolvimento das atividades permite a criação de qualquer outro projeto eletromecânico, também ligado à internet das coisas.

O projeto de um robô é uma obra, em minha visão, das mais complexas.

Um robô engloba o conhecimento de múltiplas disciplinas e recursos que serão também necessários para o desenvolvimento de outros mecanismos da Internet das Coisas (IOT).

Para a execução desta obra, dividiremos o projeto em tópicos e para cada tópico dividiremos em três áreas distintas:

1. Mecânica,
2. Eletrônica
3. Software

Cada tópico aborda uma parte do projeto, permitindo que o projeto seja entendido como foi montado.

Seguindo o processo linear da obra, e executando todas as atividades, mesmo sem conhecimento avançado, qualquer um pode e conseguirá desenvolver seu próprio robô, mesmo não dispondendo de uma impressora 3D.

Tornamos esta obra totalmente aberta, o que permite modificações. O projeto no momento da publicação do livro encontra-se no site e poderá ser baixado. Porém, a sua evolutiva passa a ser distinta da publicação, podendo sofrer diversas melhorias.

Para efeito didático, recomendo a todos utilizar os arquivos da source forge, e, posteriormente, realizar a atualização sobre as mudanças do meu site.

Sugestões construtivas

Apesar do projeto recomendar a construção através de impressão 3D, todas as peças também podem ser construídas através de modelos caseiros, usando materiais de fácil acesso.

As peças do corpo do robô foram baseadas em globos de isopor, como da ilustração abaixo.

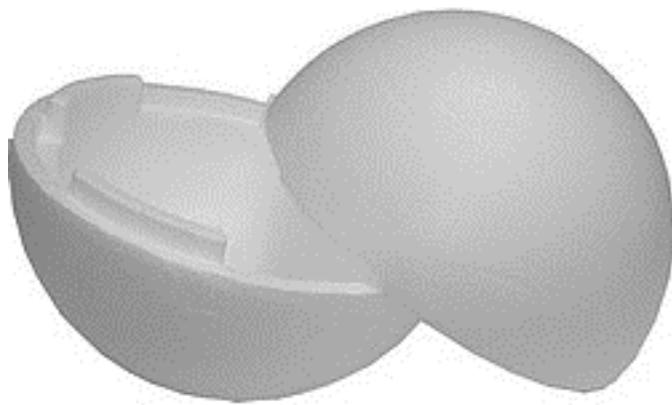


Figura 1 - Globo de isopor

Estas peças podem ser compradas facilmente em qualquer papelaria, sendo seu custo muito baixo, bastando que o leitor adquira tais globos e faça os chanfros definidos nos modelos.

Recomendo que, ao ser feito com globos de isopor, o leitor o reforce com massa epóxi para que ganhe resistência mecânica. A este revestimento deve ser feita uma prévia cobertura de folhas de alumínio para que o epóxi não entre em contato direto com o isopor.

O uso desta técnica reduz custos na montagem das peças e pode ser facilmente utilizado, já que as medidas do robô foram originalmente baseadas nestes modelos de isopor.

As especificações de chanfros podem ser obtidas pela leitura dos desenhos técnicos.

O que há nos arquivos?

Junto com este livro fornecemos todas as informações necessárias para montagem do robô.

O Robotinics tem a seguinte composição das pastas:

- Arduino – Fontes dos programas para compilação do Arduino
- CAD – Desenhos de Vista para confecção das partes mecânicas
- Impressora 3D – Arquivos destinados à impressão das peças
- Linux – Aplicações destinadas à compilação do Raspberry PI
- EAGLE – Fontes das placas de PCB, para sua composição e modificação
- PCB – Espelhos para impressão das matrizes
- SolidWorks – Arquivos dos projetos em SolidWorks para modificação
- Tools – Ferramentas auxiliares no desenvolvimento e gestão do robô

Eu preciso ter todos os softwares para montar meu robô?

A resposta é “não”. Os softwares e aplicações aqui demonstram a concepção de um robô a partir do zero. Ou seja, a partir do projeto até a etapa de montagem.

Todo o projeto já foi concebido, e o leitor deste livro poderá omitir toda a composição dos softwares caso queira apenas criar um robô.

No entanto, para aqueles que pretendem customizar ou desenvolver outros projetos com base no modelo de robô apresentado, as ferramentas aqui mencionadas têm a relevância conforme o tipo de customização que se queira realizar.

Atualizações

O projeto do Robotinics é um projeto open source. Todos os itens deste projeto são incluídos no site <http://maurinsoft.com.br/projetos/robotinics/>. Todas as ferramentas e informações são disponibilizadas neste site.

Erros e Inconformidades e Isenção de Garantia.

O Projeto de construção do robô é um projeto dinâmico, o que diz que as peças e conexões do robô são corrigidas constantemente. Todo o trabalho foi realizado visando garantir que nenhum arquivo esteja fora da especificação, porém havendo alguma inconformidade, peço aos leitores que enviem um e-mail para marcelomaurinmartins@gmail.com, que terei o maior prazer em auxiliar no que for possível.

Apesar de prolongado trabalho de revisão do projeto e por este projeto ser aberto (open source), não há garantias sobre quaisquer forma sobre a montagem, nem tão pouco o funcionamento quando houver adaptações dos equipamentos ou especificações do projeto e técnicas aqui abordadas.

Participação no Projeto

O Projeto do robô é aberto para qualquer um que queira contribuir, pois a participação de todos criará um robô mais eficiente.

Caso tenham interesse em participar, fica o link no sourceforge para sua requisição:

<https://sourceforge.net/projects/robotinics/>

O que há de comum entre um robô e um dispositivo IOT

Internet of Things (IoT) ou Internet das coisas é, para muitos, a quarta onda da revolução, chamada revolução das coisas.

A internet das coisas, permite que equipamentos comuns sejam conectados à internet, porém, a conexão em si não agrega benefício. A internet das coisas prevê que as informações coletadas sejam analisadas e permitam que equipamentos possam se interconectar.

Outro aspecto muito importante é que permite que as informações geradas possam ser processadas e analisadas, gerando melhora dos processos e consecutiva maior eficiência deles.

Um exemplo prático, e bem simples, é o uso de semáforos inteligentes. Imagine que criemos semáforos ligados à internet, e que cada equipamento seja capaz de perceber o fluxo de veículos em ruas e cruzamentos. Ao enviar essas informações a uma central, podemos aumentar ou diminuir o tempo de parada nos cruzamentos, melhorando o fluxo em horários críticos.

Os usos destas técnicas não são tão diferentes de um robô, pois o semáforo terá, além dos sensores, o poder de comunicação com outros dispositivos, o que é basicamente o que um robô faz com os diversos componentes eletrônicos e mecânicos nele instalado.

Na prática um robô é apenas um equipamento de IoT que projeta além de diversos dispositivos de coleta de informação, também diversos dispositivos de saída de informação, bem como projetos complexos de tomada de decisão. Tudo isso conectado na Internet.

O processamento das informações de um robô é mais um exemplo do uso de seus sensores a fim de prover meios de garantir que a comunicação com outros equipamentos ligados na internet e com pessoas seja mais eficiente.

Neste sentido, robôs são exemplos práticos de IoT, tendo todos os requisitos para aplicação nesta contextualização.

Esta obra apenas enfoca um tipo, não tendo, no entanto, obrigatoriedade em ser referência sobre este tema.

Por fim, aqueles que tem por ambição no desenvolvimento e projeto de equipamentos ditos IoTs, com certeza fará bom uso das técnicas e recursos aqui apresentados.

Entendendo o que é um Robô

Um robô é um conjunto de elementos eletromecânicos dispostos de tal forma que permitem a interação com o ambiente.

Os robôs podem ser categorizados quanto ao seu gerenciamento:

Autômatos – Sem controle externo, operando de forma independente de qualquer fator decisório externo.

Tele presenciais – Quando são “vestidos”, por assim dizer, permitindo que o operador possa utilizar o robô como sua própria roupa, semelhante ao definido no filme “Avatar”, de James Cameron.

Remotamente controlados – Quando são gerenciados a certa distância, porém com nível de gerenciamento dado na terceira pessoa. Ou seja, você pilota, mas não sente que é você. Entende claramente a distinção entre você e o robô.

Tipos de Ambientes

Os robôs apresentam vários formatos e tamanhos, porém podemos realizar separações considerando o ambiente com o qual irá interagir. Quanto ao ambiente, podemos dividi-los em diversas categorias:

- **Drones** – Robôs aéreos, possuem hélices ou turbinas para superar a gravidade.
- **Robôs aquáticos** – São equipados e construídos para existir em ambientes aquáticos. Ex.: submarinos robóticos e barcos.
- **Terrestres** – São equipados com pernas ou rodas, dependendo do projeto, porém existem para se locomover na superfície terrestre.

Entre outros.

Construção genérica de um robô

Qualquer robô, independentemente da forma, modelo ou mesmo finalidade, é constituído de partes bem definidas:

- **Alimentação** – Fornece a energia para alimentar o robô, permitindo autonomia para que este possa realizar seu objetivo.
- **Estrutura física** – Através do seu projeto construtivo (corpo), o robô interage com o mundo e este com ele. A estrutura física, apesar de variada, é constante em qualquer robô, e é uma das partes mais difíceis do projeto.
- **Estrutura eletrônica** – Todo movimento é fruto de dispositivos eletrônicos, tais como motores, Servo motores, motores de passo, acionadores de pistões hidráulicos e pneumáticos, e tantas outras. Desta forma, as estruturas eletrônicas sentem o mundo, gerando os sinais e, por fim, gerando o movimento ou reação esperada para o meio físico.
- **Central de processamento** (cérebro) – Qualquer robô, mesmo os que são controlados, precisa de um sistema de processamento local. Esta central de processamento permite que o robô colete as informações do ambiente, bem como controle os dispositivos a ele vinculados, dando condições de que haja interação com o ambiente.
- **Software** – O projeto de construção de um robô é um feito complexo, pois o número de interfaces existentes e as possibilidades para cada ação são praticamente infinitas. O software torna gerenciável este processo, criando padrões elaborados, porém controlados, para que cada dispositivo eletrônico possa ser interpretado e controlado.

Primeiros passos

Iniciaremos nossa jornada na robótica, mas não antes de identificarmos o que iremos fazer.

O planejamento de um robô talvez seja a atividade mais importante.

Como os robôs são projetos complexos e às vezes de longa duração, a definição de onde se quer chegar não é só interessante, mas crucial, tornando claro o resultado esperado e permitindo que não desvie do foco do projeto, bem como objetivando para que você não gaste horas de seu tempo em coisas que não estavam planejadas.

Não tenho aqui o intuito de criar uma biografia de gestão de projetos, pois é claro para mim que não sou a pessoa e nem essa é a obra mais indicada para tratar do assunto. Porém, é importante ter em mente o que se planeja. Para tanto, iremos gerar uma definição inicial, que será confrontada no final do projeto, para que possamos identificar se conseguimos realizar a tarefa.

Objetivo:

- 1) Criar um protótipo de robô terrestre controlado remotamente através de comandos.
- 2) Ser fácil e escalável, podendo ser adaptado a diversos outros projetos.
- 3) Permitir ser customizada para diversas soluções.
- 4) Ser acessível tanto na fabricação quanto na compra de peças e sobressalentes.
- 5) Permitir a impressão das peças mecânicas utilizando impressora 3D ou montagem artesanal.

Pois bem, identificamos os pontos importantes entre muitos que precisamos ter no nosso projeto. Agora é trabalhar para conseguir tal empreitada.

1. Conceitos Básicos

Os conceitos básicos vão além do desenvolvimento do robô, permitindo que você entenda o fundamental para começaremos o desenvolvimento do projeto.

Nos conceitos básicos, apresentaremos uma visão geral sobre o ambiente de projeto Mecânico, Eletrônico e de Software.

1.1 Mecânica

O desenvolvimento de projetos mecânicos não é tarefa fácil. Vamos compilar aqui uma série de dicas no intuito de auxiliar e orientar nas melhores práticas de desenvolvimento de projetos.

Os projetos mecânicos visam garantir que o objetivo construtivo seja alcançado. Para auxiliar os projetos mecânicos(Fialho), utilizamos ferramentas de CAD (Desenho Assistido por Computador), que é a tradução literal de CAD do Inglês (*Computer Aided Design*).

Existem diversas soluções para modelagem e desenvolvimento de projetos 3D, e entre as ferramentas pagas e abertas estão estas:

- SolidWorks (Pago)
- FREECAD
- OPENSCAD
- Google SketchUp

SolidWorks

O SolidWorks é hoje uma das ferramentas mais completas no desenvolvimento e validação de projetos mecânicos.

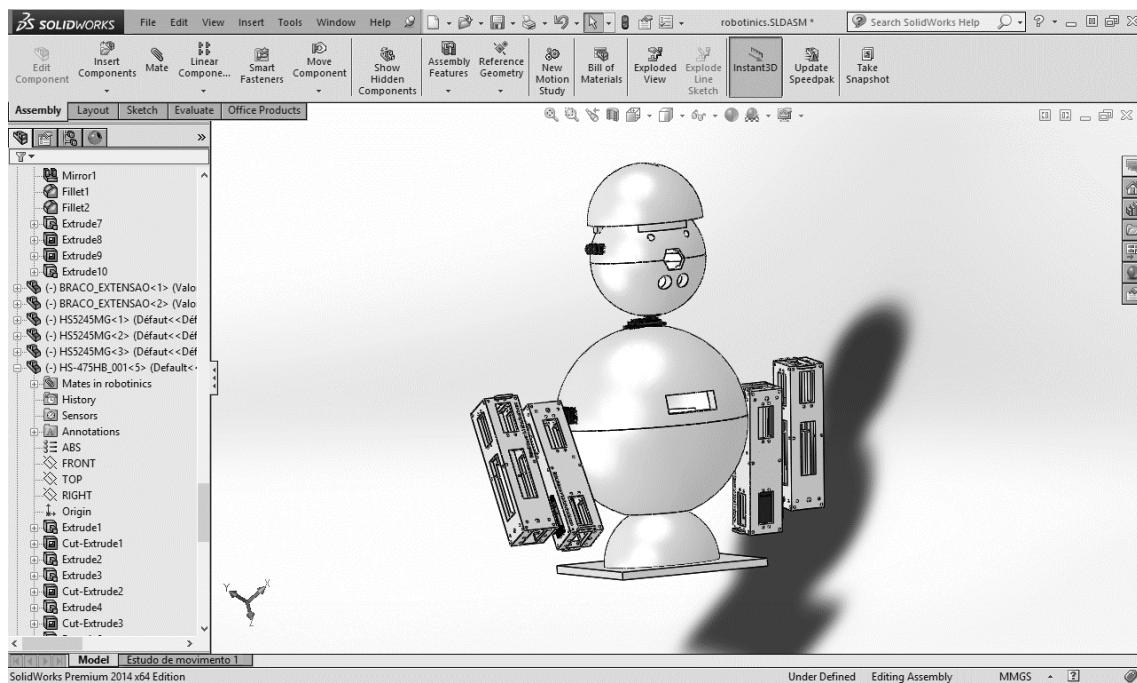


Figura 2 -Tela do Solidworks

Através do SolidWorks é possível, além de desenvolver a solução completa sobre o prisma mecânico, fazer diversos testes, tais como:

- Teste de esforço
- Análise de materiais
- Dimensionamento
- Estudo de movimento
- Acoplamento eletromecânico
- Geração de visualização de componentes eletrônicos

Através do SolidWorks também é possível gerar os arquivos STL, que são lidos pelas impressoras 3D e utilizados para gerar os modelos.

O objetivo deste livro não é fornecer treinamento para desenvolvimento no SolidWorks. Caso o leitor desejar, nem ao menos precisará utilizar esta ferramenta, pois todos os arquivos STL estão disponibilizados, podendo imprimir diretamente, sem necessidade de modelar. Porém, caso faça a modificação do projeto, o uso desta ferramenta acelera o desenvolvimento do projeto mecânico.

Serão apresentados aqui alguns aspectos relativos à construção mecânica do robô, apresentando, para tal, esta ferramenta.

Gerando arquivos STL para impressão no SolidWorks 2014

A criação dos arquivos STL a partir de um projeto concebido é tarefa extremamente fácil no SolidWorks.

Para tanto, siga os passos aqui descritos:

- 1) Entre no SolidWorks
- 2) Selecione a opção File > Open

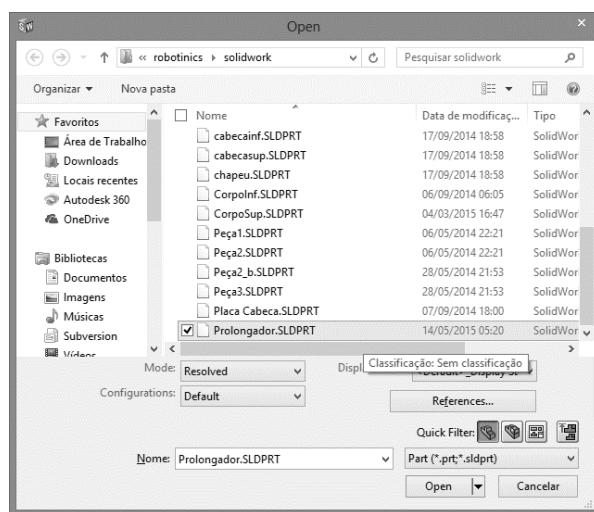


Figura 3 - Opção de abertura de arquivo no SolidWorks

- 3) Indique o arquivo da parte mecânica do projeto que se deseja imprimir. Em geral, ele tem a extensão .sldprt ou .prt

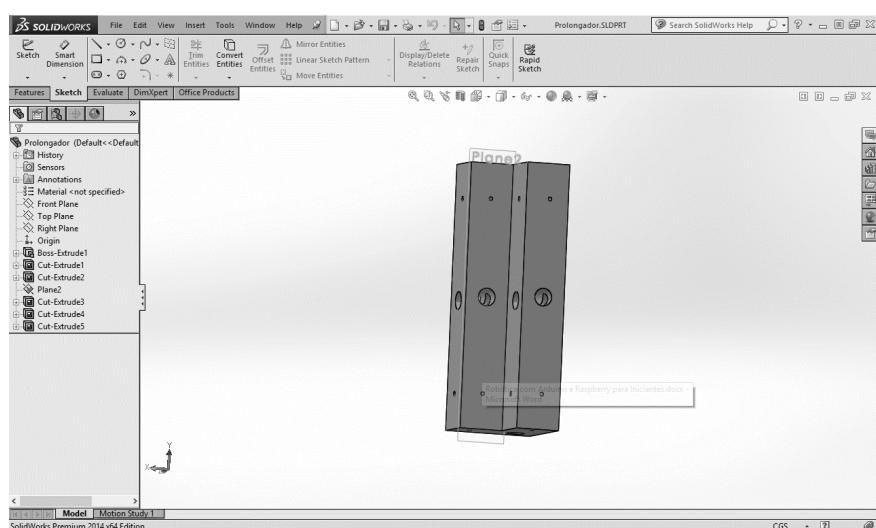


Figura 4 - Parte que se deseja imprimir aberta no SolidWorks

- 4) Selecione a opção File > Save as

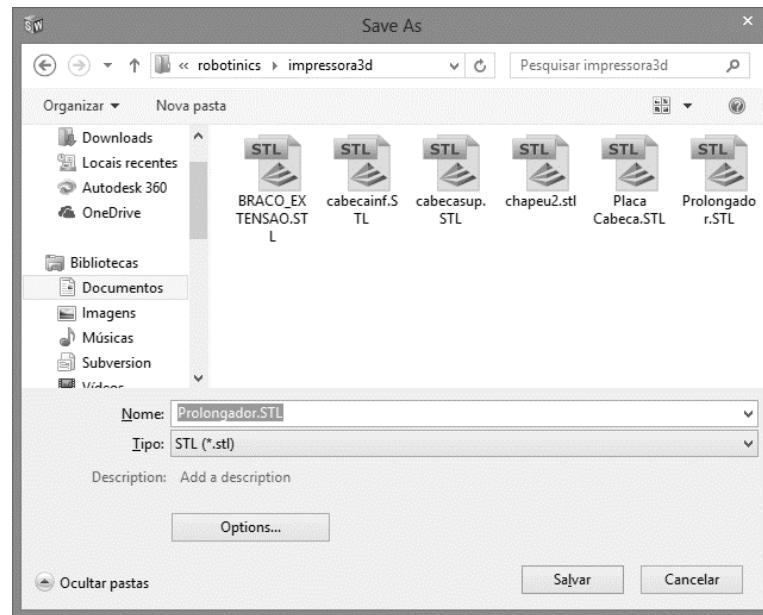


Figura 5 - Opção de salvamento do arquivo

- 5) No tipo, selecione a opção STL (*.stl). Desta forma, o SolidWorks entende que você quer salvar no formato para impressão 3D
- 6) No Diretório de Impressora3D, salve com o nome adequado. Neste caso, Prolongador.STL

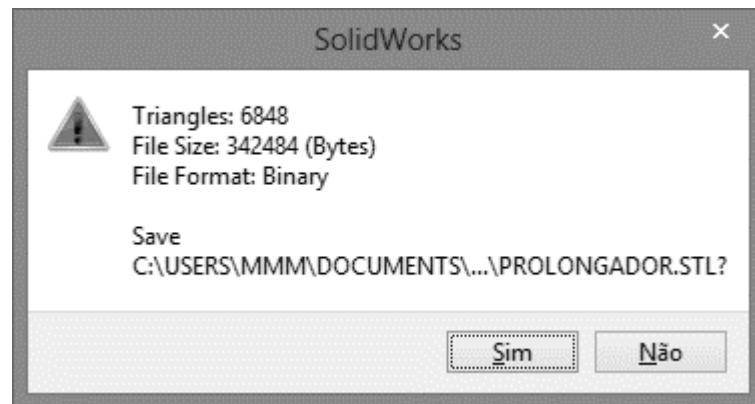


Figura 6 - Confirmação de salvamento STL

- 7) O SolidWorks automaticamente irá triangular a peça e pedir a confirmação do salvamento do Arquivo

Impressoras 3D

Impressoras 3D são dispositivos que através de comandos numéricos (CAM) permitem a criação de peças sólidas.

As impressoras 3D existentes para o mercado pessoal permitem gerar modelos físicos de plástico ABS ou PLA. Para construir este modelo físico é utilizado o arquivo STL, que contém um conjunto de vetores construtivos. Ao ser lido pelo software da impressora 3D, este arquivo STL é convertido em um conjunto de comandos posicionais chamados GCODE.

O GCODE é um padrão utilizado na modelagem de peças e muito empregado em CNCs.

Como funciona uma impressora 3D

Existem hoje dois grandes grupos de impressoras 3D.

A mais comum utiliza um filamento plástico em forma de bobina, que é aquecido em um bico térmico (extrusora). O aquecimento controlado permite que este plástico se liquefaça e caia em uma base, sendo depositado, camada por camada, até que o formato da peça seja alcançado.

O outro modelo é por solidificação por luz. A base é imersa em um líquido fotossensível que, ao ser bombardeado por um feixe de laser, reage e endurece.

Desta forma, o laser é direcionado aos locais onde deve reagir, endurecendo. É criado, desta forma, o modelo que se deseja.

Utilizaremos, para imprimir este trabalho, a impressora da Vinci, do fabricante XYZPrinting (<http://la.xyzprinting.com>). Trata-se de uma impressora por bobina de ABS (extrusora), que imprime peças em dimensões de 200 mm x 200 mm x 190 mm, com resolução máxima de 0.1 mm.



Figura 7 - Impressora 3D da Vinci XYZPrinting

Software de Impressão

O software da impressora 3D permite, a partir do arquivo STL, gerar o arquivo GCODE e encaminhar para a impressão.

O software de impressão se comunica diretamente com a impressora, sendo a única ferramenta capaz de fazer isso.

A impressora XYZPrinting é conectada através da porta USB, sendo reconhecida através do software XYZWare, que é fornecido com o produto.

Este software pode ser baixado diretamente do site do fabricante ou instalado a partir do CD que acompanha a impressora.

Montagem de peças utilizando o software XYZWARE

A montagem de peças (impressão) é um processo extremamente simples.

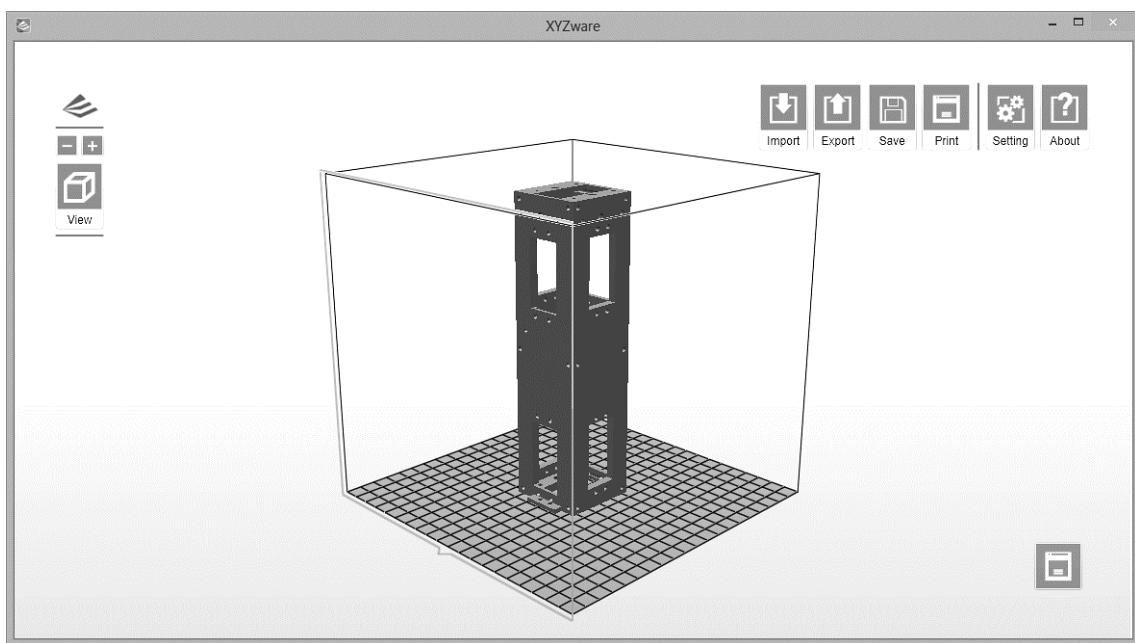


Figura 8 - Software XYZWare preparado para imprimir uma peça

Existem situações na modelagem de peças 3D em que se faz necessário o uso de bases de apoio para sustentação, como uma coluna que sustenta uma laje. A coluna é colocada no desenho, garantindo que, ao ser posto o plástico liquefeito, ele se mantenha em posição até endurecer.



Figura 9 - Opções para finalização da impressão

A opção **Supports** cria estas estruturas automaticamente quando verifica um chanfro em que o filamento não terá sustentação.

A opção **Raft** cria uma cama de filamentos para evitar que a peça encoste diretamente na base de vidro. Esta cama de filamentos é muito útil porque, dependendo da forma que a peça é montada, a base de vidro pré-aquecida remolda a face que está encostada na base, criando imperfeições no objeto.

A geração destas colunas de sustentação pode ser feita automaticamente, quando o software já a cria, ou de forma manual, quando é obrigado pelo software que você aponte o local deste.

Cartucho de Filamento



Figura 10 - Cartucho de filamento da XYZPrinting

Para imprimir, a impressora 3D utiliza um filamento em forma de rolo. Este filamento pode ter diversas cores, sendo a cor da peça resultado da cor do filamento que se encontra na impressora.

Cálculos Mecânicos

Nem tudo pode ser previsto no início de um projeto, porém existem alguns conhecimentos que devem ser prévios antes de iniciarmos esta longa jornada para o desenvolvimento de nosso robô. Existem cálculos que devem ser considerados e pensados, pois a sua realização garante que a escolha dos componentes eletrônicos e mecânicos atenderá as necessidades do projeto.

Nota do Autor:

O que é interessante neste livro é que as informações técnicas aqui presentes são, de fato, ensinadas no ensino médio e fundamental, e que realmente percebi que muito do que é ensinado na escola não é percebido em termos de praticidade e aplicabilidade.

Dois itens mecânicos devem ser avaliados quanto à construção de um robô:

- Dimensionamento dos motores
- Dimensionamento dos servos motores

Para tanto, vamos considerar os cálculos para tais itens.

Dimensionamento dos Motores

O dimensionamento dos motores garante que o robô possa mover-se livre de auxílio externo.

Quando trabalhamos com veículos simples, como um projeto caseiro, dificilmente realizamos todos os cálculos relacionados ao dimensionamento de motores. Porém, quando pensamos em um robô pesado, com várias peças e diversos componentes, a estimativa deste componente faz-se necessária, pois um erro no projeto acaba prejudicando o desenvolvimento e a continuidade deste.

Por vezes, em projetos de hobistas, utilizamos a experimentação como referência para dimensionamento. Porém, ao ser calculado, garantimos com alguma segurança que a prática seja mais assertiva. Não obstante que a teoria negue a prática, pois, ao fim, é à prática que prova a teoria.

A apresentação deste tópico é baseada no excelente trabalho (Dias, 2011) que muito me auxiliou a formar este tópico.

Cinemática - Estudo do movimento

A cinemática é a ciência que estuda o movimento dos corpos. Quando aprendemos esta matéria na escola, dificilmente pensamos em aplicações práticas do dia a dia.

Ao analisar nosso robô, podemos entender as seguintes forças envolvidas em seu movimento:

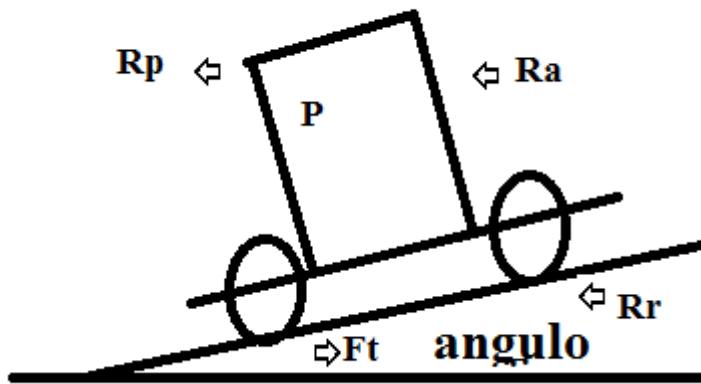


Figura 11 - Forças relacionadas ao movimento do robô

Onde:

Ft – É a Força Trativa, ou força transmitida nas rodas e que resulta em movimento

Ra – Resistência Aerodinâmica, que é a força de resistência (Força Oposta) que o ar gera ao ser movimentado

P – Carga Total, que representa a massa (peso do robô) que está sendo movida

Rp – Resistência de Rampa. Quando aplicado um ângulo de rampa, uma força é gerada de forma proporcional à sua inclinação.

Rr – Resistência de Rolamento, que é a resistência aplicada em uma roda.

Considerando que ao acionar o motor o robô precisará se mover, precisamos levar em conta que a força modular será tal que:

$$F_t > R_a + R_p + R_r$$

Isto significa que, na prática, a Força Trativa (Ft) ou Força de tração) deve superar todas as forças de resistência, tais como resistência do ar, resistência de rolamento etc.

Partindo deste princípio, poderemos considerar tais componentes de força, vendo cada um isoladamente.

Cálculo de Força Trativa

Nesta etapa avaliaremos os cálculos necessários para composição dos elementos de tração das rodas, verificando o dimensionamento das rodas e sua tração.

A força trativa é a força gerada pela movimentação da roda, que está em contato com o solo.

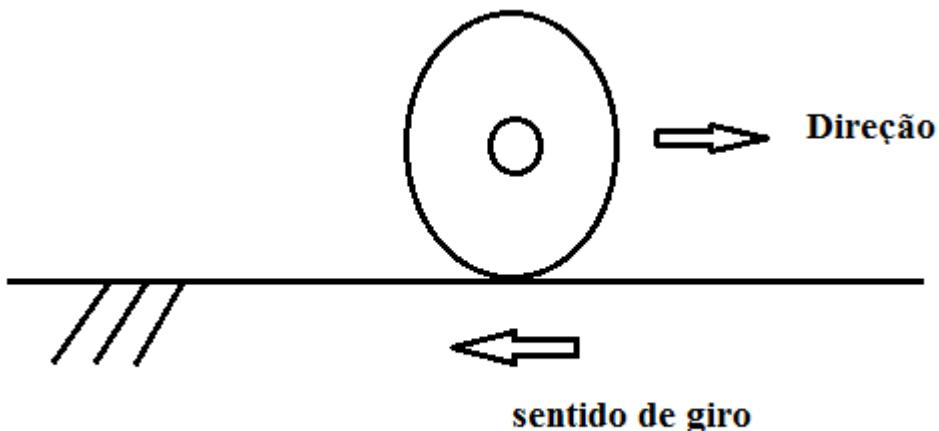


Figura 12 - Ilustração que mostra efeito de força trativa

O dimensionamento de carga dirá primeiramente o tipo e número de rodas necessárias para tracionar o robô. Na prática, quando dimensionamos um componente, temos dois caminhos:

1 – Dimensionar baseado nas características do robô, estimando a força necessária para deslocar e depois escolhendo o componente mais adequado

2 – Dimensionar baseado no componente de tração (motor), verificando suas características e determinando se é compatível com a necessidade

Logicamente, a escolha de um componente de tração deve passar por um filtro prático, sendo que elementos visivelmente não compatíveis não podem ser, de forma nenhuma, questionados.

Estamos utilizando o segundo caso. Primeiro, pela facilidade na aquisição deste componente, e, em seguida, pelo seu custo. Porém, toda a lógica aplicada neste trabalho pode ser facilmente empregada inversamente para o outro caso.

Sendo assim, um componente elétrico, como um motor, já possui uma série de especificações técnicas definidas. Vale a pena levantar de antemão as especificações de um componente antes de qualquer outro processo.

Iremos comentar alguns destes itens, explicando o que cada um significa.

Para tanto, utilizaremos as especificações do motor com caixa de redução e roda plástica.

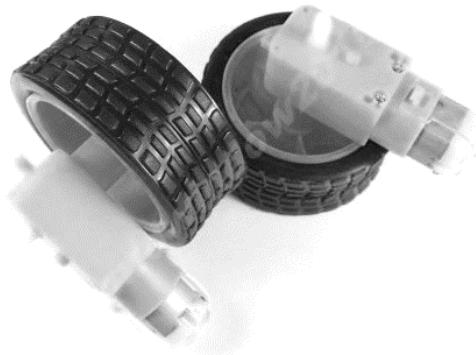


Figura 13 - Motor DC com caixa de Redução + Roda

Se pesquisarmos na internet, encontraremos a especificação deste, conforme apresentado:

- Tensão de Operação: 3-6 V (valor máximo 12 V não recomendado)
- Rotação: 170 RPM 120 RPM (6 V)
- Consumo: 100ma – 3 V/120ma - 6 V
- Torque: 0,35 Kgf cm a 3 V (0,0343Nm)
- Torque: 0,80 Kgf cm a 6 V (0,078Nm)
- Redução: 1:120

Torque

O torque, ou momento de força, é a força do motor no seu eixo.

Vale a pena ressaltar que o torque é a força que o motor entrega na caixa de redução, e não a força dispensada pela roda.

O torque é dado em Nm (Newton metro) ou Kgf cm (Quilograma força por centímetro).

Em motores DC, o torque máximo é dado em especificação técnica, conforme passado anteriormente.

No motor DC com Caixa de Redução, o torque máximo varia em função da tensão aplicada, conforme apresentado:

- Torque: 0,35 Kgf cm a 3 V
- Torque: 0,80 Kgf cm a 6 V (0,078Nm)

Cálculo da Força Trativa

Apesar da especificação do conjunto já apresentar a Força Trativa diretamente na especificação, poderíamos calcular, através da seguinte equação:

$$F = \frac{(T \times R_{engrenagem} \times eficiência)}{R_{pneu}}$$

Onde:

T (Torque) = Torque máximo entregue pelo motor

Rengrenagem = Relação de engrenagem, apresenta a relação entre a entrada entregue pelo motor e a saída dada pela caixa de redução.

Eficiência = Apresenta a relação entre o torque fornecido pelo motor e a tração entregue pela caixa de redução (câmbio).

R_{pneu} = Raio do pneu. Conforme o raio do pneu, pode variar a força de tração.

Lembrando que, conforme figura abaixo, a força de tração será multiplicada por quatro, pois a tração será nas quatro rodas.

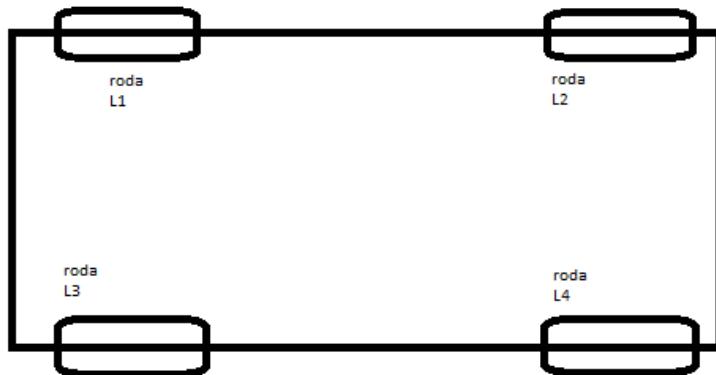


Figura 14 - Layout da base do robô com quatro rodas motoras

Sobre o estudo de tração e análise das forças, devemos atribuir e determinar:

- ✓ As rodas são motoras, ou seja, cada uma destas apresenta um motor DC, e somadas apresentam o torque total do motor.
- ✓ Todas as rodas e motores são do mesmo tipo e modelo, apresentando torques iguais. Desta forma, desprezam-se as eventuais diferenças de torque.
- ✓ Cada roda distribui uniformemente o peso e a tração, não apresentando diferença no esforço resultante.

- ✓ O peso do robô pode ter até 3kg, conforme periféricos instalados.

Torque de Tração Máxima

Para cálculo de torque de tração máxima, devemos considerar o torque máximo que um motor pode gerar em um caso ideal.

O torque de tração máxima é uma referência para ser utilizada como comparação.

Na especificação o fabricante já apresenta o valor da F_t (força trativa), não necessitando ser calculado, pois já foi dado.

Porem caso seja necessário a fórmula para este fim é:

$$F_t = Fm \times r$$

Força de Tração por Roda:

Conforme já apresentamos anteriormente:

$$F_{t(6V)} = 0,078Nm$$

O momento força das quatro rodas com redução e já aplicado à roda é:

$$F_{t(6V)} = 0,078Nm \times 4$$

$$F_t \text{ das quatro rodas}(6V) = 0,312Nm$$

Em projetos eletromecânicos, as vezes algumas especificações são omitidas pelos vendedores. Sendo sua localização uma verdadeira garimpagem, pois nem mesmo é claro o fabricante. No caso do projeto deste motor com engrenagem de redução.

$$F_t (6v)= 0,312Nm \text{ ou } 3,185 \text{ Kgf cm}$$

Referência:

$$1Nm = 10.19716213Kgf cm$$

Dica:

Site para conversão de torque

<http://webcalc.com.br/Conversoes/form/torque>

Cálculo de força para aclice

O cálculo de força para aclice determina o valor a ser considerado para que o robô consiga superar uma rampa com aclice de 10 graus de elevação.

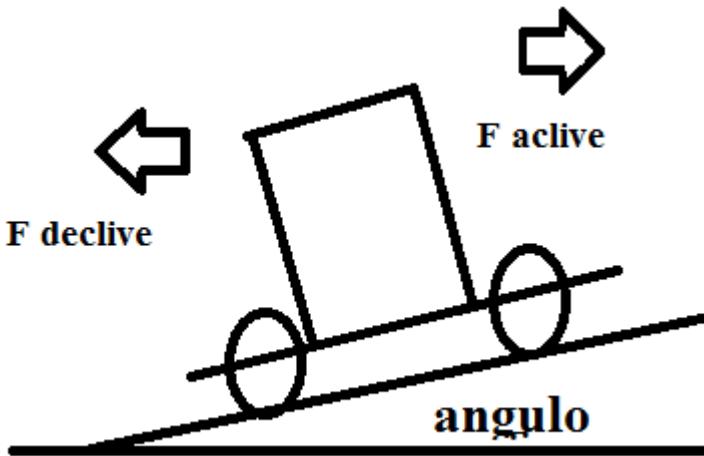


Figura 15 - Robô subindo uma rampa

A fórmula para esta é

$$F_{aclice} = m \times g \times \text{SEN } \alpha$$

Onde F_{aclice} = força inercial, ou força que deve ser superior para que supere a gravidade e faça o robô subir a rampa.

Onde:

- m = massa do robô 3kg
- g = constante gravitacional $9,8m/s^2$ da terra ao nível do mar
- α = Ângulo da rampa

Dado:

$$1N = 1Kg \times 9,8m/s^2$$

Consideraremos que tanto o A como F_a são desprezíveis.

Sendo assim,

$$F_{aclice} (90 \text{ graus}) = 3Nm/s^2$$

Onde aplicaremos agora aos graus de inclinação.

Podemos calcular a força conforme tabela abaixo:

Ângulo de inclinação	Força resultante
5 graus	$3N \times \text{Sen } 5 = 0,26\text{Nm}$
10 graus	$3N \times \text{Sen } 10 = 0,52\text{Nm}$
15 graus	$3N \times \text{Sen } 15 = 0,77\text{Nm}$

➤ $F_{\text{aclive}}(5) = 2,56\text{Kgf} \rightarrow 0,26\text{Nm}$

➤ $F_{\text{aclive}}(10) = 5,1\text{Kgf} \rightarrow 0,52\text{Nm}$

➤ $F_{\text{aclive}}(15) = 7,6\text{Kgf} \rightarrow 0,77\text{Nm}$

Desta forma, para efeitos práticos, ajustaremos o aclive para **5 graus** de inclinação, entendendo que o valor máximo seria 0,26N, onde teríamos valor máximo de **0,312Nm**.

Força de Atrito de Rolamento

É a força gerada para pela deformação da Roda e pelo atrito entre a roda e o chão e pela deformação do pneu.

A força de atrito de rolamento cria uma resistência física para o movimento devido, esta força é a força que faz você gastar mais dinheiro com combustível, quando seu carro está com o pneu murcho, por exemplo.

Para efeitos práticos não iremos calcular esta força, porem ela existe mas apresenta valores desprezíveis.

Nota do autor:

Se não calculamos, porque apresentamos esta força?

Primeiramente, o cálculo de força de atrito de rolamento envolve variáveis que com certeza fogem do escopo deste livro.

Porem para veículos de rodas maiores e mais pesados, como carros deve ser considerado.

Força de Arrasto Aerodinâmico

O valor de arrasto aerodinâmico para um robô de diâmetro pequeno e velocidade reduzida torna-se desprezível.

Sendo sua formula dada: (Silveira, 2011)

$$F_a = \frac{1}{2} \cdot \rho \cdot C \cdot S \cdot v^2$$

Onde:

- ρ é a intensidade do ar, que é aproximadamente $1,22\text{kg/m}^3$ ao nível do mar
- C parâmetro adimensional, que varia em função do formato ($C=0,11$)
- S é a área do corpo frontal
- v é a velocidade do ar em relação ao corpo

Cálculo de Esforço de Arrasto

O Esforço de Arrasto, na verdade é a somatória das forças de oposição ao movimento.

O Cálculo de Esforço de Arrasto, no caso do robô rolamentado, leva em consideração a soma das forças anteriores, sendo:

- Cálculo de acente (5 graus ajustado) =7,6N
- Força de rolamento (desprezível)
- Força de Atrito Estático (desprezível)
- Cálculo de arrasto aerodinâmico (desprezível)

Esforço Arrasto = 0,26Nm

Totalizando **0,26Nm** quando estiver em uma rampa.

Como não foi apontada, farei uma breve explicação sobre a Força de atrito estático, esta é gerada pelo atrito mecânico das partes (motor, caixa de redução, etc).

Agora, comparando nossa força de tração máxima (6 V) de **0,312Nm** e nosso esforço de arrasto de **0,26Nm**, podemos perceber que as forças, apesar de próximas, permitem que o robô se movimente.

O que pudemos perceber, inclusive, no ensaio experimental!

Como melhorar nossa força de tração!

Bom uma maneira simples de realizar essa melhora é adicionar mais rodas ao sistema.

Ou então a troca do motor e da caixa de engrenagem de redução, por equipamento com maior Força Trativa.

A utilização deste conjunto atual, mesmo não obedecendo as especificações iniciais idealizada (15 graus de rampa), fez-se necessários devido ao custo.

As vezes em projetos, as especificações são alteradas ligeiramente devido a questões como essa.

Não devemos, no entanto, considerar que houve falha, pois, questões como custo, devem ser tão importantes como especificações técnicas aprimoradas.

As especificações iniciais são uma projeção para o ideal, segue a linha do bom senso, onde a limitação de acente não é um impeditivo para o perfeito funcionamento do robô.

Cálculo de Esforço dos Braços

A construção de um braço mecânico exige analisar vários itens:

- ✓ Design dos segmentos do braço e seus segmentos
- ✓ Análise dos pontos de sustentação
- ✓ Elementos de tração (dimensionamento de carga)
- ✓ Elementos de fixação

Cada ponto destes itens deve ser considerado, e sua montagem aplica considerável esforço no projeto.

Porém, em muitos projetos de hobistas tais conceitos não são avaliados, pois em protótipos o principal intuito não é a criação de um projeto replicável, e sim a criação ou validação de um conceito (peça única).

Neste sentido, muitos pontos são resolvidos e identificados durante a fase de criação, não tomando o devido cuidado com sua documentação e eventual replicação.

Em projetos industriais e/ou complexos, elementos não podem ter o mesmo trato, pois temos que garantir, além da replicação de elementos com equivalência técnica para cada peça, que eles sejam padronizados.

Analizando problemas envolvidos

A carga e pesos relativos à própria estrutura podem criar problema no projeto, pois se não avaliado de forma a pensar em sua estrutura e também sobre a carga que irá levar, de fato pode não atender a esta especificação.

Neste tópico serão abordados os dimensionamentos dos elementos de carga e tração associados à construção de um braço robótico.

Motores e Servo motores são elementos de carga e tração. Estes são alimentados por tensões DC e muitas vezes pensados como projeto eletrônico, porém a avaliação das características necessárias para que estes cumpram a tarefa de levantar um braço é uma atribuição mecânica, devendo ser primeiro definido como mecânico e, em seguida, pesquisado e projetado pela eletrônica já contemplando estes aspectos.

É importante considerar que um braço de robô é apoiado em um único ponto, e que este tem como responsabilidade efetivamente gerar uma força em oposição à força gerada pela gravidade.

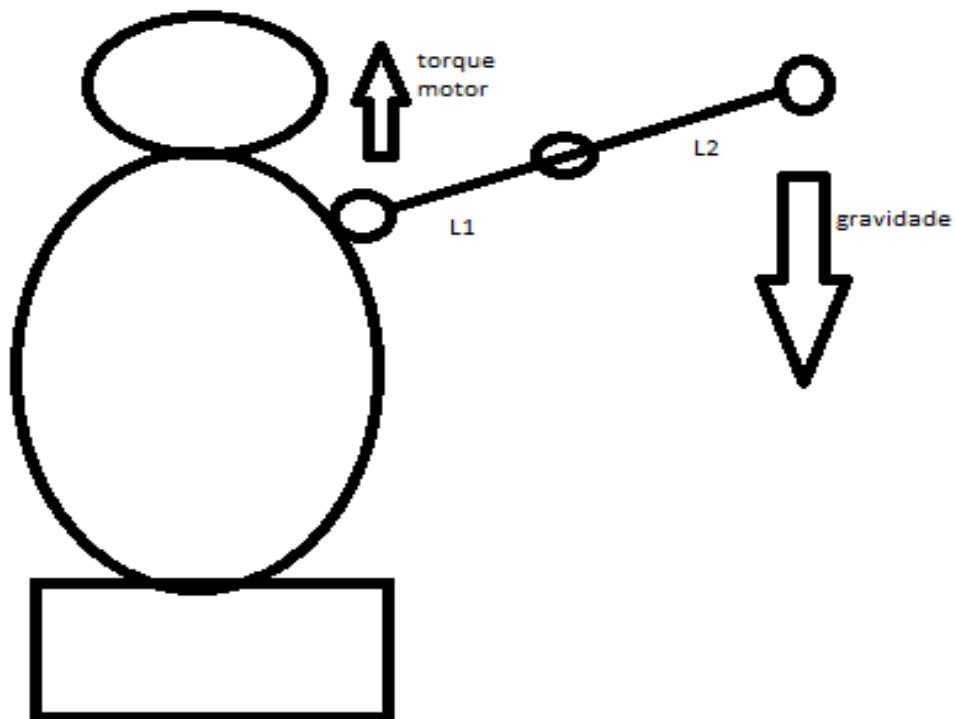


Figura 16 - Representação física de um braço e suas forças mecânicas

As especificações mecânicas de esforço demonstram os pontos de esforço do projeto mecânico.

A gravidade irá atuar no corpo como uma alavanca, fazendo uma força no sentido oposto ao do Servo motor. Por contrapartida, o Servo motor precisa superar esta força para manter ou levantar o braço.

O dimensionamento de carga visa realizar o estudo destas forças a fim de identificar pontos onde haverá maior esforço no modelo do braço. Este estudo é importante para avaliar os valores necessários para que o braço cumpra seu objetivo, que é, de fato, se mover.

Para determinação dos cálculos iniciais, o primeiro passo é a medida e tomada de pesos de todos os componentes mecânicos que são componentes do braço.

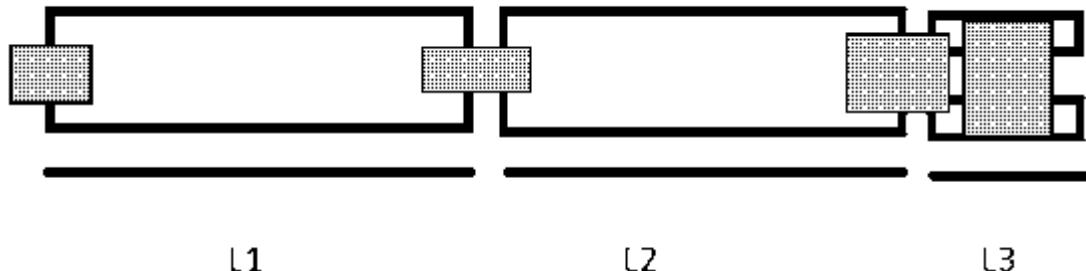


Figura 17 - Visualização dos segmentos do braço do robô

Segue lista de componentes:

- Braço Extensão (L1) – peso: 116g
- Braço Base (L2) – peso: 101g
- Garra (L3) 3 x Servo motor – peso: 55g

O leitor pode identificar que os Servo motores estão identificados como blocos de cor distinta. E ao identificar que a lista de componentes apresenta apenas três servos, pode questionar sua exatidão. De fato, os servos componentes do braço são quatro, porém há um fixado ao corpo que movimenta os demais, totalizando apenas três servos sendo tracionados pelo quarto servo.

Este quarto servo não entra na composição do peso, pois está fixado diretamente ao corpo, não sendo constituinte do braço. No entanto, é ele que sofre maior esforço, pois traciona todos os outros componentes.

O dimensionamento de carga dos servos pode ser pensado como um eixo, onde o elemento mais próximo ao ponto de apoio como o L1 precisa superar maior carga (Torque) e os elementos mais próximos a L3 precisam superar menor carga (Torque).

Para cálculo de Torque resultante, será dado:

- $L_1 = 190 \text{ mm; } 116 \text{ g}$
- $L_2 = 190 \text{ mm; } 101 \text{ g}$
- $L_3 = 50 \text{ mm; } 70 \text{ g}$

Os pesos de cada segmento foram dadas através de determinação do software SolidWorks. Para tanto, se atribuiu o material constituinte como ABS.

Cálculo de esforço

Força L1 = Torque de esforço do servo fixado ao corpo

$$Mt(L_1) = \left(\frac{L_1}{2} \times Q_1\right) + \left(\frac{L_2}{2} \times Q_2\right) + \left(\frac{L_3}{2} \times Q_3\right);$$

Onde:

- L1,L2,L3, são os comprimentos dos segmentos respectivos
- Q1,Q2, Q3, são os pesos dos segmentos respectivos
- Mt é o torque necessário para superar a gravidade.

Calculando

$$Mt(L_1) = (19.0 \times 0.116) + [(19.0+19.0) \times 0.101] + [(19.0+19.0+5.0) \times 0.110];$$

$$Mt(L_1) = 2.204 + 3.838 + 4.74;$$

$$Mt(L_1) = 10.8 \text{ kgf.cm}^2$$

Mt(L2) = Torque de esforço do servo fixado no cotovelo do robô (L₂)

Mt(L2) = (Segmento L₂ x massa L₂) + (Segmento L₃ x Massa L₃);

$$Mt(L2) = 3.838 + 4.74$$

$$Mt(L2) = 8,578 \text{Kgf cm}^2$$

Mt(L3) = (Segmento L₃ x Massa L₃);

$$Mt(L3) = 4.74 \text{ kgf cm}^2;$$

Dados os valores de referência:

- Servo MG996R torque 9.40 kgf cm²
- Servo Mg995 torque 13 kgf cm²

Desta forma, o Servo motor indicado é o de 13kgf cm² para as juntas L₁ e L₂, e as demais podem ser de 9.4kgf cm².

Sendo assim, deve-se aplicar para os pontos L₁ e L₂ o Servo motor MG995, pois os valores devem ser superiores à força opositora da gravidade.

Quando houver necessidade de carga ou movimentação de objetos, este deve ser considerado como elemento L₄ e também equacionado à fórmula acima.

Para efeitos didáticos, vamos desconsiderar tal componente.

[“Mr. Scott, I Need MORE Power”](#)

Star Trek – Gene Roddenberry

1.2 Básico sobre Eletrônica

Na apresentação deste tópico visualizaremos os principais conceitos que serão base para os demais capítulos. Neste sentido, daremos uma quebra no conceito do braço robótico e das rodas, pois precisaremos apresentar conceitos mais introdutórios até que se chegue ao ponto novamente de continuarmos com este assunto.

Energia

Tudo no universo é movido por energia. A energia movimenta as pessoas, são absorvidas pelas plantas e o robô também a consome.

O armazenamento de energia em pessoas e animais se dá através de processos bioquímicos, já as plantas adquirem a energia pela fotossíntese, que nada mais é que o armazenamento da energia em compostos orgânicos pela captura da energia luminosa.

Nos robôs, o processo não é tão complexo. Podemos utilizar duas fontes de energia:

1. A energia da rede elétrica – Através de adaptadores elétricos.
2. Através de baterias recarregáveis, que armazenam a energia, estocando para momentos em que a rede elétrica não estiver disponível.

Ao contrário do que pode se imaginar, as duas fontes não são excludentes entre si. Muito pelo contrário – são complementares.

A rede elétrica tem como vantagem poder fornecer ao robô grande quantidade de energia por longos períodos de tempo. Porém, ela exigirá que o robô se fixe em um local, o que nem sempre é possível.

A **bateria** tem capacidade limitada de armazenamento, porém permite que o robô possa se locomover para ambientes que não são servidos pela rede elétrica. Desta forma, o robô precisa seguir um ciclo contínuo de uso e recargas.

A identificação da atuação do robô permite definir se precisará de baterias ou apenas da rede elétrica.

A pergunta que deve ser feita é se o meu robô precisará se locomover. Se a resposta é afirmativa, possivelmente precisará de baterias. Para dimensionarmos a capacidade energética de nosso robô, precisamos começar explicando o tópico elétrico.

Eletônica de Potência

Entendamos que o nosso robô é representado pela Resistência R1, que neste circuito possui uma alimentação, seja externa ou interna, com tensão definida em V1.

Conforme a Lei de Ohm, temos a seguinte fórmula:

$$V = R \times I$$

Onde **V** é a tensão aplicada no circuito

R é a resistência a esta tensão

E **I** é o corrente elétrica dada em Amperes.

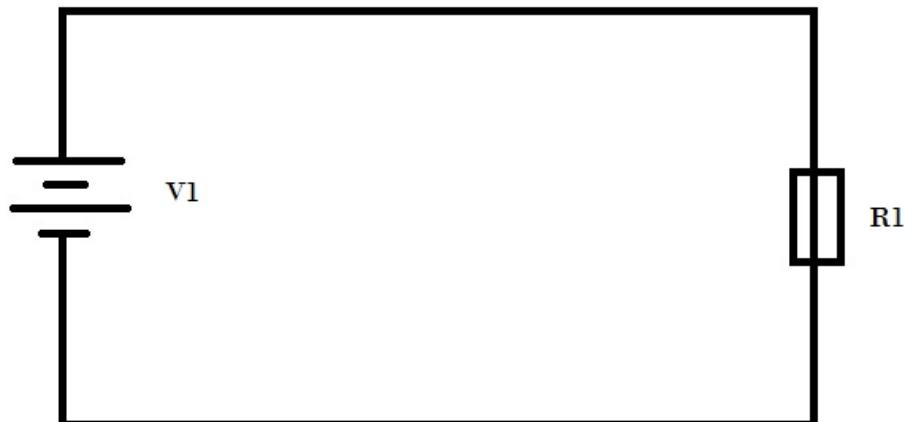


Figura 18 - Visão Geral de Consumo de Corrente

Por padrão, circuitos eletrônicos utilizam redes DC (Corrente Continua), que são redes elétricas que não variam sua polaridade em função do tempo. O padrão DC permite criar circuitos de pequena tensão e é adequado a circuitos integrados.

Por definição, equipamentos como Arduino e Raspberry utilizam tensões DC de 5 V. Porém, alguns equipamentos eletrônicos precisam de tensões maiores, como 12 V DC.

Um projetista, ao criar uma fonte ou ao usar uma já existente, não faz isto a esmo. A escolha da fonte precisa ser analisada, sendo que geralmente trabalharemos com a tensão de entrada sendo um pouco maior do que a do nosso sistema, retribalhando esta tensão para os demais circuitos, que utilizam tensão diferente.

Dimensionamento da rede elétrica do robô

Ao analisar um projeto robótico, como foi descrito anteriormente, temos que verificar as tensões dos componentes que são utilizados.

Analizando previamente os componentes iniciais:

Elemento	Tensão
Motor DC	12 V
Servomotor	5 V
Leds	5 V
Arduino	12 V
Raspberry	5 V

Cálculo de Tensão/Corrente e Resistência

$$V = R \times I$$

Uma aplicação prática desta fórmula é o dimensionamento de resistores para LEDs.

Ao aplicarmos LEDs no robô, não podemos fazê-lo de forma direta, pois, se assim ocorrer, queimaremos o LED.

É necessário associar o LED a um resistor, a fim de limitar a corrente passante no circuito e garantindo que o brilho seja o correto.

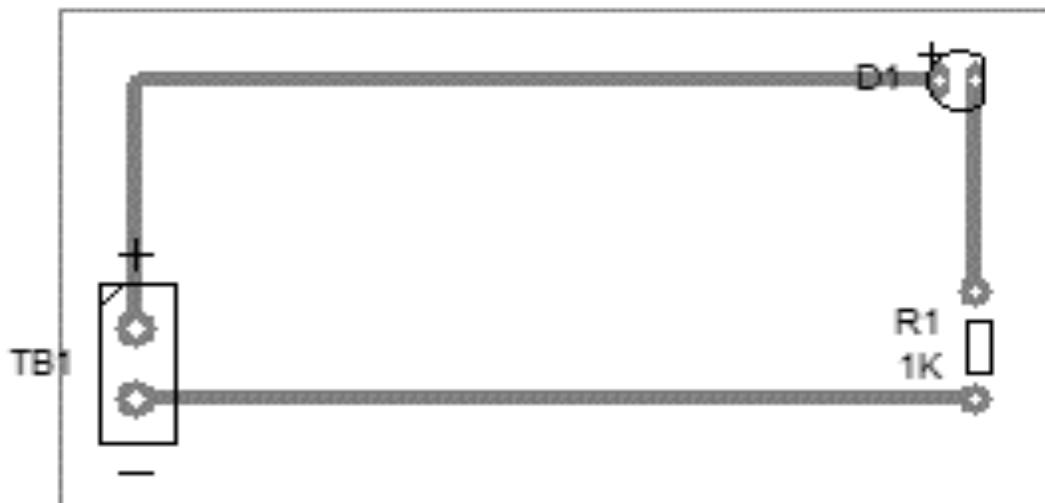


Figura 19 - Conexão led x resistor

Onde:

TB1 – Conector da fonte

R1 – Resistor

D1 – LED

Para atribuir o valor da resistência, primeiro, precisamos descobrir o tipo de LED queremos. Cada tipo de LED possui uma tabela de corrente (consumo). LEDs vermelhos e verdes usam 0,05 A (Amper), ao passo que LEDs brancos de alta intensidade usam 0,02 A.

Tipo de LED	Voltagem (V)	Corrente (A)
Azul, Ultravioleta, Branco alta luminosidade	3,7 V	0,02A
Verde, amarelo, laranja, Azul	1,6 V	0,05 A
Infravermelho	1,2 V	0,02A

Sabendo que a corrente do mesmo é 0,02A ou 200 mA e a tensão usada no sistema é de 12 volts, e a tensão consumida no led é de 3,7V.

Para dimensionar o resistor, fazemos a seguinte conta:

$$R = \frac{V - VI}{II}$$

Onde:

V é a voltagem do Circuito

VI é a voltagem de trabalho do led

II é a corrente de trabalho do led

R a resistência do resistor (o que se deseja descobrir).

$$R = \frac{12 - 3,7}{0,02} \Rightarrow \frac{8,3}{0,02} \Rightarrow 415\Omega$$

O uso de resistores com valores próximos pode ser utilizado.

Por exemplo, se tivermos resistores de 500Ω, irá funcionar, mas o brilho será um pouco menor. Caso utilizemos 5 V em vez de 12 V, como sugerido, ficaríamos da seguinte forma:

$$R = \frac{5 - 3,7}{0,02} \Rightarrow \frac{1,3}{0,02} \Rightarrow 65\Omega$$

Entendendo o Capacitor

O capacitor funciona como uma pequena pilha, que, quando disposto em paralelo com o circuito, proporciona a manutenção da carga necessária no momento de uma queda da tensão da Fonte V. Não iremos entrar no contexto desta queda, pois será abordado no Capítulo 2.

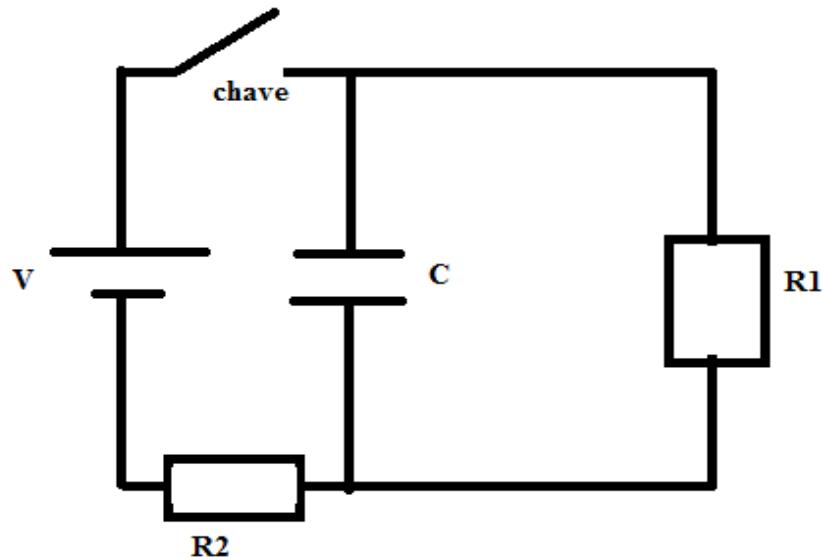


Figura 20 - Exemplo de uso de capacitor

Imagine que temos uma fonte V que alimenta o resistor R1, R2 e o capacitor C. Quando fechamos a chave, passa uma corrente elétrica no circuito, que carrega eletricamente o capacitor C até saturar.

No ponto da saturação não circula corrente elétrica no capacitor, ficando este como nulo no circuito. Porém, os Resistores R1 e R2 mantém o consumo de corrente.

Ao se abrir a chave, a fonte deixa de atuar no circuito. Através de análise de circuito, observamos que o Resistor R2 também será anulado.

Porém, o circuito ainda funcionará, pois, a carga acumulada no capacitor C manterá a corrente circulando no Resistor R1.

Cálculo de Capacitor

Em função, da explicação acima, podemos presumir que o capacitor possui um tempo de carga e descarga associado, sendo este tempo dado em função da resistência R2.

Conforme apresentado no livro “Construa Seu Próprio Computador Z80”, segue a fórmula para Cálculo de Capacitor:

$$C = \left(\frac{dt}{dv} \right) * I$$

- C = Valor do capacitor em farads
- I = corrente máxima do regulador
- dt = tempo de carga do capacitor
- dv = tensão de ripple admissível

Porém, para calcularmos os valores de dt e dv, precisamos considerar alguns elementos.

Tensão de ripple (dv)

$$V_c = V_{\text{pico}} - V_{\text{ripple}}$$

V_c = É a tensão que se deseja utilizar

V_{pico} = É a tensão máxima (pico) que se verifica

V_{ripple} = É a tensão mínima necessária para o perfeito funcionamento do sistema

Este cálculo será utilizado no Capítulo 2, porém, por questões didáticas, todas as principais equações são apresentadas neste capítulo.

Cálculo de Potência

Para acharmos a potência de um equipamento, aplicamos a fórmula abaixo:

$$P = V \times I$$

onde:

P é a potência dada em watts

V é a tensão dada em volts

I é a corrente dada em Amperes

Como estimar a potência de um circuito de forma prática e com poucos cálculos

A mais simples forma é por medição, utilizando um multímetro. Lê-se a corrente consumida, multiplicando pela tensão.

Porém, a medição desta forma está sujeita a erros, pois, tratando-se de um robô que possui elementos como servo motores, a corrente consumida pelos circuitos eletromecânicos pode variar em função dos esforços realizado pelos servos.

A forma mais segura é encontrar o consumo de cada item do projeto e somá-lo até obter a corrente total.

Exemplo de Cálculo de Potência

Quantidade	Item	Corrente
1	Arduino Mega	0,02A (Estimado)
1	Raspberry PI	0,700A
10	LEDs Verdes	0,05 *10 = 0,5A

Baseado no consumo da tabela acima, podemos prever a corrente do sistema. Para tanto, somamos as correntes.

$$\text{Corrente Total} = 0,02 + 0,7 + 0,05 \rightarrow 0,77A$$

Porém, para avaliarmos a potência, temos que considerar as tensões aplicadas.

Ficando:

$$\text{Potência Arduino} = 0,02 A * 5 V \Rightarrow 0,1 \text{ VA}$$

$$\text{Potência Raspberry} = 0,7 * 5 V \Rightarrow 3,5 \text{ VA}$$

$$\text{Potência LEDs} = 0,5 * 1,6 = 0,8 \text{ VA}$$

$$\text{Potência Total} = 4,4 \text{ VA.}$$

Potência Aparente, Reativa e Ativa

Em um mundo perfeito não há perdas, e toda energia elétrica fornecida em um motor é convertida em energia mecânica. De fato, isso não ocorre na prática, e parte da energia aplicada em um circuito se perde.

Desta forma, temos em um sistema elétrico:

- Potência Aparente (VA) – É a energia que a fonte precisa fornecer
- Potência Reativa (VAr) – É a energia perdida pelo circuito, esta perda pode ser devido a vários motivos.
- Potência ativa (W) – Potência convertida e transmitida pelo circuito. No caso do motor, o movimento, ou potência disponibilizada no circuito já descontada as perdas.

Vamos entender um pouco essas medidas e o que elas significam:

Quando informamos a capacidade de uma bateria, sempre fazemos pela Potência aparente, pois é a potência que será entregue pela bateria (potência máxima), indiferente das perdas no circuito.

Quando tratamos já na potência em um circuito (ou potência consumida), sempre consideraremos Potência Ativa (W), pois é a potência que será necessária para tocar o circuito, indiferente das perdas até chegar no circuito.

Na prática, quando tratamos do dimensionamento de uma bateria, super estimamos os valores de potência (potência aparente) pois entendemos que uma bateria que fornece 10VA jamais poderá fornecer 10W no circuito que a consome, pois existe a perda na Potência Reativa (Var).

Desta forma, no mínimo a bateria deve ter Potência Aparente maior que a Potência Ativa do circuito.

Estimativas justas, geralmente implicam em falha no projeto eletrônico e devem ser cuidadosamente pensadas.

Cálculo de Consumo

O consumo de um aparelho é dado em função do tempo em que está ligado, e este consumo mede a quantidade de energia que passou e foi consumida pelo circuito.

O consumo é dado pela seguinte fórmula:

$$C = Pxt,$$

Onde:

C = Consumo em watts/hora

P = Potência do circuito em watts

t = Tempo decorrido em horas

Alguns dos leitores irão se questionar sobre o cálculo da bateria, e muitos perguntarão onde foi parar. Bom, todo o estudo que fizemos até o presente momento serviu como base para o dimensionamento das nossas baterias.

Existem muitas baterias no mercado, cada qual com sua capacidade. Vamos agora definir que a bateria tem uma capacidade dita em Ah (Amper hora), pois as tensões dela são estabelecidas.



Figura 21 - Bateria NK 18650

Desta forma, uma bateria de 4,8 Ah, trabalhando com tensões de 3.7 volts, tem potência aparente da pelo cálculo:

$$P=4,8Ah*3.7V \rightarrow 17,76 V Ah$$

Em cada bateria (17,76VAh), sendo necessário um case de três baterias em série para se alcançar a tensão de 11,1 V, necessário para o funcionamento dos equipamentos (vistos posteriormente).

Onde:

$$Tempo = \frac{P_{Fornecida}}{P_{Consumida}}$$

Tempo = $53,28 \text{ VA} / 4,4 \text{ VA} \Rightarrow 12,1 \text{ horas ou 12 horas aproximadamente.}$

No cálculo acima, uma bateria totalmente carregada deve fornecer um total de aproximadamente 12 horas de energia para um circuito hipotético. Porém não é assim que funciona na prática.

Pois ao consumir a energia da bateria em um circuito a tensão da bateria vai decaindo até chegar a zero.

Quando chega a 30% consumida ou carga de 70%, considera-se esgotado a bateria, pois a tensão cairá para 7,7 V, aplicando na prática, 3,6 horas de uso, apenas considerando os LEDs, o Raspberry e o Arduino.

Uma prática normal, ao desenvolver o robô é sempre medir a tensão da bateria. Lendo valores inferiores a 9V, a bateria deve ser recarregada.

Ao calcular um robô, devemos considerar todos os componentes, que omitimos apenas por questões didáticas. Na prática de uso, e através de testes, constatamos que, após 15 minutos com todos os equipamentos, já não era possível utilizar a bateria.

Ligações em Série ou em Paralelo de Bateria

As ligações em série e paralelo são muito utilizadas em pilhas e baterias, porém às vezes sua prática é um pouco negligenciada. Vamos entender um pouco melhor este conceito, pois nos próximos capítulos iremos aplicá-los:

Vamos considerar nossa bateria inicial:

Definindo os valores e quantificando.

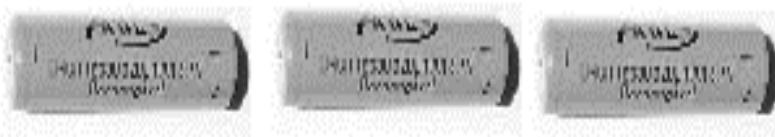


Figura 22 - Baterias em série

Neste tipo de bateria, a energia armazenada (capacidade) é **6,8 Ah**, e sua tensão é **3,7 volts**.

Quando alinhamos a bateria em série, suas tensões são somadas, porém sua capacidade se mantém. Desta forma, se medirmos a tensão neste sistema, chegaremos a **11,1 volts**, mas sua capacidade será mantida em **6,8 Ah**.

Entretanto, quando aplicamos as mesmas baterias em paralelo, verificamos o efeito contrário.



Figura 23 - Ligação em paralelo

Neste esquema, a tensão será mantida e a capacidade será somada, ficando definido como se segue:

Tensão do sistema: **3,7 V**

Capacidade do sistema: $6,8 \text{ Ah} + 6,8 \text{ Ah} + 6,8 \text{ Ah} = \mathbf{20,4 \text{ Ah}}$

Entenda porque funciona assim

Podemos explicar esse comportamento quando entendemos que a pilha trabalha com energia. Sendo assim, convertemos as duas medidas em uma única VA (Potência Aparente). Para tanto, voltamos na fórmula de Potência Elétrica.

$$P = V \cdot I;$$

Desta forma, calcula-se a potência de cada bateria,

$$P = 3,7 \text{ V} * 6,8 \text{ Ah} \Rightarrow 25,16 \text{ VA h}$$

Se cada bateria tem 25,16 VA h, a soma das potências das três baterias serão (25,16 V Ah + 25,16 V Ah + 25,16 V Ah) \Rightarrow **75,48 V Ah**

Desta forma, fazendo a ligação em série, teremos tensão de 11,1 volts, aplicando a mesma fórmula:

$$P = V \cdot I \Rightarrow 11,1 * 6,8 \Rightarrow \mathbf{75,48 \text{ VA h}}$$

Desta forma, a potência das pilhas não se modificou.

Na ligação em paralelo, a tensão seria a mesma, 3,7 volts, porém a capacidade seria somada, ficando 20,4 Ah, aplicando a fórmula de Potência:

$$P = V \cdot I \Rightarrow 3,7 \text{ V} * 20,4 \Rightarrow \mathbf{75,48 \text{ V Ah}}$$

Desta maneira, a energia potência da bateria se mantém.

Ferramentas básicas para eletrônica

Como qualquer projeto, o desenvolvedor de eletrônica necessita de ferramentas básicas para produzir e medir placas. Neste tópico veremos algumas ferramentas e passaremos o conceito básico sobre elas.

Multímetro

O multímetro torna-se equipamento obrigatório para qualquer um que deseja aprender eletrônica. O multímetro permite medir de forma simples e descomplicada os seguintes itens:

- Tensão DC – Tensão de Corrente Contínua, usada nos robôs
- Tensão AC – Tensão de Corrente Alternada, usada em residência
- Corrente consumida – A corrente consumida pelo sistema
- Testador de Sinal – Testa continuidade de fios, verificando se o mesmo fecha um circuito
- Resistência – Permite medir a resistência de componentes

Alguns multímetros, além destes itens, também possuem capacímetro, que é a medição de capacitores (dato em F), e frequêncimetro, que é a medição de frequências.

Ferro de Solda

O ferro de solda permite fazer uniões de componentes eletrônicos através da fusão de solda (estanho). O processo é realizado por meio do aquecimento da ponta do soldador, que é encostada ao componente que se quer soldar, em conjunto com o estanho, para que seja feita a fusão deste.

O ferro de solda é ferramenta obrigatória para qualquer um que desejar trabalhar com eletrônica.

Estanho

O estanho é uma liga metálica com baixo ponto de fusão (232 C), que permite fundir fios metálicos a fim de criar circuitos eletrônicos.

Dry film

Dry Film é uma película fotossensível que torna possível criar placas com acabamento e qualidade profissional. O Dry film facilita muito a confecção de PCBs (Placas de Circuito Elétrico), pois permite utilizar qualquer impressora na geração da matriz para criar os filmes.

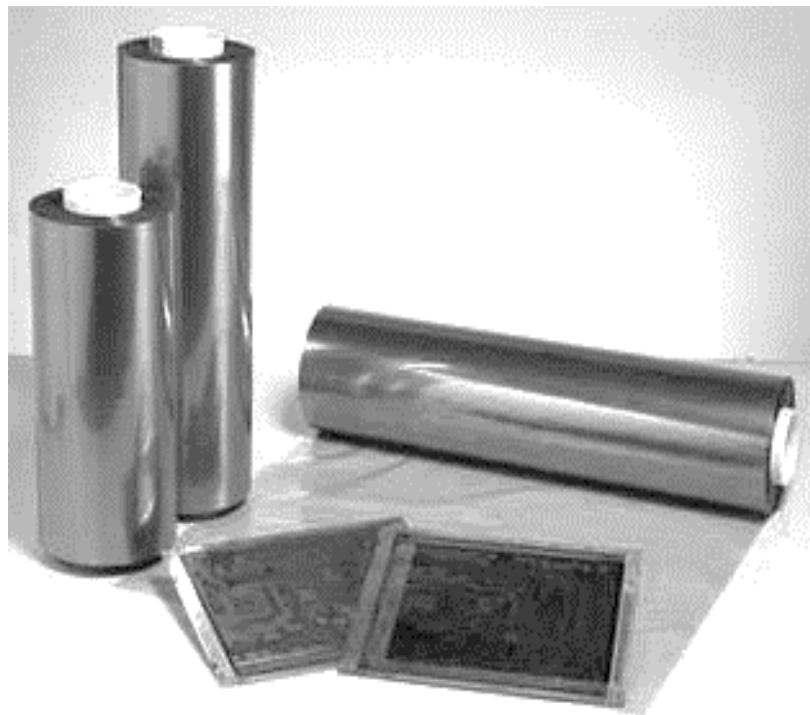


Figura 24 - Película de Dry Film facilita muito a montagem de PCBs

Película de Transparência jato de tinta para retroprojetor

A película de transparência é utilizada para a montagem da matriz que será utilizada na confecção das PCBs.

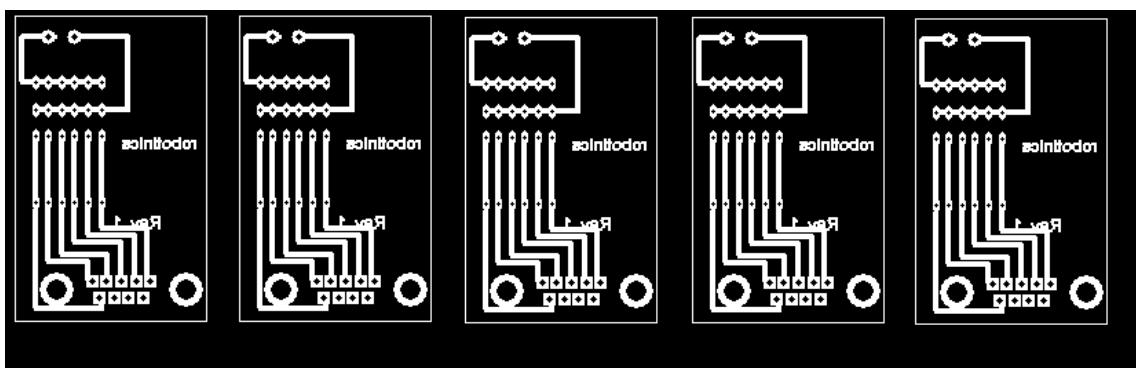


Figura 25 - Imagem de negativo impresso para confecção de placa PCB

O processo de montagem de uma placa PCB pela técnica de Dry Film equivale aos processos de silk screen. A parte clara permite a passagem de luz, que reage com o Dry Film e solidifica, mantendo-se após a lavagem do produto e protegendo a trilha da ação de corrosão.

Percloreto de Ferro

O Percloreto de Ferro é um composto químico utilizado para corrosão do cobre utilizado nas placas de PCB.

O percloreto, cuja fórmula FeCl_3 , quando em pó apresenta cor amarela, porém, quando dissolvido em água, sofre hidrólise, liberando calor (Wikipédia, s.d.).

Placas de PCB

As placas de PCB são placas de fibra de vidro ou equivalente, com uma camada de cobre recobrindo uma ou duas de suas faces.

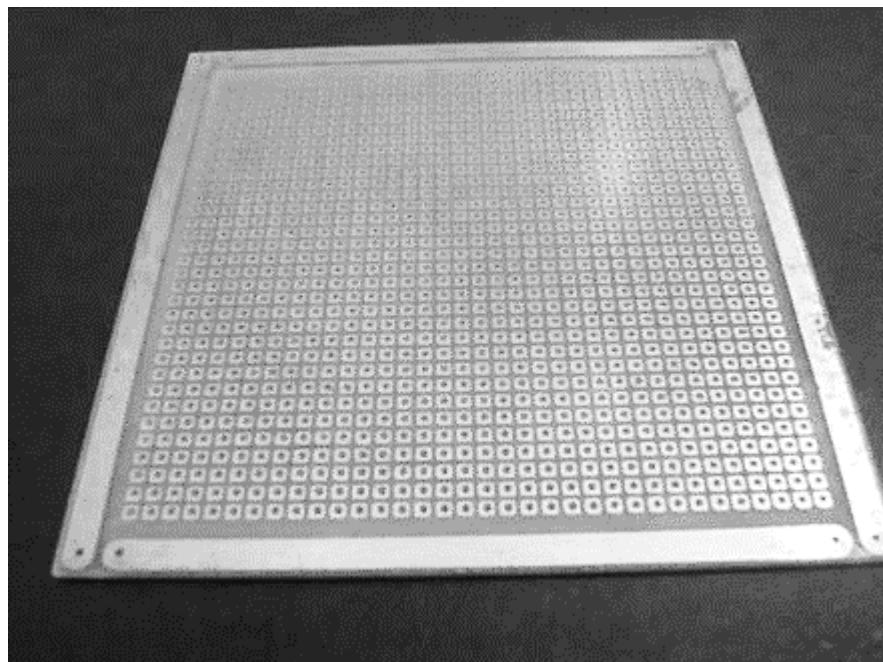


Figura 26 - Exemplo de placa PCB

As placas de PCB são utilizadas na confecção das placas customizadas deste livro.

Eagle

O Eagle é uma ferramenta proprietária da CadSoftUSA. Ela possui uma licença gratuita, porém limitada. É oferecida em versões para diversas plataformas, tais como Windows, Linux e Mac. Mais detalhes do Eagle podem ser vistos no site www.cadsoftusa.com.

O Eagle permite o desenvolvimento de esquemas elétricos, bem como circuitos de PCB, até sua finalização e exportação em padrão CAD. Esta praticidade permite criar e testar circuitos eletrônicos de forma simples e fácil.

O livro foi escrito na versão 7.3.0, porém as versões são atualizadas constantemente.

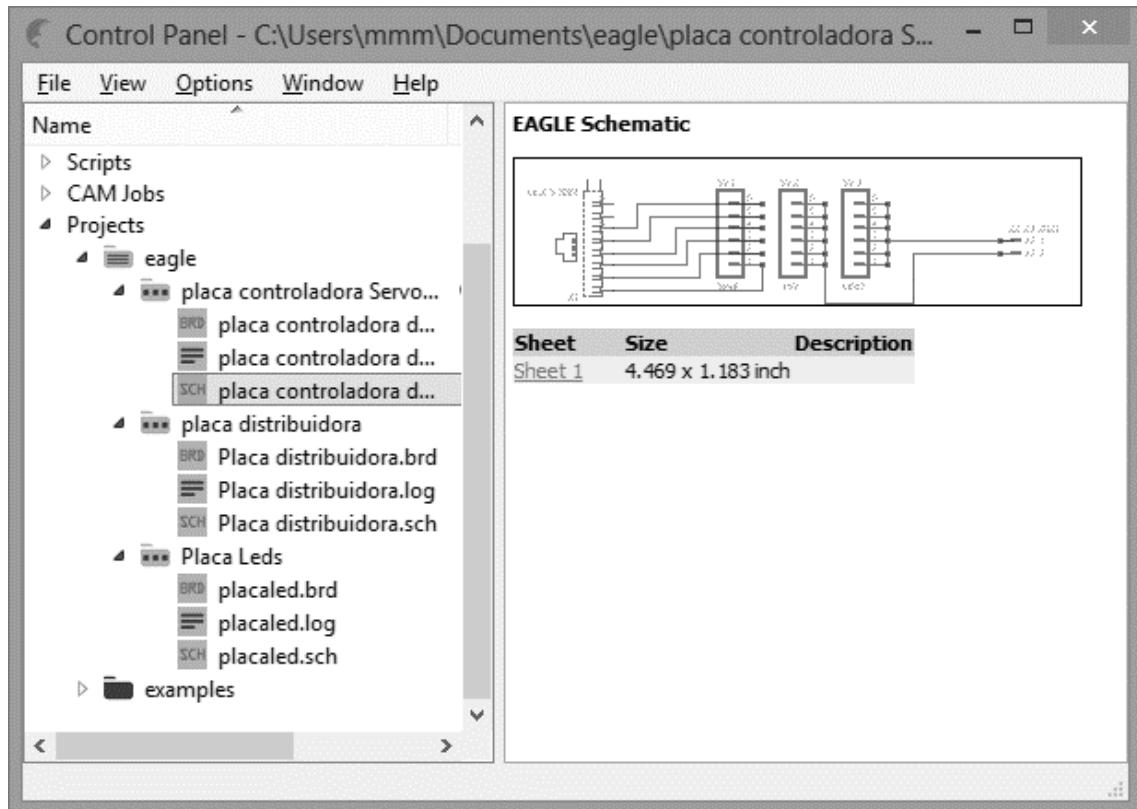


Figura 27 - Software Eagle, da CadSoft

O Eagle trabalha com conceito de projetos, sendo que existem duas grandes visões:

Esquemático – É a montagem das ligações ou parte das ligações. Não reflete, no entanto, a montagem da placa, e sim as ligações lógicas dos componentes.

Board (Placa) – É a montagem da placa, onde todas as estruturas lógicas do esquemático são importadas e é feito o posicionamento das peças nas placas.

Na versão gratuita o Eagle limita placas de 10 x 10 cm.

O autor recomenda o uso do Eagle, inclusive pela facilidade na integração com SolidWorks.

Não estaremos aqui abordando os detalhes para desenvolvimento do Eagle, apenas comentando alguns procedimentos para sua utilização.

A utilização do Eagle consiste em alguns passos simples:

- 1) Criação do Schematic (esquemático da placa), onde são abordados os aspectos lógicos (componentes e suas ligações elétricas)

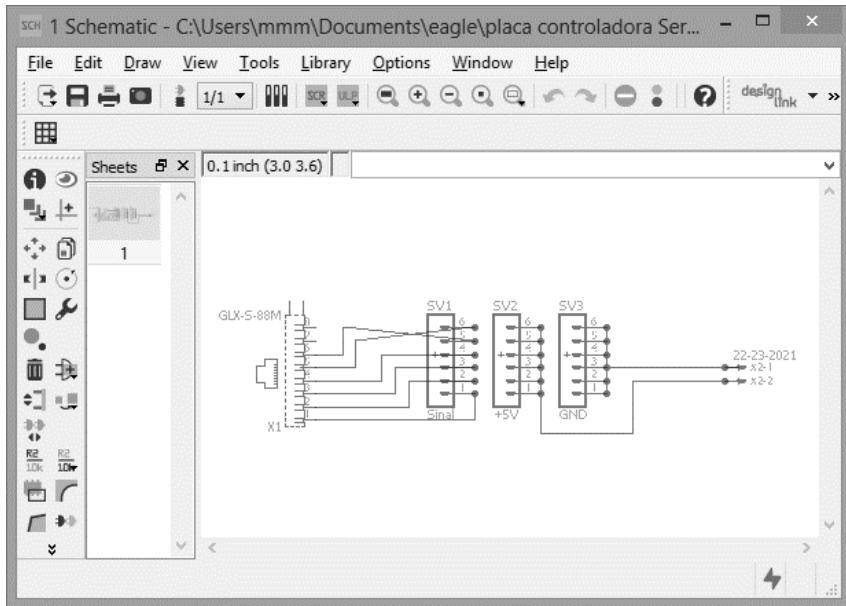


Figura 28 - Visão do Schematic do Eagle

- 2) Após sua criação, exporta-se o Schematic para criar a visão da Board (PCB). Para tanto, usa-se o botão
- 3) Com o projeto da Board (PCB) criada, são definidas as trilhas (nets), terminando o projeto.

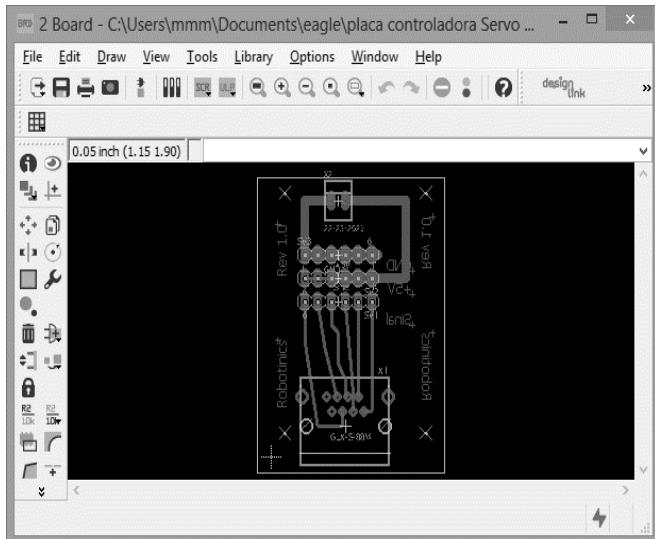


Figura 29 - Visão da Board

1.3 Software

O software é a programação do hardware. Para entender este conceito, pensemos no corpo humano.

Nós somos pele, ossos e sangue. Cada um de nós tem todos esses atributos, mas cada um é único. Apesar de nosso hardware (cabelo, ossos, sangue) ser semelhante, nosso software (consciência) é diferente.

O software é bem isso mesmo. Ele não é material, não se pode pegar. Ele existe atrelado ao nosso hardware (corpo), porém não se limita a ele. No caso dos computadores, o software não pensa, pelo menos ainda. O software contém um conjunto de instruções que desenvolve ações previamente programadas, reagindo de forma planejada a estímulos dados pelo hardware e realizando reações controladas, em que cada ação é planejada previamente, contendo uma resposta previamente projetada.

Neste sentido, o software controla:

- Os sinais de entrada
- Os sinais de saída

Tudo através de um programa que lê as entradas, interpreta e, por fim, estabelece os padrões de saída.

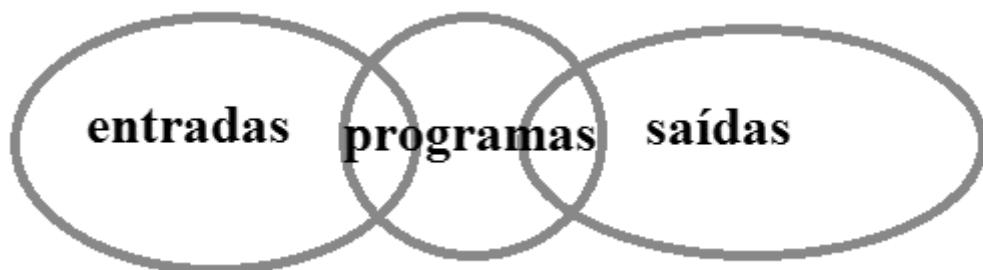


Figura 30 - Ligação entre entradas e saídas

Tipos de Desenvolvimento

O desenvolvimento de um projeto robótico envolve várias atividades de desenvolvimento. Podemos destacar, entre as atividades de desenvolvimento de um robô:

- ✓ Desenvolvimento das placas de leitura dos sensores de I/O
- ✓ Desenvolvimento do programa da CPU
- ✓ Desenvolvimento das interfaces de conexão
- ✓ Ferramentas de teste

Desenvolvimento das placas de leitura dos sensores – Programamos todos os dispositivos de entrada e saída (I/O).

Desenvolvimento dos programas da CPU – Controlamos as programações de alto nível, que controlam não só o robô, mas também as lógicas projetadas para suas ações. Nesta etapa do projeto podemos utilizar os dispositivos de I/O, consumindo-os em redes de processamento, prevendo ações e gerenciando saídas.

São ações desenvolvidas na CPU:

- Análise e interpretação de sinais sensoriais
- Projetos de rede neural (análise de padrões)
- Armazenamento de rotinas de memória e armazenamento
- Interfaces de conexão com usuários
- Conexões e gerenciamento de dispositivos de acesso remoto, como bluetooth, wifi e ethernet.
- Serviços de conexão, como SSH, Telnet, Web, socket server
- Reconhecimento de padrões visuais (opencv)
- Interface de gerenciamento por toque (touch screen)
- Informações aparentes em LCDs e telas

Desenvolvimento de Interfaces de Conexão são relacionadas às interfaces externas ao robô, como webservices ou dispositivos programadores ou gerenciadores. Estas interfaces podem estar situadas em notebooks, tablets, celulares ou mesmo na nuvem, permitindo que o robô faça ou receba comandos e ou atualizações.

Ferramentas de Teste são, como o próprio nome diz, ferramentas desenvolvidas durante a fase de desenvolvimento para testar parte ou componente do robô. Nela podemos identificar e calibrar elementos, servindo de meio para configuração. Estas ferramentas podem ser enquadradas no desenvolvimento de interface de conexão, mas seu uso é tópico e restringe-se à fase de desenvolvimento e testes.

Ações voluntárias e involuntárias

Assim como uma pessoa, um dispositivo possui ações voluntárias e involuntárias. As ações voluntárias garantem que consigamos operar o dispositivo da forma requerida. Entretanto, algumas atividades são involuntárias, não dependem de planejamento e sua ação faz com que a intervenção garanta a segurança no controle do dispositivo, bem como previne sua quebra.

Entre ações involuntárias, podemos prever:

- ✓ Parada de movimento por detecção de obstáculo em distância menor que a mínima necessária
- ✓ Desligamento compulsório em caso de bateria baixa
- ✓ Controle de posicionamento de Servo motores, evitando movimentos bruscos ou em ângulos não permitidos
- ✓ Limitação de controle de carga consumida¹

Algumas destas ações são controladas não por um dispositivo alto nível, como o microprocessador, mas sim por um microcontrolador.

Projetando o equipamento

Estaremos, a partir deste ponto, tratando cada um dos componentes abordados nos itens anteriores.

A forma e composição de um projeto de software pode variar em função da estrutura física proposta, bem como da finalidade apresentada.

Não temos a pretensão de dizer que a nossa visão é a única e nem a melhor. Por experiência, pude perceber que a melhor prática é a que tem melhor resultado com menor custo. Neste sentido, fatores como experiência do programador, custo de componentes e área de aplicação às vezes são mais importantes que a própria engenharia de software ou tecnologia envolvida.

✓ ¹ Controle de Carga Consumida – É o processo de software que evita o consumo de carga momentânea elevada, ele verifica movimentos que estão sendo realizados pelos servos, e o consumo total estimado, desta forma evita que a tensão da bateria não caia a níveis críticos, o que pode ocasionar o famoso reset do robô. Também realiza o monitoramento da carga residual da bateria, garantindo que o robô irá funcionar em qualquer situação. Também encerrando o funcionamento quando a bateria alcance nível critico.

Programação no Arduino

O Arduino é o hardware que lê os sinais do robô (Input), tais como temperatura, proximidade da parede e velocidade, ativa os dispositivos de saída (Output), tais como servo motores, motores DC, luzes, etc.

Ao conjunto de dispositivos de entrada e saída controlados pelo arduino, chamamos de dispositivos I/O.

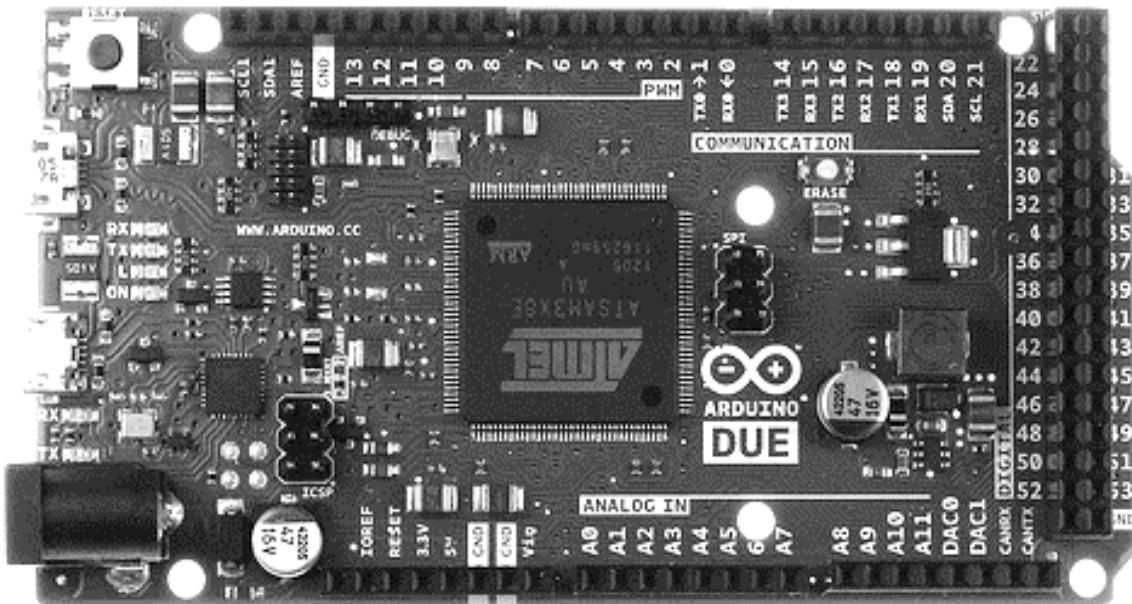


Figura 31 - Arduino Mega

A programação do Arduino se dá através de um código pseudo C (Wiring), que permite ser carregado no computador.

Este código é carregado por um cabo USB para a memória do microcontrolador.

O programa gravado dentro do micro controlador é chamado de Firmware, que é basicamente um software embarcado.

A programação do micro controlador exige um conhecimento mais específico, e sua função é o **gerenciamento do hardware**, conforme apresentado no tópico anterior.

São responsabilidades do Arduino:

- Leitura de sinais analógicos como tensão das baterias, corrente passante do sistema etc.
- Leitura de sinais digitais (sensor proximidade, I2C etc.)
- Controle dos movimentos dos servo motores
- Controle do movimento das rodas e sentido (controlador ponte H)

- Acionamento das saídas digitais (LEDs, reles)
- Controle das ações involuntárias e automáticas²

Para desenvolvimento do fonte e respectiva compilação do firmware, o Arduino utiliza uma IDE, que é um conjunto de ferramentas especialmente criadas para o desenvolvimento dos programas em Arduino.

Esta IDE pode ser baixada no site do fabricante, www.arduino.cc.

Download

Para compilar o Arduino, deve-se baixar gratuitamente a versão do compilador no site do fabricante (www.arduino.cc).

A interface é muito simples e permite, de forma intuitiva, a programação conforme apresentado a seguir.

² São ações automáticas que não necessitam de comando externo, tais como mecanismos de parada automática ao perceber uma parede, desligamento automático em caso de bateria fraca.



Figura 32 - Exemplo da IDE do Arduino

Compilar – Compilar é transformar o código escrito em linguagem C e converter em código de máquina, que é a linguagem que o equipamento conhece.



Para compilar, basta você utilizar o botão .

Carregar – A Carga é o processo de transferir o programa compilado para o equipamento. Este processo é feito através da conexão do equipamento no computador utilizando um cabo USB.

Configuração do ambiente

Para permitir que um programa funcione e compile corretamente no equipamento, é necessário informar ao compilador qual a versão do seu hardware.

Para tanto, seleciona a opção:

Ferramentas > Placa, indicando a placa que está conectada ao computador.

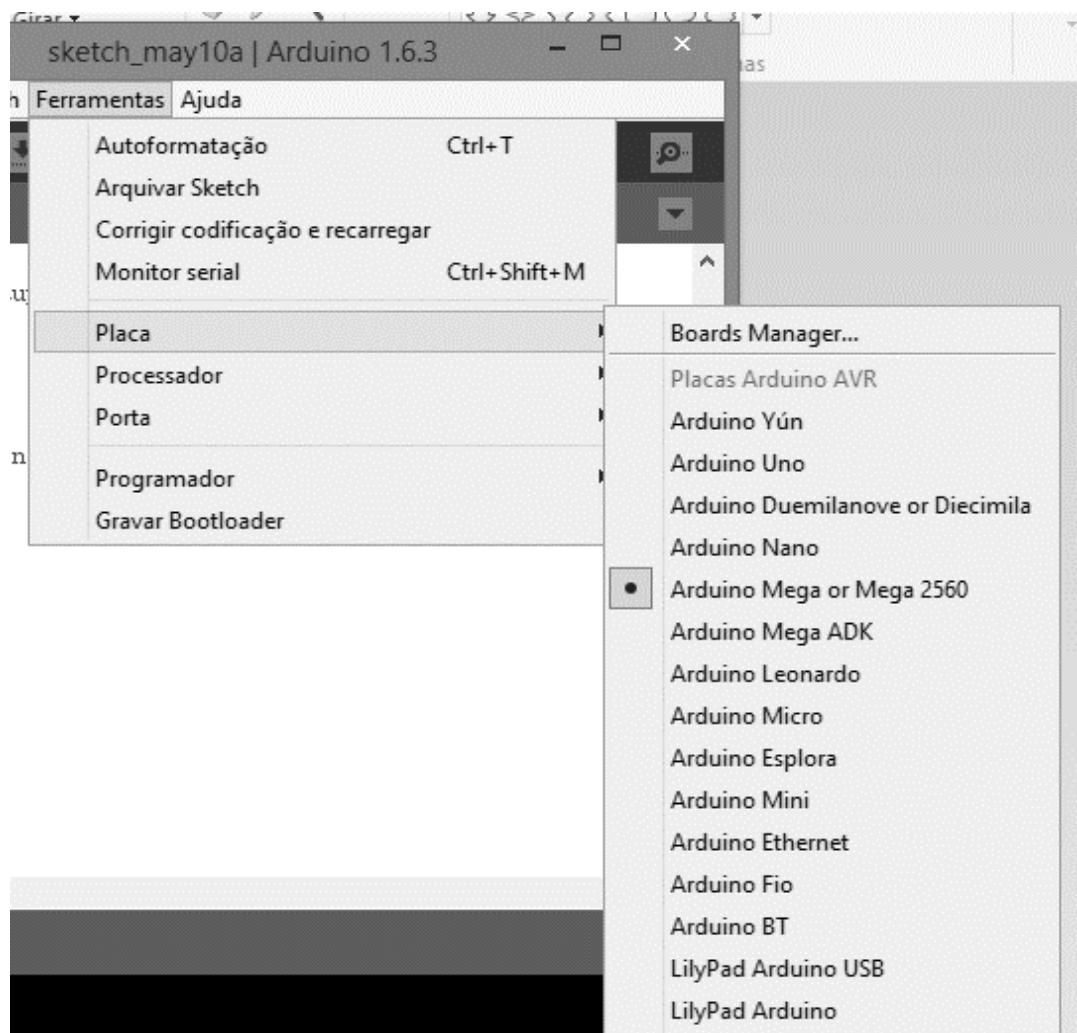


Figura 33 - Seleção da Versão do Hardware

Outra informação importante é a configuração da porta USB que será utilizada.

Para isso, selecione a opção **Ferramentas > Porta**, escolhendo a porta que foi criada para este dispositivo.

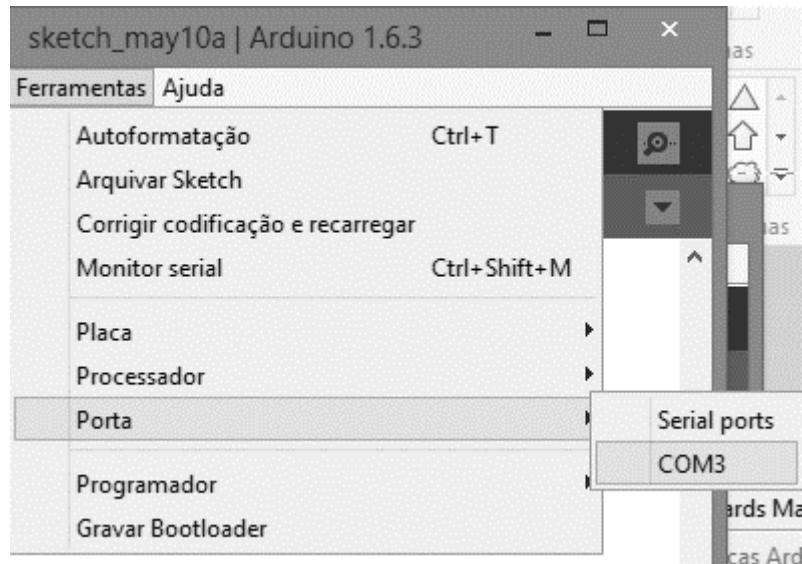


Figura 34- Seleção da Porta de Comunicação USB/Serial

Programação no Raspberry

Ao contrário do Arduino, que possui um número limitado de alternativas para desenvolvimento, o Raspberry permite que se faça uso de diversas linguagens, tais como PHP, Java, C, C++, Free Pascal, Python, Lazarus, entre outros.

A programação em Raspberry é muito ampla, e também permite a instalação de diversos outros programas através da conexão com o site do fabricante e download dos pacotes.

O Raspberry PI é um exemplo de microprocessador de baixo custo e com ótimo custo-benefício.

Porém, atualmente há uma infinidade de outros projetos de microprocessadores semelhantes ao Raspberry que também podem ser utilizados também sem perdas ao nosso projeto, tal como o Cubieboard, Odroid, entre outros.

O Raspberry tem a função de **Desenvolvimento dos programas da CPU³**, sendo utilizada para permitir interfacear os comandos que pretendem que sejam realizados pelo robô.

Para armazenamento, o Raspberry utiliza um cartão de memória SD, que pode variar de 2 até 32GB até o presente dia.

As placas possuem duas entradas USB e um cabo de alimentação micro USB. Além disso, possuem saída de áudio estéreo, saída de vídeo HDMI e RCA. Ao longo deste livro faremos um passo a passo desta ferramenta.

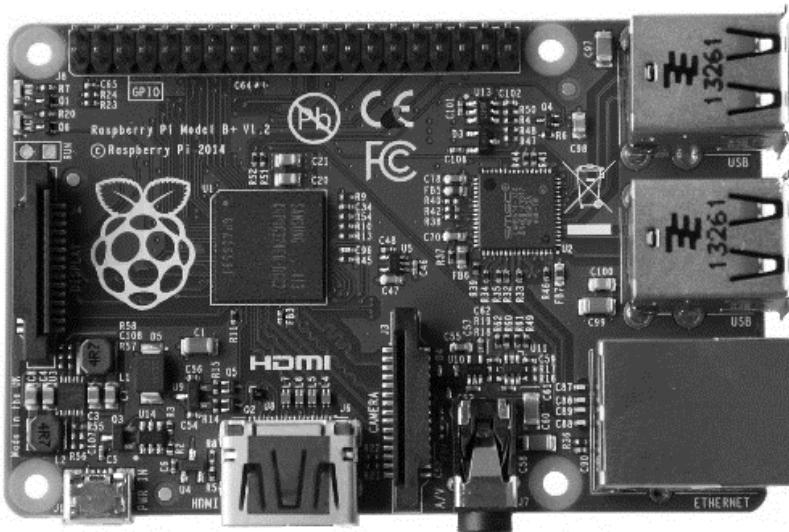


Figura 35 - Foto do Raspberry Pi 2

³ CPU – Unidade de Processamento Central ou do inglês “Central Processing Unit” de onde veio a sigla.

Sistema Operacional

Ao contrário do Arduino, o Raspberry possui um sistema operacional que é pré-instalado em máquina. Este sistema operacional tem a responsabilidade de gerenciar o hardware, permitindo que diversas aplicações sejam gerenciadas, também interfaceando dispositivos como placa de rede, áudio e GPIO (Conexões de Propósito Geral).

Download

O Raspberry PI permite que diversos sistemas operacionais sejam instalados. Para isso podem ser baixados uma imagem do sistema operacional através do site www.raspberrypi.org.

Recomenda-se o uso do RASPBIAN, que é a distribuição recomendada pela própria fabricante.

Como instalar o Raspbian OS

Para instalar o sistema operacional é necessário baixar uma cópia da ISO (cópia do sistema para instalação).

1. Baixe a ferramenta Win32DiskImager (<http://sourceforge.net/projects/win32diskimager/>)
2. Instale a ferramenta
3. Coloque o cartão SD no computador
4. Indique no drive a unidade do SD criada
5. Abra o arquivo ISO e descompacte na pasta do cartão SD
6. Retire o cartão do PC e conecte na entrada do Raspberry PI

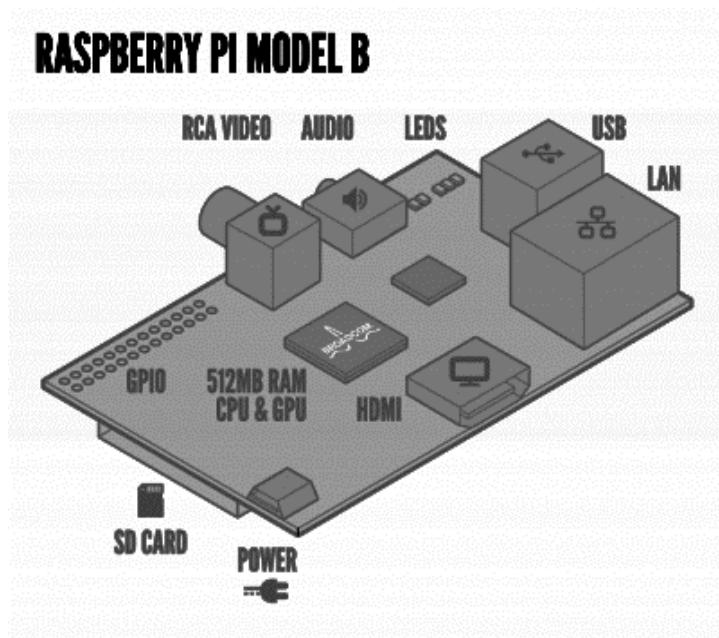


Figura 36 - Dispositivos do Raspberry

Acessando

Para acessar o dispositivo, conecte-o na porta HDMI de uma televisão, conectando o Raspberry a um teclado e mouse USB.

Ao conectar pela primeira vez, surgirá um menu para que você configure o equipamento e o sistema operacional.

Para logar no sistema, o usuário padrão é **pi** e a senha **raspberry**.

Instalando pacotes de aplicações

Para instalar os pacotes necessários ao robô, basta digitar o seguinte comando: *apt-get install <nome aplicação>*

Será necessário a instalação dos seguintes aplicativos:

```
apt-get install ssh  
apt-get install arduino  
apt-get install gcc  
apt-get install make  
apt-get install cmake  
apt-get install mysql-server  
apt-get install apache2  
apt-get install libapache2-mod-php5 php5 php-pear php5-xcache php5-mysql php5-curl php5-gd  
apt-get install lazarus  
apt-get install espeak  
apt-get install vncserver  
apt-get install motion  
apt-get install vim  
apt-get install build-essential  
apt-get install ffmpeg  
apt-get install pkg-config
```

Ferramentas Auxiliares

O uso de ferramentas auxiliares para teste e desenvolvimento conectado ao PC é muito importante para que o desenvolvedor possa sincronizar e desenvolver corretamente o projeto.

Filezilla

Ferramenta de transferência de dados entre o PC e o Raspberry PI.

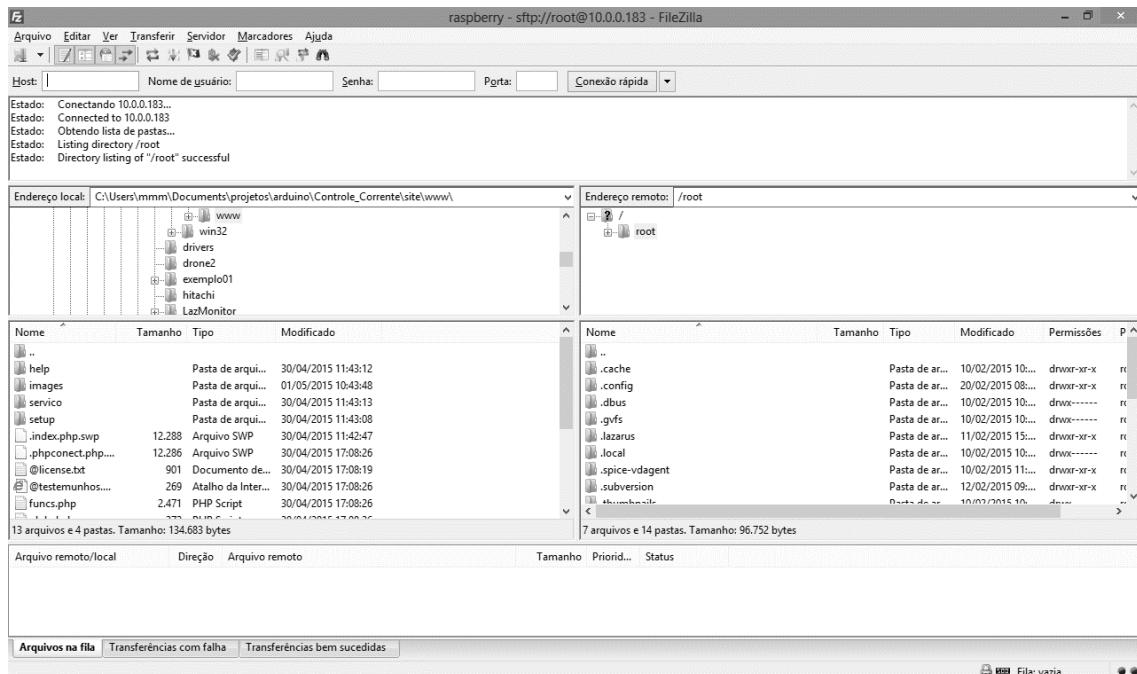


Figura 37 - Ferramenta de transferência de arquivos

Conforme apresentado na ilustração a cima, o Filezilla permite transferir arquivos de um PC para o robô, bem como utilizar edição no PC, sincronizando automaticamente.

Download: <https://filezilla-project.org>

mRemoteNG

Ferramenta similar ao PUTTY, permite a conexão e execução de comandos no robô

Download: <http://www.mremoteng.org>

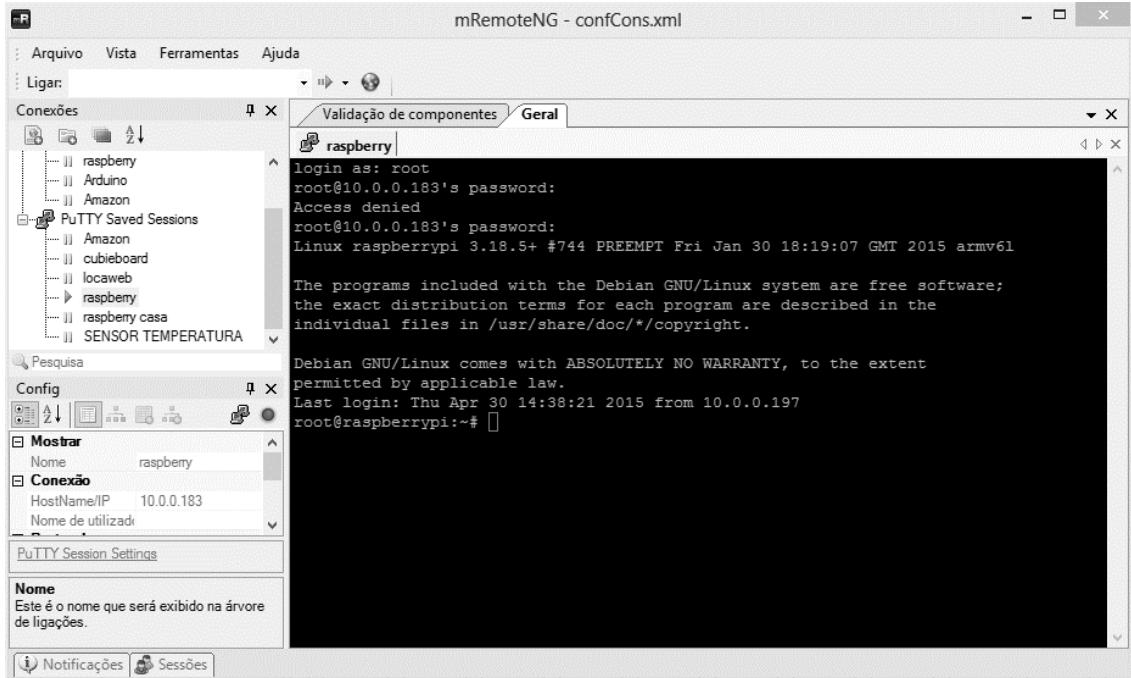


Figura 38 - Exemplo de conexão feita facilmente pelo mRemoteNG

O mRemoteNG facilita em muito a utilização e desenvolvimento de projetos, pois integra diversas conexões, tais como SSH, VNC e Telnet, entre outras.



Figura 39- Tipos de conexões existentes no mRemoteNG

Desenvolvimento em C para Raspberry

Este tópico especial para desenvolvimento em C mostrará os fundamentos e é um módulo à parte justamente pela importância do C no desenvolvimento de robôs.

Para darmos inicio ao nosso programa, usaremos um hello.c. Edite o seguinte código, salvando como hello.c.

Use o vim, que é a interface melhorada do vi. Para tanto, digite na pasta o seguinte comando:

```
$> vim hello.c
```

Para salvar no vim, basta pressionar a tecla ESC, em seguida o comando :wq, que significa w – salvar , q – sair



```
#include <stdio.h>

int main(void)
{
    printf("Hello World\n");
}

"hello.c" 8L, 67C           8,0-1          All
```

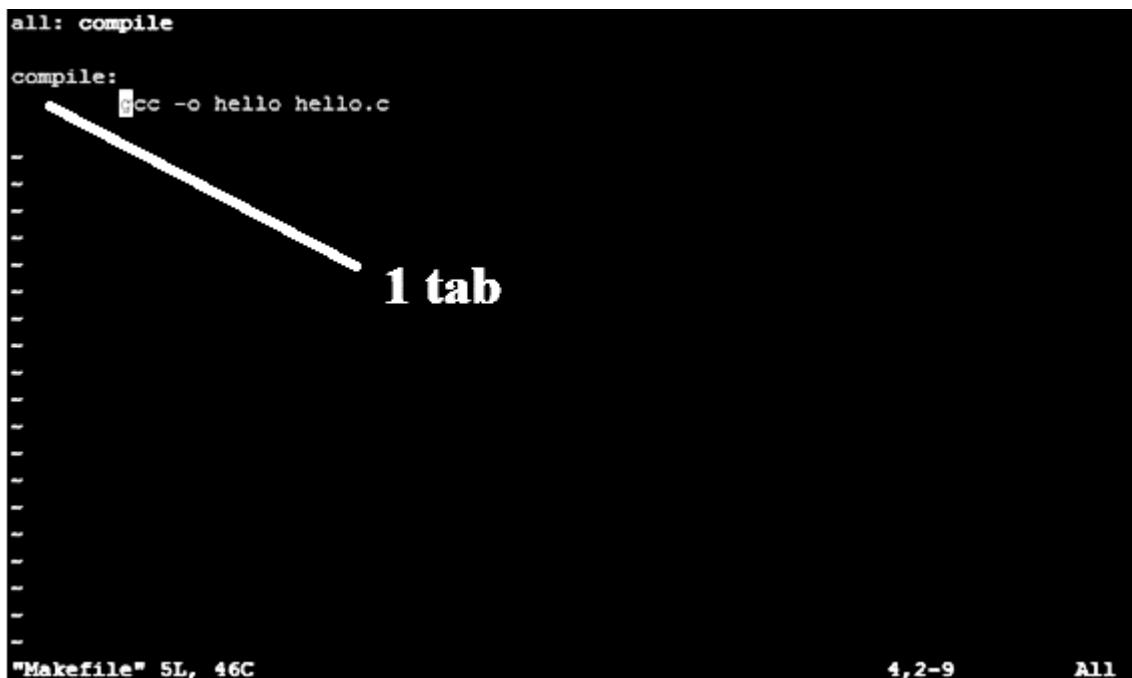
Figura 40 - Exemplo de uso do VIM para edição

Makefile

Crie o arquivo Makefile:

```
all: compile  
  
compile:  
    gcc -o hello hello.c
```

Editando conforme figura abaixo.



The screenshot shows a terminal window with a black background and white text. It displays a Makefile with the following content:

```
all: compile  
  
compile:  
    gcc -o hello hello.c
```

A large white arrow points from the word "compile:" to the command "gcc -o hello hello.c". Below the arrow, the text "1 tab" is written in large white letters. At the bottom of the terminal window, there is some status information: "Makefile" 5L, 46C on the left, 4,2-9 in the center, and All on the right.

Figura 41 - Código Makefile para compilação

Salve e na pasta dos arquivos execute o comando. Não esqueça de que os espaços no arquivo Makefile são um tab, e não espaço. **Caso digite espaço, o Makefile não funcionará corretamente.**

```
-bash-3.2$ vim hello.c
-bash-3.2$ vim Makefile
-bash-3.2$ make
gcc -o hello hello.c
-bash-3.2$ ./hello
Hello World
-bash-3.2$ █
```

Figura 42 - Resultado final do programa

Se não compilar, verifique cuidadosamente o código-fonte em busca de erros de sintaxe.

Para rodar o programa, basta digitar na pasta onde está o comando

```
./hello
```

O resultado será :

```
Hello Word!
```

```
$>
```

Desta forma encerramos o processo básico de desenvolvimento em C para Linux.

Caro leitor, o foco deste livro não é ensinar os fundamentos do C, e sim apenas introduzimos os preceitos desta linguagem para quem não tem nenhuma experiência prévia na linguagem.

Recomenda-se, caso o leitor não conheça a linguagem C, que utilize qualquer obra que explana sobre os fundamentos da linguagem.

Mais adiante abordarei partes de código-fonte em C, explicando não a linguagem, mas sim as bibliotecas e funções necessárias para o uso do robô.

Também será fornecido, neste material, os fontes e projetos necessários para composição do seu robô.

Muitas bibliotecas, onde estão

Uma das perguntas mais comuns para os desenvolvedores é “onde estão instaladas as libs que meu programa utiliza?”.

Existem várias formas de solucionar este problema, e o pkg-config é uma delas.

Para rodar o pkg-config, siga o exemplo:

```
pkg-config --cflags --libs pocketsphinx
```

Para entender melhor para que serve esse comando, vamos olhar um exemplo de Makefile:

```
CC=gcc  
CFLAGS=-I/usr/local/include -I/usr/local/include/sphinxbase -I/usr/local/include/pocketsphinx  
export CFLAGS  
LIBS=-L/usr/local/lib -lpocketsphinx -lsphinxbase -lsphinxad  
  
all: compile  
compile: exe1 exe2  
  
exe1:  
    $(CC) -o teste teste.c $(CFLAGS) $(LIBS) -DMODELDIR=""
```

Quando monto um Makefile na mão, preciso incluir os vínculos de todas as libs que utilizo no meu programa.

Para tanto, uso no meu Makefile o parâmetro LIBS e CFLAGS para indicar os caminhos da lib.

Em cada lib eu preciso identificar os caminhos físicos.

Agora imagine utilizar este script para compilar em ambientes que não consigo controlar onde estão instalados estas libs, nem a versão instalada da lib?

Percebeu como tornou-se difícil gerenciar a aplicação!

Bom, o pkg-config resolve esta situação, apresentando uma forma prática e didática para compilar dinamicamente meu programa, pois ele entrega os caminhos das libs de forma simples.

O exemplo a seguir mostra, de forma resumida, como ficaria a lib, já incluindo os links do pocketsphinx:

```
gcc -ggdb `pkg-config --cflags --libs pocketsphinx` -DMODELDIR="" teste2.c -o teste2
```

PHP

O PHP é um acrônimo de Hipertext Preprocessor, destinada a realizar programação em especial dentro de páginas Web.

Em nosso projeto o PHP será utilizado para permitir que o robô tenha uma interface web, enviando e controlando funcionalidades pelo Browser.

Não entraremos no detalhamento da linguagem, nem abordaremos o processo de instalação a fundo, pois este é instalado pelos scripts que posteriormente passaremos.

Desenvolvimento em Windows e Linux X86

O desenvolvimento das interfaces de conexão, bem como ferramentas de teste, são elementos necessários, como apontados anteriormente. Dependendo do sistema operacional utilizado, podemos empregar vários tipos de linguagens de programação, entre elas C, C++, Delphi, Lazarus, PHP, C#, Java, Shell Script e Python.

A escolha da melhor ferramenta não é uma tarefa fácil, e deve acontecer baseada principalmente em critérios práticos, como experiência do desenvolvedor e outros aspectos técnicos e psicológicos.

Estaremos aqui utilizando as seguintes linguagens para interface:

- Interface Web, PHP
- Programação de ferramentas desktop para configuração e testes – Lazarus

Nos próximos capítulos daremos início às atividades de desenvolvimento do nosso robô, criando as estruturas fundamentais para tal empreitada.

Lazarus

O Lazarus é uma RAD Open Source desenvolvida sobre os moldes do Delphi. Ele permite a compilação de uma aplicação em diversas plataformas:

- Linux X86
- Linux Arm (Raspberry)
- Windows 32 bits e 64 bits
- Mac OS
- Android
- Entre outras plataformas

O uso do Lazarus neste projeto é permitir o desenvolvimento de aplicações acessórias no desktop (computador) para comunicar com o robô, de modo a controlá-lo remotamente, bem como testar seu funcionamento.

Download: <http://www.lazarus-ide.org>

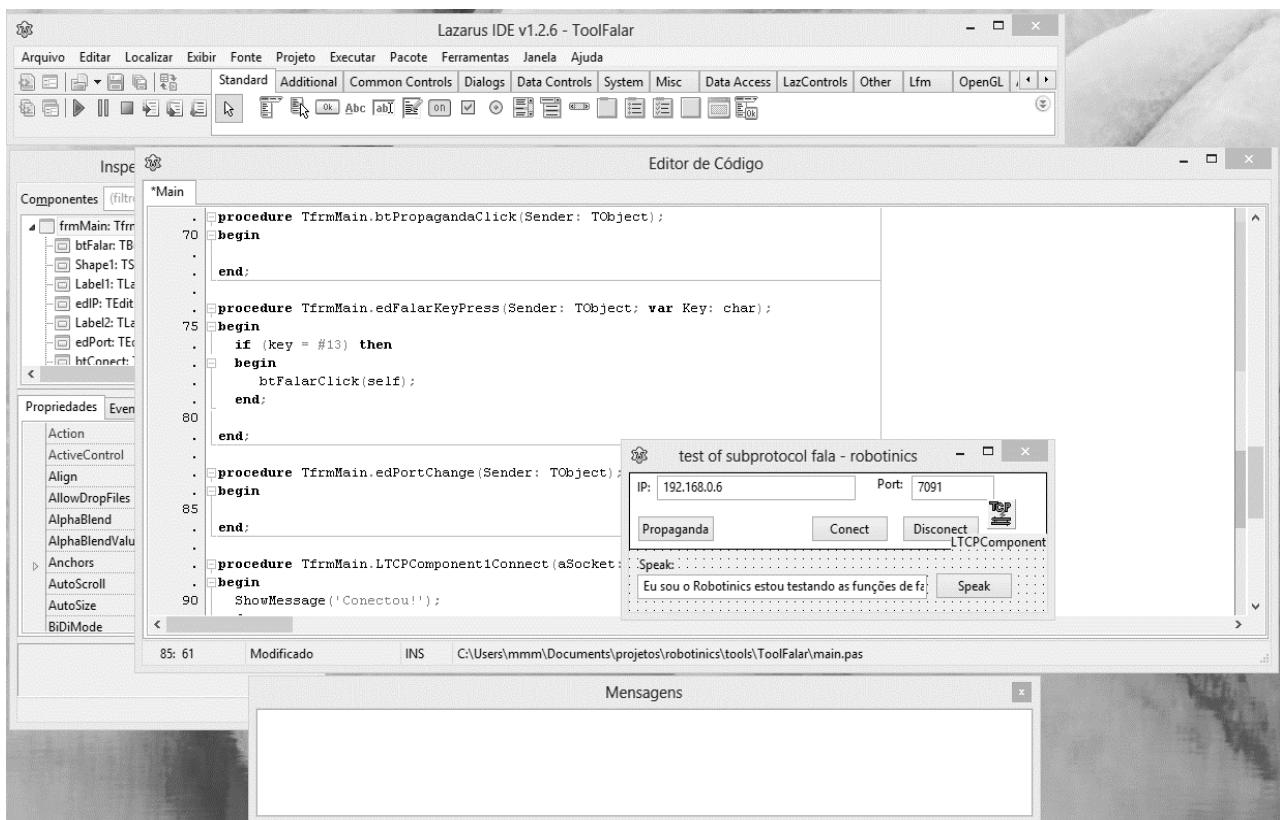


Figura 43 - Interface do Robotinics – Módulo de Comunicação de Fala

Bibliotecas

As bibliotecas são pacotes de componentes que permitem a utilização de recursos adicionais, como comunicação Ethernet, Serial e componentes visuais, entre outros no Lazarus.

5dpo comunicação serial: <http://wiki.lazarus.freepascal.org/5dpo> – Comunicação Serial

LNET: <http://sourceforge.net/projects/lazarus-ccr/files/INet> – Biblioteca de comunicação socket

O Lazarus possui diversas bibliotecas gráficas que podem ser incluídas no pacote e compiladas. Existe farto material na internet que possibilita a configuração destas libs. Recomendo que assistam meu canal no YouTube, onde demonstro a configuração destas ferramentas: www.youtube.com/user/marcelomaurin.

O uso e configuração do Lazarus, bem como a composição desta linguagem, foge do tema deste livro, sendo indicadas outras fontes de referência para tal.

Explicaremos nesta obra os fragmentos relevantes ao desenvolvimento em Lazarus, para confecção de interfaces no PC que façam a comunicação com o robô.

Todos os projetos serão incluídos para teste e download.

2. Construindo seu robô

A partir de agora iremos construir os diversos módulos do robô.

No primeiro capítulo foi apresentado todo o conceito básico introdutório. Caso não tenha lido o primeiro capítulo, recomendo fortemente que o faça.

2.1 Mecânica da Base do Robô

Nesta etapa iremos construir a base, onde será depositada toda a eletrônica da básica do robô. No projeto mecânico definiremos as estruturas mais fundamentais para compor um robô móvel por rodas. Para tanto, vamos definir a base inferior e a base superior.

Base Inferior

A base inferior é responsável pela fixação das rodas e apoio. Formada em acrílico, tem a responsabilidade da sustentação.

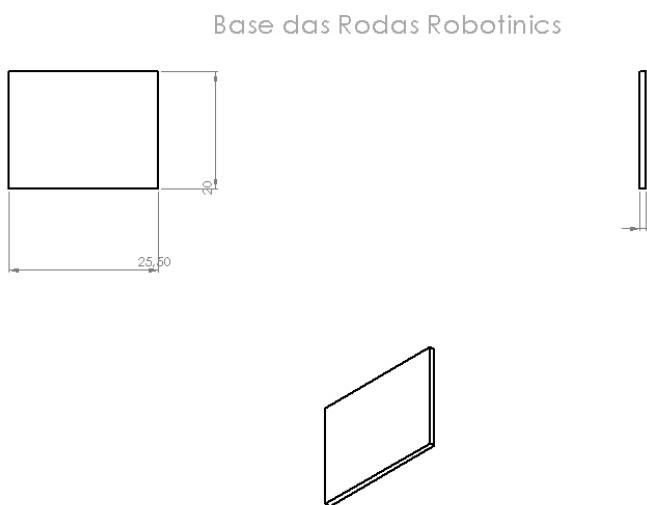


Figura 44 - Desenho em visão da base inferior

Medidas da placa acrílica 25,5cmx 20cm com 1 cm de espessura.

Arquivos e Modelos

Aqui estão contidos os arquivos originais da estrutura em SolidWorks e sua respectiva impressão 3D .

Arquivo SolidWorks: BaseRodas.SLDPR

Material: Acrílico

Base Superior

A base reta superior se liga ao corpo do robô, permitindo a fixação e passagem dos fios e equipamentos.

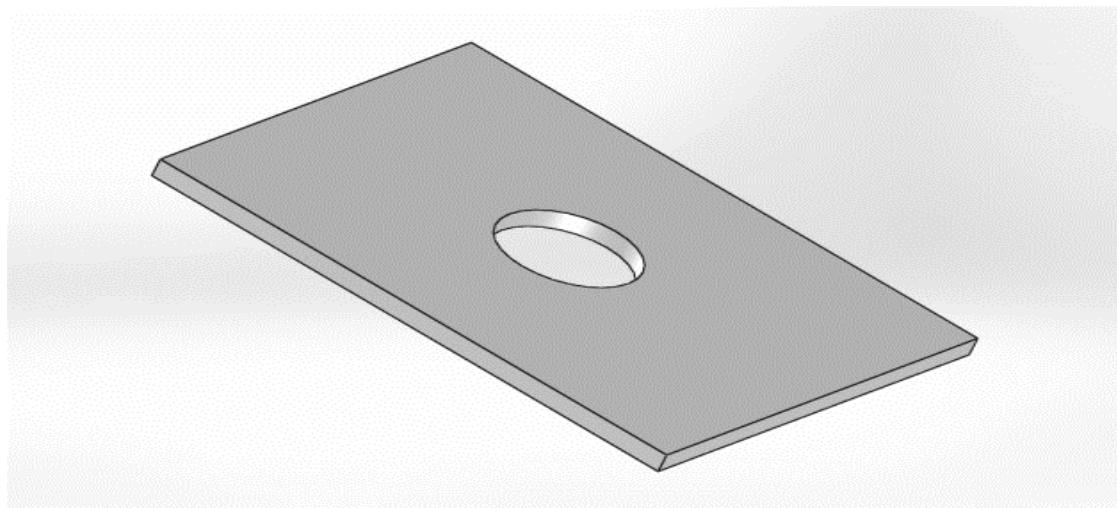


Figura 45 - Visão da base superior criada no SolidWorks

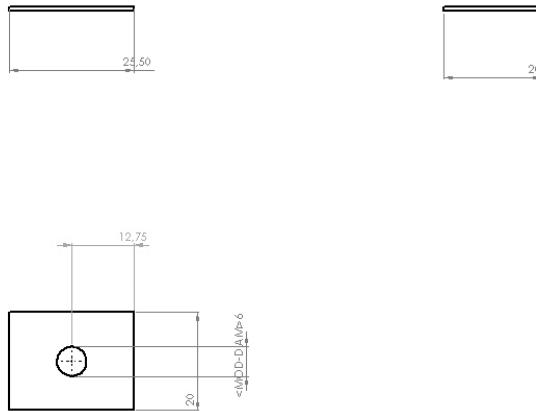


Figura 46 - Especificações técnicas da base reta superior

Medidas da placa acrílica 25,5cmx 20cm com 1 cm de espessura.

Arquivos e Modelos

Aqui estão contidos os arquivos originais da estrutura em SolidWorks.

Arquivo SolidWorks: robotinics\solidwork\BaseReta.SLDPR

Material: Acrílico

Rodas

No Capítulo 1 abordamos todos os cálculos necessários para composição de força e potência para tração nas rodas. Neste sentido não entraremos nos detalhes da especificação, ficando limitados aos detalhes construtivos.

Testei estas rodas até o peso total de 3 kg. Porém, nesta configuração as rodas e os eixos sofrem um esforço muito forte.

Recomendo que estas rodas sejam utilizadas em peso de um robô de até 3 kg.

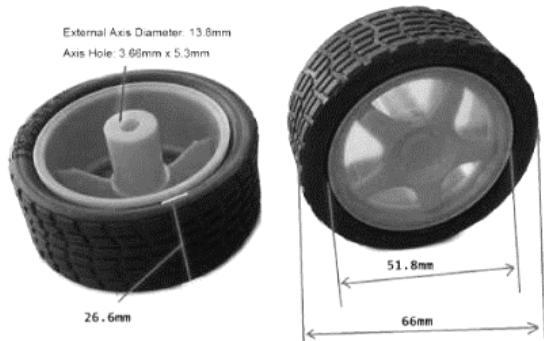


Figura 47 - Rodas do robô

Motor com caixa de redução

O motor é responsável pela geração do movimento e permite que o robô se movimente.

As especificações técnicas do motor e da caixa de redução foram passadas e calculadas no Capítulo 1, ficando apenas os aspectos construtivos aqui passados.

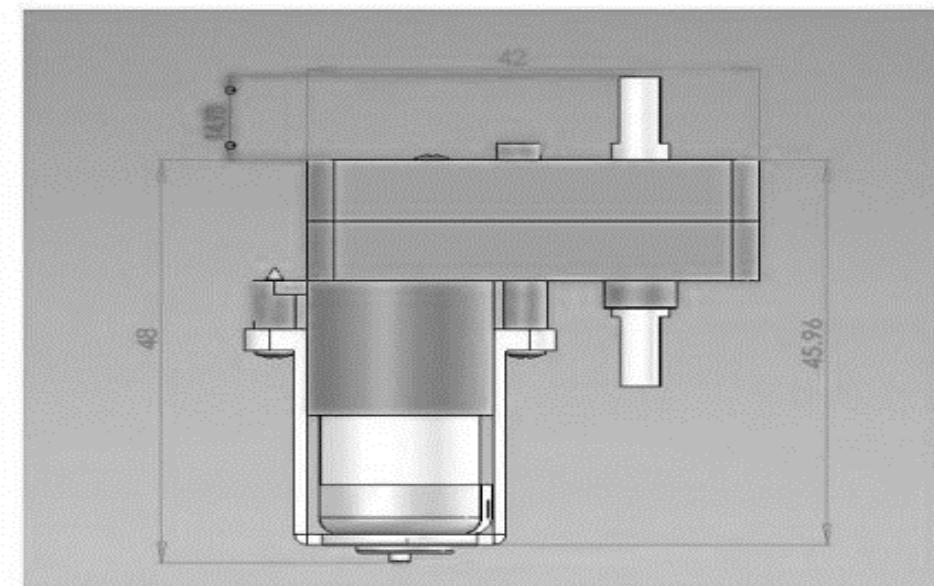


Figura 48 - Dimensionamento da caixa de redução

Utilizaremos quatro motores com caixa de redução para fixação do motor, conforme descrito na especificação inicial.

Suporte para motor

O suporte será fixado na base do robô para fixação dos motores na placa. A resistência deste suporte depende do peso que será sustentado.

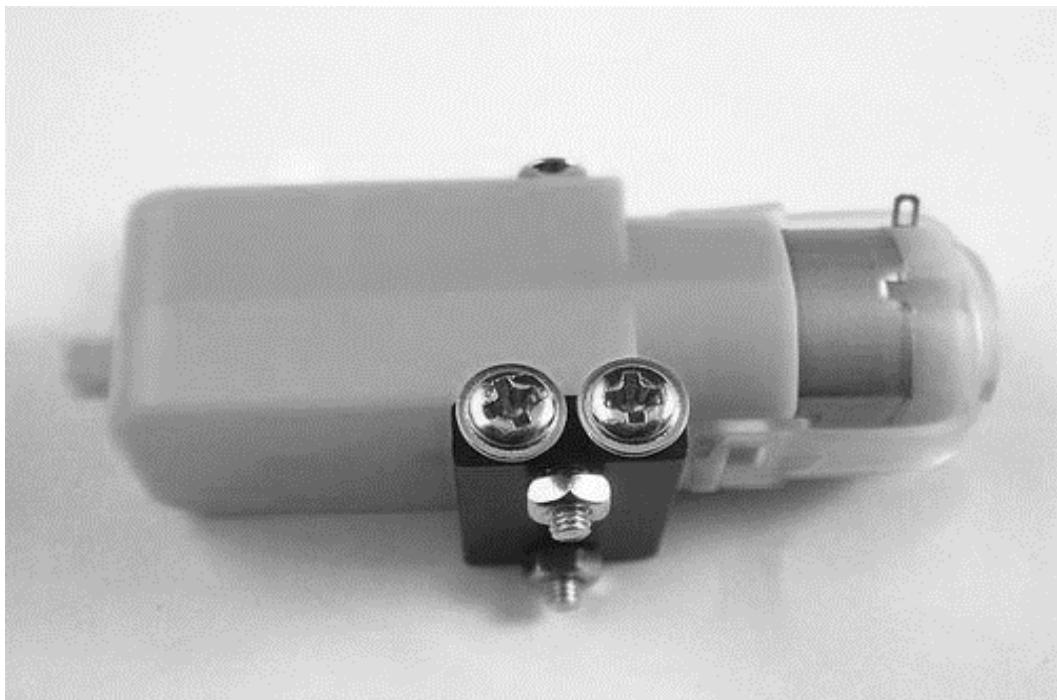


Figura 49 - Suporte e fixação para motor e caixa de redução

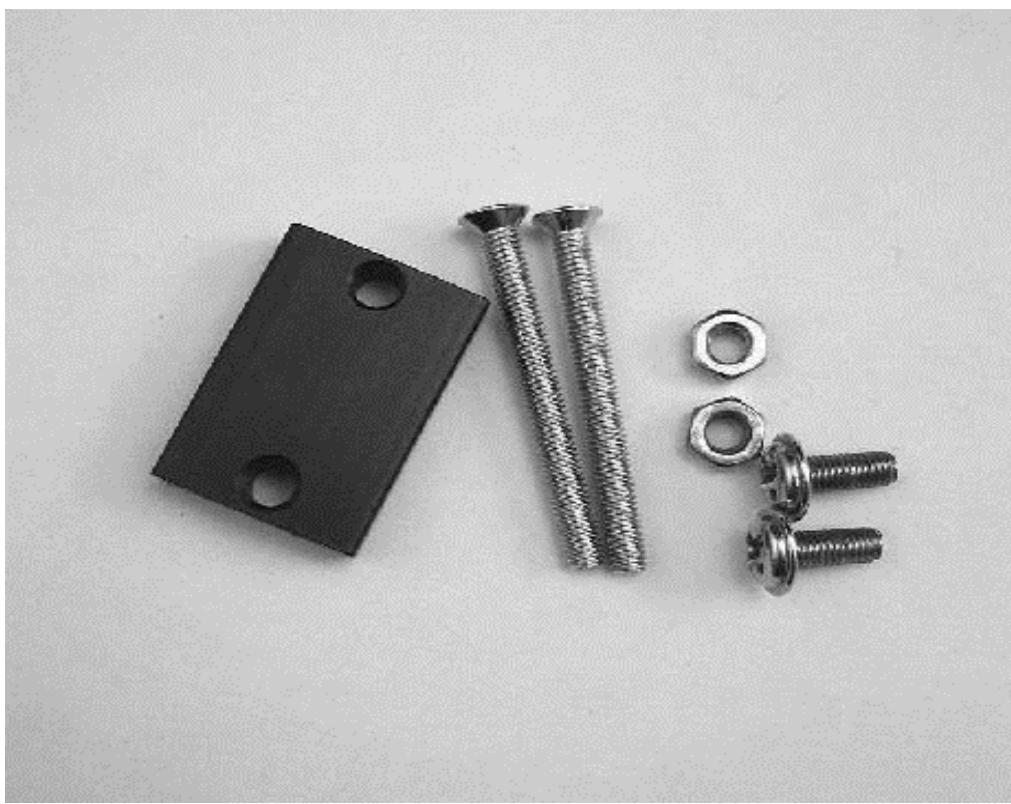


Figura 50 -Elementos de fixação do motor

O suporte para o motor fixa e garante que o motor se prenda à base inferior. Sua fixação e firmeza garantem que a roda não se solte durante a movimentação do robô.

De fato, todo o peso do robô será distribuído entre as quatro rodas, devendo este suportar tal esforço.

Os parafusos são fixados na lateral do motor e unidos com na base acrílica, segurando e suportando o peso do robô, através deste suporte de ferro.

Prolongadores da Base

Têm a responsabilidade de fixar o robô na base superior, dando apoio e sustentação ao robô.

Os prolongadores também chamados de alongador, podem ser comprado facilmente na internet.

Permite acoplar placas laterais a fim de fechar o robô, dando acabamento ao mesmo.



Figura 51- Vista dos prolongadores

Material: Metal

Tamanho: 5,6cm

Fixação de parafusos de 2.8mm

Montagem mecânica

Nesta última etapa montaremos a base, fixando todos os itens mecânicos e eletrônicos que estão sendo definidos neste capítulo.

A montagem mecânica é a parte final de qualquer processo produtivo e auxilia a indústria em como deve proceder e quais as sequências necessárias para montagem da base robótica.

Procedimento de Montagem

1. Pegue os motores com caixa de redução, fixando estes nos suportes dos motores
2. Fixe os suportes dos motores à base inferior
3. Fixe o prolongador à base inferior
4. Fixe o suporte de porta-pilhas na base inferior
5. Fixe a placa do circuito do carregador à base inferior
6. Fixe o step down à base inferior
7. Com os fios apropriados, faça a ligação elétrica entre a placa do carregador e o step down
8. Com os fios apropriados, faça a ligação elétrica entre a placa do carregador e o suporte de porta-pilhas.
9. Fixe o relê à base inferior
10. Conecte os três fios do relê às respectivas entradas da placa do carregador
11. Conecte o fio apropriado na saída de energia
12. Conecte o fio apropriado na entrada de energia

2.2 Eletrônica

Planejamento da Fonte Externa

O planejamento de uma fonte externa é o primeiro e principal componente a ser considerado em qualquer projeto eletrônico. Não existe nenhum projeto eletrônico que possa funcionar sem energia. A qualidade da entrega da energia influencia diretamente no circuito eletrônico – tanto para aspectos como confiança e estabilidade de funcionamento.

Para explicar esta afirmação, basta lembrar que o robô possui diversos mecanismos, tais como Servo motores, que dependem da carga elétrica para sua movimentação.

Também a fonte elétrica deve possuir capacidade superior a capacidade total consumida pelo robô, mais a capacidade utilizada para carga da bateria.

Outro aspecto importante é com relação a picos de energia.

Os equipamentos eletrônicos são construídos para trabalharem em faixa de tensão pré-estabelecidas. No entanto, fontes mal projetadas podem fornecer picos de energia que por vezes podem ultrapassar estas faixas, bem como, efetivamente, chegar a valores inferiores a tensões mínimas de trabalho.

Projeto de Fonte AC/DC

O projeto de fontes de alimentação está cada vez mais em desuso, pois a compra de fontes externas de baixo custo “e às vezes baixa qualidade” acaba por inviabilizar o projeto eletrônico de uma fonte.

Porém, seu entendimento fornece subsídios para a escolha da melhor solução, evitando problemas futuros com fonte de má qualidade.

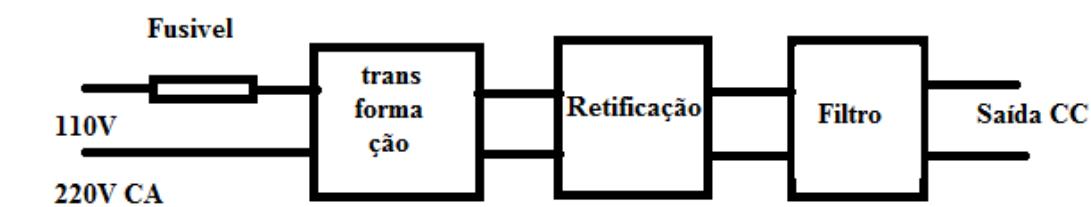


Figura 52 - Projeto de uma fonte básica

Para efeito prático, serão abordados os principais elementos de uma fonte de alimentação, apresentando como é realizado, conforme apresentado por (Ciarcia).

Existem três grandes blocos de trabalho na composição de uma fonte de alimentação:

- Transformação
- Retificação
- Filtro

Transformação – Como o próprio nome diz, transforma uma tensão em outra. No nosso caso, de 220 V ou 110 V em 12 V. O trafo⁴ apenas eleva ou diminui a tensão, preparando esta para ser tratada.

Ao se analisar a saída de um transformador, percebe-se que a tensão foi abaixada, porém ainda é CA (corrente alternada).

Retificação – A retificação consiste em transformar uma tensão AC (alternada) em CC (corrente contínua). De forma geral a retificação utiliza uma ponte de diodos para retificar. Hoje já existem CLs que componentizam uma ponte de diodo.

⁴Trafo é a abreviação de transformador

Filtro – O filtro tem a responsabilidade de manter o nível de tensão suficiente entre os ciclos de carga, permitindo que a tensão não decaia tanto que não permita o funcionamento dos equipamentos.

A tensão AC possui ciclos de alternância, chamado ripple⁵ de tensão. Este ciclo varia, sendo de 50 Hz ou 60 Hz, conforme o país. No Brasil, o ciclo é de 60 Hz.

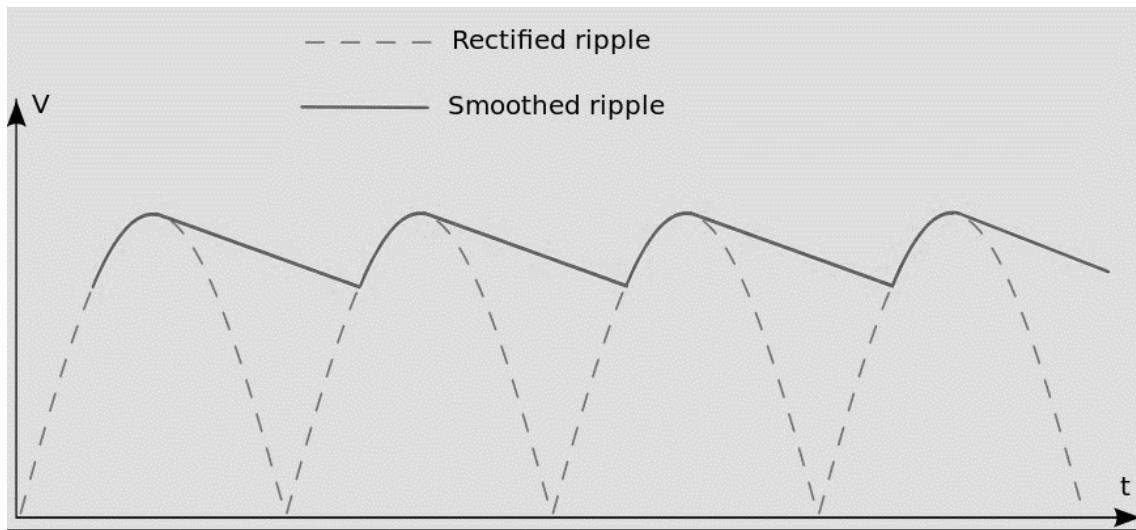


Figura 53 - Ripple de tensão após ser filtrada e o ripple na saída da ponte de diodos

Dimensionamento de um Capacitor

Conforme apresentado no Capítulo 1, no item de cálculo de capacitores, verificamos que $C = (dt/dv) * I$.

Para dimensionar o DV(tensão de ripple admissível), basta verificar a tensão de trabalho que precisaremos ter. Esta tensão de trabalho deve sustentar o circuito.

Nossa fonte pode ter que trabalhar com 12 V, porém, como visto anteriormente, a tensão oscila constantemente, e precisamos atribuir um valor médio.

Desta forma, trabalharemos com tensão de 10 V.

A fórmula para cálculo de VRipple é dado:

$$V_c = V_{\text{pico}} - V_{\text{ripple}}$$

Onde convencionou tensão V_c (Média) 10 V, $V_{\text{pico}} = 12 \text{ V}$;

$$10 = 12 - V_{\text{ripple}} \Rightarrow V_{\text{ripple}} = 12 - 10 = 2 \text{ V}$$

Desta forma, podemos calcular o valor aproximado do nosso capacitor.

⁵Ripple – Conforme apresentado no livro (Ciarcia) é chamada tensão pico a pico.

Conforme (Ciarcia):

Dt é dado, 8,3ms (120 Hz)

Corrente máxima do Regulador = 5A

$$C = (dt / dv) * I \Rightarrow 8,3 * (10^{-3}) * 5 / 2 \Rightarrow C = 20,75 * 10^{-3} \text{ ou aproximadamente } \mathbf{20,75\mu F}$$

Projeto de um carregador de bateria

Para o desenvolvimento de um carregador, é exigido entender como funciona uma bateria.

Entenda como funciona uma bateria:

Uma bateria é uma máquina química. Sobre ela reside uma substância química, ácida ou alcalina, que reage com um ou dois metais, criando uma corrente elétrica resultante.

Nem toda bateria é recarregável. As baterias comuns não permitem recarga, pois o componente que reage não consegue voltar à sua composição inicial quando é oferecida uma carga.

Porém, alguns compostos, quando oferecida uma carga elétrica, voltam à sua composição química inicial. A estes chamamos de baterias recarregáveis.

Toda bateria tem uma resistência interna, e esta resistência é sentida de duas formas na prática.

- A primeira é um aquecimento, resultante da recarga, que deve ser controlado para garantir a vida útil da bateria.
- A segunda, e, em consequência da primeira, é o consumo de parte desta energia para vencer a resistência (potência reativa).

Nem todas as baterias são iguais. As baterias apresentam três informações importantes, para avaliação de como recarregar. Cada informação pode contribuir para modificar a forma de sua carga. Assim, é importante não modificar ou realizar o procedimento de um tipo em outro tipo de bateria.

São estas informações:

- Tipo do metal – Referência aos metais constituídos NiMH (níquel hidreto), Li-Ion (lítio ion), NiCd (níquel cádmio)
- Valor de tensão – Refere-se ao valor da tensão da bateria (operação)
- Capacidade – Capacidade de armazenar energia, dada em (A h)

Tipo de Metal

Primeiramente, a bateria possui uma liga metálica. Esta liga apresenta características, que devem ser verificadas, como tempo de recarga, valor de sobretensão, entre outros.

A que utilizaremos é a bateria NK 18650, que possui Li-ion ou lítio como metal de desgaste.

Nesta bateria, para entender, consideramos a sobretensão em torno de 10% do valor da célula nominal.

O efeito memória nada mais é do que a propensão de alguns tipos de bateria a diminuir sua vida útil em decorrência de “vícios” no ciclo de carga e descarga. Uma vantagem das baterias de Li-ion é que elas não apresentam o efeito memória.

Valor de Tensão por Pilha

Dependendo do tipo de pilha ou bateria, o valor de cada pilha pode variar. Para o tipo que estamos utilizando a tensão é de 3.7 volts por unidade de pilha. Desta forma, se medirmos no multímetro uma pilha carregada, perceberemos que a mesma possui tensão varia em 3.7 volts.

À medida que a pilha descarrega, sua tensão cai, e quando chega a 70% (2,5 V) da tensão normal dizemos que ela está descarregada.

A definição da tensão de recarga para esta bateria é necessária, para que esta tensão consiga vencer a resistência natural da pilha, conseguindo carregá-la à tensão máxima permitida.

Para baterias deste tipo, a carga recomendada é trabalhar com tensões 10% superior à tensão normal, ou seja, 4,07 volts. Quando se trabalha com tensão até 10% do valor, damos o nome a esta recarga de carga lenta.

A vantagem de uma carga lenta é que podemos mantê-la indefinidamente na bateria, sem risco de queima da mesma. Pois quando esta entrar em equilíbrio com a tensão, a recarga resultante será zero.

Em valores superiores a 10%, são chamadas recargas rápidas, pois o incremento de tensão também agiliza a recarga, porém aumenta o stress da bateria, resultando em um maior aquecimento.

As recargas rápidas podem prejudicar a vida útil de uma bateria, criando, inclusive, problemas de explosão quando “esquecidas” no ciclo de recarga.

Capacidade

A capacidade é a quantidade de energia que será permitida passar. Em recarga lenta, recomenda-se que a quantidade de carga da bateria seja fornecida em um ciclo de oito horas, ou 12,5% da carga total. Em uma bateria que possui 6.8 Ah de carga máxima, a carga passante para recarga seria de 0,85A por hora.

Recomendo veementemente que não seja aplicada recarga rápida, e que seja utilizada apenas recarga lenta, pois garante economia na substituição das baterias e maior vida útil das mesmas.

Carregador de Bateria

Para recarga da bateria é necessário primeiramente realizar os cálculos do tópico anterior.

Após entender a tensão que deve ser aplicada e a corrente fornecida, é hora de tornar isso prático. Uma forma muito simples é através da aquisição de um step down.

O step down fornece uma regulagem de tensão, não obstante, também no modelo XL4005 a regulagem de corrente.

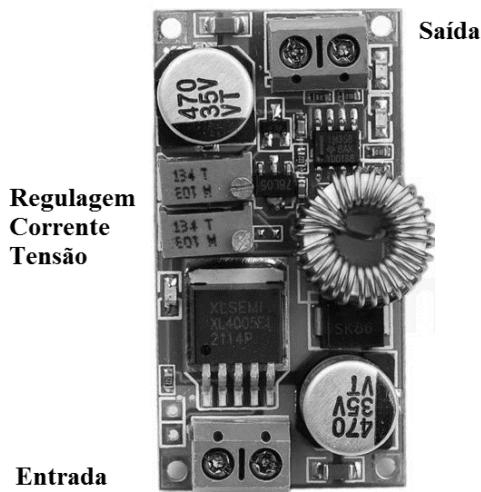


Figura 54 - Step down XL4005

Especificações Técnicas

- Tensão de entrada: 5 V até 32 volts
- Tensão de saída 0.8 até 30 volts
- Corrente de saída: 5A máximo
- Eficiência: acima de 95%
- Ripple de saída: <50mV
- Frequência de chaveamento: 300kHz
- Regulagem de corrente: $\pm 0.5\%$
- Regulagem de voltagem: $\pm 2.5\%$
- Temperatura de operação: -40°C a +85°C
- Dimensões: 51 mm x 26 mm x 14 mm

Esquema Elétrico de Recarga

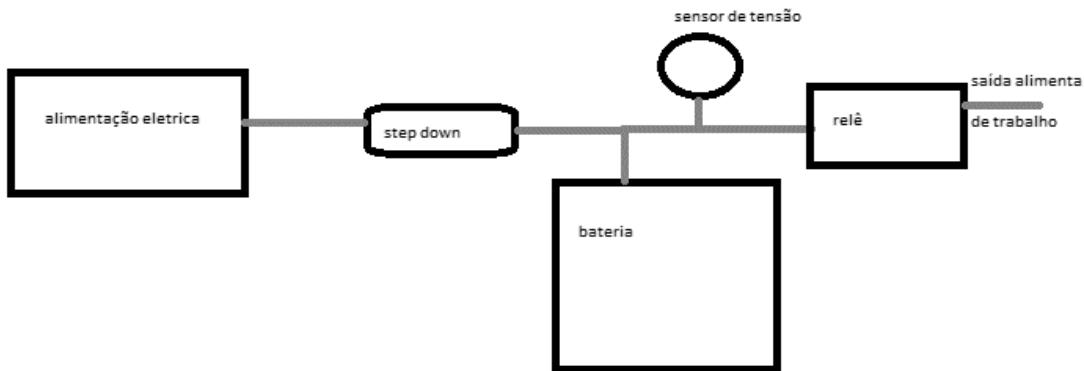


Figura 55 - Diagrama de conexão do carregador de bateria

A tensão e corrente do step down estão avaliados para operação de carga em bateria em repouso, ou seja, não alimentando o sistema do robô. Por isso a importância do relê aberto.

Outro ponto importante, é que o relê deve ser montado na posição NF para saída de alimentação de trabalho. Desta forma somente será aberta caso realmente seja indicado pelo Arduino. Este detalhe garante que o sistema por padrão será alimentado pela bateria, sendo o contrário apenas quando indicado pelo firmware.

O sensor de tensão acompanha a carga da bateria até que a tensão de operação seja obtida.

O Sensor de tensão permite fazer um acompanhamento através da interface do robô da carga da bateria.

Bateria Recarregável

A bateria recarregável de 3.7 V permite fornecer carga de 5,2 Ah por unidade.



Figura 56 - Pilha de Li-ion utilizada em nosso projeto

Especificações técnicas

Capacidade: 5,2 Ah

Sem efeito de memória

Tensão: 3,7 V

Dimensões: 18 mm x 68 mm

Suporte de Porta Pilhas

O suporte de pilhas de 3.7 volts permite criar um circuito em série das pilhas, dando origem a uma tensão de 11.1 V e carga total de 15,6 Ah.



Figura 57 - Suporte Porta Pilhas

Sensor de Tensão

O sensor de tensão, como o próprio nome diz, lê a tensão aplicada no equipamento

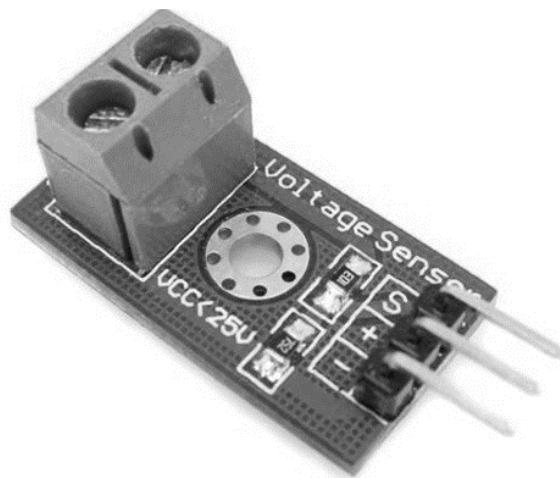


Figura 58 - Sensor de tensão

Especificações técnicas

- Faixa de Leitura: DC0-25 V
 - Resolução de leitura: 0.00489 V

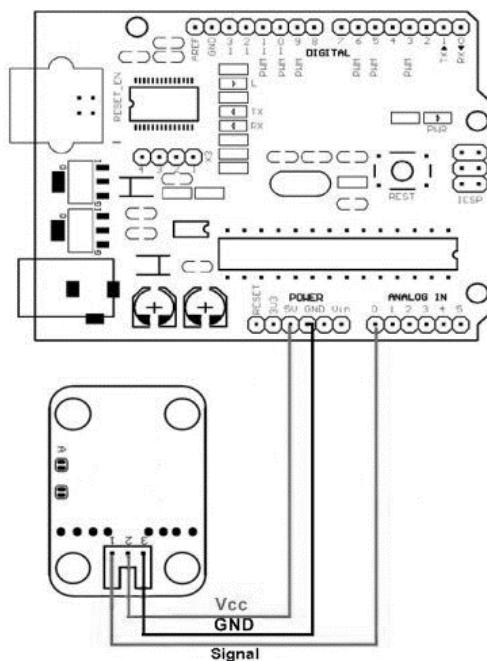


Figura 59 - Esquemático de ligação com Arduino

Programa exemplo no Arduino para controle do sensor de tensão:

```
#include <Wire.h>

int val11;
int val2;

void setup()
{
    pinMode(LED1,OUTPUT);
    Serial.begin(9600);
    Serial.println("Voltage: ");
    Serial.print("V");
}

void loop()
{
    float temp;
    val11=analogRead(1);
    temp=val11/4.092;
    val11=(int)temp;
    val2=((val11%100)/10);
    Serial.println(val2);

    delay(1000);
}
```

Como é usado o sensor de tensão

O sensor de tensão é usado para leitura das tensões da bateria.

O robô, quando alimentado por bateria, tem alimentação de trabalho de 11,5 V. Quando a tensão cai para 8,05 volts (70% da carga) ou menos, a bateria precisa ser recarregada.

O sensor de voltagem tem a função de identificar quando a carga da bateria está no fim. Este controle é feito pelo firmware do Arduino, que lê a bateria e, ao perceber a queda de tensão, notifica visualmente, através do LCD, a necessidade de uso de fonte externa.

Ele também lê a corrente passante na fonte externa. E, quando já possui alimentação, imediatamente abre o relé, permitindo a recarga automática do robô. Abordaremos estes detalhes construtivos do software no tópico específico.

Relê

O relê permite o chaveamento da bateria, caso a mesma esteja com pouca carga. Desta forma, se a mesma estiver com pouca carga e a alimentação externa estiver ligada, o relê desliga a alimentação da bateria, permitindo que o robô seja carregado.

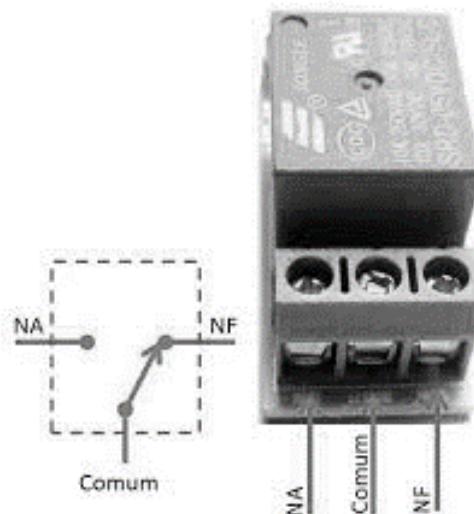


Figura 60 - Conexões elétricas do relê

O Relê possui 3 (três) conexões:

- Comum
- NA – Normalmente Aberta
- NF – Normalmente Fechada

A comum é o ponto que irá entrar o sinal elétrico ou tensão.

A NA, é o conector que normalmente estará ligado, mesmo quando o relê estiver desligado.

A NF, é o conector que só terá sinal, quando houver o sinal para acionamento.

É prática comum ao se desenvolver em Arduino, esquecer no momento da montagem, qual é o NF e o NA, e por ventura inverter esses.

O que normalmente ocorre, é que o sistema passa a ter comportamento invertido, ao mandar o sinal para o relê o circuito abre, e ao não enviar, passa a ter corrente na saída.

Se um dia, perceber esse tipo de situação, não procure erro no software, é simplesmente uma inversão no relê. Fica a dica.

A ligação elétrica no Arduino é muito simples. São três pinos:

- GND – negativo
- 5 V
- Sinal

Quando o processador envia um sinal para o pino, o relé aciona o pino NA (normalmente Aberto, assando a ter corrente).

Programa exemplo no Arduino para acionamento de relê:

```
// A função setup inicia a porta 13 como saída
void setup()
{
    // initialize digital pin 13 as an output.
    pinMode(13, OUTPUT);
}

// A função loop aciona e desliga o relê a cada segundo, repetindo sempre que a função termina
void loop()
{
    digitalWrite(13, HIGH); //Aciona o relê
    delay(1000);          // espera um segundo
    digitalWrite(13, LOW); // desliga o relê
    delay(1000);          // espera um segundo
}
```

Círculo do Carregador

Nesta etapa desenharemos o circuito responsável por unir todos os componentes do nosso carregador de baterias.

Utilizando o Eagle vamos desenvolver a placa de circuito impresso.

Para tanto, precisamos entender os blocos de ligação do circuito do carregador.

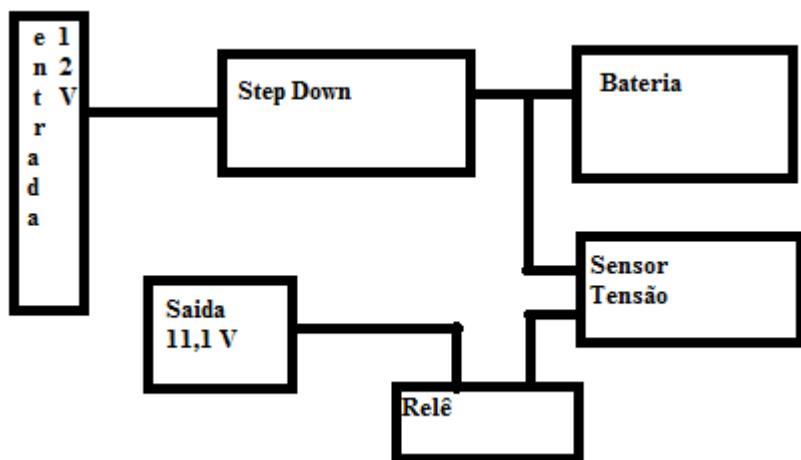


Figura 61 - Diagrama esquemático de recarga da bateria

Entrada de 12 V

Fornecido pela energia da rede externa, permitindo que a bateria seja recarregada

Step Down

Corrig e ajusta as tensões e cargas para permitir a recarga planejada da bateria

Bateria

Elemento que sofrerá a recarga

Sensor de tensão

Mede a tensão da bateria durante a carga e durante o uso

Relê

Responsável pelo controle do acionamento quando a bateria estiver sendo carregada

Saída

Fornece a tensão utilizada pelo circuito, permitindo que o robô funcione mesmo quando não estiver sendo alimentado externamente

Placa do circuito do carregador

O projeto desta placa não deve ser encarado com o melhor projeto. Existem outros formatos, por exemplo.

Criada para permitir a conexão dos componentes integrados visando sua respectiva ligação com os componentes.

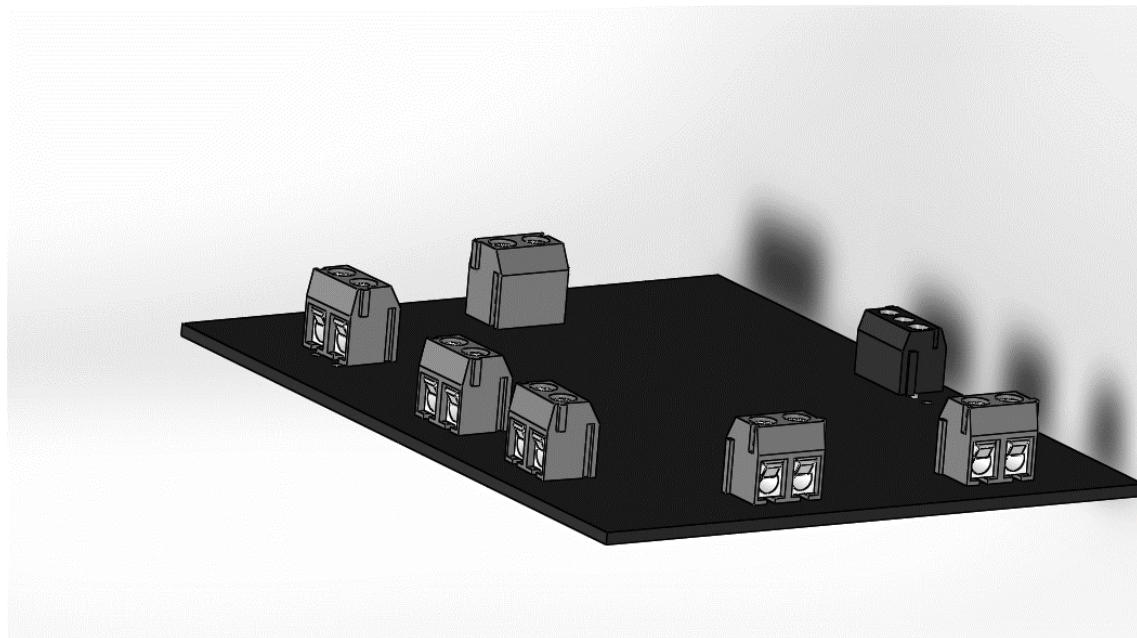


Figura 62- Visão da placa pronta

Não demonstraremos a construção desta placa, pois foge do foco deste livro. Entretanto, forneceremos o espelho da placa para que possa ser impresso em folha transparente e criado através de processo dry film.

Os espelhos dos circuitos apresentados encontram-se na pasta dryfilm do projeto.

Espelho: /robotinics/dryfilm/Espelho Placa do Circuito do Carregador.bmp

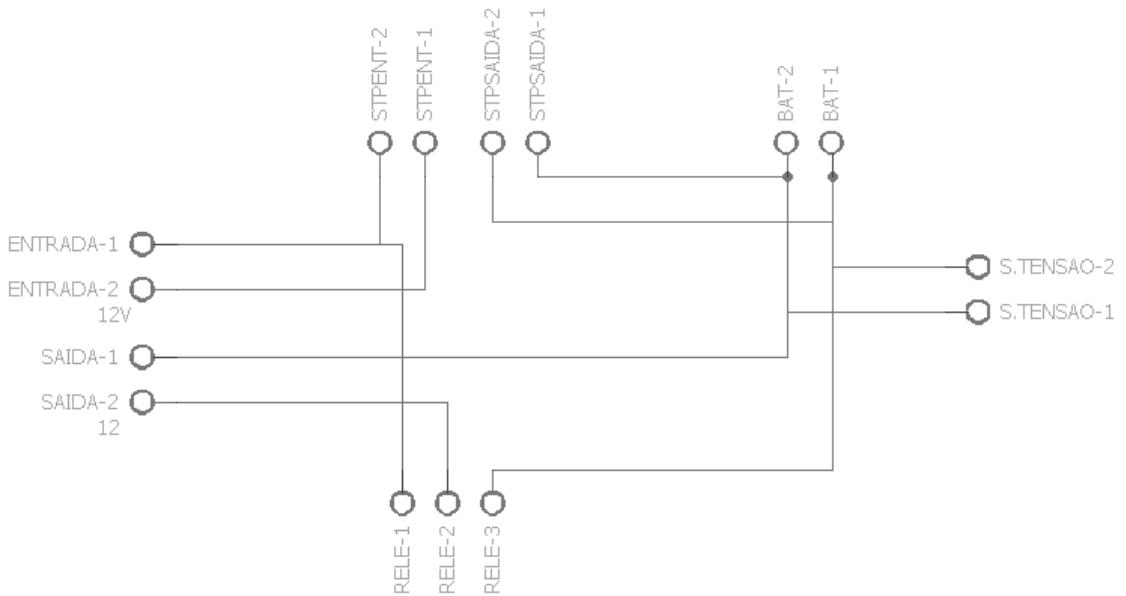


Figura 63 - Visão do Schematic

- O BAT – Permite ligar os negativos da bateria e da rede
- STEPENT – Entrada de tensão do step down
- STPSAIDA – Saída de tensão do step down, já com a tensão regulada
- Entrada – Entrada de tensão da rede elétrica
- Saída – Saída de tensão já com a distribuição
- Relê – Conexão com o relê
- S. Tensão – Sensor de tensão

A informação mais importante é que o circuito normalmente fechado fica acoplado com a bateria. Desta forma, caso seja ligado o robô, a primeira conexão que é feita é alimentada pela bateria.

Calculando as especificações do step down

Conforme já enunciado no tópico anterior, onde expliquei como funciona e os tipos de carga de uma bateria, agora iremos aplicar os princípios aprendidos, fazendo o cálculo do step down para gerar o carregamento da bateria.

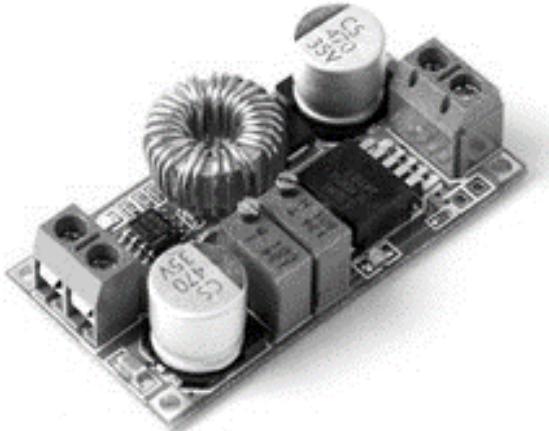


Figura 64 - Step down com regulagem de tensão e corrente

O step down permite regular a tensão e, em alguns modelos, a corrente máxima em um circuito.

Especificações técnicas do equipamento:

- Tamanho: 51 x 26.3 x 14 (mm)
- Temperatura de operação: -40 °C a +85 °C
- Regulação de tensão: $\pm 2.5\%$
- Regulação de carga: $\pm 0.5\%$
- Ripple de saída: 50 mV (max) 20 M de largura de banda
- Frequência de comutação: 300 kHz
- Eficiência de conversão: 95% (a melhor)
- Corrente de saída: máximo ajustável 5A
- Tensão de saída: 0.8 V-30 V
- Tensão de entrada: 5 V-32 V
- Retificação: retificação não sincronizada
- Propriedades do módulo: módulo de corrente e tensão constante e não isolada

Para cálculo da carga elétrica fornecida pelo regulador utilizamos a seguinte fórmula:

$$I = It / 10$$

Onde:

I – É a carga elétrica que se deseja limitar

It – Corrente total

10 – Quantidade de horas para carga máxima

Dessa forma, para calcular a carga elétrica máxima fornecida pelo regulador, usamos a fórmula:

$$It = 5,2 \text{ Ah} \times 3 \text{ baterias} = 15,6 \text{ Ah}$$

Conforme especificação da bateria passada anteriormente.

$$\text{Sendo assim, } I = 15,6 / 10 = 1,56 \text{ Ah.}$$

Assim, a carga elétrica que deve ser regulada no step down será de **1,56 Ah**.

Para cálculo de tensão, utilizamos a seguinte especificação:

A tensão normal da bateria é dada 11,1 volts, pois cada bateria possui 3,7 volts, e como estão em série e as tensões se somam, fica uma tensão total de **11,1 volts**.

Se considerarmos 1,5 volt de perda por resistência interna da bateria, podemos considerar que a tensão que deve ser aplicada na bateria é de **12,6 volts**.

Conforme apresentado no trabalho de Willian Tateyama, a carga lenta mantém sua tensão constante, enquanto a corrente decai exponencialmente.

Desta forma, a carga lenta, além de ser “lenta”, pode ser mantida, pois quando a bateria estiver com “toda a carga” a corrente será nula.

Também conforme apresentado no trabalho do (Tateyama), a carga rápida permite a passagem de alta corrente, que produz um aquecimento maior, entre 0,2C e 0,5C, o que carrega cerca de 80% da capacidade da bateria em pouco tempo e de forma linear.

Para proteção da bateria utilizamos carga lenta, pois este tipo de carga não danifica a bateria.

Como calibrar o step down

Agora que sabemos as informações necessárias para aplicar ao nosso step down, precisamos entender como calibrar o step down.

1. Primeiro, ligue uma tensão de 12 volts na entrada do step down.
2. Chaveie o multímetro na opção de volts (medindo tensão DC superior a 12 V)
3. Conecte as pontas de prova na saída do step down
4. Com uma chave de fenda pequena, rode os trimpots de regulagem de tensão até chegar no valor desejado
5. O trimpot que, ao ser movido, variar a tensão, será o de tensão, e o outro, por exclusão, será o de corrente.
6. Agora retire a ponta de prova do multímetro da saída e mude o mesmo para corrente (A). Coloque valores de 20A ou mais.
7. Reconecte as pontas de prova na saída do step down
8. Gire o trimpot da corrente até chegar no valor que calculou

Pronto – você configurou seu step down.

2.3 Software – Projeto do Software

Nesta etapa inicial, pouco se pode fazer para visualizar o robô propriamente dito, porém adiantaremos alguns pontos da visão geral do software e como este será executado.

Esta visão geral irá nortear o projeto, que será seguida nos demais tópicos.

A principal visão entre os equipamentos é a interação entre o Raspberry e o Arduino, que pode ser vista na imagem a seguir:

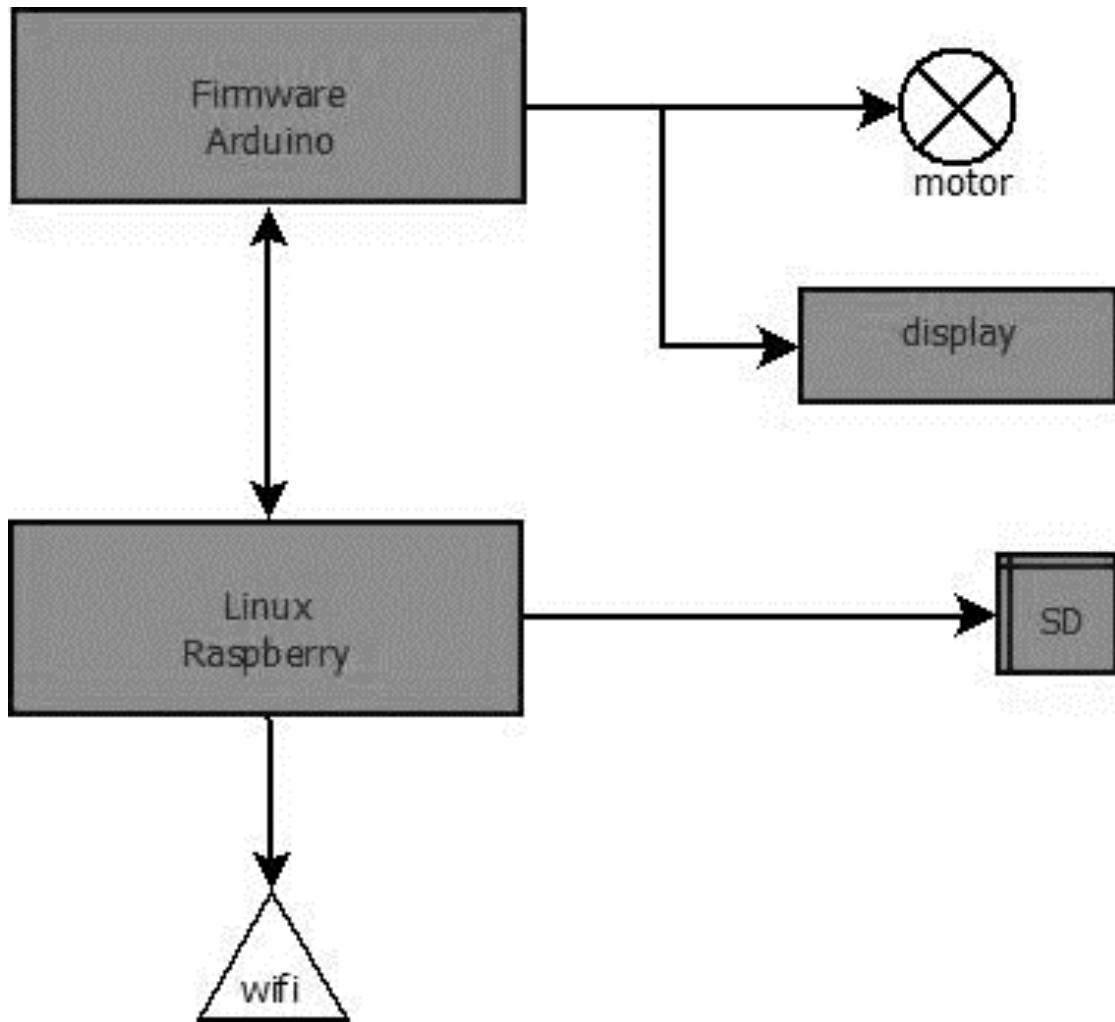


Figura 65 - Visão geral dos blocos principais para programação do robô

Na figura acima, os equipamentos Arduino e Raspberry desenvolvem papel de ligação entre os equipamentos e o mundo externo. O Arduino controla diversos dispositivos eletromecânicos, enquanto o Raspberry desenvolve papéis de armazenamento de dados no SD e conexão com mundo externo através de wifi.

Tais atividades não requerem o uso de aplicações robustas, onde a comunicação entre estes dois universos (microcontrolador e microprocessador) é exigida.

Para prover tal comunicação, garantindo também a análise e entendimentos de cada uma das tarefas fim, já definidas no início deste tópico, faz-se necessário um planejamento elaborado dos programas.

Desta forma, dividiremos o projeto do software em três grandes grupos.

1. Firmware – É o software embarcado ao hardware Arduino
2. Programas embarcados – São o conjunto de aplicativos que incluídos no Raspberry PI – Linux
3. Programas gerenciais – São o conjunto de aplicativos instalados ou disponíveis fora do robô, que visa controlar ou operá-lo remotamente

Firmware

Em eletrônica e computação, firmware é o conjunto de instruções operacionais programadas diretamente no hardware de um equipamento eletrônico (Wikipédia, n.d.).

O ciclo de funcionamento de um programa escrito em Arduino obedece a seguinte ordem:

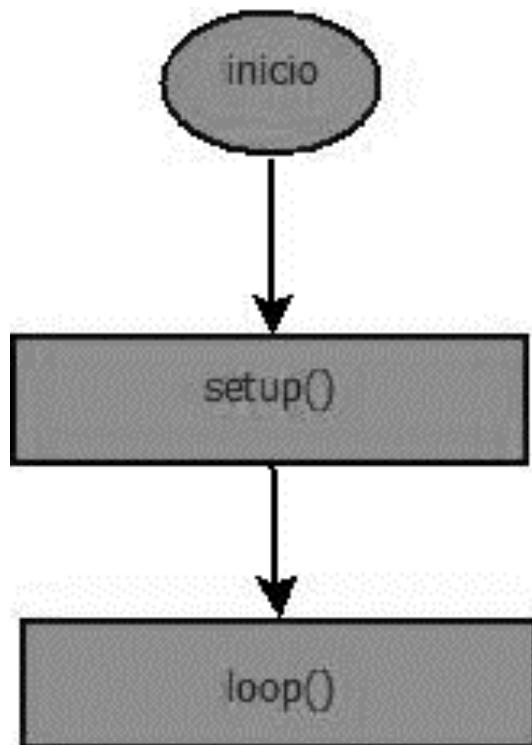


Figura 66 - Fluxo de start do Arduino

O firmware é constituído de dois grandes conjuntos de funções:

- SETUP
- LOOP

O Setup é responsável pelo start do equipamento e é chamado toda vez que o equipamento é ligado.

O Loop é a função responsável pela chamada das funções durante o ciclo de utilização do equipamento, e como o próprio nome diz, ao ser encerrada volta a rodar em um ciclo eterno.

Bloco de Execução do Arduino

O fluxo de execução da rotina loop é responsável pela execução das rotinas de repetição que irão definir o ciclo de execução cíclica.

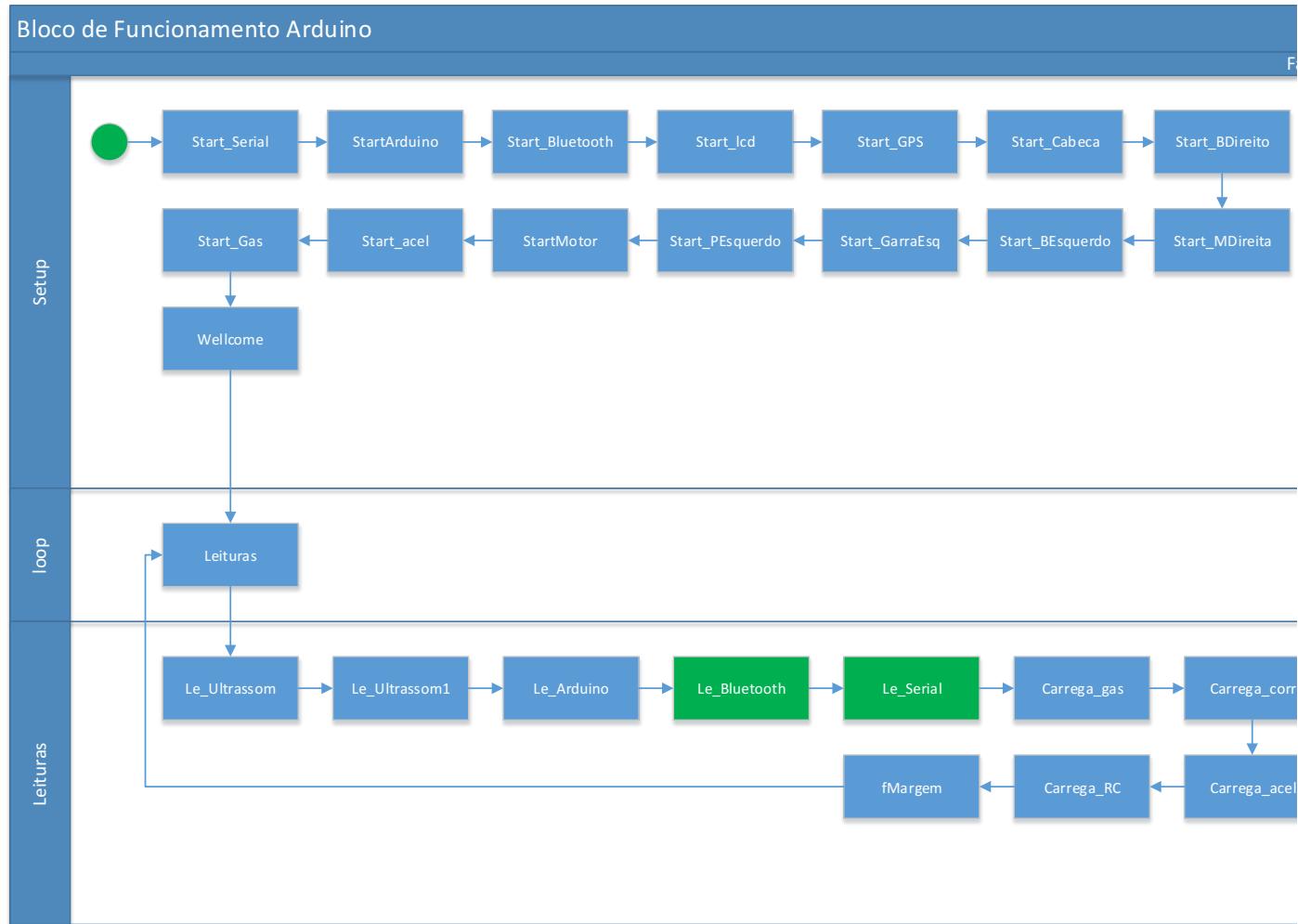


Figura 67 - Fluxograma de execução detalhado

Bloco de funções de Setup do Arduino

Funções chamadas pelo Arduino para inicialização do robô:

StartSerial();	Inicializa a serial do Arduino
StartArduino();	Inicializa porta serial de comunicação ttl com dispositivo externo
StartBluetooth();	Inicializa módulo bluetooth
Start_lcd();	Inicia dispositivo LCD
Start_GPS();	Inicia dispositivo GPS
Start_Cabeca();	Inicia dispositivo servomotor Cabeça
Start_BDireito();	Inicia dispositivo servomotor Braço Direito
Start_MDireita();	Inicia dispositivo servomotor Mão Direita
Start_BEsquerdo();	Inicia dispositivo servomotor Braço Esquerdo
Start_GARRAESQ();	Inicia dispositivo servomotor Garra
Start_MEsquerdo();	Inicia dispositivo servomotor Mão Esquerda
StartMotor();	Inicia dispositivo controlador ponte H motor
Start_acel();	Inicia dispositivo Acelerômetro
StartGas();	Inicia dispositivo sensor de Gás
StartWelcomme();	Mostra mensagem de boas vindas

Bloco de funções de Loop do Arduino

Funções chamadas pelo Arduino durante o ciclo infinito de execução.

le_ultrassom	controla as informações lidas do ultrassom da base frente
le_ultrassom1	controla as informações lidas do ultrassom da base traseira
le_bluetooth	le informação do modulo bluetooth
le_serial	le informação da serial
carrega_gas	le informação do módulo de sensor de gás
carrega_acel	le informação do acelerômetro
carrega_rc	lê informações do modulo de rádio frequência
fMargem	faz a verificação se os limites de tolerância de distância foram ultrapassados, caso as rodas estejam em movimento, cancela a movimentação da mesma.

Desta forma o loop garante que, ao fim de cada ciclo, o processo de obtenção das informações de entrada e saída serão atendidas.

Linux

O Raspberry, ao contrário do Arduino, possui um sistema operacional para gerenciar seu hardware.

Desta forma, a aplicação fica limitada não a controlar o hardware, mas a estabelecer os protocolos de comunicação entre os equipamentos, bem como gerenciar os recursos que serão compartilhados.

Nota do Autor

Sempre que mencionamos o Raspberry como solução, não significa que esta é a única! Existem outros computadores de pequeno porte, como o Cubieboard, que atendem perfeitamente ao nosso interesse. Desta forma, sempre que o Raspberry for apresentado como solução, também peço que entendam que tudo o que foi dito também se aplica aos demais concorrentes do Raspberry, tal como o Cubieboard e como tantos outros. Em algum momento apresentarei screens do Cubieboard, pois, de fato, utilizei este computador para validar alguns projetos aqui apresentados.

Veremos aqui detalhes de cada uma das aplicações e como elas se integram com o Arduino.

De forma geral, estas aplicações são iniciadas automaticamente quando o sistema operacional é carregado.

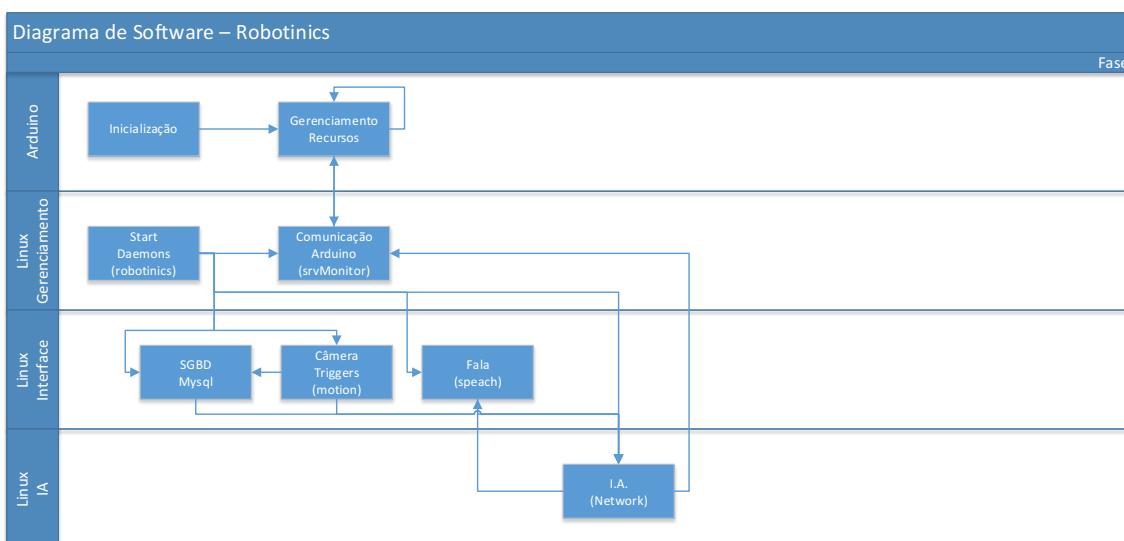


Figura 68 - Fluxograma de execução no Linux

- Robotinics – Shell script que inicializa demais módulos

- `srvMonitor` – Sistema de controle de monitoramento de comunicação com bluetooth Arduino
- `mysqld` – Sistema de banco de dados mysql 5.1
- `motion` – Sistema de controle e gestão de webcam e cftv utilizando lib v4linux
- `speech` – Serviço de fala e sintetização de voz
- `networks` – Módulo tomador de decisão

Outros módulos independentes de inicialização

- `sshd` – Serviço de comunicação por terminal pela porta 22
- `vncserver` – Serviço de terminal gráfico remoto (não inicializado pelo Robotinics)
- `phpd` – Interpretador PHP
- `apache` – Servidor web

Inicialização do Robotinics

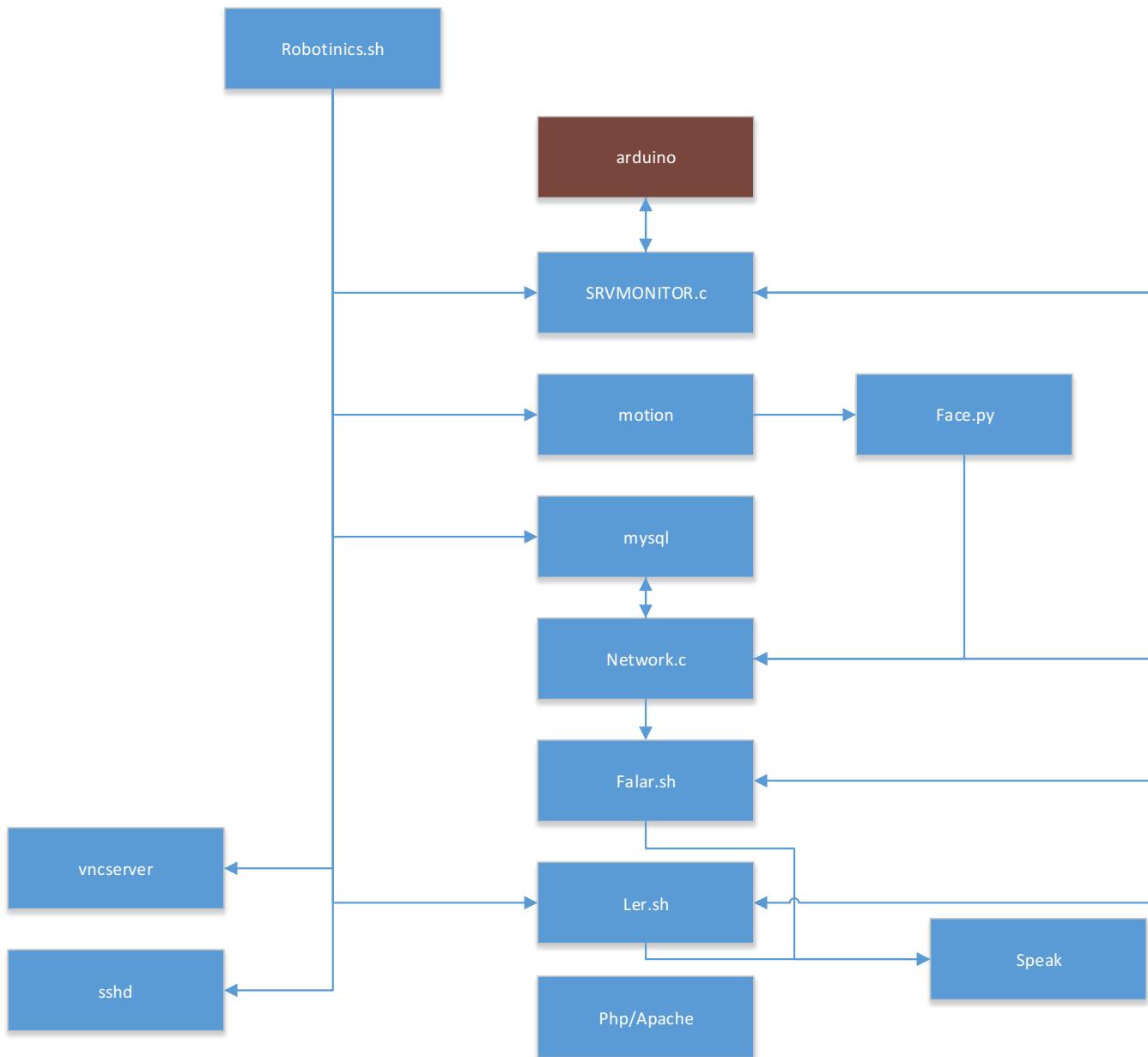


Figura 69 - Hierarquia de execução das aplicações

Robotinics.sh

O robotinics.sh é uma aplicação escrita em shell script que executa os aplicativos envolvidos no projeto, dando start (inicialização) nos serviços que são requeridos para seu funcionamento.

O Robotinics apenas executa todos os aplicativos necessários para garantir que o robô opere.

VNC SERVER

O vncserver é um serviço do Linux para acesso à interface gráfica. Através do vncserver é possível operar remotamente o Xwindows.

Sshd

O sshd é um serviço do Linux para acesso à interface texto (console). Através desta interface é possível conectar remotamente no shell executando comandos.

PHP/APACHE

O php é uma linguagem interpretada que permite através da implementação de scripts montar uma interface em web.

Motion

O motion é um aplicativo muito robusto para arquivamento de imagens e detecção de movimento.

Com o motion é possível a identificação de deslocamento de imagens, apontando, inclusive, onde ocorreu o deslocamento. O motion é amplamente utilizado em sistemas de CFTV e permite, entre outras coisas:

- Monitoramento remoto das câmeras através de interface WEB
- Vinculação de trigger (gatilhos) programados para rodar quando identificar mudanças
- Armazenamento direto em MySQL para vinculação das imagens
- Mudança de percentual de mudança para disparar triggers

MySQL

O MySQL é um banco de dados, e através dele é possível armazenar informações em grande volume. O uso deste SGDB permite criação e armazenamento de grande volume de informações através de um complexo sistema de banco de dados. O MySQL é conectado através da api mysql_client.h, que é utilizado em diversos aplicativos escritos em C.

Para termos acesso ao MySQL precisamos ter o servidor instalado em nosso robô. Para tanto, é necessário ter realizado os procedimentos descritos no Capítulo 1 de nosso livro.

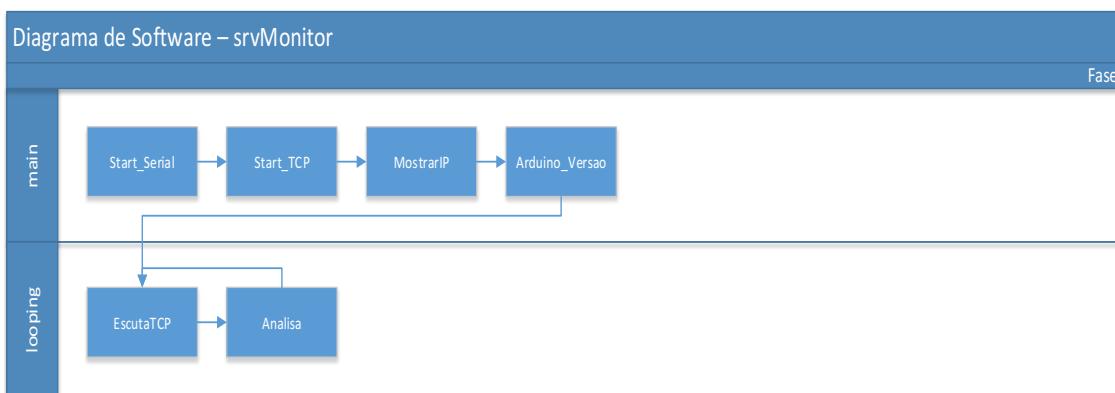
Realizando tais procedimentos podemos entrar na console através de um terminal e criar nosso banco de dados, conforme passado a seguir.

- 1) Através do seu computador, use o Filezilla, Conectando com o usuário root, no seu robô (IP do robô).
- 2) Copie a pasta mysql em /projetos/robotinics/
- 3) Entre através do Mremote, usando o usuário root, também no seu robô.
- 4) Digite cd /projetos/robotinics/mysql
- 5) Edite o arquivo Makefile, usando vim Makefile
- 6) Mudando o USER e Password do seu banco, este você deve ter anotado no momento que instalou os pacotes.
- 7) Execute o comando make create
- 8) Execute o comando make all
- 9) Para sair, digite: quit;

SrvMonitor.c

SrvMonitor é um aplicativo escrito em c para Linux que tem como responsabilidade ler as informações da serial do Linux (Arduino) e realizar a análise da mesma.

Além da função de ler a serial, o srvMonitor também gera funcionalidades para o controle das interfaces entre o Arduino e o Raspberry.



São funcionalidades do srvMonitor

- Leitura da temperatura
- Gravação de mensagens no LCD I2C
- Leitura de corrente
- Leitura de tensão
- Movimentação dos Servo motores
- Movimentação das rodas e controle da ponte H
- Acendimento dos LEDs

Todas as funcionalidades atendidas são passadas através de socket server aos demais aplicativos.

Desta forma, os recursos do Arduino podem ser geridos através de simples conexão socket entre aplicações.

Principais rotinas do srvMonitor:

- Start_Serial – Inicializa o handler da serial do Arduino
- Start_TCP – Inicializa o handler do serviço TCP, porta 8095
- MostrarIP – Mostra IP no LCD 16x2
- Mostra versão do Arduino no LCD 16x2

- Looping – Rotina de execução da lógica do aplicativo, semelhante à rotina do Arduino loop
- EscutaTCP – Rotina que escuta o que é enviado da porta TCP
- Analisa – Rotina que pega o que é enviado, valida e o que for válido e envia para o Arduino

Notas do autor:

Alguns destes aplicativos estão disponíveis apenas no repositório do Sourceforge.

Sintetização de Voz

A sintetização de voz no Linux é feita pelo software eSpeak, que é open source e permite a sintetização de voz para várias plataformas, entre elas Windows e Linux.

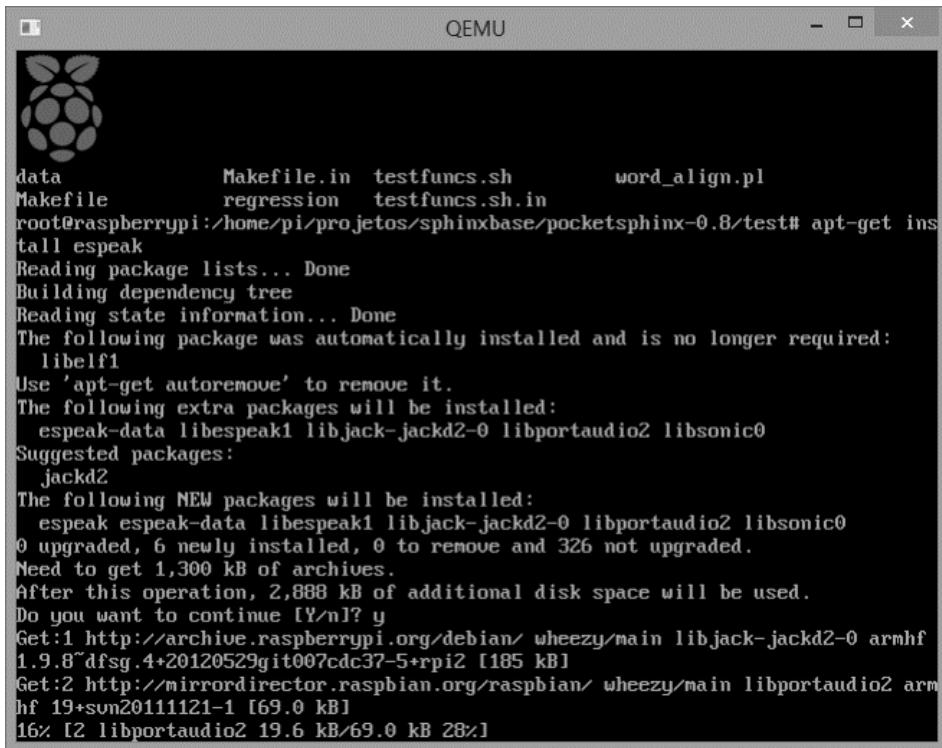
O eSpeak possui um conjunto forte de parâmetros que permite sua programação, bem como um conjunto de bibliotecas que permitem sua integração junto com aplicativos.

Repositório do projeto: <http://espeak.sourceforge.net>

Instalação do eSpeak

Para instalar o eSpeak, iremos utilizar o apt-get, digitando, para tanto:

```
sudo apt-get install espeak  
sudo apt-get install libespeak1  
sudo apt-get install libespeak-dev  
sudo apt-get install espeak-gui  
sudo apt-get install espeak-dbg  
sudo apt-get install espeak-data  
sudo apt-get install brltty-espeak
```



The screenshot shows a terminal window titled "QEMU" running on a Raspberry Pi. The terminal displays the following output from the apt-get command:

```
data      Makefile.in  testfuncs.sh      word_align.pl  
Makefile    regression  testfuncs.sh.in  
root@raspberrypi:/home/pi/projetos/sphinxbase/pocketsphinx-0.8/test# apt-get ins  
tall espeak  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following package was automatically installed and is no longer required:  
  libelf1  
Use 'apt-get autoremove' to remove it.  
The following extra packages will be installed:  
  espeak-data libespeak1 libjack-jackd2-0 libportaudio2 libsonic0  
Suggested packages:  
  jackd2  
The following NEW packages will be installed:  
  espeak espeak-data libespeak1 libjack-jackd2-0 libportaudio2 libsonic0  
0 upgraded, 6 newly installed, 0 to remove and 326 not upgraded.  
Need to get 1,300 kB of archives.  
After this operation, 2,888 kB of additional disk space will be used.  
Do you want to continue [Y/n]? y  
Get:1 http://archive.raspberrypi.org/debian/ wheezy/main libjack-jackd2-0 armhf  
1.9.8~dfsg.4+20120529git007cdc37-5+rpiz2 [185 kB]  
Get:2 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libportaudio2 arm  
hf 19+svn20111121-1 [69.0 kB]  
16% [2 libportaudio2 19.6 kB/69.0 kB 28%]
```

Figura 70- Instalação do software de sintetização de voz

Para a instalação do software, seu Raspberry precisará estar conectado à internet.

Comando eSpeak

Iremos abordar as principais opções do eSpeak.

Ao instalar o eSpeak, vamos encontrar um único executável.

Para ver os comandos disponíveis, usamos o comando:

```
espeak --help
```

Como na vida real, um italiano, ao falar uma frase em português, possuirá um “sotaque”. Para que nosso sintetizador de voz tenha o “sotaque” brasileiro, devemos definir que o padrão de voz seja o nosso. Para isso, utilizamos este comando:

```
espeak --voice=pt
```

ou

```
espeak -v pt
```

A tabela total de sotaques pode ser obtida no site <http://espeak.sourceforge.net/languages.html>

Pitch é a cadência ou tempo na fala. Quando lembramos de cadência, temos que lembrar dos narradores de jogos de futebol. O narrador tem um tempo baixo, enquanto nós temos um tempo mais alto. O valor padrão é 50.

Para controle da cadência usamos o parâmetro -p

```
espeak -p 165
```

Speed é a velocidade de palavras por minuto. O padrão é 175.

Para sua sintaxe, usamos o parâmetro -s:

```
espeak -s 120
```

Sintetização de frases diretas

Para sintetização de frases diretas, usamos a frase entre aspas.

Vemos um exemplo de frase: Olá mundo

```
espeak -v pt -p 165 -s 120 "Olá mundo"
```

Origem: <http://espeak.sourceforge.net/>

Leitura de Arquivos

Para leitura de conteúdo de arquivos, o eSpeak utiliza o comando file. Para isto, basta utilizar o parâmetro -f seguido do nome do arquivo, conforme sua sintaxe:

```
espeak -v pt -p 165 -s 120 -f README.txt
```

Configurando saída de som para o eSpeak

O Raspberry e o Cubieboard possuem saídas de som diferentes, como a saída de áudio HDMI e headfone.

Para configurar uma entrada de áudio realize o seguinte procedimento, como demonstrado no site (Cubieboard).

Instale pela console alsa-utils:

```
apt-get install alsa-utils
```

Execute pela interface gráfica o alsamixer.

Edite o arquivo /etc/asound.conf

```
pcm.!default {  
    type hw  
    card 0 //atribua 1 ou 0, conforme apontado no alsomixer  
    device 0  
}  
ctl.!default {  
    type hw  
    card 0 //Indique 1 ou 0 conforme o alsomixer  
}
```

Falar.sh

Falar.sh é um bash em shell script que utiliza a aplicação para sintetização de voz.

A sua sintaxe é:

```
falar [frase que se deseja falar]
```

Entraremos nos detalhes do software de sintetização de voz no Capítulo 4, porém, por enquanto, basta saber que este comando é responsável por este trabalho.

```
#!/bin/bash  
/sbin/espeak -v pt -p 165 -s 120 "$1"
```

Ler.sh

Ler.sh é um bash em shell script que utiliza a aplicação eSpeak,

A sua sintaxe é ler [arquivo que se deseja ler]

A diferença entre o ler e o falar é basicamente que falar pronuncia o conteúdo do parâmetro que se fornece entre aspas, e ler pega um arquivo texto indicado e lê o seu conteúdo.

Sintaxe

```
#!/bin/bash
/sbin/espeak -v pt -p 165 -s 120 -f $*
```

Network.c

Programa previsto no Robotinics, porém não implementado. Pretensamente projetado para meu mestrado. Responsável pela análise de padrões e tomada de decisões.

O network.c deve utilizar técnicas de I.A. para análise de entradas de dados, tais como imagens, sons e sensores para entender o mundo.

Os algoritmos de I.A. são complexos e permitem ampla gama de aplicações.

Iremos tratar sobre algoritmos de IA ao fim do projeto.

Integração do eSpeak com C

Segue um exemplo de uso de integração do eSpeak com C para Raspberry. O srvFala, tem a responsabilidade de ser um daemon, que fica monitorando um banco de dados. Nele é possível verificar as entradas de comandos de voz. Realizando sua respectiva sintetização.

O srvFala pode falar tanto por chamada por shell, como utilizando integração direta da API do eSpeak. O fonte fala1.c utiliza a chamada por shell, enquanto o fala2.c utiliza o acesso a API do Linux.

O Include abaixo, referência a biblioteca do speak_lib, que será utilizada.

```
#include <espeak/speak_lib.h>
```

No fragmento abaixo, temos a rotina responsável pela inicialização da rotina de fala.

```
int Setup_Speak()
{
    //must be called before any other functions
    //espeak initialize
    //output = AUDIO_OUTPUT_SYNCH_PLAYBACK
    output = AUDIO_OUTPUT_PLAYBACK;
    int Buflength = 500;
```

```

int Options=0;
char *path=NULL;

if (espeak_Initialize(output, Buflength, path, Options ) <0)
{
    puts("could not initialize espeak\n");
    return -1;
}

else
{
    puts("\nOk!\n");
}

espeak_SetVoiceByName("pt");

//espeak_SetParameter(espeakRATE, value, 0);

//Captura valores padroes
espeak_VOICE *voice_spec=espeak_GetCurrentVoice();

voice_spec->gender=2; // 0=none 1=male, 2=female,
voice_spec->age = 0; // 0=not specified, or age in years
espeak_SetVoiceByProperties(voice_spec);

espeak_SetSynthCallback(SynthCallback);
}

```

```

void Ler( char * Info)
{

```

```
char buff[1000];
sprintf(buff, "%s",Info);
if((speakErr=espeak_Synth(buff,    strlen(buff),    0,0,0,espeakCHARS_AUTO,NULL,NULL))!=
EE_OK) {
    puts("error on synth creation\n");
}
espeak_Synchronize();
}
```

No código a cima temos, 2 partes importantes:

`espeak_Synth` – Responsável pela sintetização do texto propriamente dito.

`espeak_Synchronize` – Responsável por aguardar o término da sintetização, isto é necessário, pois o processo de sintetização ocorre de forma assíncrona com o programa, ou seja, você executa a ação, e mesmo não tendo terminado o processo de fala, o programa continua sua execução. O synchronize, garante a sincronização das ações.

Arquivo: Makefile

```
#HEADERS = program.h headers.h

CC= gcc
MYSQLCFLAGS= -I/usr/include/mysql -DBIG_JOINS=1 -fno-strict-aliasing -g
MYSQLLIBS= -L/usr/lib/mysql -lmysqlclient -lpthread -lz -lm -lrt -ldl
YESPEAK= -lespeak

default: all

all: clean compile install run
compile:
    $(CC) fala1.c -g -o srvFala $(MYSQLCFLAGS) $(MYSQLLIBS) $(YESPEAK)

install:
    #Retira registro do serviço
    update-rc.d srvFala remove
    #copia para pasta de binários
    cp srvFala /usr/local/bin/
    #copia script de serviço
    cp ./srvFala_service /etc/init.d/srvFala
    #atribui direitos de serviço
    chmod +x /etc/init.d/srvFala
    #registra o serviço
    update-rc.d srvFala defaults

clean:
    rm -f srvFala
    rm -f /usr/local/bin/srvFala

run:
    srvFala
```

O que há de especial sobre o código do Makefile é a linha a seguir:

```
YESPEAK= -lespeak
```

Nela, incluímos a biblioteca para ser linkada ao código, sem esse código não conseguimos compilar o programa.

No diretório srvFala você encontrará o código completo tanto para utilizar integrando por script shell, como uso direto na api.

3 Base do Robô

Nesta etapa iremos montar as peças base do robô, que será 100% funcional. Porém, o robô será apenas montado da “cintura” para baixo, não tendo todas as especificações técnicas. Entretanto, será possível realizar diversas operações nele.

A base do robô pode ser utilizada como modelo para diversos outros projetos.

3.1 Mecânica - Finalização da Etapa Mecânica da Base do Robô

Nesta etapa as especificações construtivas da base do nosso robô foram entregues, mas ele ainda não foi completamente montado.

Há agora a confrontação no projeto da fixação de todos os chanfros e furos responsáveis pela fixação das placas e componentes. Também se inicia a definição e cálculo das medidas dos fios e cabos passantes para que possamos estabelecer as medidas ideais para sua montagem.

Este levantamento e análise das distâncias das peças é fator imprescindível no processo mecânico, pois estabelece meios de prever como os fios e cabos estarão dispostos e de que maneira estes serão fixados a seus respectivos pontos de fixação.

Visão de Montagem

Utilizando o SolidWorks podemos criar uma visão de montagem que nada mais é do que a montagem das diversas peças criadas em nosso projeto, compondo um pacote no qual podemos ter a ideia real do produto.

A visão de Montagem (assembly):

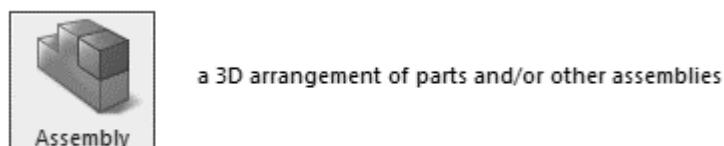


Figura 71 - Opção de criação de uma nova montagem das peças no SolidWorks 2014

Para efeito didático, usaremos uma visão já criada.

Realizamos, então, as seguintes opções:

- Entre no SolidWorks 2014
- Selecione File > Open

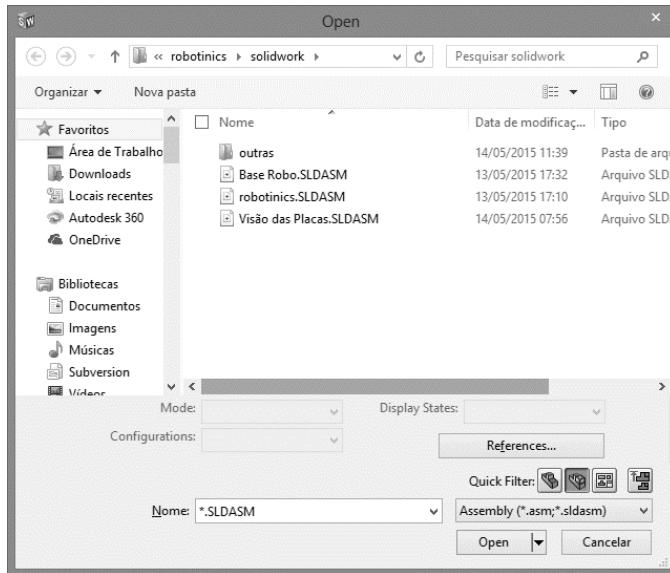


Figura 72 - Seleção do arquivo de visão das placas

- Selecione o tipo Assembly
- Selecione o arquivo Visão das Placas.SLDASM

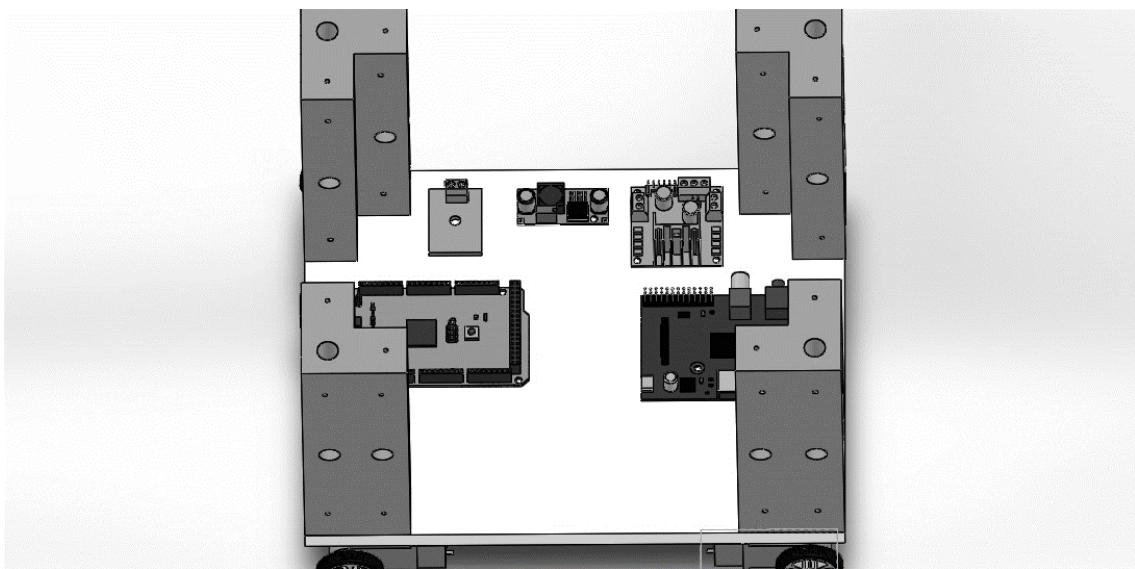


Figura 73 - Visão das placas aplicadas à base

Desta forma é possível fazer a composição dos componentes do robô verificando sua disposição antes de imprimir as placas.

Outra vantagem desta técnica é verificar eventuais colisões entre peças e componentes.

3.2 Eletrônica – Componentes Principais

Nesta etapa o robô ganha vida. É nela em que são feitas também as conexões elétricas que permitem o perfeito funcionamento do mesmo.

São acrescidas as placas principais, bem como as unidades de alimentação.

Visão Geral da Alimentação do Robô

Com estes itens, o robô começa a ganhar forma, permitindo que seja ligado e comece a responder a nossos comandos.

Através deste diagrama iremos explicar os detalhes do funcionamento e as necessidades de cada um dos itens apresentados.

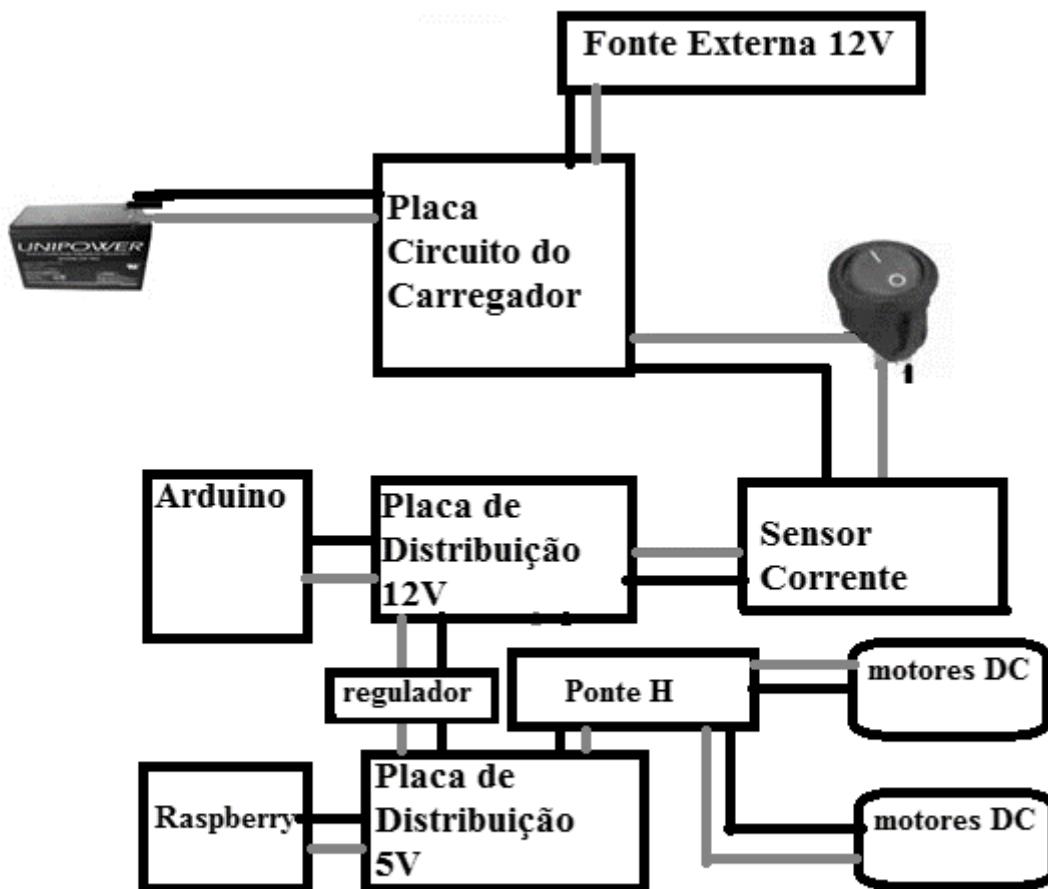


Figura 74 - Diagrama geral de alimentação do robô

Iremos omitir os itens que já foram mencionados nos módulos anteriores, pois estaríamos incorrendo em redundância e talvez cansando o leitor. Desta forma, caso haja dúvidas no trato dos itens anteriores, a melhor opção será rever os títulos 1 e 2.

Alimentação Externa

A alimentação externa torna possível gerar níveis de corrente que permitam, além do funcionamento, também a recarga do robô.

Nosso robô utilizará uma fonte DC de 12 V com 2 A de carga.

A conexão com a fonte será aplicada neste capítulo, ainda que provisoriamente em um jack⁶ externo, pois as conexões do jack serão apresentadas no próximo capítulo.

Sensor Corrente

O sensor de corrente lê a corrente consumida pelo robô, convertendo em valores digitais. Estes valores são indicados por amostragem, sendo calculados o consumo médio do robô e o tempo estimado das baterias.

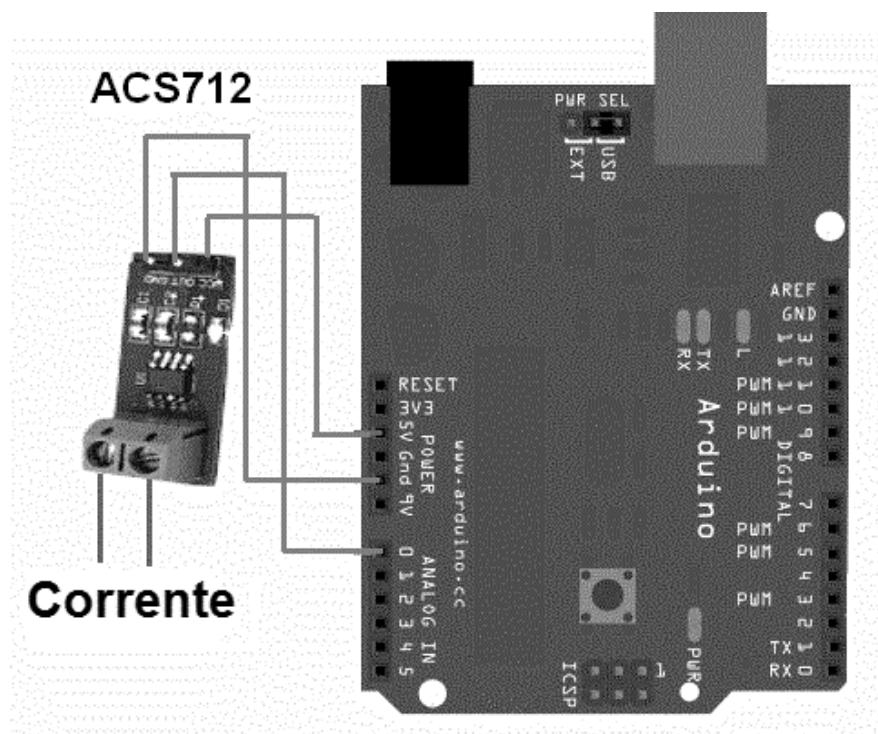


Figura 75 - Esquema elétrico do sensor de corrente

⁶ Conector Jack – É um tipo de conector que une dois segmentos de fios, podendo retirar facilmente.

Exemplo de uso deste componente:

```
float samplesnum = 1000;  
float adc_zero = 510; //relative digital zero of the arudino input from ACS712  
long currentacc = 0;  
long currentac = 0;  
long adc_raw = 0;  
long currentAD = 0;  
  
void setup()  
{  
    Serial.begin(9600);  
}  
void loop()  
{  
  
    for(int i=0; i<samplesnum; i++)  
    {  
        adc_raw = analogRead(0);  
        currentacc += (adc_raw - adc_zero) * (adc_raw - adc_zero); //rms  
    }  
    currentAD = (currentacc * 75.7576)/ 1024; //D to A conversion  
    currentac = sqrt(currentAD)/samplesnum; //rms  
    Serial.println(currentac);  
}
```

Regulador 5 V

A alimentação padrão do sistema primário do robô é 12 V DC, porém parte do sistema elétrico é alimentado por 5 V. A esta necessidade faz-se necessário o uso de um sistema de regulagem para 5 volts.

São características requeridas neste sistema:

- Regulagem de tensões 12 volts, porém com tolerância de 14 até 7 volts
- Fornecimento mínimo de corrente de 2^a
- Capacidade de filtrar ruídos na tensão, eventualmente gerados pelos motores e outros dispositivos
- Capacidade de manter tensão minimizando rippler de tensão⁷

O conversor de 12 V de entrada para 5 V garante um fornecimento estável de 5 V para todos os equipamentos.

Entre os equipamentos que utilizam 5 V, estão:

- Raspberry PI
- Servomotores
- LEDs
- Shields do Arduino e sensores

⁷Ripple de tensão - Tipicamente a tensão de ripple na eletrônica é um valor residual e periódico obtido de uma fonte de tensão que, por sua vez, é alimentada por uma de uma corrente alternada. Este ripple é derivado da incompleta supressão da onda alternada no interior da fonte de tensão (Wikipédia).

Placa de Controle dos Motores DC

De forma geral, este componente utiliza o CI⁸ L298N.

A “Shield” de controle de motor DC fornece chaveamento das polaridades (ponte H) e fornecimento de carga elétrica dos motores DC de 12 V.

Vantagens

- Inversão das polaridades do motor nas saídas dos motores DC 9 V
- Proteção e isolamento entre as portas digitais e tensões de alimentação 12 V
- Controlador dos motores L298N (ponte h)
- Chave Seletora A (ENA)
- Chave Seletora B (ENB)
- Portas Seletoras de Sentido (IN1, IN2, IN3, IN4)
- Saída de MOTOR DC1
- Saída de MOTOR DC2

Desvantagem

- Não permite ajuste de velocidade

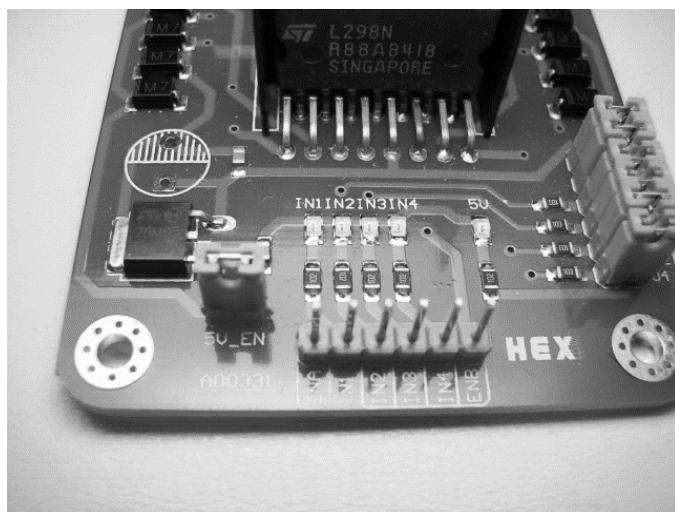


Figura 76 - Imagem dos pinos da placa de Controle DC

⁸CI – Referência a Circuito Integrado

Esquemático de ligações

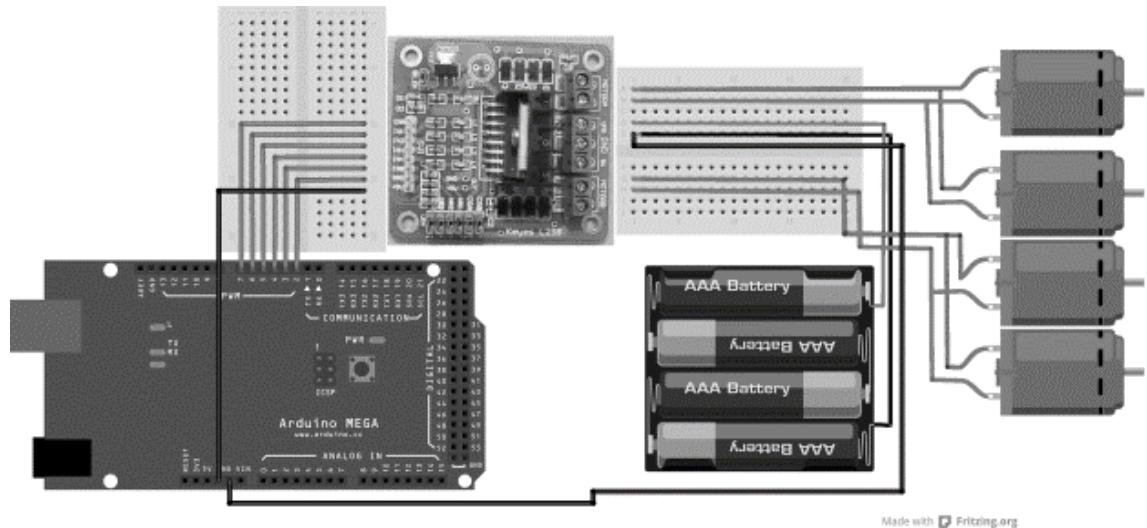


Figura 77 Esquemático de ligação elétrica no Arduino

Exemplo de software Arduino para controle da ponte H

```
#define CW 2
#define CCW 3

#define ENA 8
#define ENB 13

#define black 2 // In1
#define brown 3 // In2
#define orange 4 // In3
#define yellow 5 // In4

void setup() {

    DDRB = 0x3f; // Digital pins 8-13 output
    PORTB = 0x00; // all outputs Dp8-13 set to off

    pinMode(CW, INPUT);
    pinMode(CCW, INPUT);

    digitalWrite(CW, 1); // pullup on
    digitalWrite(CCW,1); // pullup on

}
```

```

void loop() {

    if (!digitalRead(CW)) forward(480, 0);
    if (!digitalRead(CCW)) reverse(480, 0);

} // end loop

void reverse(int i, int j) {

    // Pin 8 Enable A Pin 13 Enable B on
    digitalWrite(ENA, HIGH);
    digitalWrite(ENB, HIGH);

    j = j + 10;
    while (1) {

        digitalWrite(black, 0);
        digitalWrite(brown, 1);
        digitalWrite(orange, 1);
        digitalWrite(yellow, 0);
        delay(j);
        i--;
        if (i < 1) break;

        digitalWrite(black, 0);
        digitalWrite(brown, 1);
        digitalWrite(orange, 0);
        digitalWrite(yellow, 1);
        delay(j);
        i--;
        if (i < 1) break;

        digitalWrite(black, 1);
        digitalWrite(brown, 0);
        digitalWrite(orange, 0);
        digitalWrite(yellow, 1);
        delay(j);
        i--;
        if (i < 1) break;

        digitalWrite(black, 1);
        digitalWrite(brown, 0);
        digitalWrite(orange, 1);
        digitalWrite(yellow, 0);
        delay(j);
    }
}

```

```

    i--;
    if (i < 1) break;
}

// all outputs to stepper off
digitalWrite(ENA, LOW);
digitalWrite(ENB, LOW);

} // end reverse()

void forward(int i, int j) {

    // Pin 8 Enable A Pin 13 Enable B on
    digitalWrite(ENA, HIGH);
    digitalWrite(ENB, HIGH);

    j = j + 10;
    while (1) {

        digitalWrite(black, 1);
        digitalWrite(brown, 0);
        digitalWrite(orange, 1);
        digitalWrite(yellow, 0);
        delay(j);
        i--;
        if (i < 1) break;

        digitalWrite(black, 1);
        digitalWrite(brown, 0);
        digitalWrite(orange, 0);
        digitalWrite(yellow, 1);
        delay(j);
        i--;
        if (i < 1) break;

        digitalWrite(black, 0);
        digitalWrite(brown, 1);
        digitalWrite(orange, 0);
        digitalWrite(yellow, 1);
        delay(j);
        i--;
        if (i < 1) break;

        digitalWrite(black, 0);
        digitalWrite(brown, 1);
        digitalWrite(orange, 1);
    }
}

```

```

digitalWrite(yellow, 0);
delay(j);
i--;
if (i < 1) break;

}

// all outputs to stepper off
digitalWrite(ENA, LOW);
digitalWrite(ENB, LOW);

} // end forward()

```

Tabela de Funcionamento dos pinos

IN1	IN2	IN3	IN4	SELA	SELB	DC1	DC2
1	0	0	0	1	0	+	- 0
0	1	0	0	1	0	- +	0
0	0	1	0	0	1	0 +	-
0	0	0	1	0	1	0 -	+

Nota:

A notação +- refere-se ao posicionamento dos polos com relação ao motor, onde -+ representa a inversão deste.

O valor zero em DC1 e DC2 representa que ambos os bornes estão em GND. Ou seja, não passa corrente.

Regulador de Tensão 5 V

O regulador de tensão usado é o turnigy SBEC 26 V, conforme a foto abaixo.



Figura 78 - Componente de regulagem de tensão

- Tipo: **Switching**
- Proteção de entrada: **Proteção contra inversão de polaridade**
- Saída (constante): **5 V/5A ou 6 V/5A**
- Entrada: **8 V-26 V (lipo 2 7 cell)**
- Peso: **18 g**

A vantagem deste tipo de componente está em seu baixo custo e alta potência.

O regulador de tensão consegue realizar a conversão entre 12 V DC para 5 V DC, garantindo, além da qualidade do sinal, tratamento contra problemas na sobretensão.

Este tipo de regulador tem um baixo custo, alto rendimento e é muito utilizado inclusive em projetos de aviões, drones etc.

Tem capacidade de fornecer alta corrente.

Placa de Distribuição de 5 V ou 12 V

A placa de distribuição de 5 V ou 12 V permite que através de uma única entrada 5 V ou 12 V de alimentação sejam fornecidas várias portas de saída para alimentar diversos dispositivos.

A placa apenas distribui o quadro elétrico.

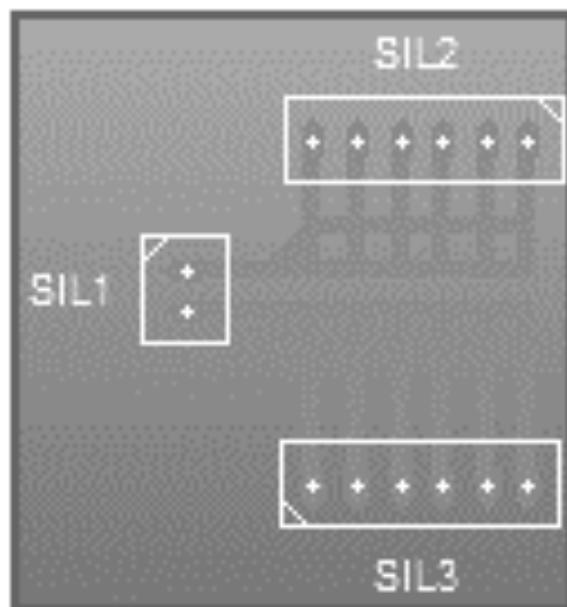


Figura 79 - Layout da placa

O SIL1 é o componente que recebe a entrada de tensão proveniente do retificador 5 V.

SIL2 e SIL3 são os conectores de distribuição de energia para os componentes.

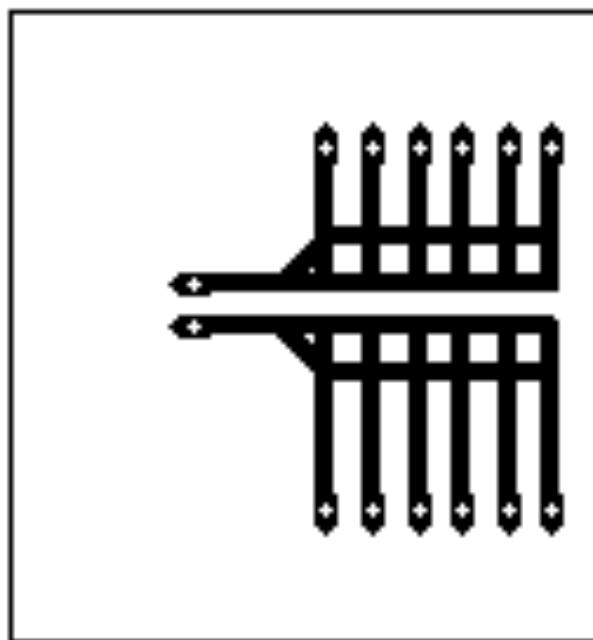


Figura 80 - Visão das trilhas

Arquivos:

Espelho: /pcb/

Shield Expansão Arduino – Sensor Shield

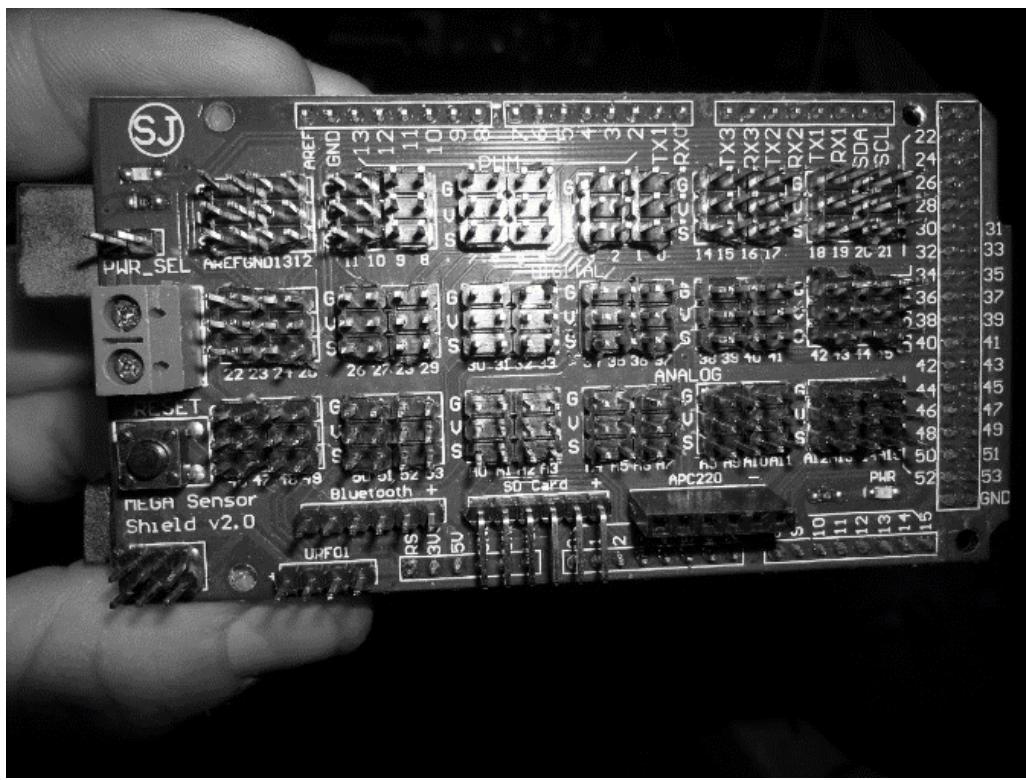


Figura 81 - Placa de expansão Sensor Shied

O pacote Arduino possui uma placa (shield) que permite separar as conexões já permitindo que todo o conector seja previamente preparado com alimentação.

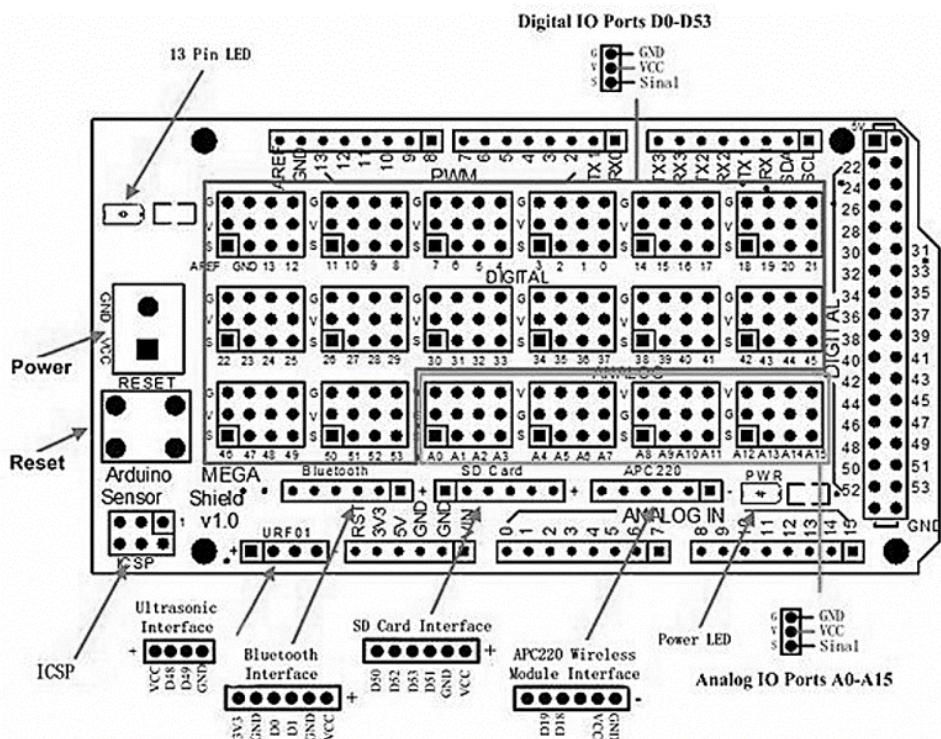


Figura 82 - Pinagem do Arduino Shield

A vantagem estratégica desta shield é a economia de tempo na confecção de placa específica, pois esta já prepara os barramentos para os servomotores.

Sensor Ultrassom

O ultrassom é um dos primeiros sensores que iremos utilizar no nosso projeto.

O ultrassom permite detectar obstáculos sólidos e determinar a sua distância do sensor.

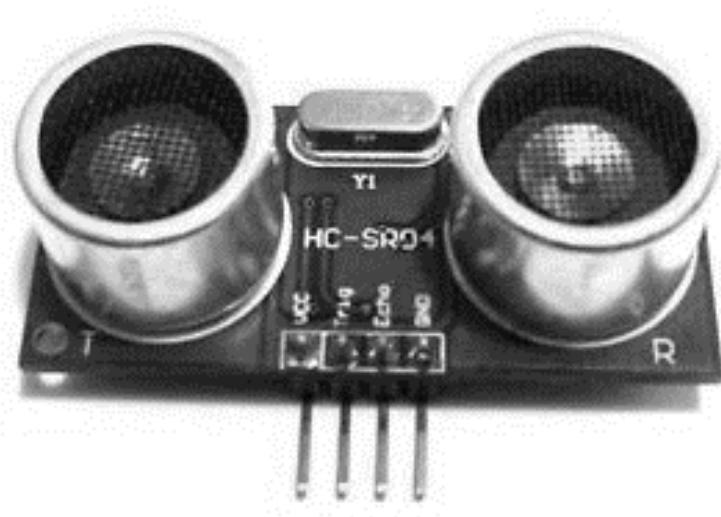


Figura 83 - Visão do Sensor de Ultrassom

Utilizamos o HC-SR04 devido a seu baixo preço e fácil aquisição.

O princípio de funcionamento de um ultrassom é bem simples.

O ultrassom possui dois dispositivos: um emite um pulso sonoro e outro ouve o som.

O tempo entre a emissão do som e a recepção do som determina a distância.

Lembramos que a fórmula de espaço percorrido é dada sobre a seguinte fórmula, em que a velocidade é constante:

$$S = S_0 + V \cdot t$$

V = É dada 340,29 m/s.

Desta forma,

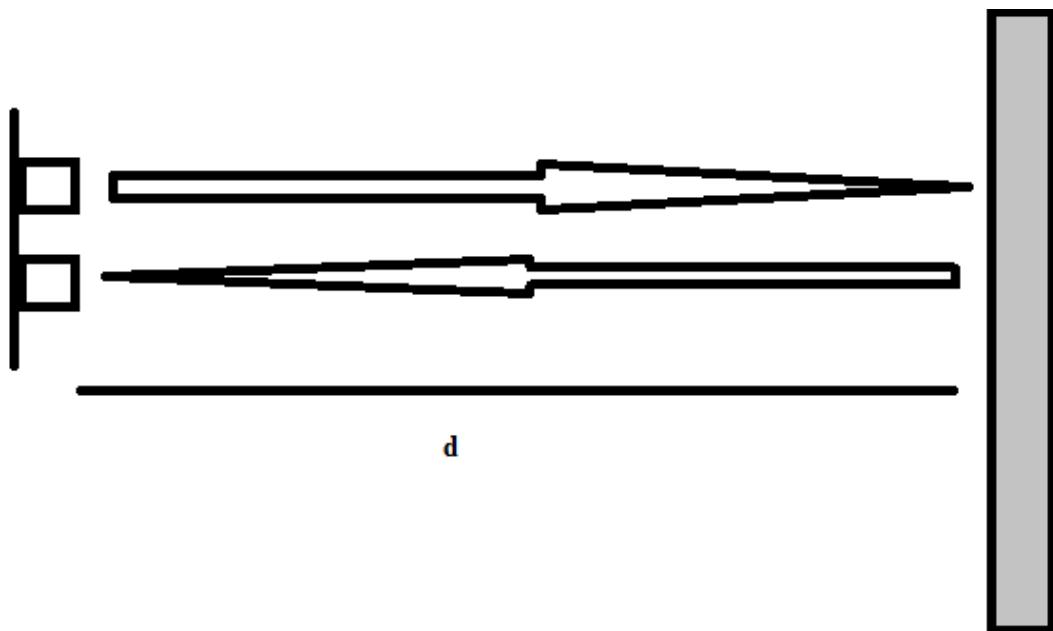


Figura 84 - Exemplo de funcionamento do ultrassom

O sensor emite o pulso que é refletido de volta para o sensor.

d é a distância em cm e a distância percorrida é $d * 2$.

Desta forma, S é a distância percorrida. Iremos imaginar que o sensor esteja a 50 cm do objeto medido. Assim, a distância percorrida na ida e na volta é de um metro.

Para calcularmos o tempo que demora, basta aplicar a fórmula.

$S = S_0 + V * t \Rightarrow 1 = 0 + 340,29 * t \Rightarrow 340,29t = 1 \Rightarrow t = 1/340,29 \Rightarrow t = 0,00293$ segundos ou 2,93 milésimos de segundo.

Provavelmente o leitor deve pensar que é pouco tempo, porém, olhando o datasheet do equipamento, que é a especificação técnica para o sensor, percebe que o equipamento tem sensibilidade ainda maior.

Parâmetros Elétricos

- Tensão de trabalho: DC 5 V
- Corrente de trabalho: 15 mA
- Frequência de trabalho: 40 Hz
- Alcance máximo: 4 m
- Alcance mínimo: 2 cm
- Ângulo de ataque: 15 graus

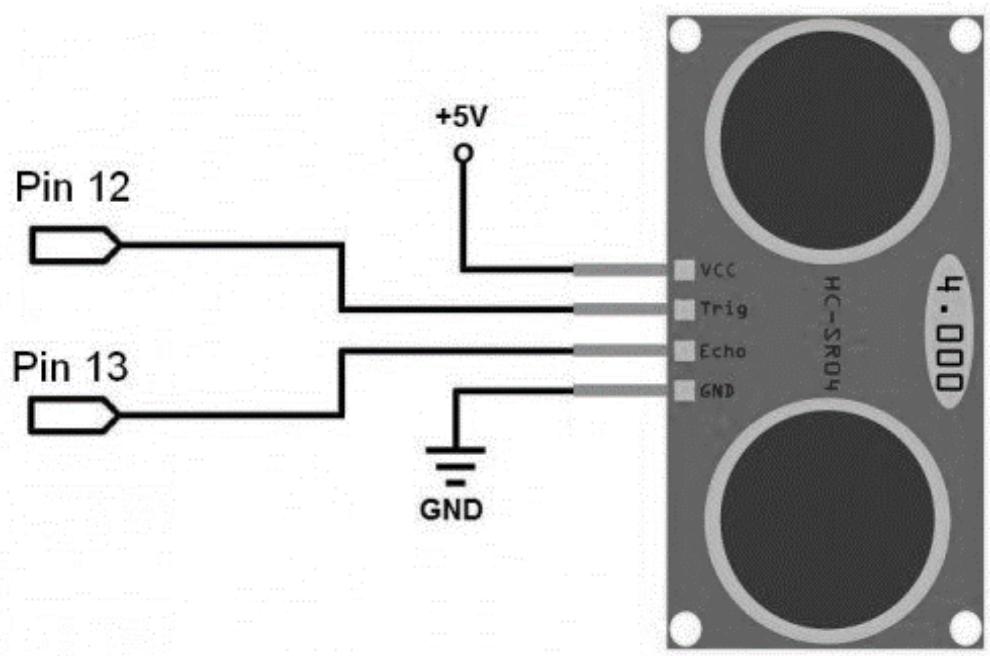


Figura 85 - Esquemático de ligação do ultrassom

Exemplo de código

```
#include <Ultrasonic.h>

Ultrasonic ultrasonic(12,13);

void setup()
{
    Serial.begin(9600);
    pinMode(echoPin, INPUT);
    pinMode(trigPin, OUTPUT);
}

void loop()
{
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    int distancia = (ultrasonic.Ranging(CM));
    Serial.print("Distancia (cm): ");
    Serial.println(distancia);
    delay(800);
}
```

As conexões elétricas serão mostradas posteriormente, porém, vale lembrar que o sensor deve ser ligado na tensão de 5 V e GND e os dois pinos de trig e echo são definidos e conectados no Arduino.

Para integrar com o Arduino usamos a lib Ultrasonic.h, que pode ser baixada no site <https://github.com/JrodrigoTech/Ultrasonic-HC-SR04>.

Serão montados três sensores de ultrassom no robô, onde o sensor fica localizado.

São estes:

- Ultrassom0 – Parte traseira do robô
- Ultrassom1 – Frente
- Ultrassom2 – Cabeça do robô

Neste momento instalaremos apenas os ultrassons 0 e 1 (primeiro e segundo), pois o da cabeça será instalado em ocasião oportuna.

Infelizmente, nossos sensores ainda não possuem posições para encaixe, pois as bases que serão utilizadas ainda não foram construídas. Desta forma, posicionaremos provisoriamente os sensores na base de acrílico colados com cola quente, para que possamos realizar alguns testes. No próximo capítulo iremos instalá-los já em seus locais definitivos.

As conexões de 5 V e GND serão feitas nas expansões de 5 V, já previamente definidos e documentados.

Pinout de ligação do Arduino Shield

O pinout é a tabela de pinos endereçados para os dispositivos.

A tabela de pinout fornece uma organização e orientação na confecção e montagem dos cabos e conexões.

De posse desta tabela, o montador do robô simplesmente precisa conectar o cabo correto no pino correto. O processo de montagem da tabela de pinout auxilia no processo de montagem e evita erros.

Pin	Descrição
06	pinCabeca – Pino de controle do Servo motor da cabeça
44	pinBDireito – Pino de controle do Servo motor braço direito
08	pinMDireita – Pino de controle do Servo motor mão direita
09	pinBEsquerdo – Pino de controle do Servo motor braço esquerdo
10	pinGarraesq – Pino de controle Servo motor garra esquerda
11	pinGarraDir – Pino de controle Servo motor garra direita
46	pinPEsquerdo – Pino de controle Servo motor garra esquerda
A6	pinVoltagem – Pino de leitura analógica para leitura da voltagem
A4	analogInGas – Pino de leitura do sensor de gas (reservado)
A5	analogOutGas – Pino analógico para controle LED sensor de gás (reservado)
A0	analogInCorr – Pino de leitura sensor de corrente
17	PIN0_RFRX – Pino de radiofrequência 433 RX (reservado)
16	PIN0-RFTX – Pino de radiofrequência 433 TX (reservado)
2	PIN0_TRIGGER – Sensor ultrassom trigger
3	PIN0_Echo – Sensor ultrassom echo
40	Pin0_trigger1 – Sensor ultrassom trigger 1
35	Pin0_echo1 – Sensor ultrassom echo
38	Arduino2TX – Porta Serial virtual TX
37	Arduino2RX – Porta Serial Virtual RX
26	ENA – Sensor ponte H
28	IN1 – Sensor Ponte H
30	IN2 – Sensor Ponte H
32	IN4 – Sensor Ponte H
34	IN3 – Sensor Ponte H
36	ENB – Sensor Ponte H
I2C	Sensor LCD
A10	RF 433 Controle Remoto – D0 (reservado)
A9	RF 433 Controle Remoto – D1 (reservado)
A8	RF 433 Controle Remoto – D2 (reservado)
A7	RF 433 Controle Remoto – D3 (reservado)
33	Relê Bateria
18	TX bluetooth (reservado)
19	RX bluetooth (reservado)
39	Laser do Robotinics (reservado)

É muito comum em projetos eletrônicos que alguns pinos no pinout não sejam utilizados no projeto padrão, mas **reservados** para uso futuro, quando o hardware e o firmware já contemplem os recursos esperados.

Neste exemplo o leitor verá diversos pinos que estarão como **reservados**, porém não terão uso neste livro.

O uso destes pinos envolve dispositivos que por questões didáticas não serão avaliados neste material.

3.3 Software

Vamos apresentar agora o software do Arduino.

Neste módulo detalharemos o processo de testes e análise das conexões do robô.

Para este capítulo ocorrer deve-se, primeiramente, ter realizado todas as duas etapas anteriores e também lido todos os capítulos.

Desta forma, ao realizar a carga do firmware, poderemos observar os resultados esperados.

Carregando o firmware no Arduino

Para carregar o firmware no Arduino, primeiramente deve-se ter realizado a instalação do programa, conforme apontado no Capítulo 1.

Ao finalizar o instalador, entre no programa Arduino.



Figura 86 - Programa Arduino

Carregando o Fonte do Robô

No Arduino realize a seguinte opção:

1. Selecione o menu **Arquivo > Abrir**. Neste momento surgirá uma tela perguntando onde está o fonte do projeto.
2. Vá no projeto do Robotinics, selecionando a subpasta Arduino.

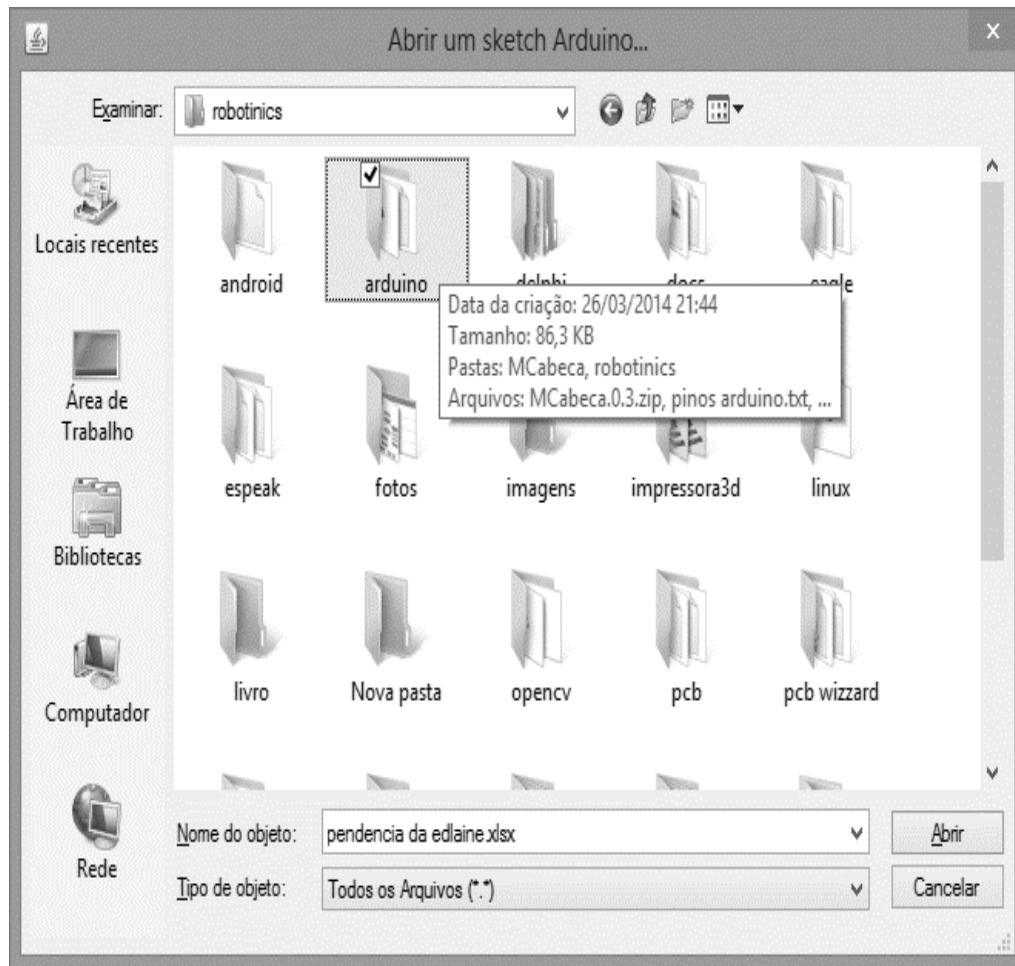
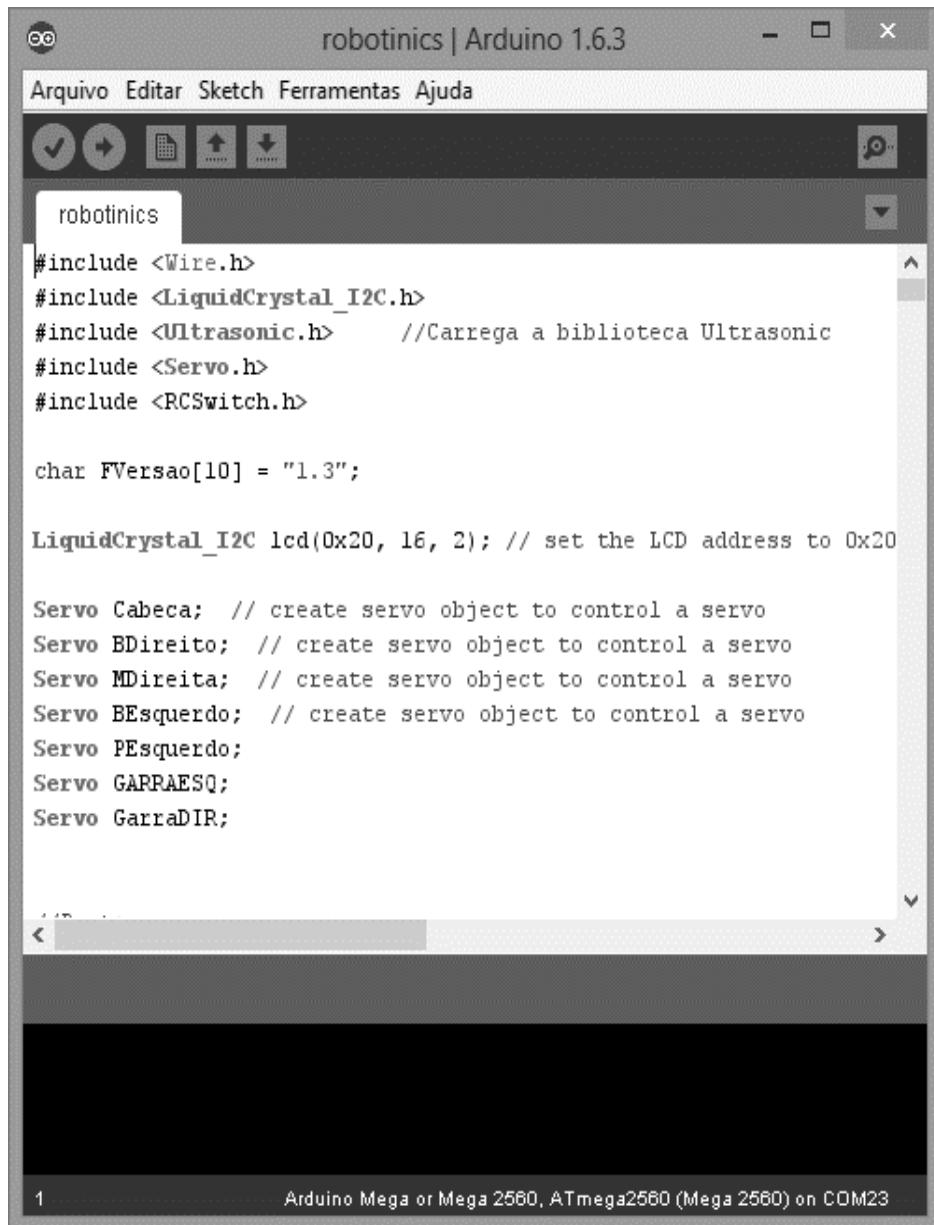


Figura 87 - Selecionando fonte do projeto

3. Selecione a subpasta Robotinics e abra o arquivo robotinics.ino



The screenshot shows the Arduino IDE interface with the title bar "robotinics | Arduino 1.6.3". The menu bar includes "Arquivo", "Editar", "Sketch", "Ferramentas", and "Ajuda". Below the menu is a toolbar with icons for file operations. The main code editor window contains the following C++ code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Ultrasonic.h>      //Carrega a biblioteca Ultrasonic
#include <Servo.h>
#include <RCSwitch.h>

char FVersao[10] = "1.3";

LiquidCrystal_I2C lcd(0x20, 16, 2); // set the LCD address to 0x20

Servo Cabeca; // create servo object to control a servo
Servo BDireito; // create servo object to control a servo
Servo MDireita; // create servo object to control a servo
Servo BEsquerdo; // create servo object to control a servo
Servo PEsquerdo;
Servo GARRAESQ;
Servo GarraDIR;
```

The status bar at the bottom indicates "1" and "Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM23".

Figura 88 - Ambiente de Edição do programa do arduino

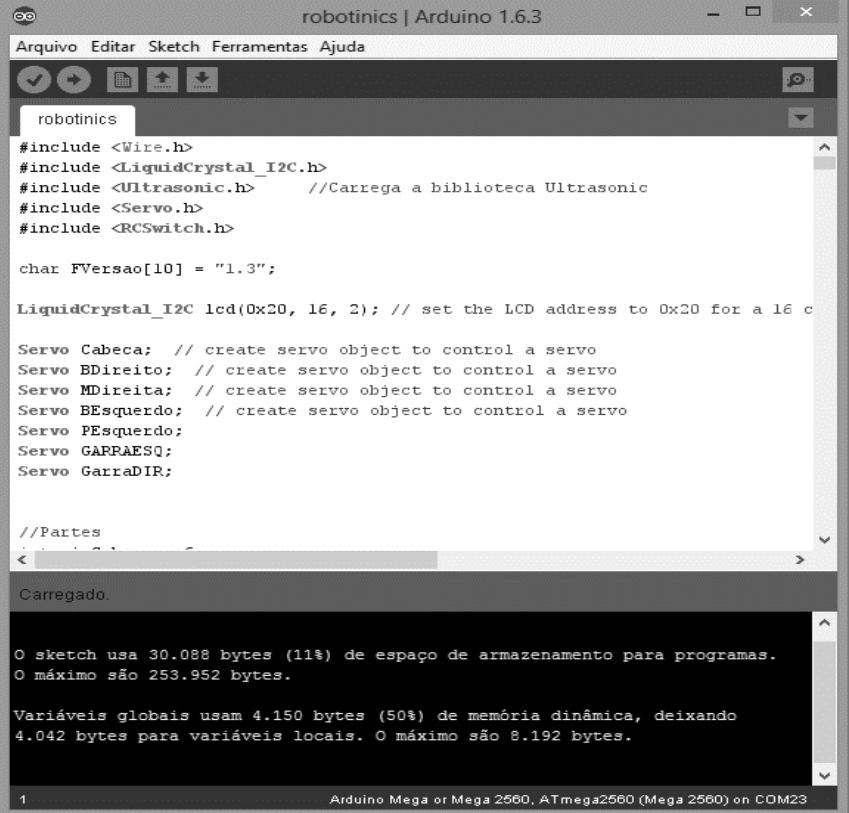
4. Clique no botão Verificar  e aguarde a confirmação de sucesso de compilação.

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** robotinics | Arduino 1.6.3
- Menu Bar:** Arquivo, Editar, Sketch, Ferramentas, Ajuda
- Sketch Editor:** The code for the sketch is displayed. It includes includes for Wire.h, LiquidCrystal_I2C.h, Ultrasonic.h, Servo.h, and RCSwitch.h. It defines a character array FVersao[10] with the value "1.3". A LiquidCrystal_I2C object lcd is created with address 0x20. It also declares several Servo objects: Cabeca, BDireito, MDireita, BEsquerdo, PEsquerdo, GARRAESQ, and GarraDIR.
- Compile Output:** The output window shows the compilation results:
 - Compilação terminada.
 - O sketch usa 30.088 bytes (11%) de espaço de armazenamento para programas. O máximo são 253.952 bytes.
 - Variáveis globais usam 4.150 bytes (50%) de memória dinâmica, deixando 4.042 bytes para variáveis locais. O máximo são 8.192 bytes.
- Status Bar:** Shows the board as Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM23.

Figura 89 - Compilação realizada com sucesso

5. Se compilado com sucesso, faça a carga do firmware no equipamento clicando no botão  Carregar.



The screenshot shows the Arduino IDE interface with the title bar "robotinics | Arduino 1.6.3". The menu bar includes "Arquivo", "Editar", "Sketch", "Ferramentas", and "Ajuda". Below the menu is a toolbar with icons for file operations. The main area displays the code for the "robotinics" sketch. The code includes #include statements for Wire.h, LiquidCrystal_I2C.h, Ultrasonic.h, Servo.h, and RCSSwitch.h. It defines variables for servo objects (Cabeca, BDireito, MDireita, BEsquerdo, PEsquerdo, GARPAESQ, GarraDIR) and an LCD object (lcd). A section titled "//Partes" contains comments. The status bar at the bottom right shows "Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM23". The status bar also indicates the sketch uses 30.088 bytes (11%) of program storage space, with a maximum of 253.952 bytes. Global variables use 4.150 bytes (50%) of dynamic memory, leaving 4.042 bytes for local variables, with a maximum of 8.192 bytes.

Figura 90 - Carga de firmware realizada com sucesso

6. O resultado é que o firmware será carregado no Arduino, que possuirá agora o programa para testes.

Atenção

O projeto será apresentado contendo diversas dependências de componentes. Verifique se possui todos os componentes. Caso não possua, entre na pasta tools e pegue os arquivos contendo todos os componentes e instale na pasta library, que se encontra dentro da pasta do Arduino.

Após instalar as pastas contendo as bibliotecas, é necessário reiniciar o Arduino, realizando os procedimentos já mencionados.

Não se esqueça de modificar as configurações de porta do Arduino mega e o tipo de Arduino (mega), para que o compilador crie o binário do firmware correto.

Gerenciando e dando os primeiros comandos

Para verificar o perfeito funcionamento do robô, entre no Monitor Serial, clicando na opção



Surgirá uma tela como anunciada abaixo:

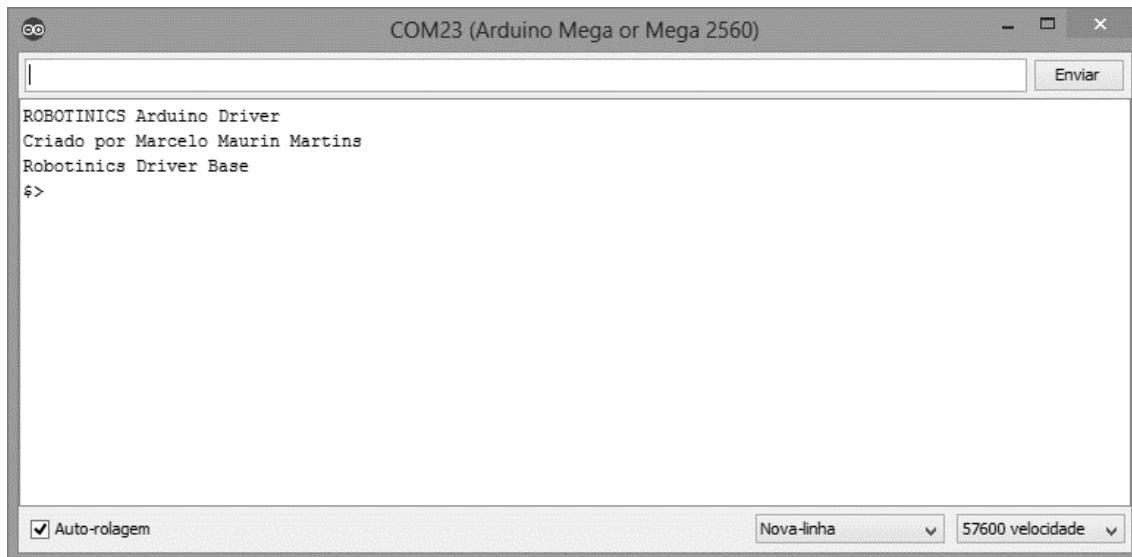


Figura 91 - Debugando retorno da USB

O Robotinics funciona como um terminal. Se houver comandos, ele responderá emitindo a ação correspondente.

Desta forma, qualquer interface poderá atuar se comunicando com o Arduino e realizando movimentos programados.

Todos os comandos são MAIÚSCULOS justamente para diferenciar de comandos UNIX.

Os comandos do ROBOTINICS podem ser listados pelo Comando MAN, que é a abreviação de Manual.

Lista de Comandos do Arduino Robotinics

MAN

Manual de comandos. Mostra uma explicação breve sobre descrição dos comandos permitidos.

Função: Help();

VER

Imprime a versão do firmware instalado do robô.

Função: Ver();

RE

Controle de Ré, que permite acionar a ré do robô. O acionamento da ré permite que o robô ande para trás, porém ele utilizará um sensor ultrassom para evitar colisão.

Este sensor manterá uma distância de 20 cm. Assim, se o usuário tentar acionar ou mantiver o robô em ré o comando será ignorado.

O movimento será mantido até que outro comando de movimento seja chamado.

Função: Re();

PARA

Controle de Parar. Quando em movimento, este comando faz o robô parar.

Função: Para();

FRENTE

Controle de movimento para Frente, gera o movimento para frente até que encontre um comando oposto ou o sensor de ultrassom frontal encontre um obstáculo a 20 cm.

Função: Frente();

GESQ

Controle de Girar à Esquerda. Desta forma, as rodas da direita apresentam movimento em um sentido e as da esquerda em sentido invertido.

Função: GiraEsq(0);

GDIR

Controle de Girar à Direita, que faz com que as rodas da direita apresentem movimento em um sentido e as da esquerda em sentido invertido.

Função: GiraDir(0);

GGARRADIR

Controle de Girar a Garra Direita.

Função: GGarraDir(0);

GPUNHOESQ

Controle de Girar a PUNHO ESQ.

Função: GPUNHOESQ(0);

CLS

Limpa mensagens 1 e 2 da tela de LCD.

Função: CLS();

MSG1

Imprime uma mensagem no LCD frontal do robô. A mensagem é informada na primeira linha do LCD.

Sintaxe:

MSG1: <texto>

Função: não há

MSG2

Imprime uma mensagem no LCD frontal do robô. A mensagem é informada na segunda linha do LCD.

Sintaxe:

MSG2: <texto>

Função: Não há

SERIAL

Envia uma mensagem na porta TTL destinada à integração com outros Arduinos e/ou periféricos.

Sintaxe:

SERIAL:<texto>

Função: EnviaArduino(pBuffer.substring(5));

LCDCLEAR

Limpa a tela do LCD frontal, equivalente ao CLS.

Função: não há

GPS

Lê coordenadas do GPS quando disponível no robô.

Função: Le_GPS();

ACEL

Lê as informações do acelerômetro do robô quando disponível.

Função: Le_ace();

ULTRA0

Lê a medida do Ultrassom Frontal

Função: Le_Ultrasom(1);

ULTRA1

Lê a medida do Ultrasom Rê

Função: Le_Ultrasom1(1);

GAS

Lê Sensor Gás Metano, quando disponível no robô.

Esta opção detecta vazamentos de gás localizados em dutos ou mesmo em casa quando o cliente não estiver, permitindo que emita um alerta ao proprietário.

Função: Le_gas();

CORR

Lê a corrente consumida pelo robô. Com isso, é possível estabelecer o consumo e também o tempo estimado das baterias

Função: Le_corr();

TESTE

Teste de movimento. Neste comando o robô aciona de forma programada um script, realizando todos os movimentos de todos os membros. O teste é muito utilizado para apresentações em que há necessidade de se mostrar o poder dos movimentos sem, no entanto, desenvolver scripts complexos de movimento.

Função: TesteMovimento();

GBESQ

O comando GBESQ ou Gira Braço da Esquerda permite realizar movimentos programados do braço através da indicação do ângulo que se deseja atacar.

Sintaxe:

GBESQ = Ângulo

Função: GBEsq(Ângulo);

GBDir

O comando GBDIR ou Gira Braço da Direita permite realizar movimentos programados do braço através da indicação do ângulo que se deseja atacar.

Sintaxe:

GBDIR = Ângulo

Função: GBDir(Ângulo);

GPUNHOESQ

Permite girar o punho esquerdo do robô indicando um ângulo que se reflete em posição do punho em relação ao braço.

Sintaxe:

GPUNHOESQ = Ângulo

GPUNHODIR

Permite girar o punho direito do robô indicando um ângulo que se reflete em posição do punho em relação ao braço.

Sintaxe:

GPUNHODIR = Ângulo

GCABECADIR

Permite movimentar a cabeça para o lado direito, atacando um ângulo definido no comando.

Atribuindo 0 como sendo olhando em frente.

Sintaxe:

GCABECADIR= Ângulo

Função: GCabecaDir(Ângulo);

GCABECAESQ

Permite movimentar a cabeça para o lado esquerdo atacando um ângulo definido no comando.

Atribuindo 0 como sendo olhando em frente.

Sintaxe:

GCABECAESQ= Ângulo

Função: GCabecaEsq(Ângulo);

GPGARRADIR

Controle de Girar a Garra Direita.

Sintaxe:

GPGARRADIR = Ângulo

Função: GGarraDir(Ângulo);

GPGARRAESQ

Controle de Girar a Garra Esquerda.

Sintaxe:

GPGARRAESQ = Ângulo

Função: GGARRAESQ(Ângulo);

GPPUNHOESQ

Controle de Girar PUNHO Esquerdo, atacando um ângulo definido.

Sintaxe:

GPPUNHOESQ= Ângulo

GPPUNHODIR

Controle de Girar PUNHO DIR, atacando um ângulo definido.

Sintaxe:

GPPUNHODIR= Ângulo

Função: GPUNHODIR(Ângulo);

SETMONITOR

Status de monitoramento = ON/OFF

O monitoramento é um debug que é habilitado para se testar as implementações e correções do robô.

Nele são mostrados os diversos parâmetros lidos, como tensão, corrente e demais sensores no LCD.

O uso desta ferramenta dificulta a impressão de mensagem no LCD, pois o ciclo de atualização das mensagens pela ferramenta é constante.

Função: SETMONITOR(true);

Checklist de validação da Etapa

Iremos agora testar os movimentos do robô, a fim de garantir que estejam funcionando corretamente.

Este teste visa garantir que toda a montagem do robô foi concluída e que ele esteja dentro das especificações funcionais.

É muito importante, ao finalizar uma etapa, que não deixemos nenhum problema residual. Em um projeto, os problemas identificados em uma etapa do projeto e deixados costumam nos assombrar durante toda a vida útil do produto.

Então a melhor coisa a fazer é validar e testar cada etapa individualmente, garantindo, com isso, que o que foi montado esteja de acordo com o que havia sido planejado.

Para auxiliar neste processo, vamos colocar o nosso robô em uma base, ou plataforma, para que as rodas girem livremente e não se movimentem.

Desta forma podemos executar os comandos do robô sem medo de que ele quebre um vaso ou alguma outra coisa da casa.

Para tanto, entramos mais uma vez no Arduino, executando o terminal.

Teste1

Digite o comando: FRENTE e veja se as rodas estão se movimentando no sentido correto.

Caso estejam correndo fora do padrão, possivelmente você invertiu os fios na ponte H que ligam o motor DC à ponte H.

Passe a mão sobre o ultrassom frontal e verifique se está parando o movimento.

Caso não esteja ou o motor não acione, possivelmente as conexões do ultrassom estão incorretas.

Teste 2

Realize os comandos de movimento RE, GDIR e GESQ em sequência.

No movimento de RE, o comportamento do ultrassom deve ser o mesmo apresentado para o comando FRENTE.

Neste teste, além do teste anterior, deve ser observado se as rodas estão rodando livremente e com a força esperada.

Para realizar este teste, passe o dedo suavemente sobre as rodas, verificando se elas resistem ao toque suave.

À medida que pressiona o dedo nas rodas fornecendo maior resistência, a tendência das rodas é ir diminuindo a velocidade, e desta forma será possível avaliar se o robô irá realmente se movimentar.

Problemas nesta etapa estão geralmente ligados à fonte externa. Fontes com baixa potência geralmente refletem pouca força nos motores e por vezes até o reset do equipamento quando forçado.

Bluetooth.exe

O Programa Bluetooth é um simulador de movimentos por conexão bluetooth.

Para auxiliar o leitor no funcionamento e execução dos comandos do robô, foi criado um programa que pode ser conectado tanto na porta USB do Arduino ou via bluetooth.

O software foi escrito em Lazarus e necessita da biblioteca sdpo para funcionar.

Fonte: robotinics\tools\bluetooth

Abordaremos a conexão em bluetooth nos próximos capítulos.

Este programa será utilizado em diversos capítulos para que possamos controlar diretamente o Arduino, sem a necessidade de passar pelo Raspberry.

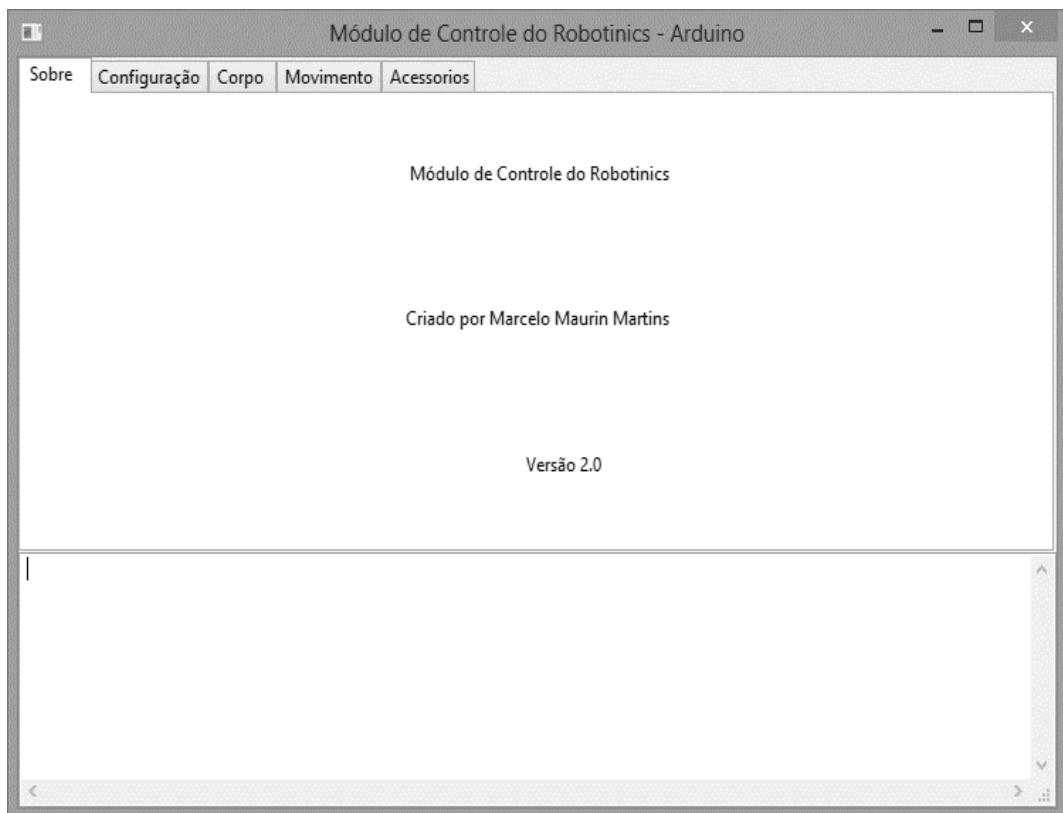


Figura 92 - Módulo de Controle do Robotinics - Arduino

Para acessar o programa de gerenciamento do robô, siga os seguintes passos:

- 1) Entre na pasta do projeto Robotinics
- 2) Entre na pasta Tools/Bluetooth
- 3) Execute o programa Bluetooth.exe

- 4) Conecte o Arduino na porta USB de seu computador ou pareie o bluetooth do seu computador com o bluetooth do Arduino.
- 5) Identifique a porta COM que foi criada ao conectar o dispositivo desejado.
- 6) Selecione a porta através da aba de configurações.



Figura 93 - Módulo de controle, configuração do dispositivo

Para movimentar o robô, informe a porta correta e clique no botão Conectar.

- 7) Ao pressionar o botão Conectar surgirá um texto conforme apresentado acima.

8) Clique na aba Movimento



Figura 94 - Comandos de movimentação do robô

9) Clique no botão Frente.

10) O robô irá simular a execução do comando, como se fosse digitado.

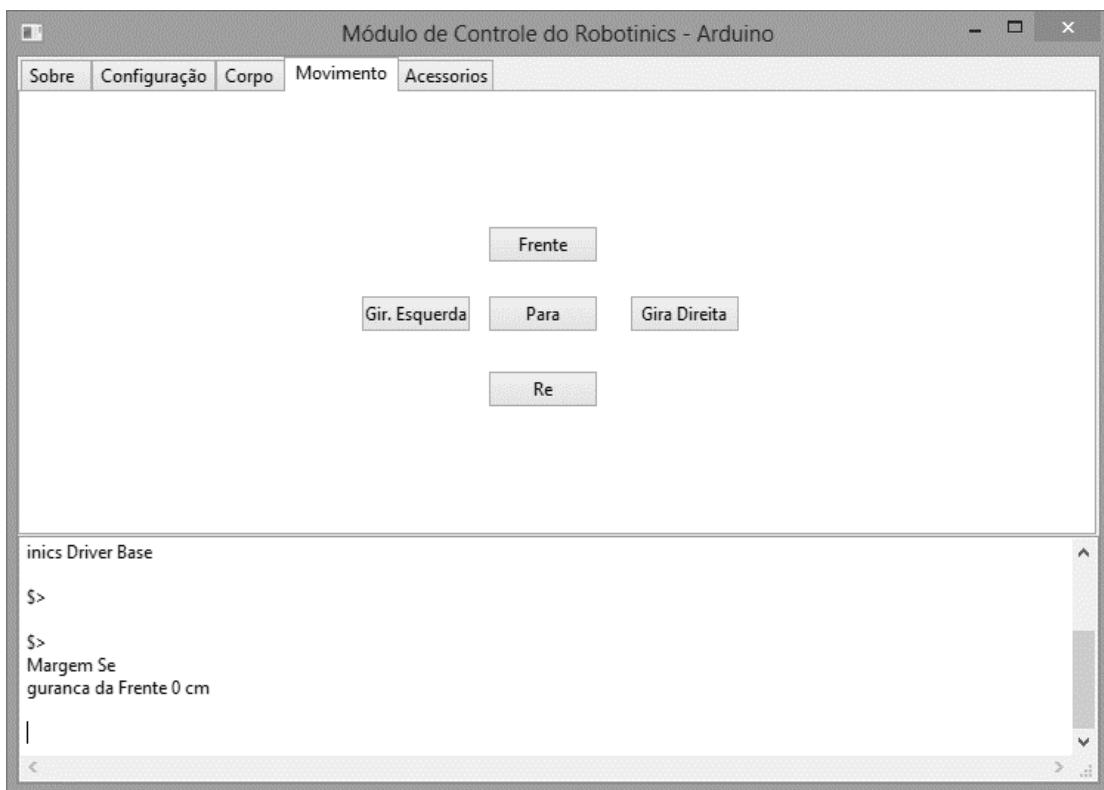


Figura 95 - Visualização do comando realizado

Desta forma é possível realizar todas as operações do robô.

4. Montagem do Corpo inferior

Agora que já possuímos uma base do nosso robô, precisamos agregar um tronco a ele.

O tronco permitirá posicionar alguns elementos que não estavam posicionados no capítulo anterior, mas que utilizamos.

4.1 Mecânica

Conectaremos alguns elementos construídos no SolidWorks e agregaremos ao nosso robô.

Serão duas peças agregadas, sendo o formato final do robô como apresentado.

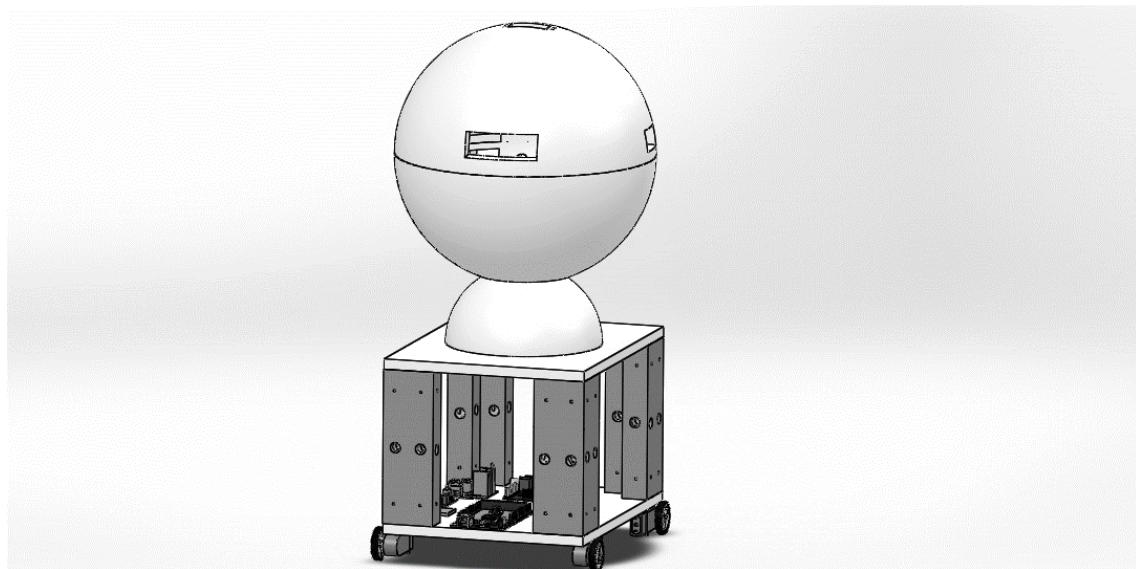


Figura 96 - Visão das peças já montadas no robô

Nota do autor:

Neste desenho pode ser visto um suporte diferente do apresentado na inicial.

Este suporte foi incluído nos arquivos, mas retirado e mantido os de metal por questões estéticas.

Criação das placas eletrônicas no SolidWorks

Uma das atividades mais importantes em projetos eletromecânicos é a montagem dos elementos mecânicos e não mecânicos.

O SolidWorks possui um conjunto de ferramentas que proporcionam a integração com ferramentas eletrônicas, permitindo que criemos visões das PCBs já montadas,

Conseguindo assim a visão da montagem física, auxiliando o montador na execução da atividade de montagem.

Estaremos abordando neste capítulo além deste detalhamento, também a montagem do corpo inferior.

Base pequena

A base pequena inferior é a estrutura que é ligada à base do robô e ao corpo.

Este elemento de fixação estrutural acopla o sensor de ultrassom traseiro, porém a principal função deste é ser base de fixação e suporte para as demais peças.

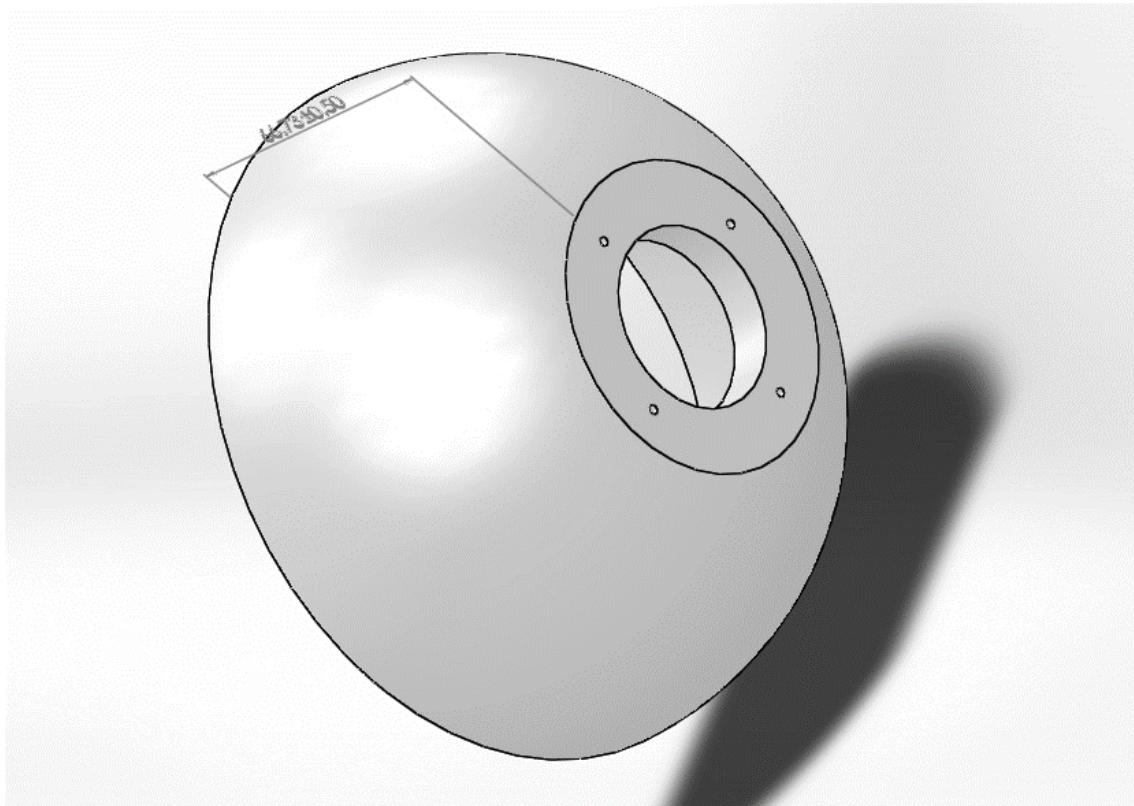


Figura 97 - Visão da peça no SolidWorks

Esta base ainda não contempla isso, mas em um futuro próximo poderá ser fixada sobre um eixo rolamentado, permitindo assim movimentos rotacionais, rotação do robô, com base fixa.

São características da base pequena inferior:

- Chanfro na base para passagem de fios
- Furos para fixação na base de acrílico
- Chanfro para fixação do sensor ultrassom traseiro
- Chanfro para fixação de conector elétrico 12 V

Arquivos e Modelos

Aqui estão contidos os arquivos originais da estrutura em SolidWorks e os respectivos arquivos para impressão.

Arquivo SolidWorks: robotinics\solidwork\ basePequenaInf.SLDprt

Arquivo de Impressão: robotinics\impressora3d\ BasePeqInf.STL

Corpo Inferior

A base grande inferior faz parte do corpo do robô.

No corpo do robô são fixados diversos componentes eletrônicos, tais como o Arduino e diversas Shields.

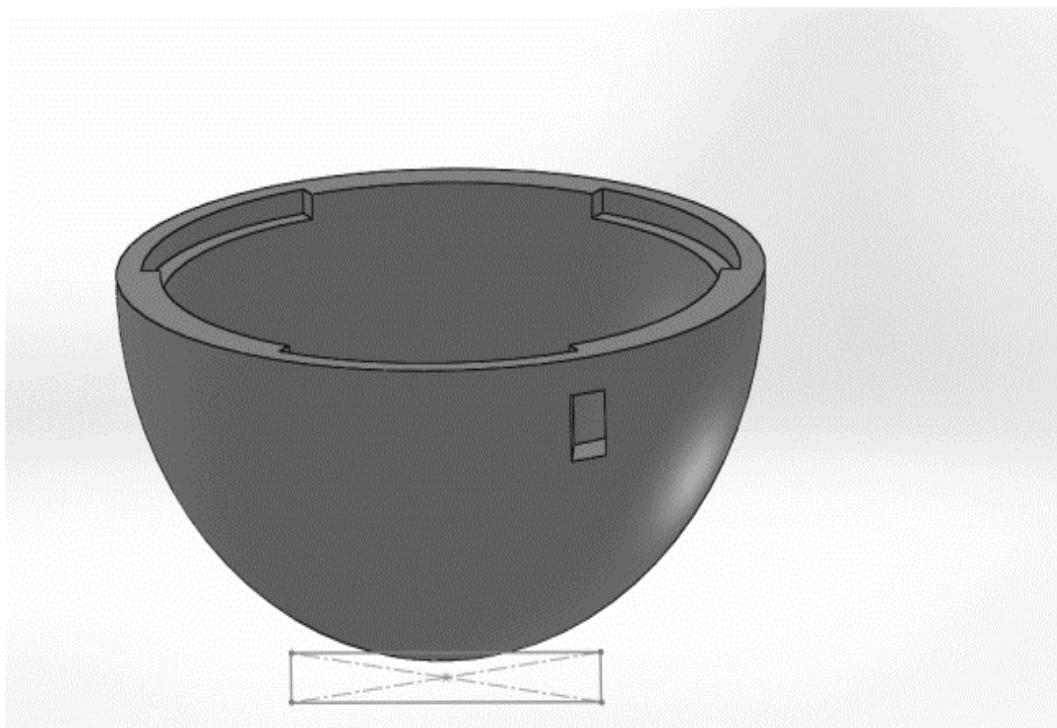


Figura 98 - Visão do corpo inferior, incluindo chanfro da tomada

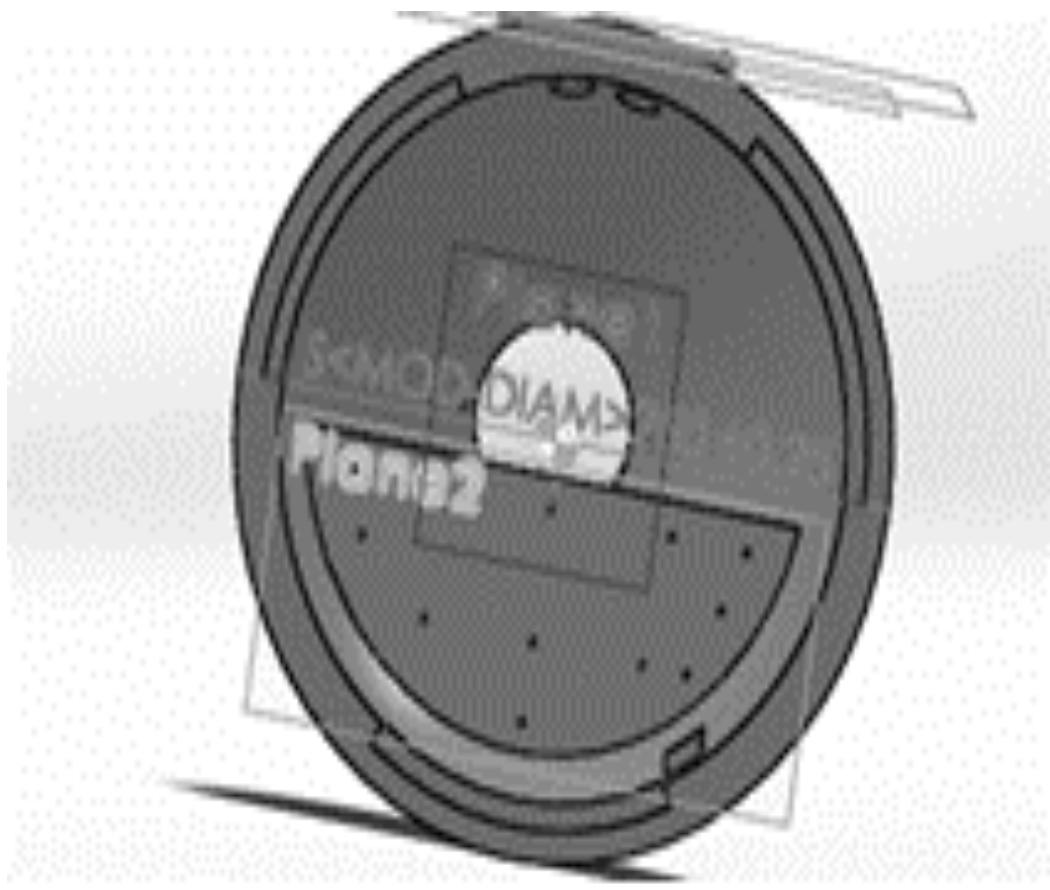


Figura 99 - Visão do Suporte com a base de fixação do arduino

Arquivos e Modelos

Aqui estão contidos os arquivos originais da estrutura em SolidWorks e os respectivos arquivos para impressão.

Arquivo do SolidWorks: robotinics\solidwork\CorpoInf.SLDPR

Arquivo de Impressão 3D: robotinics\impressora3d\CorpoInf.STL

Corpo Superior

O corpo superior é a base de sustentação de quase todos os Servo motores.

O corpo superior permite a fixação dos braços e da cabeça do robô, bem como a passagem de todos os fios para o corpo para às extremidades do robô.



Figura 100 - Corpo superior

Arquivos e Modelos

Aqui estão contidos os arquivos originais da estrutura em SolidWorks e os respectivos arquivos para impressão.

Arquivo do SolidWorks: robotinics\solidwork\CorpoSup.SLDPRT

Impressão 3D: robotinics: \impressora3d\CorpoSup.STL

Montagem

Na montagem as peças serão encaixadas de forma a fixar as mesmas.

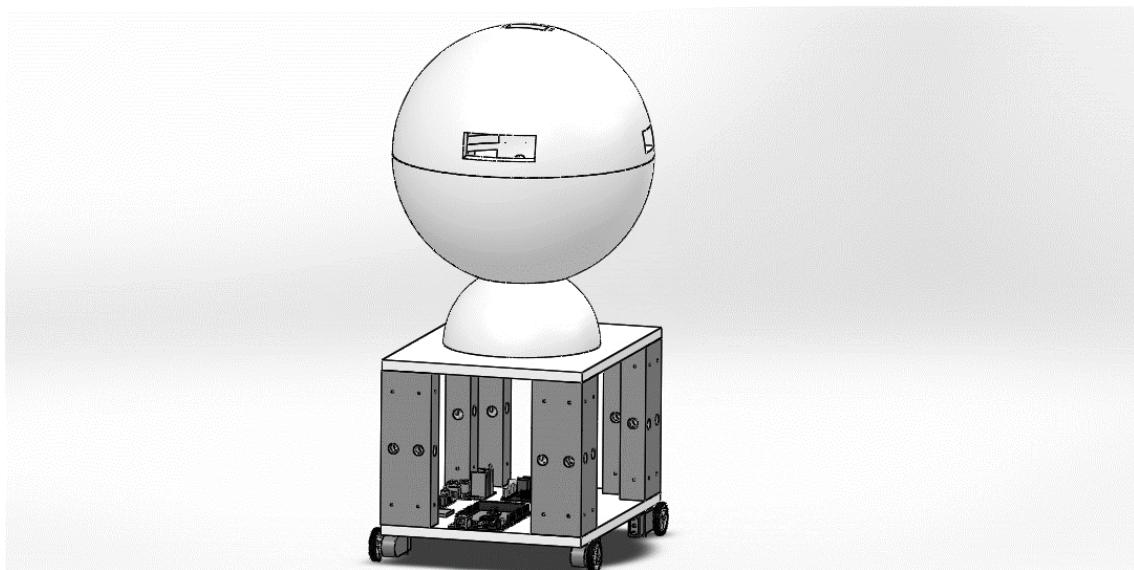


Figura 101 - Visão do robô montado até o final deste capítulo

4.2 Eletrônica

A eletrônica desta etapa começa a se tornar um pouco mais complexa do que foi nos módulos anteriores. Agora começaremos a ligar os fios, fazendo a montagem de diversas Shields (sensores do Arduino) que até o presente momento não foram contempladas nem mencionadas em nosso projeto.

Bluetooth

O módulo bluetooth apresenta a possibilidade de integração entre diversos equipamentos compatíveis, tais como:

- Celulares
- Tablets
- Computadores

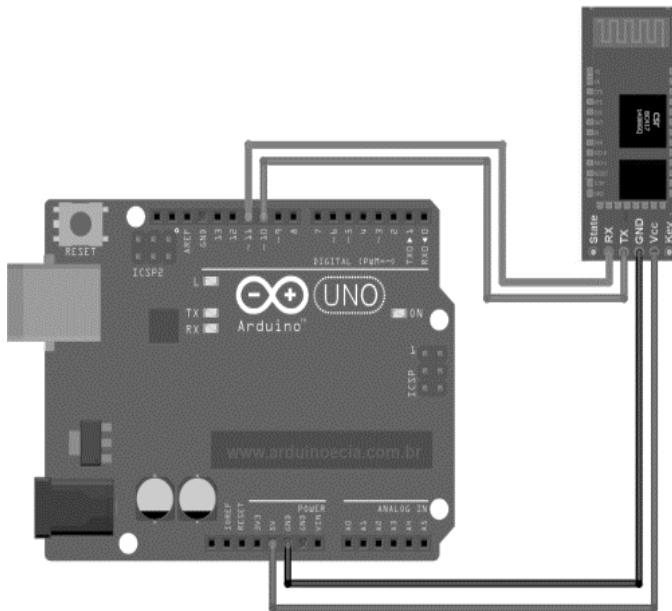


Figura 102 - Visão das ligações elétricas

A conexão a com o Arduino é muito simples. O bluetooth trabalha com comunicação TTL, ou seja, TX e RX. Desta forma você tem um fio para transmitir dados (TX) e outro para receber (RX). A alimentação é ligada diretamente no Arduino, conforme ilustração.

Caso seja necessária ligação externa, a tensão necessária para o funcionamento é 5 V.

Utilizaremos o HC-05.

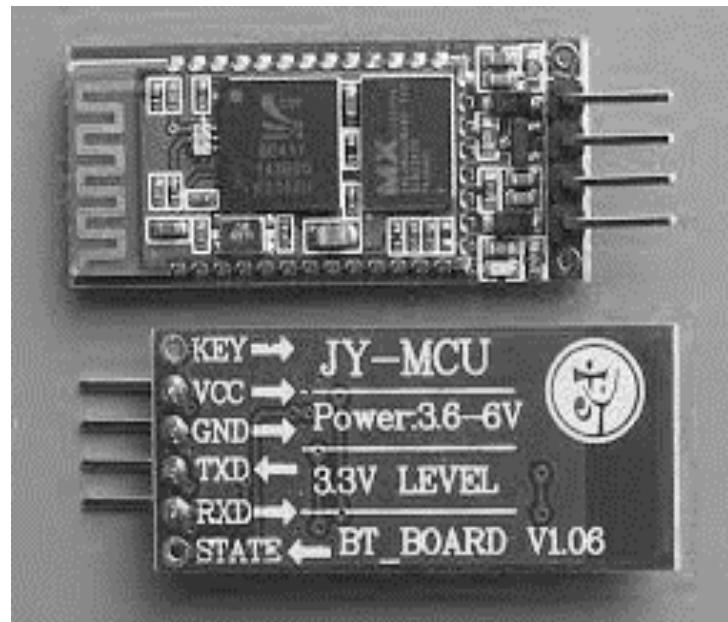


Figura 103 - - Imagem de um Bluetooth

Características:

- Permite enviar e receber dados TTL através da tecnologia Bluetooth
- Funciona com qualquer adaptador Bluetooth USB
- Cobertura de sinal de aproximadamente 10 m
- Configurável via comando AT

LCD 16x2 I2C

O LCD 16x2 apresenta, entre as vantagens, baixo consumo elétrico e ótimo custo-benefício.

O Padrão I2C permite o acoplamento de múltiplos dispositivos através de um único barramento, modificando apenas o endereço de comunicação.

A comunicação através do I2C se dá por dois fios, diminuindo em muito a passagem de trilhas e circuitos. O controlador I2C é soldado diretamente ao LCD, reduzindo o custo e facilitando o processo construtivo.

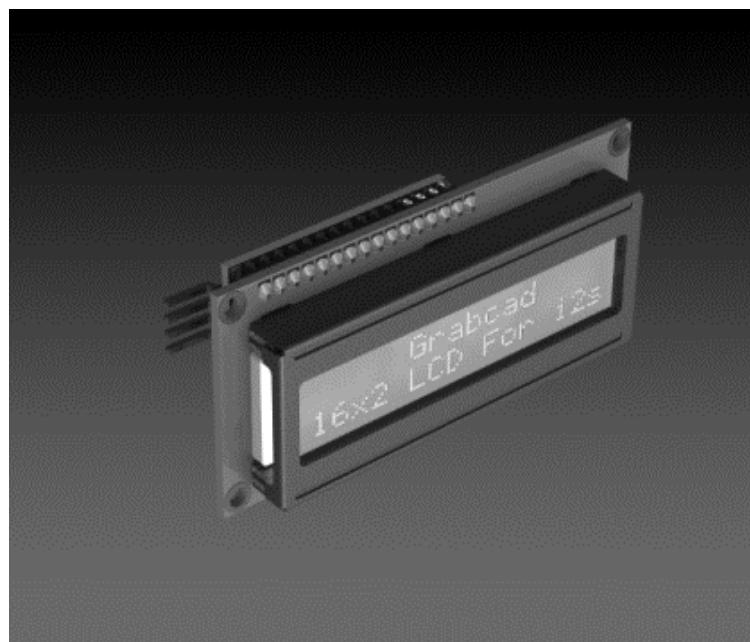


Figura 104 - Visão do LCD 16x2

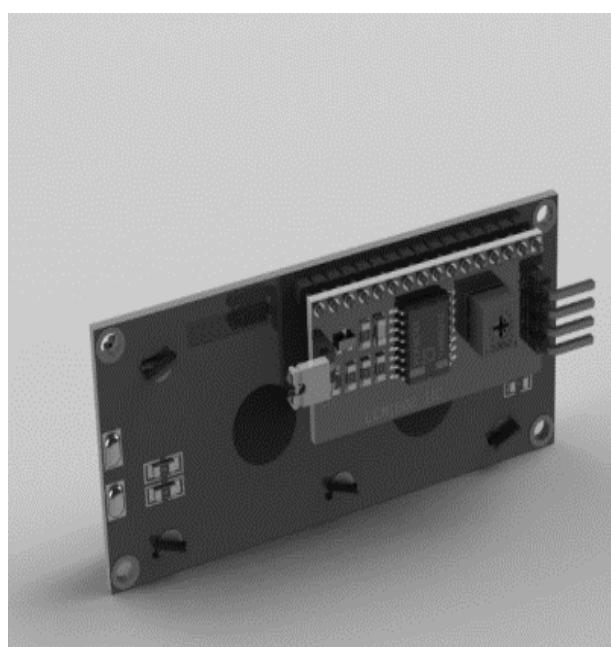


Figura 105 - Visão da traseira do LCD I2C

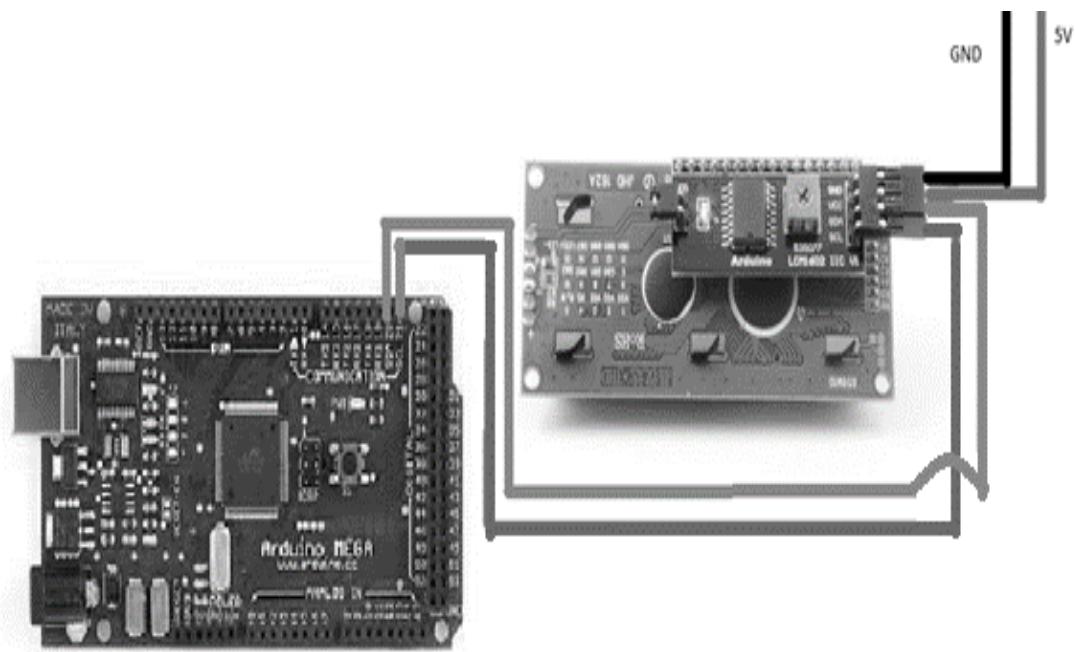


Figura 106 - Esquema elétrico do Arduino com LCD

Exemplo de I2C com Arduino

A biblioteca LiquidCrystal_I2C.h é responsável por fornecer as funções para permitir a comunicação com o LCD. Sem esta biblioteca o uso do dispositivo no Arduino fica mais difícil.

O download desta biblioteca pode ser conseguido pelo link abaixo:

<https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library>

O exemplo a seguir demonstra o uso da biblioteca LiquidCrystal_I2C.h, que permite a comunicação com o Arduino.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define BACKLIGHT_PIN 13

LiquidCrystal_I2C lcd(0x20); // Indica Endereço I2C
void setup()
{
    // Switch on the backlight
    pinMode ( BACKLIGHT_PIN, OUTPUT );
    digitalWrite ( BACKLIGHT_PIN, HIGH );

    lcd.begin(16,4);      //Iniciando o LCD Informa 16 colunas x 4 linhas
    lcd.home ();          // Ir para Inicio da Tela, SetCursor(0,0);
    lcd.print("Olá ");
    lcd.setCursor ( 0, 1 ); // Indo para segunda linha
    lcd.print (" Robotinics ");
}

void loop()
{
```

Servo motor

Como funciona um Servo motor

O Servo motor nada mais é do que um motor DC acoplado a um potenciômetro.

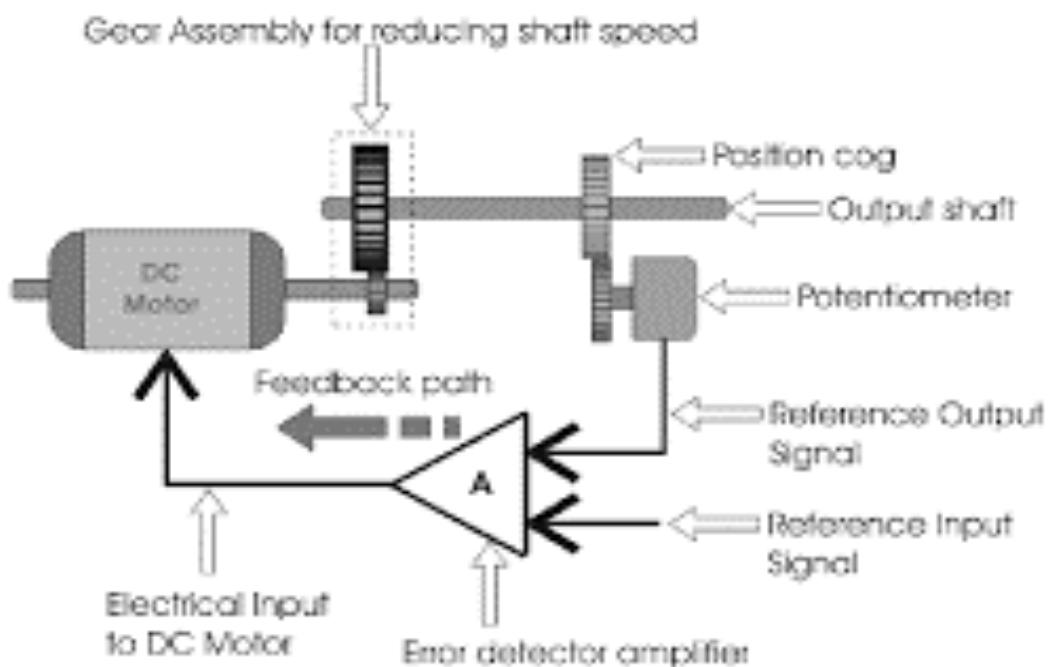


Figura 107 - Visão de perfil de um servomotor

O motor, ao girar, também movimenta o potenciômetro, que, por sua vez, gera uma mudança na resistência, que é medida por um CI que interpreta a variação da resistência, criando uma identificação resistência x posição.

O CI do Servo motor associa a resistência medida a um sinal PWM. O PWM é a tradução de Pulse Width Modulation, ou Modulação de Largura de Pulso. O Arduino tem a capacidade de gerar sinal PWM, mas não em todos as portas – apenas em portas específicas.

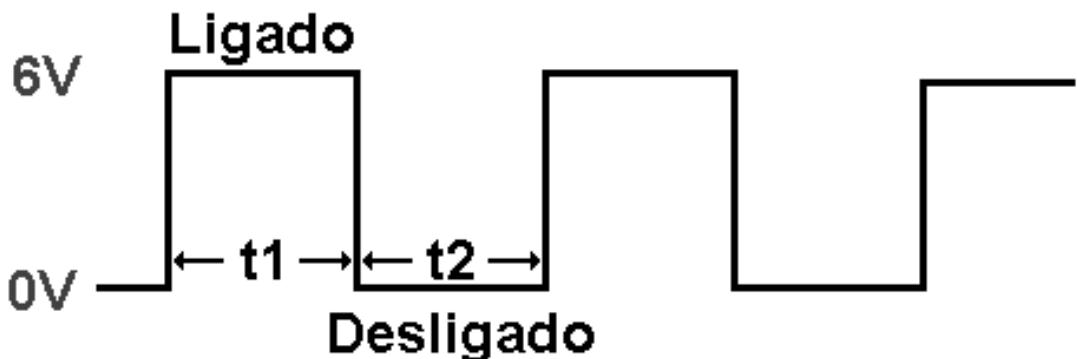


Figura 108 - Exemplo de Sinal PWM

Quanto maior o sinal PWM, maior será a largura de t_1 .

O Arduino Mega fornece 15 portas PWM:

- Digital 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
- Digital 44, 45, 46

O Servo motor, de forma geral, pode oferecer rotação de 180 ou 360 graus. Os mais comuns são os modelos de meia volta ou 180 graus de rotação.

Servo Tower Pro Mg995 Digi Hi-Speed – 15 kg

O Servo motor tem função de movimentar as estruturas mecânicas, como braços e garras.



Figura 109 - Servo motor de 15 kg

Especificação Técnica

Servo Tower Pro MG995 Digi Hi-Speed –

- Velocidade 0,13 s
- Peso 55 g
- Torque 13 kg-cm 4,8 V / 15 kg-cm 6,0 V
- Velocidade: 0,17 seg 4,8 V / 0,13 seg 6,0 V
- Voltagem: 6 V
- Roletado
- Engrenagens: metal
- Peso: 55 g
- Medidas: 40 x 20 x 43 mm

O Servo motor opera com tensões de 5 V no robô, apresentando giro de 180 graus.

O Servo motor opera com sinal PWM, sendo gerido pelo Arduino em uma das portas PWM⁹ existentes.

Todo Servo motor tem três fios. As cores podem variar de modelo para modelo, porém, nos modelos utilizados, as cores são:

- Marron – GND
- Vermelho – 5 V
- Laranja – PWM

⁹A modulação por largura de pulso (MLP) - mais conhecida pela sigla em [inglês](#) PWM (Pulse-Width Modulation) - de um sinal ou em [fontes de alimentação](#) envolve a modulação de sua [razão cíclica](#) (duty cycle) para transportar qualquer informação sobre um canal de comunicação ou controlar o valor da alimentação entregue à carga.

Tower Pro MG996R Digital Metal Servo

O servomotor MG996R é um servomotor com capacidade de torque de 10 kgf,

Tower Pro MG996R Digital Metal Servo

Especificações técnicas



Figura 110 - Servomotor de 10 kg

Modulação PWM: Analógica

Torque:
4.8 V:
130.5 oz-in (9.40 kg-cm)
6.0 V:
152.8 oz-in (11.00 kg-cm)

Velocidade:
4.8 V:
0.17 sec/60°

	6.0 V: 0.14 sec/60°
Peso:	1.94 oz (55.0 g)
	Comprimento: 1.60 in (40.6 mm)
	Largura: 0.78 in (19.8 mm)
Dimensão:	Altura: 1.69 in (42.9 mm)
	Massa: 55g
Tipo de Engrenagens:	Metal
Suporte à Rotação:	Dual Bearings

Algumas informações estão em Inglês, pois foram coletadas de datasheets.

Placa de Controle dos Servos Motores

A placa de controle dos servos motores tem por responsabilidade o fornecimento elétrico e sinais PWM¹⁰ do Servo motores.

Esta placa será utilizada no robô para ser ligada aos Servo motores.

Serão necessárias três placas para controlar o robô. Cada placa deve ser conectada em um lado do robô, e a terceira no pescoço ou abaixo do pescoço.

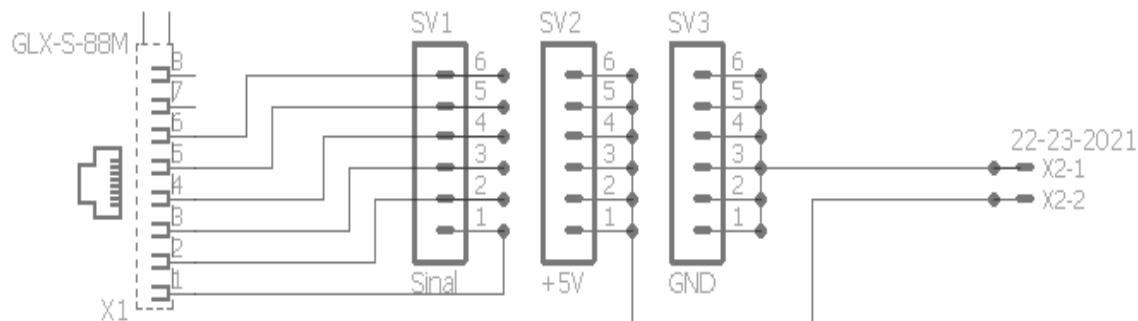


Figura 111 - Esquemático da placa controladora de Servo motores

¹⁰PWM – A modulação por largura de pulso (MLP) – mais conhecida pela sigla em [inglês](#) PWM (Pulse-Width Modulation) – de um sinal ou em [fontes de alimentação](#) envolve a modulação de sua [razão cíclica](#) (duty cycle) para transportar qualquer informação sobre um canal de comunicação ou controlar o valor da alimentação entregue à carga (Wikipédia).

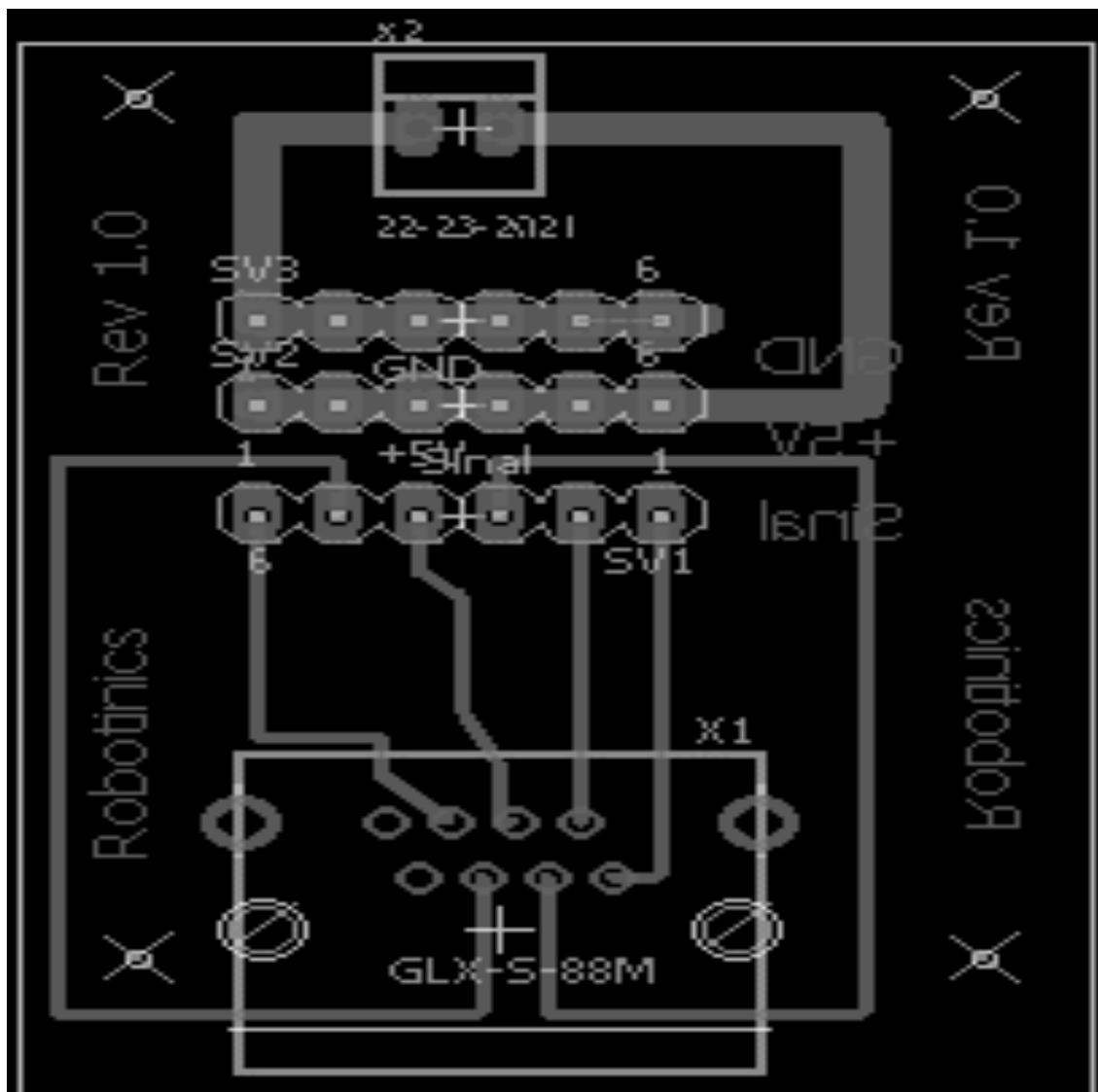


Figura 112 - Visão do layout da placa

Arquivos:

Espelho para Fabricação: /pcb/Espelho_placa02.bmp

Arquivo PCB Wizzard: /pcb/Placa Controle Servo Motor.pcb

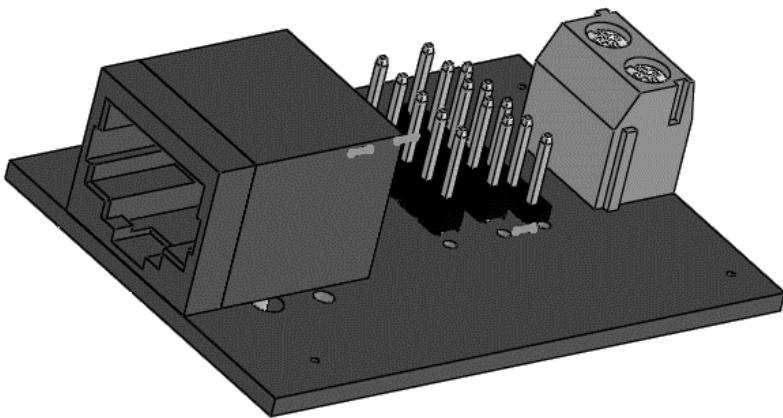


Figura 113 - Vista da placa de controle dos Servo motores

A placa controladora de Servo motor transmite os sinais através de um conector tipo RJ45. Outra placa, igual a esta, recebe os sinais do Arduino, que, por sua vez, se comunica através de um cabo de rede.

A vantagem nesta solução reside na facilidade de clipagem (montagem do cabo) e pela flexibilidade dele quando movimentado.

A lista de componentes utilizados nesta placa e nas demais será apresentada no Capítulo 5, onde discorremos sobre os elementos de ligação e cabos.

Placa controladora de LED

A placa controladora de LED fornece conexão elétrica para os LEDs do robô.

Os LEDs operam em especificações diferentes da maioria dos outros componentes do robô.

Os LEDs queimam quando ficando em correntes superiores a 0,3A, devendo, então, ser limitados pelo uso de resistores.

Para cálculo do valor de resistor, utiliza-se a seguinte fórmula:

$V = R \cdot I$, onde:

V é a tensão aplicada – no nosso caso, $V = 5$ Volts

R = Resistência desejada ou que se pretende utilizar

I = Corrente suportada pelo LED – a corrente de operação

A tabela abaixo apresenta o cálculo de corrente:

Côr	Queda de Tensão	Corrente Máxima
Vermelho	1.8 V	0.02 A
Verde	2.1 V	0.02 A
Amarelo	2.0 V	0.015 A
Laranja	2.0 V	0.02 A
Azul	3.1 V	0.02 A
Branco	3.1 V a 4.0V (depende do fabricante)	0.02 A
Infra-vermelho	1.1 V	0.02 A

Figura 114 - Tabela de corrente de LEDs

Desta forma, utilizando um LED verde de 0,02A:

$$V = R \cdot I \Rightarrow 5V = R \cdot 0,02 \Rightarrow R = 5/0,02 \Rightarrow R = 250\text{ohms}$$

Utilizamos um resistor de 330 ohms, pois o valor da corrente gerado nesse resistor será:

$$V = R * I \Rightarrow 5 = 330 * I \Rightarrow I = 5 / 330 \Rightarrow I = 0,15A$$

Desta forma, podemos utilizar qualquer cor de LED no projeto e a diferença de brilho não será expressiva.

Visão da Placa no Eagle, com todos os layers.

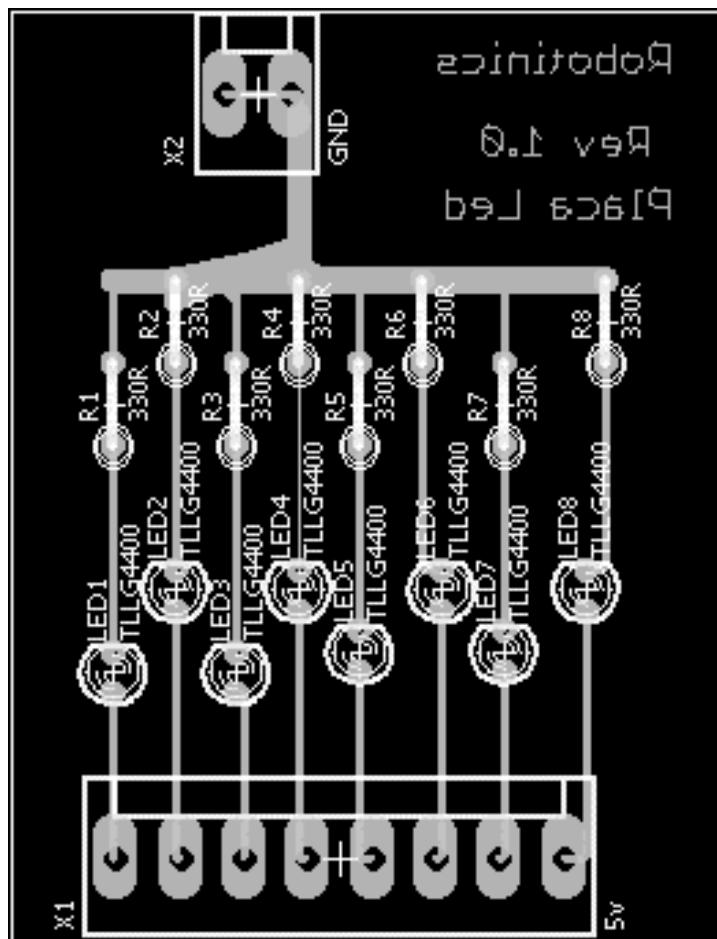


Figura 115 - Placa LED

A visão de como gerar a placa de circuito impresso no SolidWorks é apresentada no tópico a seguir.



Figura 116- Visão da placa LED consolidada no SolidWorks

LCD Touch Screen

A tela permite a visualização das informações passadas na CPU, criando uma fácil e rápida visualização do ambiente pelo usuário.

A tela também permite gerenciar as opções do robô através de controles touch screen.

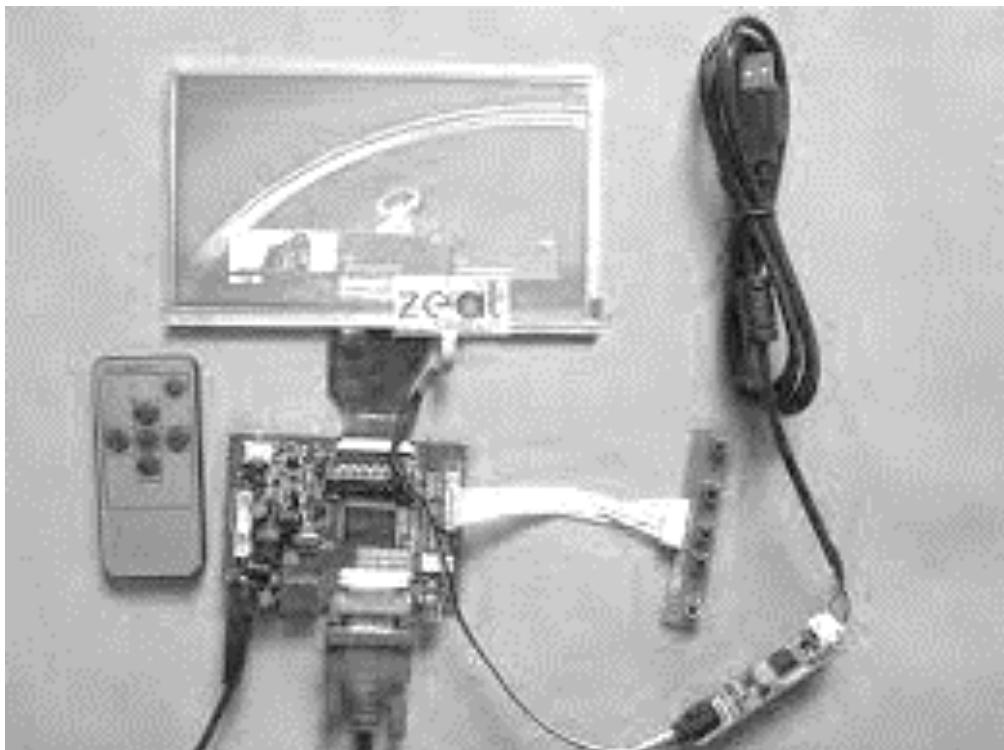


Figura 117 - Monitor LCD touch screen

A escolha da tela ou não é uma questão de projeto. No caso do Robotinics, a interface foi escolhida sobre dois aspectos:

1. Facilidade em interação, inclusive desenvolvimento
2. Melhora visual no projeto

A partir do momento em que se determinou que se construa o robô com uma tela touch (terminal) acoplado a ele, a próxima escolha é quanto à determinação do modelo de terminal.

Foram parâmetros determinísticos para seleção do modelo:

- Capacidade touch screen
- Compatibilidade com Linux
- Conector RCA e HDMI
- Ser destinado a embarcados, apresentando apenas as placas e circuitos.
- Ter níveis de tensão de alimentação coerentes com o projeto (5 V até 12 V)

A escolha desta placa foi acertada, pois apresenta todas as características desejadas.

Elementos do Terminal

O terminal é constituído por três partes:

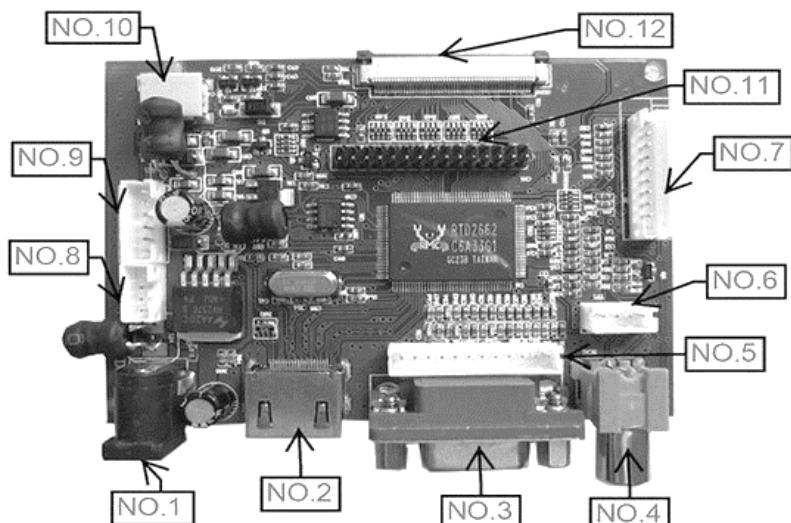
1. Placa controladora do LCD
2. LCD
3. Controlador Touch Screen USB

Descreveremos apenas a placa controladora, pois os demais componentes são satélites a esta placa e, desta forma, são facilmente entendidos.

Placa Controladora do LCD

Gerencia as funções da tela, permitindo e fornecendo todos os recursos necessários para a ligação e visualização das imagens no LCD. Sem a placa controladora do LCD, não é possível a gestão deste.

A placa possui diversos recursos, abordados na ilustração a seguir:



No.	Description	No.	Description
1	Power input	7	Key board &IR &LED indicator
2	HDMI signal Input	8	Power input
3	VGA signal input	9	Power and control signal for extra inverter board
4	AV1 signal input	10	Backlight voltage output connector
5	Extra VGA signal input	11	LVDS signal output connector
6	Extra AV1 and AV2 signal input	12	TTL signal output connector

Figura 118 - Diagrama da placa controladora LCD

Descrição dos itens:

- LCD 7 polegadas, resolução 800*480

- Entradas HDMI, VGA, 2 AV
- Controller Board VS-TY2662-V1
- 5 botões de funções
- Película touch screen
- Controlador touch screen USB
- Controle remoto

Integração Eagle SolidWorks

O Eagle é a ferramenta escolhida para gerar as placas eletrônicas customizadas em nosso projeto. O Eagle permite, em sua versão gratuita, criar placas de 10 cm por 10 cm, que é mais que suficiente para nossos projetos.

Neste livro não iremos explanar sobre a ferramenta Eagle da Cadsoft, porém mostraremos como integrá-la com o SolidWorks, permitindo que seja gerada a visão dos modelos 3D das placas.

Neste contexto, são necessárias duas etapas:

1. No Eagle, onde se exporta o arquivo necessário para ser utilizado no SolidWorks
2. No SolidWorks, onde se gera a peça que será visualizada, adicionando os detalhes e acertando as informações

Para explanar este processo, vamos utilizar o projeto da placa LEDs, que está disponível na pasta do projeto robotinics/eagle/placa leds/.

Etapa 1- Exportando placas no Eagle

Para exportar a placa gerada no Eagle, siga os seguintes passos:

1. Entre no Eagle e selecione o projeto Placa LEDs.
2. Abra o Eagle Board, dando duplo clique no item placaled.brd, conforme ilustração.

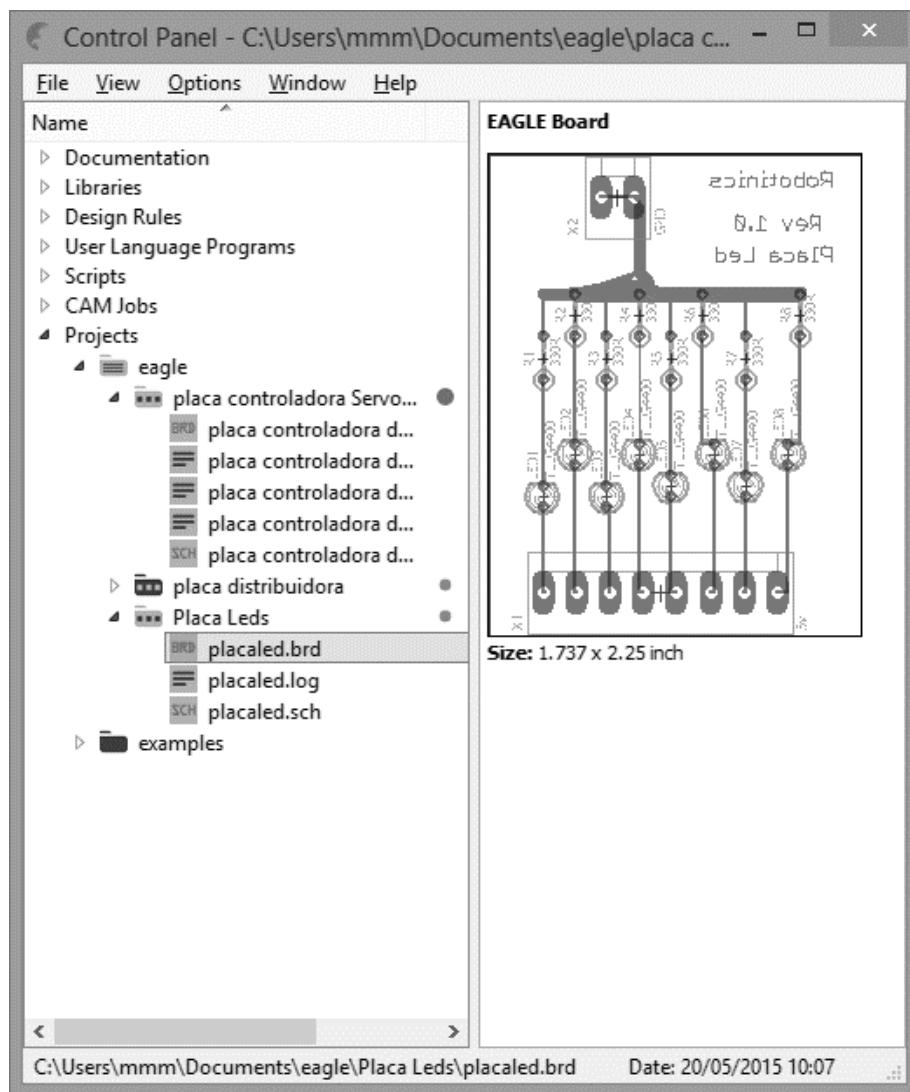


Figura 119- Software Eagle, da Cadsoft

3. No Eagle Board, selecione a opção File>Export> IDF.

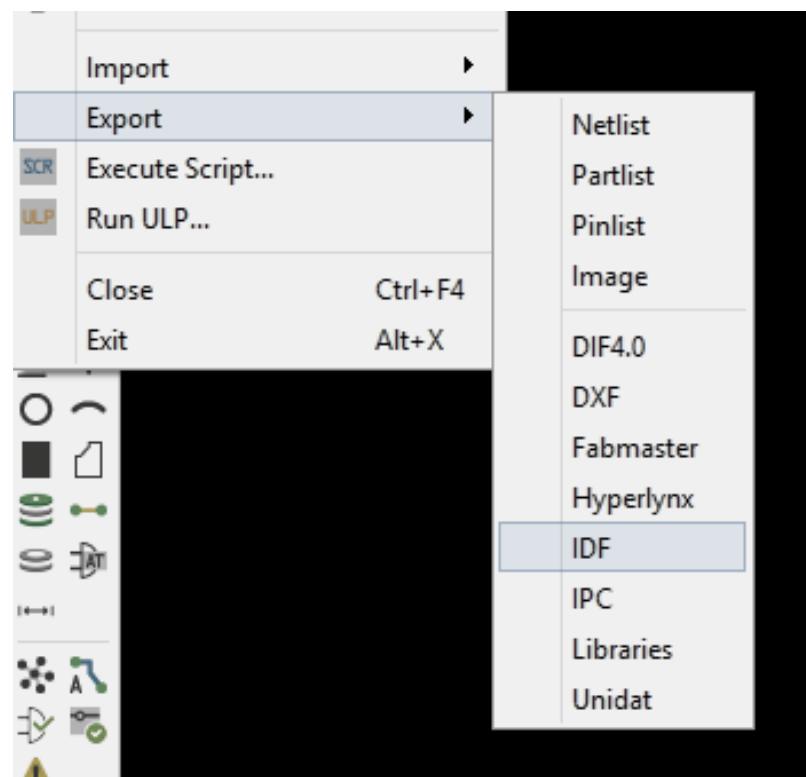


Figura 120- Gerando Exportação para SolidWorks

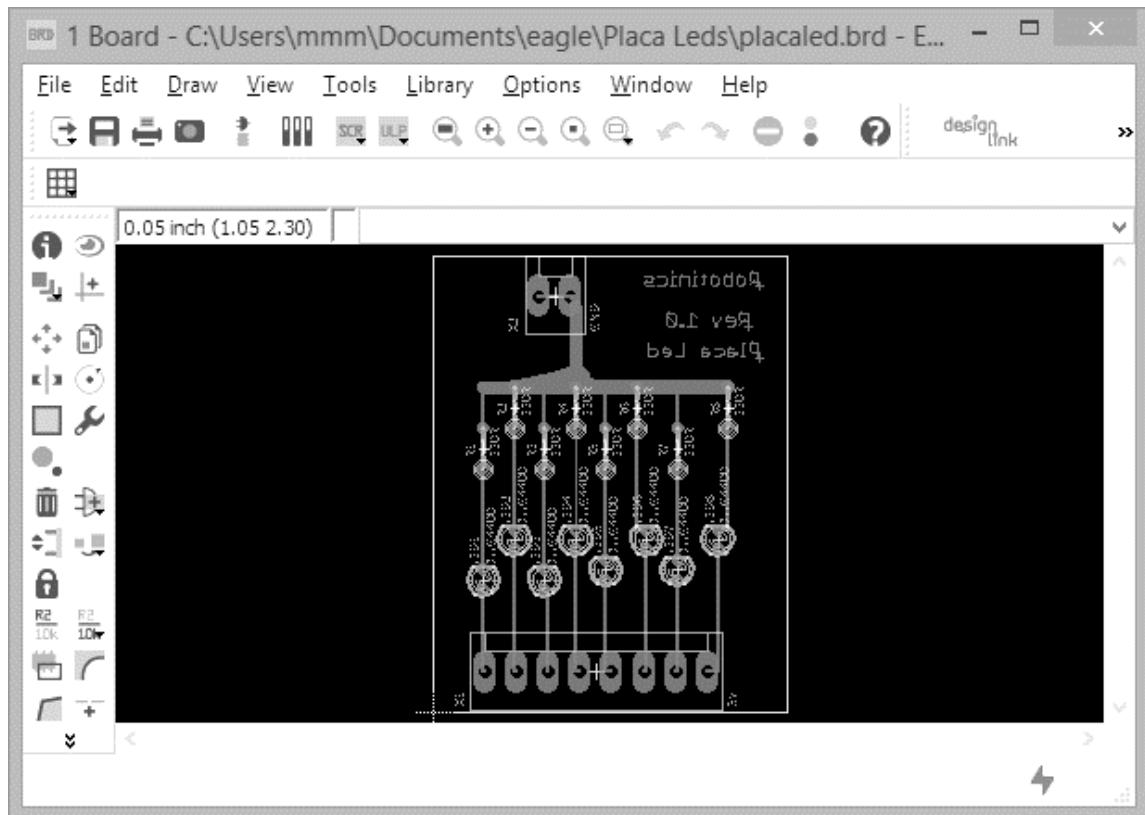


Figura 121 - Editor de placa Eagle Board

4. Após indicar a exportação para IDF, será necessário o acesso à internet, pois o software baixa uma série de arquivos para gerar o IDF.
5. Será chamada a tela IDF Exporter, que pergunta o padrão das trilhas geradas (default: 20mm) e board thickness: (default: 1,6mm), conforme apresentado na figura a seguir:

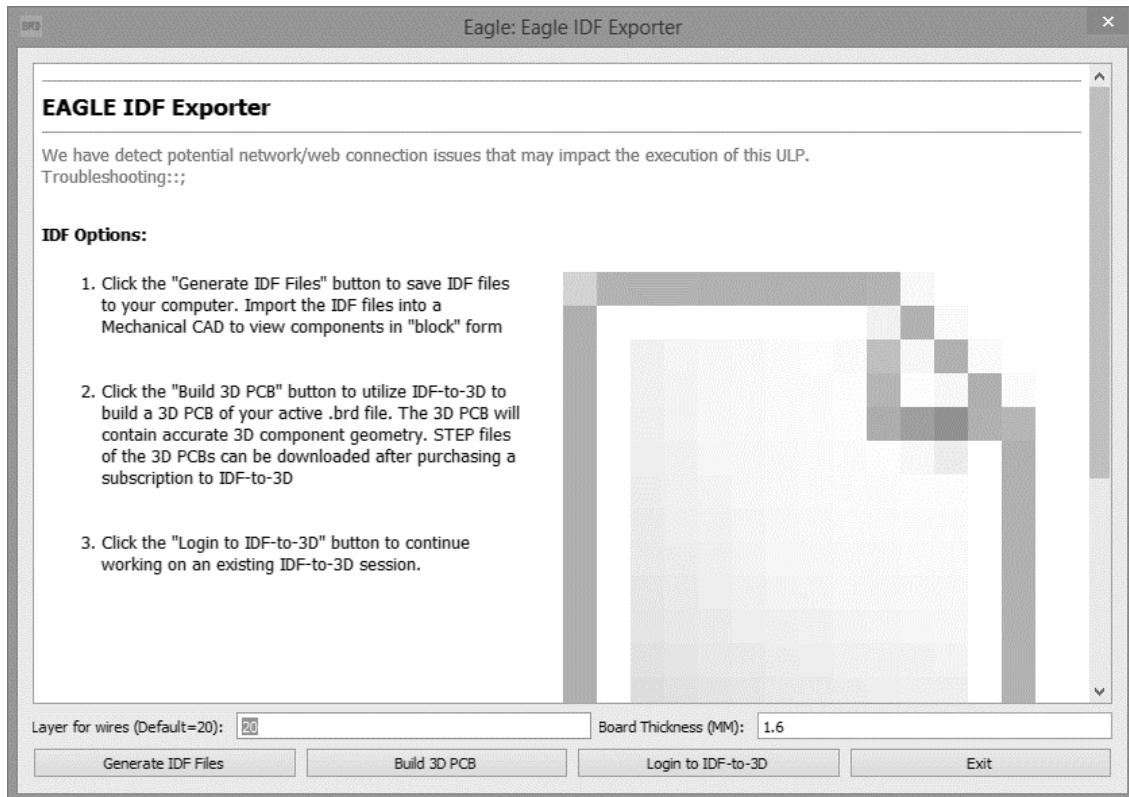


Figura 122 - Opções para geração de Exportação IDF

6. Selecione o botão Generate IDF Files
7. Indique o diretório em que será armazenado o arquivo exportado. Por definição, recomendamos criar uma subpasta Eagle e, dentro desta, outra subpasta com o nome da placa, no caso, placaleds.

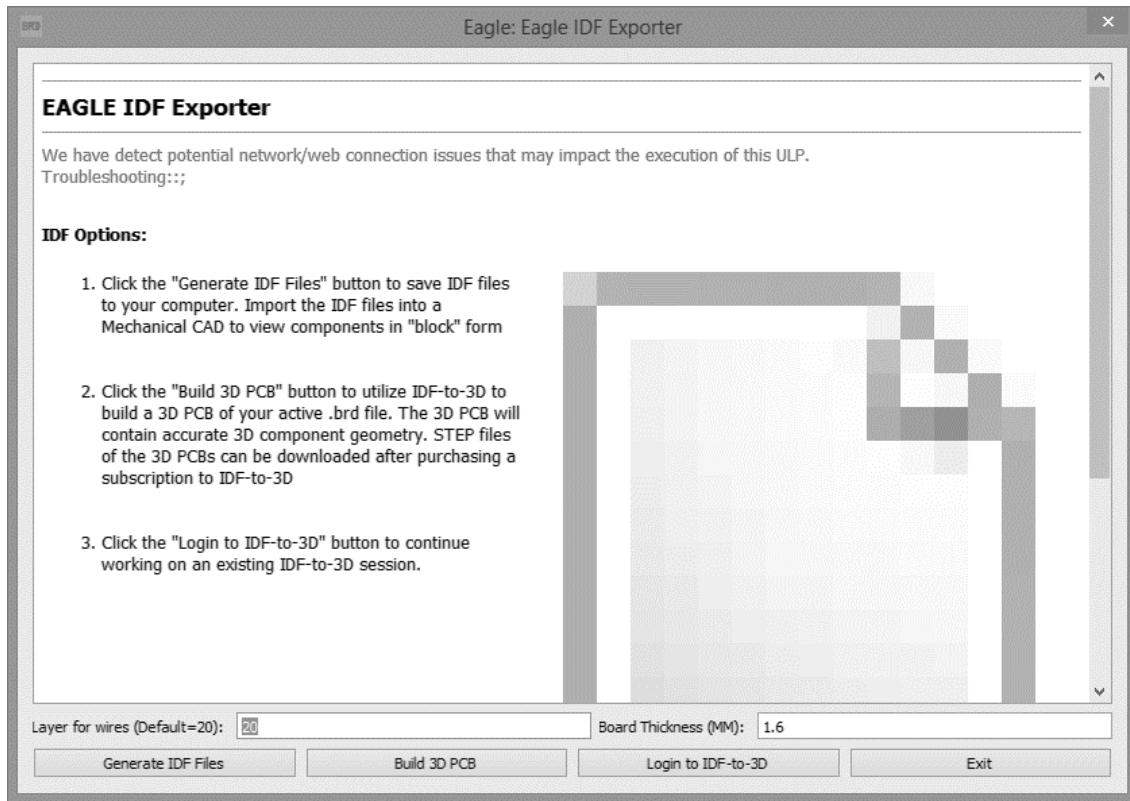


Figura 123 - Processando exportação do arquivo

8. Ao pressionar Selecionar Pasta, o processo de exportação é feito de forma automática.

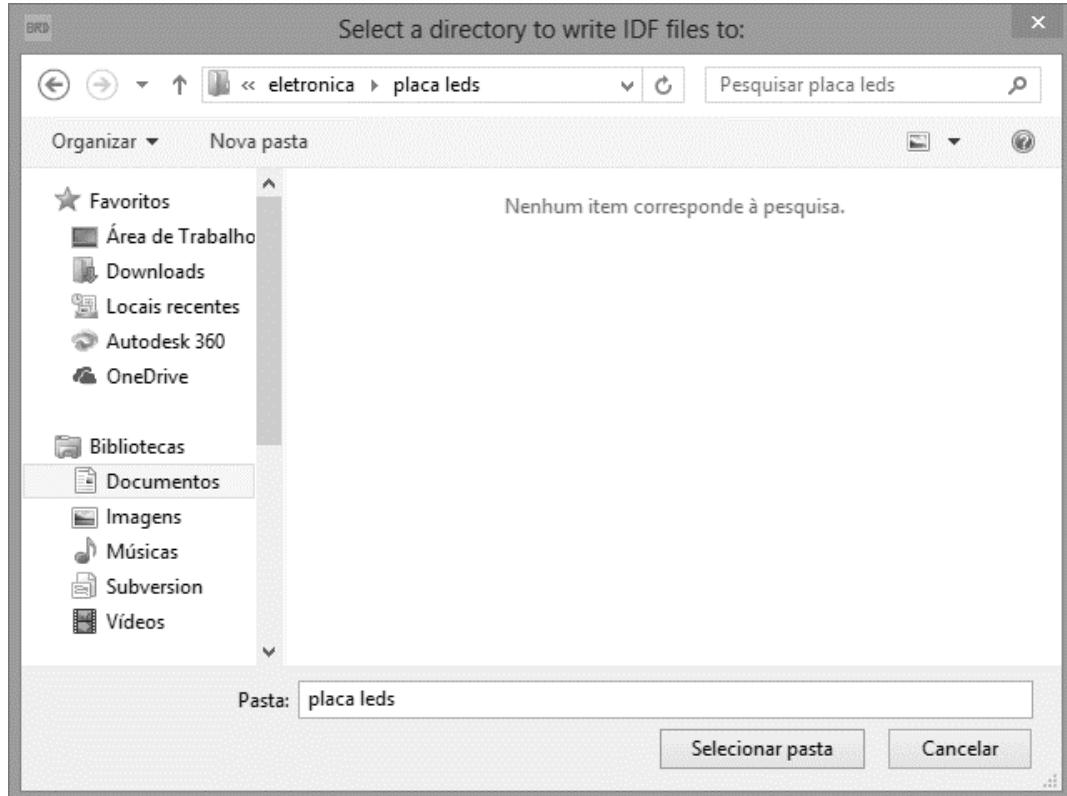


Figura 124 - Finalização da exportação para IDF

9. Ao salvar o arquivo, será indicado o local onde ele foi salvo. Anote, pois será necessário utilizar na próxima etapa.

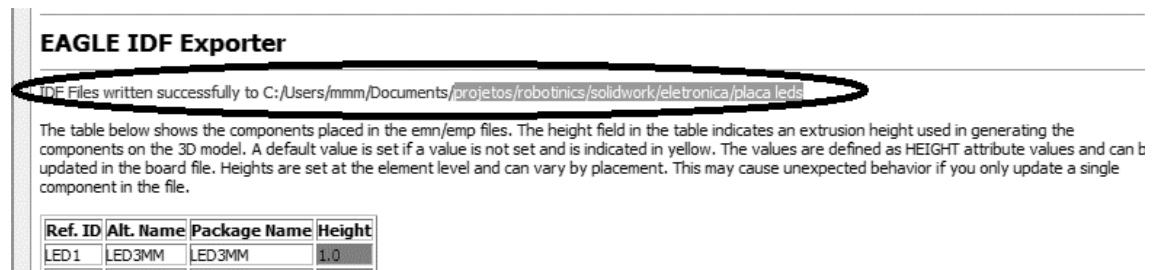


Figura 125- Local onde o arquivo foi salvo

Etapa 2 – Importação IDF no SolidWorks

Nesta etapa vamos criar o modelo 3D da placa que criamos e incluir no nosso projeto.

O SolidWorks utiliza uma ferramenta especial para montagem de placas eletrônicas – o CircuitWorks.

Adicionando o CircuitWorks

Para adicionar o CircuitWorks no seu menu, primeiramente entre em Tools > Add-ins.

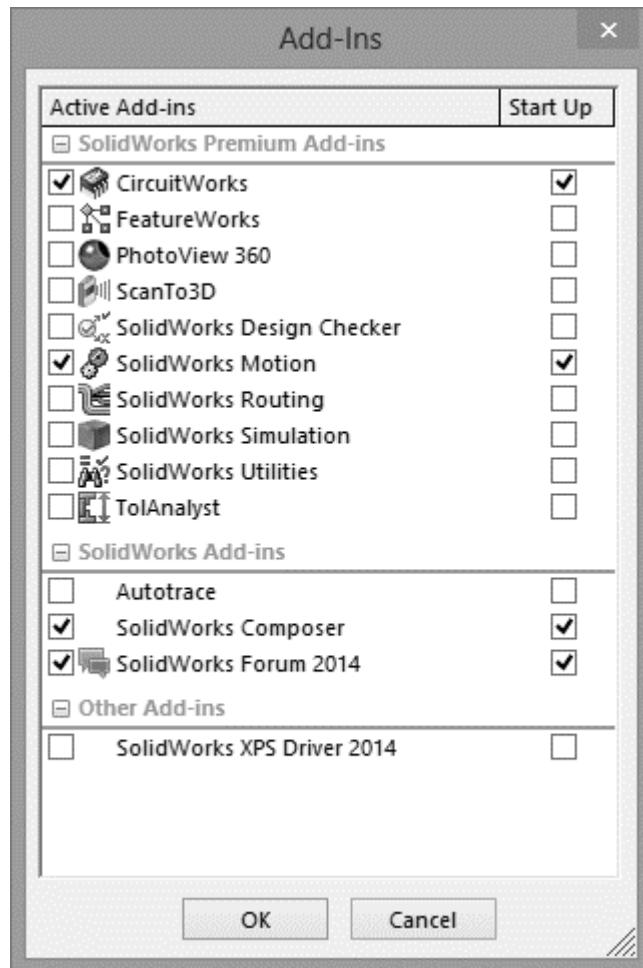


Figura 126 - Opções para adicionar Add Ins

Selecione o CircuitWorks no Active Add-ins, já incluindo também para o Start Up, ou seja, para que na próxima vez em que iniciar o SolidWorks esta ferramenta já esteja ativa.

Chamando o CircuitWorks

Ao realizar o processo de add ins, o menu CircuitWorks será incluído no menu de opções. Desta forma, aparecerá sempre como na figura abaixo:

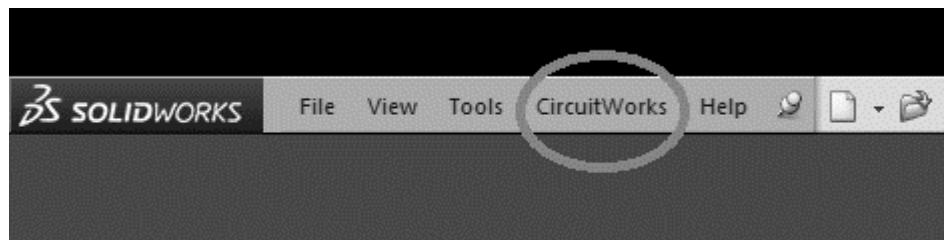


Figura 127- Menu com CircuitWorks

Entre no CircuitWorks, selecionando a opção Open ECAD File, conforme apresentado abaixo:

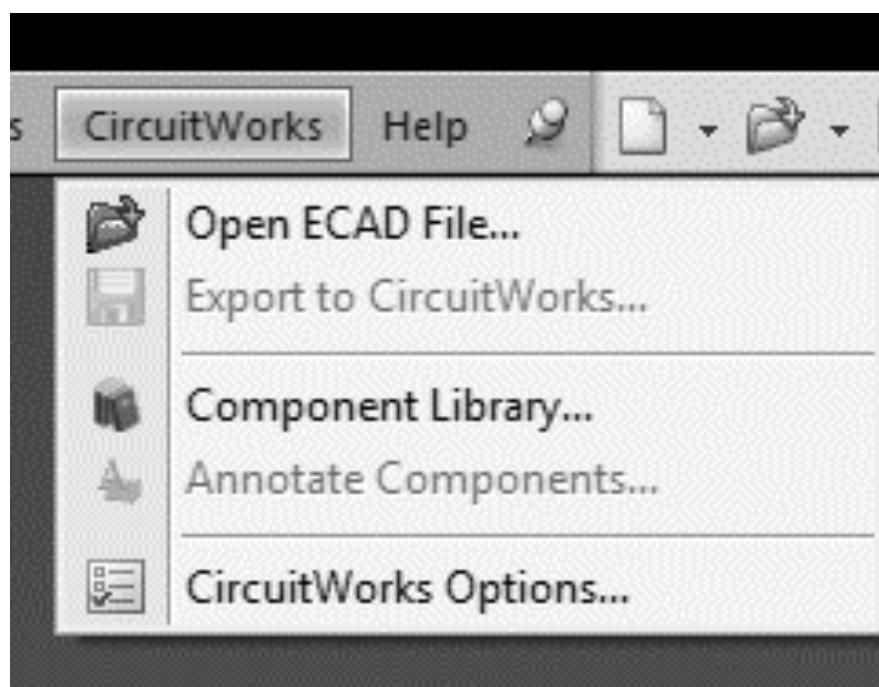


Figura 128- Open ECAD File

Agora o SolidWorks irá solicitar que indique o arquivo que quer importar.

É possível importar diversos padrões de ECAD, mas escolheremos o padrão IDF, que criamos no EAGLE. O arquivo selecionado no caso foi o placaled.emn.

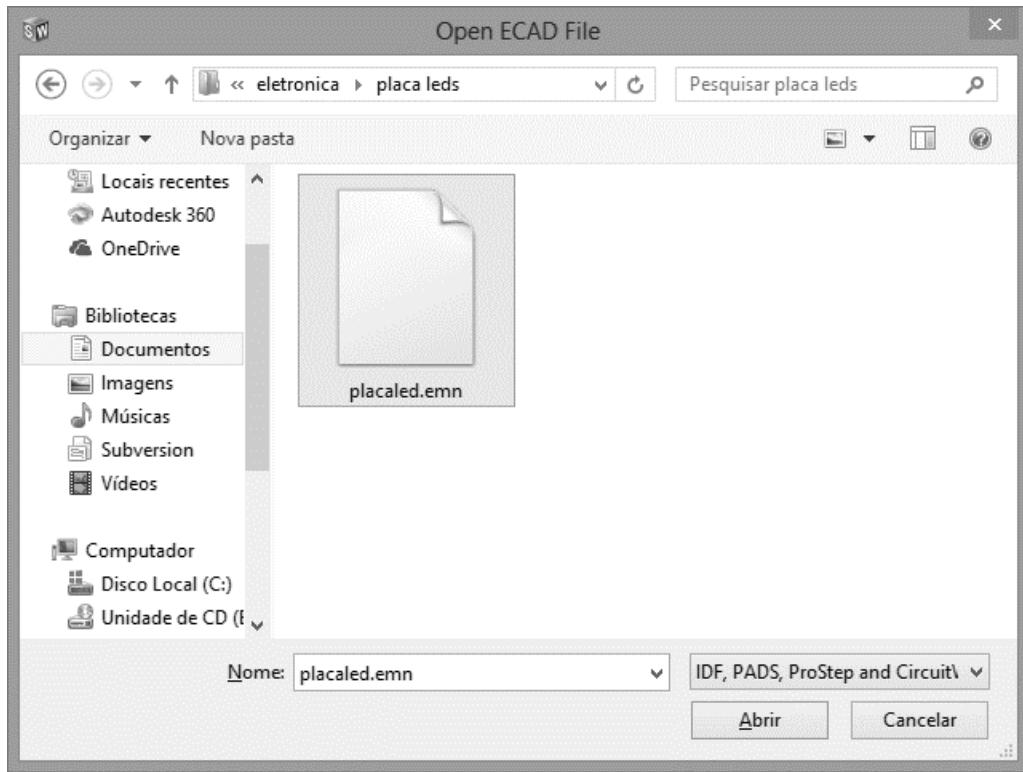


Figura 129- Arquivo placaled.emn

O CircuitWorks importa o arquivo e apresenta uma visão plana da placa, conforme imagem abaixo:

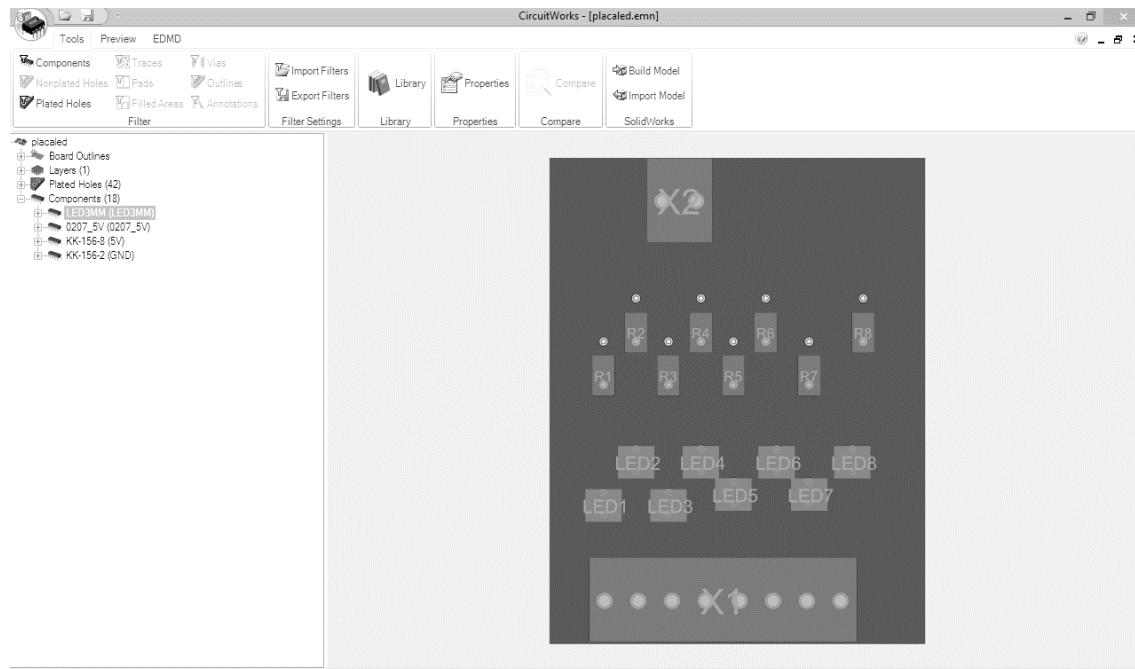


Figura 130- Visão da placa no CircuitWorks

Conforme apresentado na imagem anterior, os diversos componentes da placa são enumerados na árvore de componentes, ao lado esquerdo da tela.

Ao se selecionar o componente, o vemos em destaque (vermelho).

Neste ponto, é necessário associar o componente importado a um pacote de modelo de componente.

A SolidWorks possui uma biblioteca de componentes para serem utilizados, porém, é possível também importar outros componentes obtidos por terceiros, acrescentando este à biblioteca de componentes.

Para associar o componente listado a um componente existente, clique com o botão esquerdo do mouse no componente a ser associado, conforme figura a seguir.

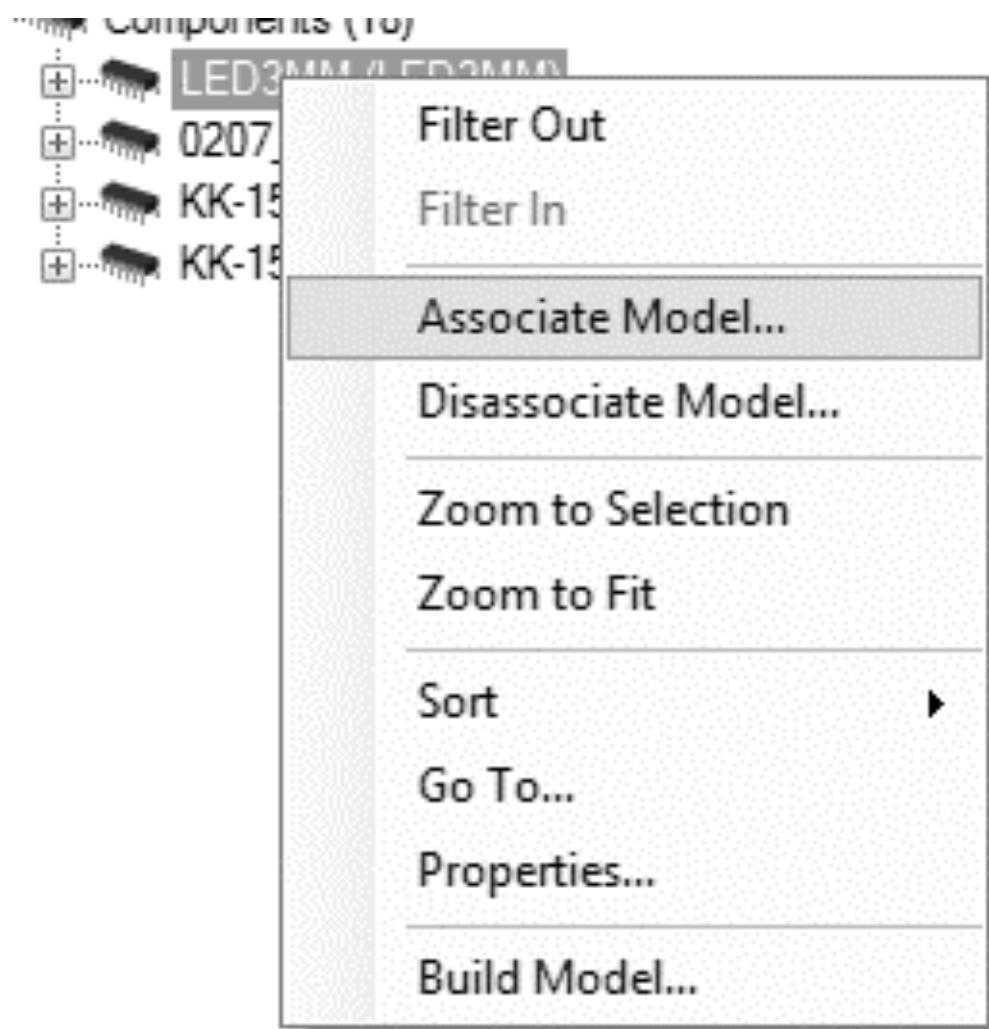


Figura 131 - Menu de opções de associação, com botão direito do mouse

Surgirá uma tela, para que indique um arquivo de parte do SolidWorks.

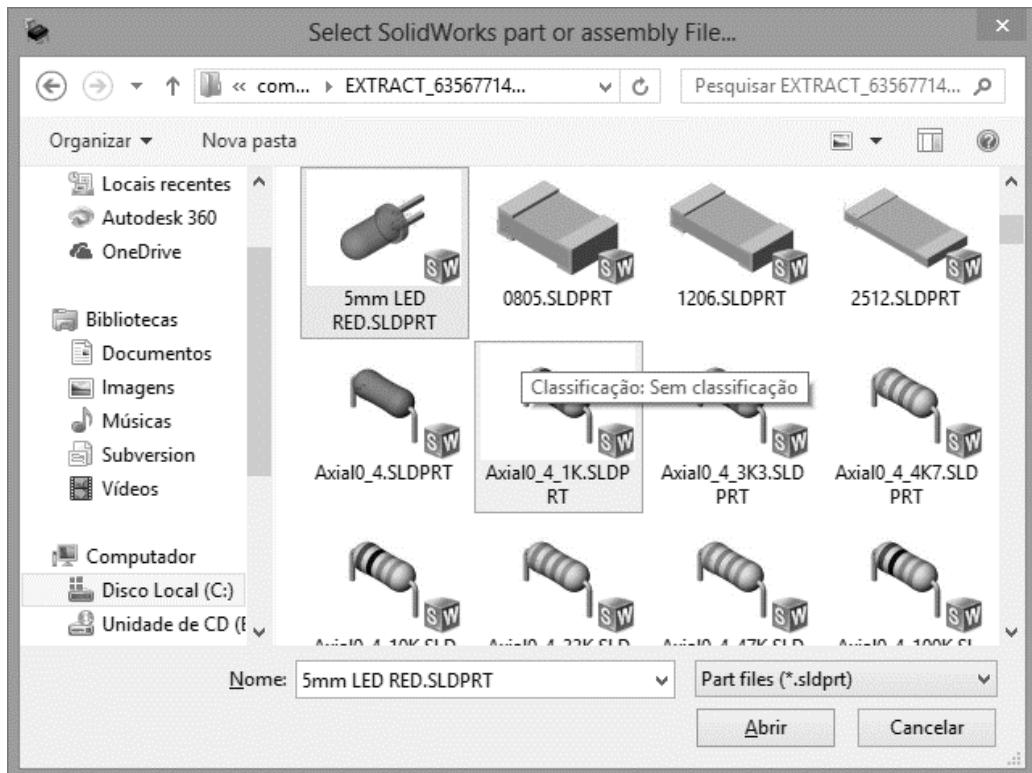


Figura 132- Opções de componentes associados

O SolidWorks possui diversos componentes. Verifique qual melhor se aplica a seu componente. Caso necessário, importe através de um site de componentes 3D, como <https://grabcad.com/home>.

No caso do LED, já possuímos um LED de 5 mm. Desta forma, associamos este a todos os componentes LEDs.

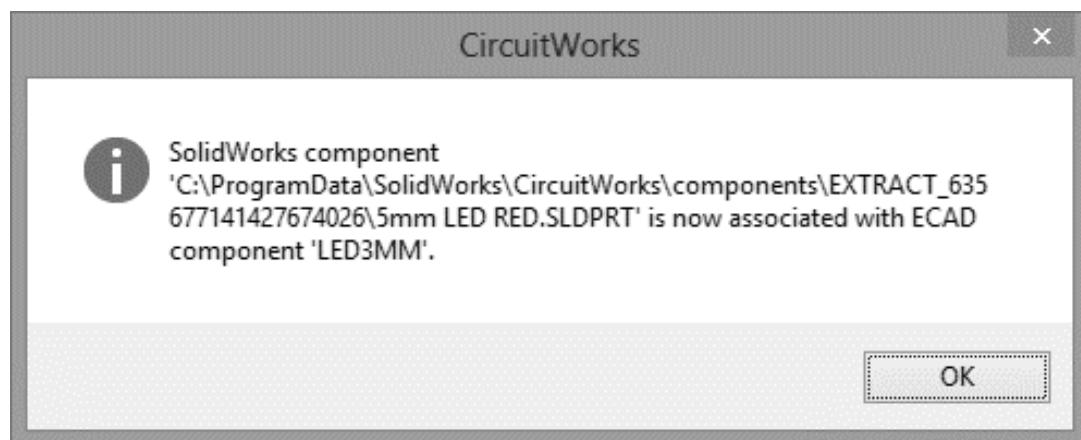


Figura 133 - Notificação de associação de componente

Pronto – realize a associação de todos os outros componentes.

Quando todos os componentes estiverem associados, clique no botão BUILD MODEL, conforme figura a seguir.

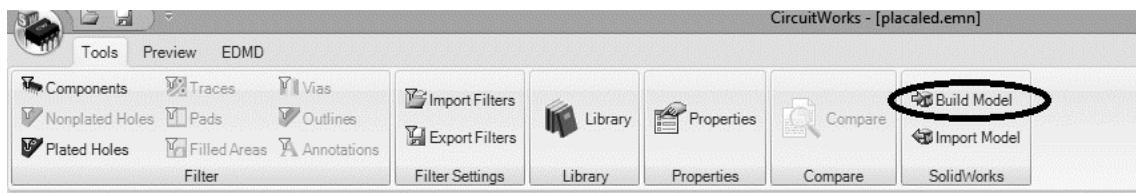


Figura 134- Menu de opções do solidworks

O SolidWorks irá verificar primeiramente se existe algum problema na geração da placa. Se houver, ele não deixará seguir.

No caso da imagem abaixo, apenas irá confirmar que tudo está correto e também a geração da peça. Clique em Build para gerar a peça.

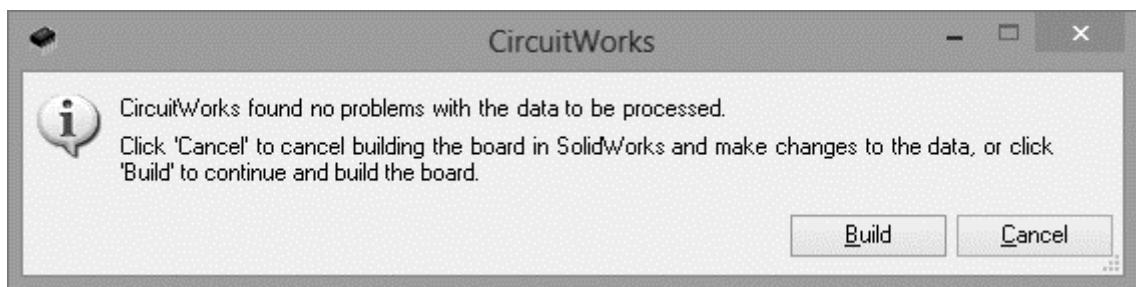


Figura 135 - Confirmação do Solidworks

Ao clicar em Build, será processado e gerado o arquivo da placa.

Ao gerar o arquivo, alguns componentes podem estar fora dos pontos ou em posição errada.

Você deve acertar seu posicionamento manualmente utilizando as opções: **move**, **rotate** e **mate**. Ao acertar o posicionamento, a placa ficará próxima ao apresentado a seguir.

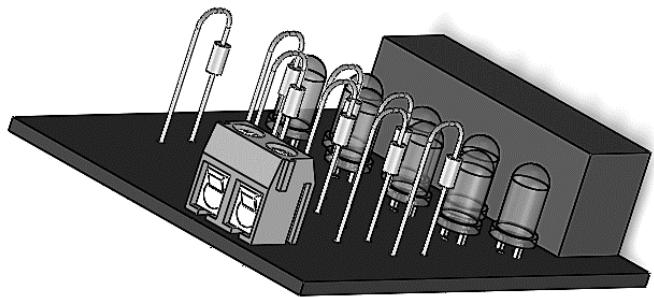


Figura 136- Visão da placa

O componente ao fundo não tem definição, pois na vinculação não foi inserido um componente padrão, mas foram modificadas as especificações de tamanho do componente para o equivalente nas dimensões do componente original.

Para finalizar, basta salvar a placa como uma montagem, identificando esta na pasta robotinics/solidworks/eletrônica/placaleds/, conforme figura abaixo:

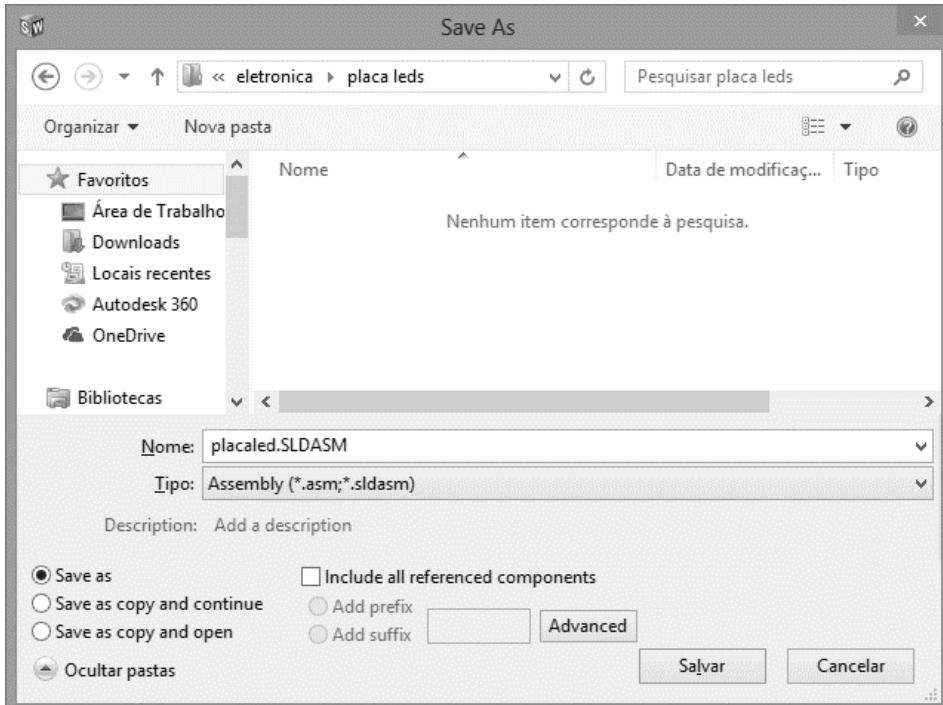


Figura 137 - Salvando placa LEDs

4.3 Software

Desenvolvimento com Raspberry PI

No tópico anterior vimos como desenvolver com o Arduino criando uma aplicação que responde a comandos. Neste contexto é possível criar agora uma aplicação que chama o Arduino e solicita comandos a ele.

Esta aplicação fica hospedada no Raspberry PI, podendo ser rodada em qualquer Linux.

MySQL

A programação e acesso ao banco MySQL em C não é prática complicada. O suporte de MySQL ao SQL é semelhante àquele encontrado no PostgreSQL (Stones, 2002), e você pode ver em mais detalhes neste livro, conforme apresentado.

Aqui iremos mostrar alguns conceitos sobre sua biblioteca, e não sobre o SQL, tampouco sobre conceitos de programação nesta linguagem.

Vamos observar este pequeno exemplo:

```
#include <my_global.h>
#include <mysql.h>

int main(int argc, char **argv)
{
    int flgErro = 0;

    //Inicializa
    MYSQL *con = mysql_init(NULL);

    if (con == NULL)
    {
        fprintf(stderr, "%s\n", mysql_error(con));
        flgErro = 1;
    }

    //Cria a conexão com o banco
    if (mysql_real_connect(con, "localhost", "root", "sua senha",
                           NULL, 0, NULL, 0) == NULL)
    {
        //Em caso de erro entra aqui
        fprintf(stderr, "%s\n", mysql_error(con));
        flgErro = 1;
    }

    //Executa comando de ação
    if (mysql_query(con, "CREATE DATABASE testdb"))
    {
        fprintf(stderr, "%s\n", mysql_error(con));
        flgErro = 1;
    }

    //Fecha a conexão com Mysql
    mysql_close(con);
    if(flgErro ==0)
    {
        exit(0);
    }
}
```

```
    }
    else
    {
        exit(1);
    }
}
```

As libs abaixo são referências da biblioteca do MySQL.

```
#include <my_global.h>
#include <mysql.h>
```

Para começar a usar o MySQL é necessário criar primeiro um ponteiro de operação:

```
MYSQL *con = mysql_init(NULL);
```

Que será inicializado com os parâmetros de conexão:

```
if (mysql_real_connect(con, "localhost", "root", "sua senha",
                      NULL, 0, NULL, 0) == NULL)
```

O localhost é o endereço do servidor mysql, no caso 127.0.0.1, ou seja, a própria máquina. O root, é o nome do usuário do mysql, e sua senha que foi utilizada na instalação.

Este comando executa um comando SQL:

```
if (mysql_query(con, "CREATE DATABASE robotinicsdb"))
```

E, por fim, fechando a conexão:

```
mysql_close(con);
```

Existem vários exemplos de utilização destes comandos em diversas fontes do nosso projeto.

A comunicação com MySQL é uma etapa importante do processo de desenvolvimento do nosso robô e de qualquer IoT.

Lendo dados no SQL

O exemplo abaixo é uma transcrição de um fragmento de programa que lê os dados do MySQL.

```
Res = mysql_query(&my_connection,"select * from security");

If (res)
{
    printf("Select error: %s\n",mysql_error(&my_connection));
}

else
{
    res_ptr = mysql_use_result(&my_connection);

    if (res_ptr)
    {
        while ((sqlrow = mysql_fetch_row(res_ptr)))
        {
            Printf("Fetched data...\n");
        }
    }
}
```

No exemplo, a função `mysql_fetch_row` recupera os dados na variável `sqlrow`, que é do tipo `MYSQL_ROW`.

`MYSQL_ROW sqlrow;`

Para ler os dados, basta referenciar como um vetor, onde `sqlrow[nro_do_vetor]`, sendo o campo `nro_do_vetor` a posição do vetor que será lida.

A função `mysql_fetch_row` captura uma linha de dados do resultado da pesquisa. Desta forma, através de um laço `while` é possível ler todos os registros.

A função irá sair do laço quando não houver mais linha de registro a ser lida.

A função `mysql_fetch_field` trabalha da mesma forma que a `mysql_fetch_row`, porém, em vez, de pegar os registros, pega as informações dos campos, como:

- Tipo de campo (numérico, string)
- Nome do campo
- Tamanho
- etc.

Outra diferença é que no `mysql_fetch_row` a saída é um ponteiro da variável do tipo `MYSQL_FIELD`.

A variável `MYSQL_FIELD` tem como propriedades os seguintes atributos:

- `name` – Nome do campo
- `type` – Tipo, que é um conjunto
- `length` – Que é o tamanho máximo do tipo
- `flags` – Conjunto de indicadores que apresentam diversas marcações para o campo.

Programação Socket

A programação socket em linux é facilmente entendida através da exemplificação do programa de servidor de fala.

srvMonitor

Para tal aplicação, usaremos no projeto os fontes robotinics\linux\srvMonitor

Nele existem dois arquivos principais:

Makefile – Responsável pela compilação do fonte

srvMonitor.c – Código-fonte do nosso programa

O *srvMonitor*, como o próprio nome diz, é responsável pelo monitoramento e controle do Arduino.

Este serviço é o primeiro de uma série de serviços do nosso robô, e permite implementar e escutar pela internet os seus comandos.

Características do nosso programa:

- Recebe comandos por TCP (Porta: 8095)
- Repassa para a USB através da porta /dev/ttyACM0, que pode ser editada

Dispositivos controlados:

- Servomotores
- LEDs
- LCD 2x16

Fluxo de Execução

O fluxo a seguir apresenta o processo de execução do srvMonitor.

Neste processo a aplicação repete os comandos liberados no socket e transfere para a porta serial do Arduino.

O srvMonitor tem, então, a responsabilidade de criar uma porta para que outras aplicações possam transmitir seus comandos para o equipamento.

Funções

Start_serial – Responsável pela criação dos handlers (cabeçalhos) e parametrização inicial da porta serial para responder comandos pela serial

Start_TCP – Cria os handlers e parametrização inicial do socket TCP para responder a comandos

MostraIP – Função que captura o IP da máquina a fim de permitir visualização no LCD do robô

Arduino_Versao – Captura a versão do firmware para controle de comando

EscutaTCP – Responsável pelo recebimento de comandos TCP enviados pelo socket

Analisa – Realiza a interpretação dos comandos capturados e armazenados na variável buff

Serialport_write – Envia dados para serial

Serialport-read_until – Recebe dados até receber o CR (enter)

Write – Envia dados para porta TCP (socket)

srvFala.c

Falar é um serviço que é responsável por atender a requisição de comandos de voz pela web.

Fonte: robotinics\linux\srvFala

Falar permite que o operador interaja pelo robô e diga qualquer comando de forma muito simples, bastando enviar o texto que se deseja na porta 7091 que seu conteúdo é dito.

Para executar o programa falar é necessária a prévia instalação do programa eSpeak e os batchs Ler.sh e falar.sh, com sua eventual instalação na pasta /usr/bin.

Não entraremos no detalhe do código total, porém mencionaremos alguns fragmentos para que o leitor entenda seu funcionamento.

Funções

main – É a função que é chamada no início da execução do programa, responsável pela configuração da aplicação

Falar – Roda o comando de leitura, permitindo que o texto seja falado

```
void Falar( char * Info)
{
    char buff[1000];
    sprintf(buff,"/usr/bin/falar.sh %s",Info);
    system(buff);
}
```

Seguindo o fragmento de programa, podemos analisar:

A linha 3 declara a variável buff, criando um espaço de 1000 bytes.

A linha 4 armazena o comando falar.sh, passando o parâmetro recebido na função da linha 1.

A linha 5 usa o comando system, que executa o comando criado na variável buff.

Motion

O que é o motion

Iremos agora configurar e ajustar a visão do robô. Para tanto, usaremos o motion.

Com o motion é possível criar um stream de vídeo para monitorar câmeras e webcâmeras remotamente.

Seu uso, inicialmente criado para cftv, foi adaptado com sucesso para a robótica.

O motion percebe mudanças na tela, permitindo que seja disparado um programa para interpretar tal imagem.

A imagem pode ser total, ou apenas da área mudada, bastando poucas configurações no arquivo.

Como instalar

O motion pode ser obtido através de download ou instalação do pacote apt-get install motion*.

Após a instalação é necessária a configuração do dispositivo que se deseja instalar (webcam ou outros dispositivos).

No pacote padrão a pasta de configuração fica localizada na pasta /etc/motion, sendo os arquivos:

- motion.cfg
- thread1.cfg
- thread2.cfg
- thread3.cfg
- thread4.cfg

O arquivo motion.cfg é responsável pela configuração da execução do programa, podendo ser de duas formas:

- ✓ Serviço – Roda independente de usuário, permitindo que o programa rode em background.
- ✓ Aplicação – Permite ser iniciado ou interrompido, ideal para ser utilizado em modo de debug, pois é possível acompanhar os retornos da aplicação.

As últimas versões deste produto podem ser obtidas pelo site do fabricante (lavrsen, s.d.): www.lavrsen.dk/foswiki/bin/Motion/WebHome.

Motion.cfg

Daremos aqui uma pequena introdução sobre a configuração do motion, apresentando de forma complementar o script de configuração que utilizamos.

Parâmetros de configuração

Daemon (on/off) – Permite colocar a aplicação em modo de service ou aplicação; caso on (ligado) fica atribuído serviço.

On_motion_detected – Trigger que é disparada quando o movimento é encontrado

Videodevice – Dispositivo que está sendo utilizado para visualizar, geralmente /dev/video0 ou /dev/video1

V4l2_palette – Placas de captura de vídeo possuem mais que um dispositivo de captura. Quando for o caso, use o valor de 0 a 8 para selecionar a porta. Webcams utilizam valor 8.

Webcam_port – Porta web que será aberta para monitorar o que o robô enxerga

Webcam_localhost – Limita o acesso da câmera na máquina localhost (127.0.0.1) ou permite visualização pela rede

Thread – Quando o robô permitir enxergar mais que uma câmera, a thread cria múltiplos arquivos de configuração. Para cada um deles é necessário criar um arquivo de configuração, chamado thread.cfg. Quando a thread não estiver definida, será criada uma única aplicação, não necessitando configurar tais arquivos.

Text_left – Texto exposto no canto esquerdo inferior

Text_right: Texto exposto no canto direito inferior

Text_double: Mostra o texto BOLD

Target_dir: Local onde as imagens serão armazenadas para posterior interpretação. O local onde a pasta será indicada precisa ter permissão do serviço. Caso contrário, não funcionará.

Setup_mode: Roda em modo de debug, apresentando na tela as informações

Área_detect: Apresenta um valor de referência. Caso este valor seja inferior ao valor da área mudada, o programa dispara os triggers de mudança, registrando a mudança. Este valor deve ser apurado em formato de debug e posteriormente registrado.

Thread1-4.cfg

O arquivo thread1.cfg e os demais criam instâncias da aplicação principal, com a diferença de parâmetros. Grande parte dos parâmetros de configuração do arquivo motion.cfg são herdados para o thread.cfg, porém os parâmetros que diferem como device são redeclarados.

Desta forma, todos os parâmetros do motion.cfg são também parâmetros do thread.cfg, não havendo, assim, necessidade de explicá-los novamente. No entanto, passaremos um exemplo de arquivo apenas para referência.

Nosso robô não possuirá duas câmeras. Desta forma, não é necessário configurar uma thread para tanto. Porém, caso seu projeto necessite de uma câmera adicional, este processo é muito interessante.

Script de Processamento da Imagem

O script de Processamento da Imagem é disparado para realização da análise de objetos.

Diferentemente do que se pode imaginar, este script não trata de reconhecer nada, mas encaminha e coordena os programas que serão chamados para reconhecer, criando um fluxo simples para execução de listas de atividades para reconhecer rosto, contornos ou outras peculiaridades da imagem.

Como funciona a visão do robô

O motion produz a captura das imagens, armazenando em banco de dados MySQL a referência e em disco a imagem e o vídeo. Também chamando o script em shell chamado analisa_img.sh para processamento das imagens e analisa_avi.sh para processamento dos vídeos. Ambos localizados em /usr/local/bin/

1. O MySQL funciona como a memória do equipamento, agrupando as imagens e gerando listas temporais
2. Web:8081 – Cria um stream de vídeo, permitindo que possamos controlar a imagem
3. Analisa_avi.sh – É chamado pelo motion e encaminha para o processamento do vídeo.
4. Analisa_img.sh – É chamado pelo motion para o processamento das imagens.

Quanto maior o fluxo das requisições chamadas nos scripts maior deverá ser a capacidade do processador, falaremos com mais detalhes nos próximos capítulos, sobre este tema.

Script analisa_avi.sh

```

#!/bin/bash

quebrar() {
    local filepath="$1"
    local filename=$(basename "$filepath")
    local ext=$(echo "$filename" | awk -F '.' '{ if (NF==2) {print $NF} else if ( NF>2) {print $(NF-1)."."$NF} }')
    local dir=$(echo "$filepath" | awk -F '/' '{ print substr($0, 0 , length($0)-length($NF)-1) }')
    echo -e "$dir"\t"$filename"\t"$ext"
}

echo $0 >> /projetos/robotics/logs/logpar0.txt
echo $1 >> /projetos/robotics/logs/logpar.txt
echo $1
ret=$(quebrar "$1")
echo "$diretorio: " "$(echo "$ret" | cut -f1)"
diretorio=$(echo "$ret" | cut -f1)
filename=$(echo "$ret" | cut -f2)
somentefilename=$(echo "$filename" | cut -d. -f1)
extensao=$(echo "$filename" | cut -d. -f2)
novoarquivo=$(echo "$1" | cut -d. -f1)
novoarquivo=$(echo "$novoarquivo"".mp4")"
echo $novoarquivo
#Converte o vídeo em formato que é permitido ser visto pelo browser
ffmpeg -i $1 -c:v libx264 -preset ultrafast $novoarquivo
echo "Processou arquivo \"$novoarquivo">>
rm -f $1

```

No exemplo acima não há comandos de tomadas de decisão configurados, tao pouco programas de reconhecimento.

5. Desenvolvimento das extremidades do robô

Nosso projeto começa a ganhar volume. Até o presente momento, nosso robô parece mais um carro do que propriamente um robô.

Neste capítulo iremos incluir suas extremidades, como cabeça e braços, agregando informações ao projeto, e finalizando no próximo capítulo, com sua montagem.

5.1 Mecânica

Ao criar os membros do robô, criamos também um problema, pois caberá aos servomotores e à estrutura mecânica a sustentação dos elementos a ele fixados.

Alguns pontos são imprescindíveis na construção e projeto destas estruturas:

- Prever as forças mecânicas que estarão atuantes no objeto
- Dimensionamento de massa e volume das estruturas que serão sustentadas
- Dimensionamento do torque necessário para tração dos corpos, incluindo eventuais objetos suportados
- Composição e projeto das estruturas mecânicas, bem como o projeto de seus componentes

Apesar de já termos detalhado no capítulo anterior o servomotor, ainda não o utilizamos em nossas aplicações.

Diferentemente de outros motores, o servomotor já possui sensor de posição incorporado, possuindo movimento de 180 graus (ou mais).

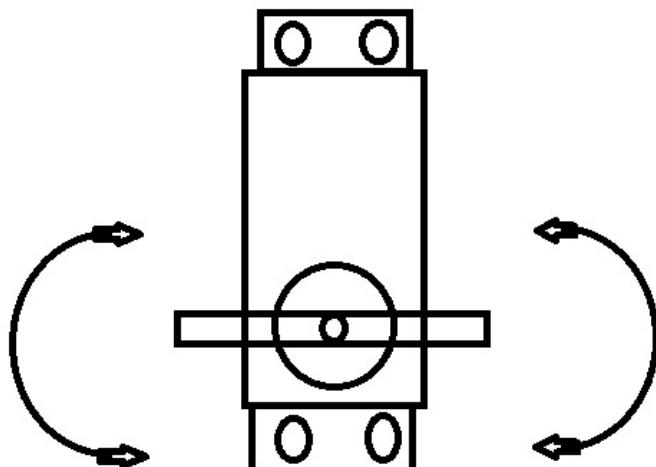


Figura 138 - Exemplo de movimento de rotação do servomotor

Desta forma, ao se definir uma posição inicial, o robô sempre a manterá. Esta informação é útil, pois permite que na montagem definamos um ponto de 90 graus e sobre ele montemos o conjunto mecânico, permitindo, desta forma, movimentos de 90 graus para cada direção.

Braço Robótico - Extensão

O projeto de um braço começa com a ligação ao seu ombro. Assim, criaremos a peça que se liga a esta.

Para tanto, utilizaremos o arquivo:

Arquivo do SolidWorks: robotinics\solidwork\BRACO_EXTENSAO.SLDPR

Impressão 3D: robotinics:\impressora3d\BRACO_EXTENSAO.STL

As características deste braço permitem que sejam conectados aos servomotores de diversas formas e gostos, podendo ser utilizado no desenvolvimento de outros projetos.

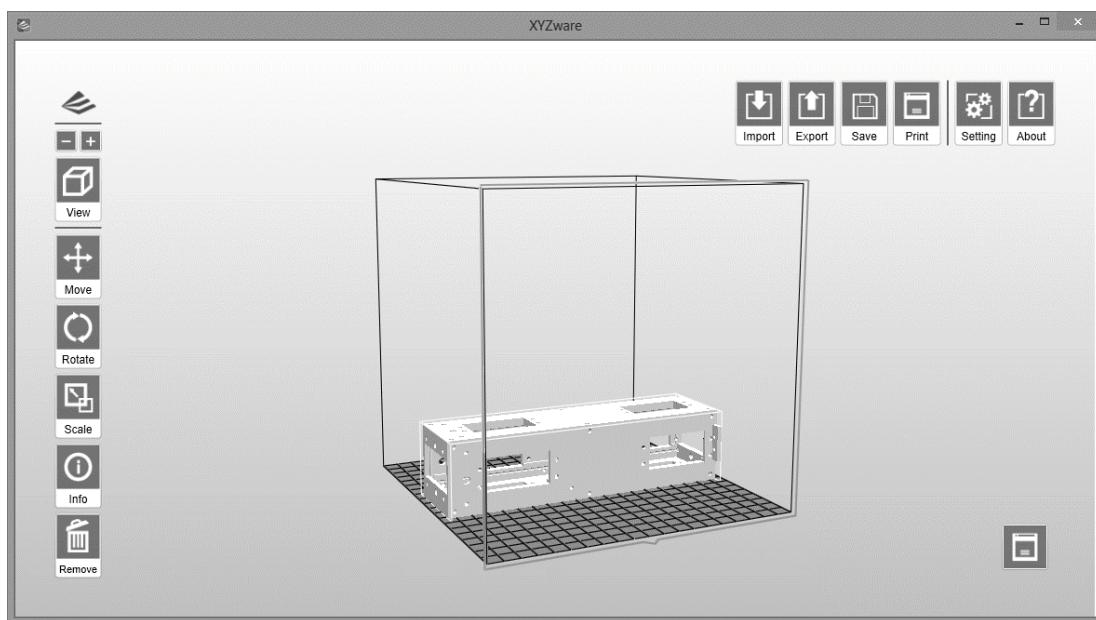


Figura 139- Vista da peça sendo impressa no software

Esta versatilidade permite o reaproveitamento do designer do branco, atribuindo melhora considerável das peças e maior flexibilidade.

Braço Robótico - Base

O braço base é conectado diretamente à garra e se conecta ao braço extensão que fixa ao ombro.

Arquivo do SolidWorks: robotinics\solidwork\BRACO_BASE.SLDPR

Impressão 3D: robotinics:\impressora3d\BRACO_BASE.STL

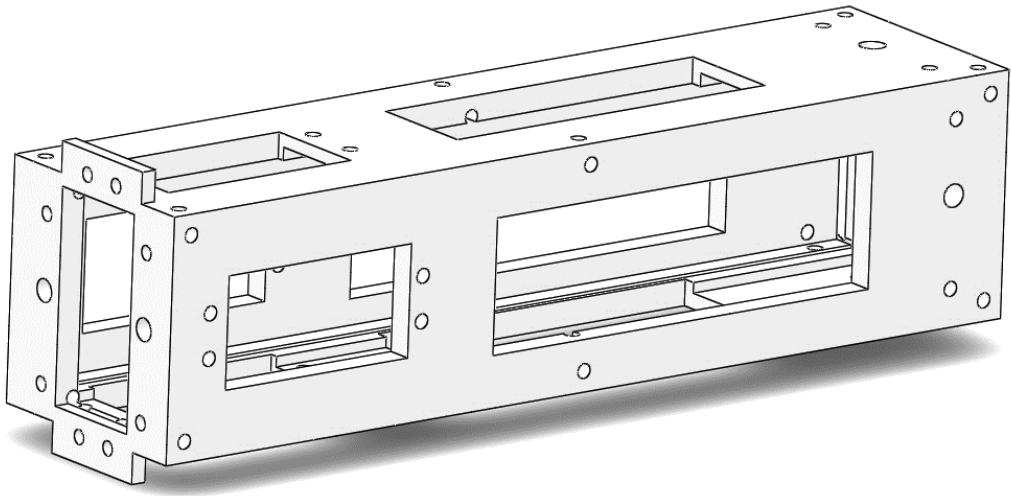


Figura 140- Visão do braço base

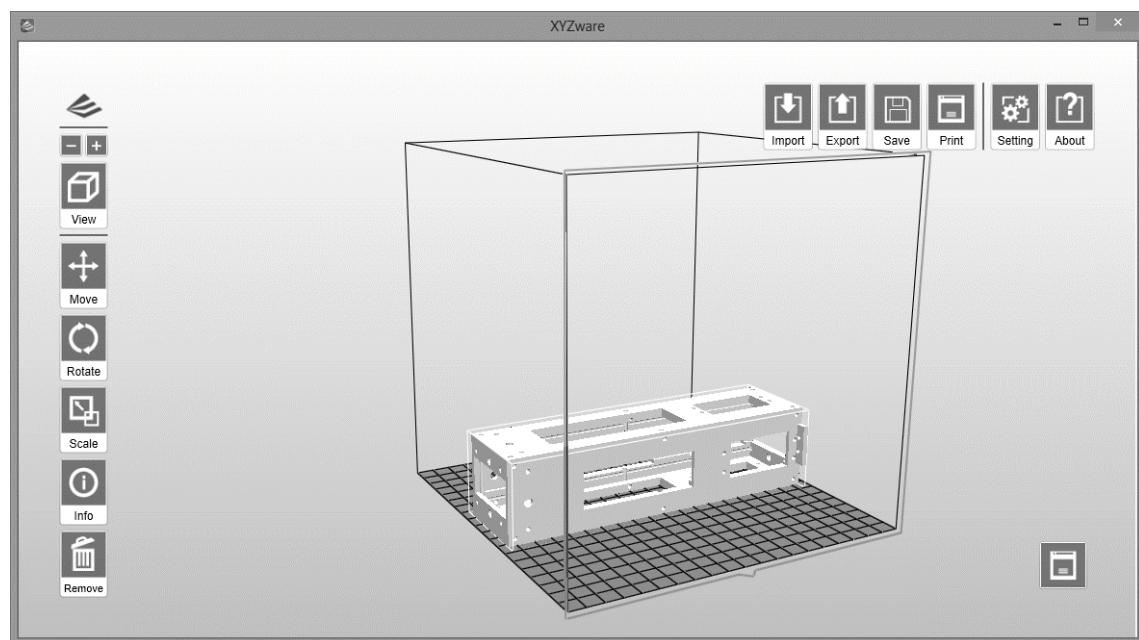


Figura 141- Visão da impressão do braço base

Montagem do Braço

A montagem do braço segue o esquemático de montagem a seguir:

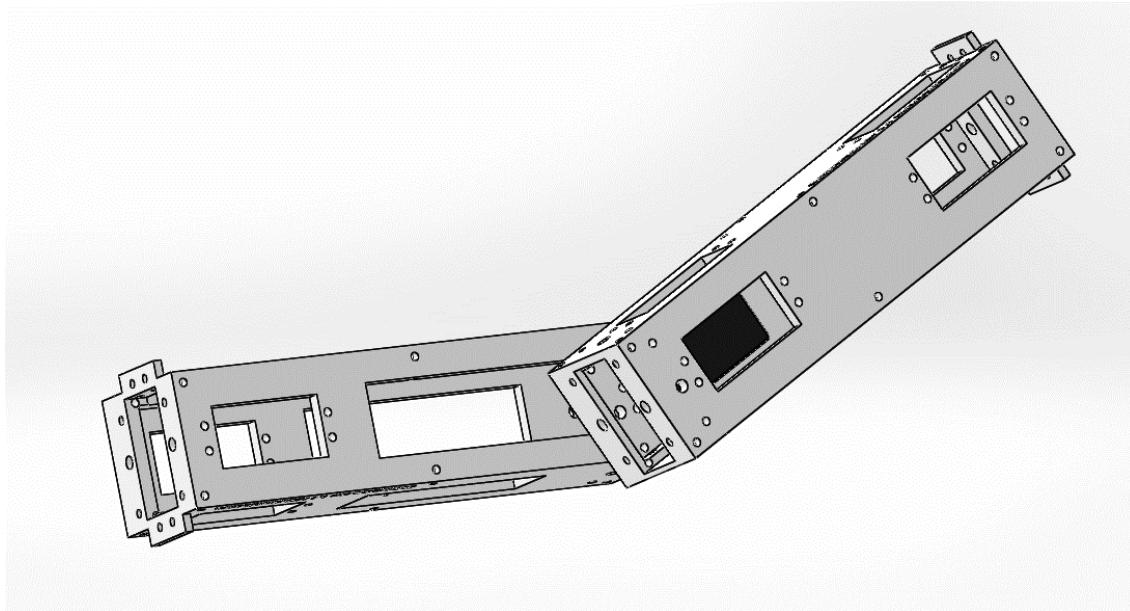


Figura 142- Montagem do braço

A base do braço é responsável pela conexão com a garra. Em alguns projetos, quando não se exige conexão longa, a base pode ser fixada sozinha.

A extensão do braço é uma prolongação do mesmo, para permitir que o robô alcance objetos um pouco mais distantes.

Um ponto negativo no uso do prolongador do braço é que, ao fazer isso, aumenta o torque necessário do servomotor que sustenta o conjunto para movimentar objetos com mesma massa, sendo que o torque é baseado na distância (d) do centro de apoio e multiplicado pela força (F).

$$T = F * d$$

O princípio aplicado já foi explicado no Capítulo 1.

Garra

Um braço robótico sem um elemento de fixação perde completamente a função e o sentido, por isso é necessário o uso de algum elemento de fixação para ferramentas.

O braço robótico permite o acoplamento mecânico de diversos elementos de fixação, e estes podem ser adaptados conforme a necessidade do projeto.

No nosso caso, utilizaremos uma garra, pois permite que o robô segure diversos objetos, como pincéis, canetas ou aplicadores de tinta etc.

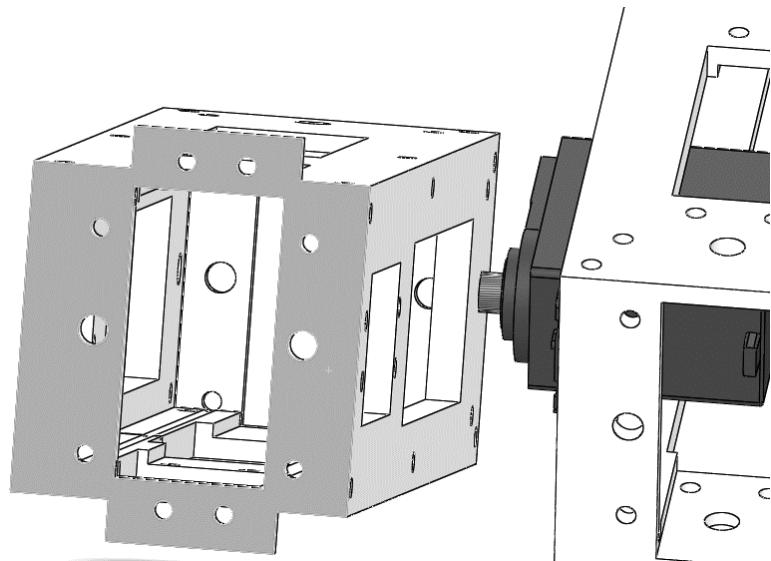


Figura 143- Visão do braço

A garra, ou “Gripper”, é responsável por pinçar objetos. Existem diversos tipos de projetos de garra, cada uma com vantagens e desvantagens.

O projeto construtivo de uma garra foge do escopo deste projeto e do livro, porém podemos recomendar o projeto Gripper Negro com Servomotores de Tesla BEM. O projeto pode ser baixado no site <https://grabcad.com/library/gripper-negro-con-servomotores-1>.

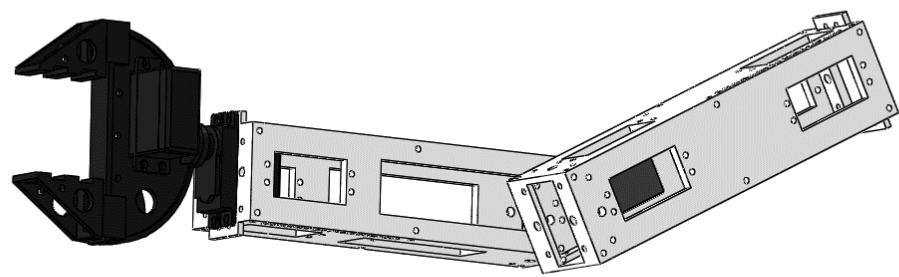


Figura 144- Visão do braço montado

Não apresentaremos a montagem do outro braço, mas sua concepção e projeto são semelhantes ao do braço esquerdo, com as faces de contato sendo as opostas.

Suporte U

O suporte em U faz a função do ombro em nosso robô. Também chamado “Brackets”, pode ser feito em diversos materiais, porém recomendo, devido ao seu esforço, que seja comprado em ferro.

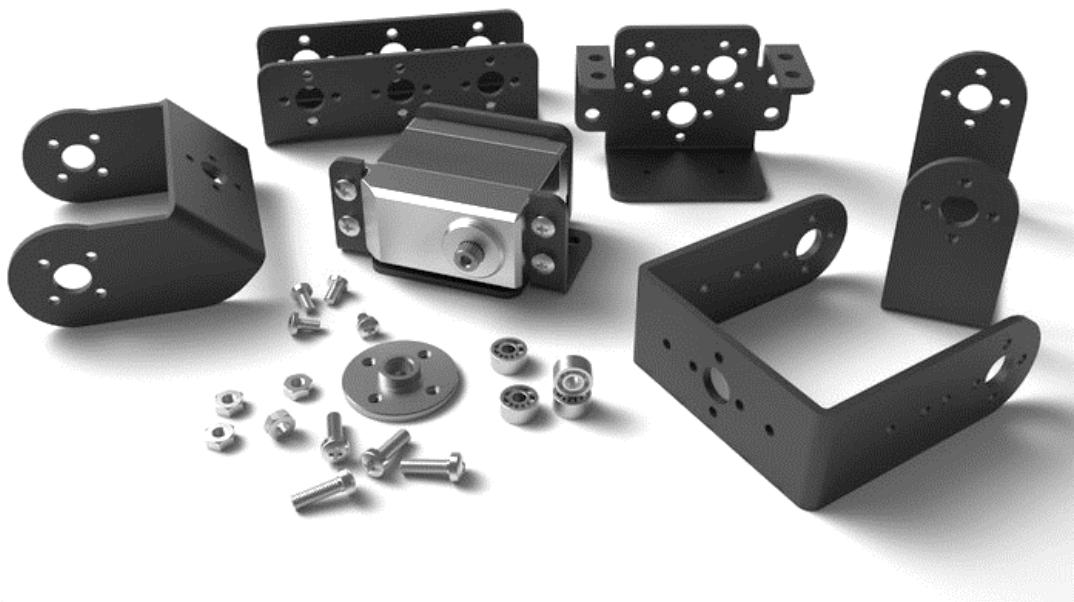


Figura 145- Servomotor e conexões

O projeto pode ser baixado no site <https://grabcad.com/library/servo-brackets-2>.

Agradecemos à Hendricks por disponibilizar o projeto.

Imagen das vistas das peças:

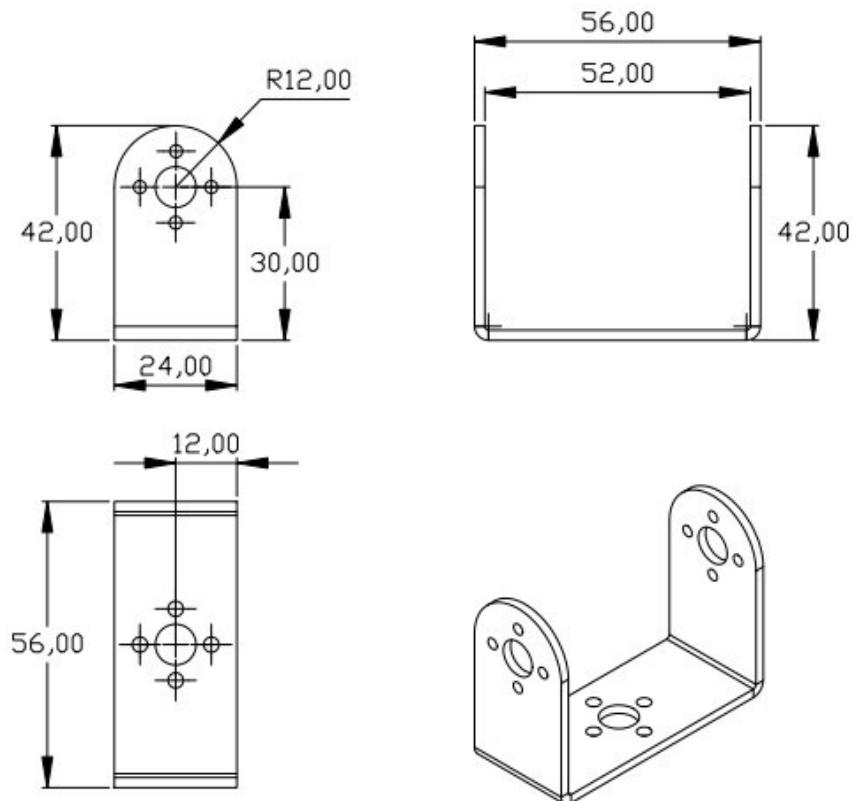


Figura 146 - Esquemático de vista da peça 1

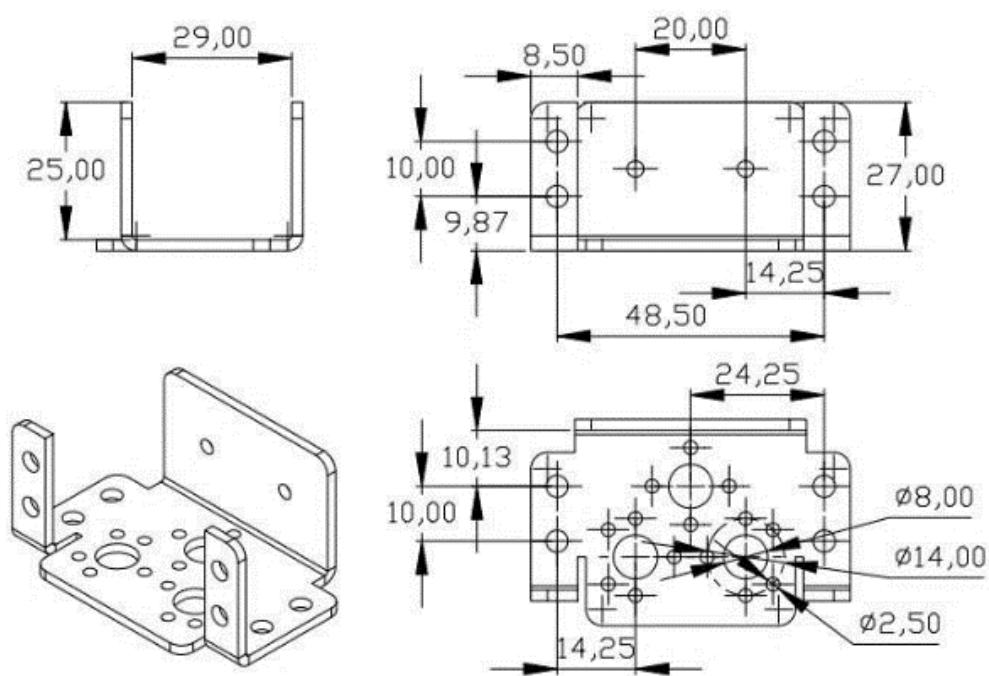


Figura 147 - Esquemático de vista peça 2

Montagem do Ombro

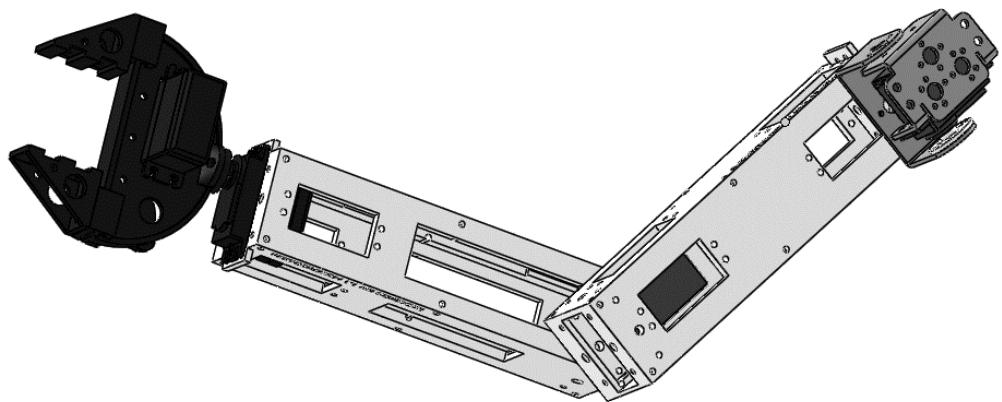


Figura 148- Visão do braço no SolidWorks

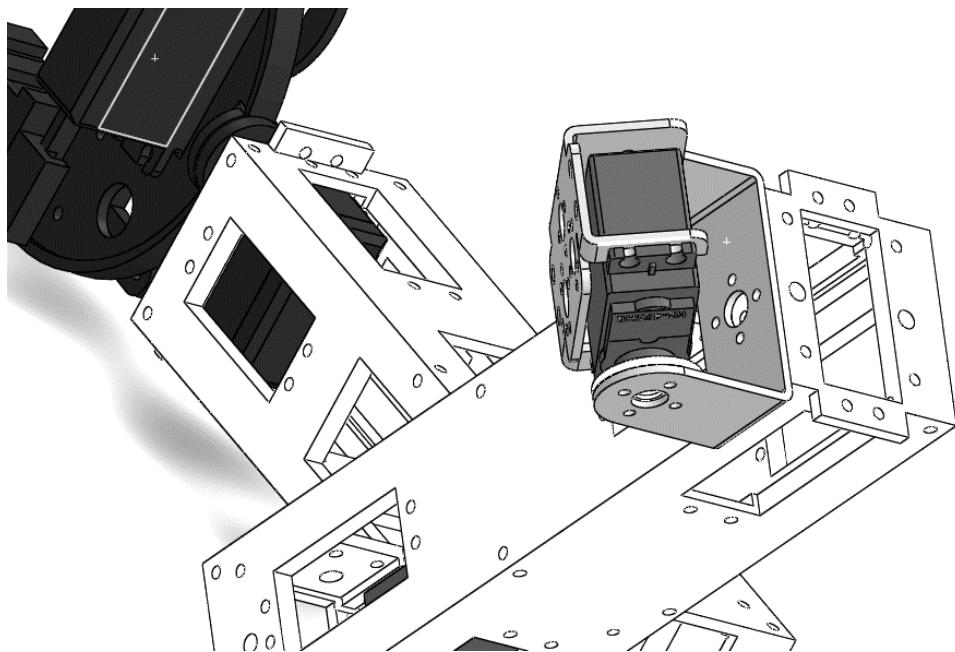


Figura 149- Conexão do braço com servomotor visto pelo SolidWorks

Esquema Denavit-Hartenberg

Não explicarei como fazer o esquema Denavit-Hartenberg, mas irei apresentar o que significa e uma ideia simplista de seu uso.

O esquema Denavit_Hatenberg primeiramente estuda o movimento de duas retas, conforme apresentado a seguir.

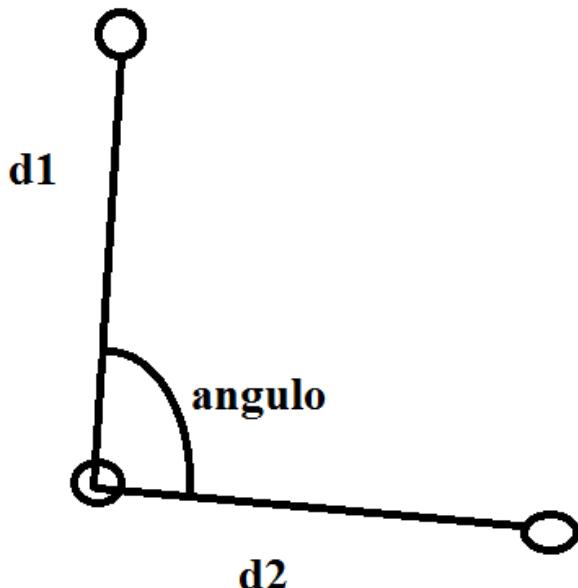


Figura 150- Representação Denavit-Hartenberg

Imaginemos um braço robótico se movimentando.

Para determinar a posição de um braço robótico de dois segmentos, precisamos apenas de duas informações:

- Distância entre os segmentos unidos
- Ângulo de rotação

Sendo assim, se tivermos um braço com três pontos de rotação, serão necessários seis parâmetros para determinar sua posição final no espaço.

Não entraremos nos detalhes do estudo deste esquema, porém, basta lembrar que, dependendo do número de segmentos alinhados em um mesmo eixo, poderá existir mais de uma solução que atenda àquela posição.

Desta forma, a indicação dos ângulos que serão aplicados nos servomotores será obtida através de um cálculo matemático complexo.

Por que não será aplicado este cálculo?

Nosso robô será operado remotamente (telepresença). Desta forma, todos os movimentos do braço serão definidos em ângulos pelo operador, e a determinação de posição final deixa de ser uma definição e passa a ser questão de escolha do operador do sistema.

5.2 Eletrônica

Componentes de Conexão

Definem-se componentes de conexão e componentes de ligação elétrica, permitindo a ligação física entre duas placas ou componentes.

A ligação elétrica é feita mediante a utilização de diversos meios. Vamos estudar alguns tipos de cabos necessários para permitir a ligação física de partes elétricas.

Para entender os componentes de conexão precisamos entender alguns de seus cálculos:

Cálculo de carga de fio elétrico em corrente contínua

Para determinar se um fio está adequado ao propósito desejado, é necessário identificar a corrente e a temperatura em que ele está trabalhando.

Estas informações são disponíveis através de diversas fontes, pois são padronizadas e tabeladas conforme apresentado:

Capacidade de Condução de Corrente (A)

Tabela 1 - Capacidade de Condução de Corrente

Seção nominal mm ²	Diâmetro	Cabos bipolares (A)	2 condutores isolados unipolares (A)
1	1.13	21	21
1,5	1.38	26	27
2,5	1.78	36	37
4	2.26	49	50
6	2.76	63	53

*Baseado em temperatura do condutor de 90°C e do ambiente de 30°C

**Referência NBR 54 10/2004

Fio paralelo 2,5 mm Vermelho e Preto

Amplamente utilizado em aparelhos de som automotivo, este fio possui cores bem definidas, evitando que seja criada confusão na elaboração das conexões.

Este fio é utilizado nas conexões com a fonte de alimentação 12 V, e todas as conexões 12 V utilizarão este fio.

A vantagem deste fio é a capacidade de conduzir corrente mais elevada, garantindo que a corrente aplicada seja conduzida.



Figura 151- Fio paralelo vermelho e preto

Conforme apresentado na tabela anterior, podemos transportar corrente máxima de 36A. Isso limita nossa potência máxima.

$$P = V \cdot A \Rightarrow P = 12 \cdot 36 \Rightarrow 432W$$

Considerando uma faixa de segurança de 70% da corrente máxima, temos 25A:

$$P = 12 \cdot 25 = 300W$$

Potência máxima com segurança para a sessão deste fio.

Fio Paralelo Preto e Vermelho 1,5 mm

O fio preto e vermelho de 1,5 tem a responsabilidade de alimentar as conexões com 5 V.



Figura 152 - Par de fio elétrico P&V

Permite fornecer potência máxima de 135 W (27A * 5 V)

Cabo Manga de Oito Vias

O Cabo Manga de oito vias tem a função de condução dos sinais elétricos para os servo-motores e LEDs. Os cabos Manga facilitam muito a configuração do robô, pois permitem criar conexões padronizadas, facilitando os encaixes.



Figura 153- Cabo Manga de oito vias

Entre as funções deste cabo, estão:

- O controle dos servo-motores pela transmissão do sinal PWM
- Acionamento de lâmpadas (LEDs)

Conecotor RJ45

O conector RJ45, ou 8P8C, é comumente conhecido como conector de rede e facilita a conexão com as placas.

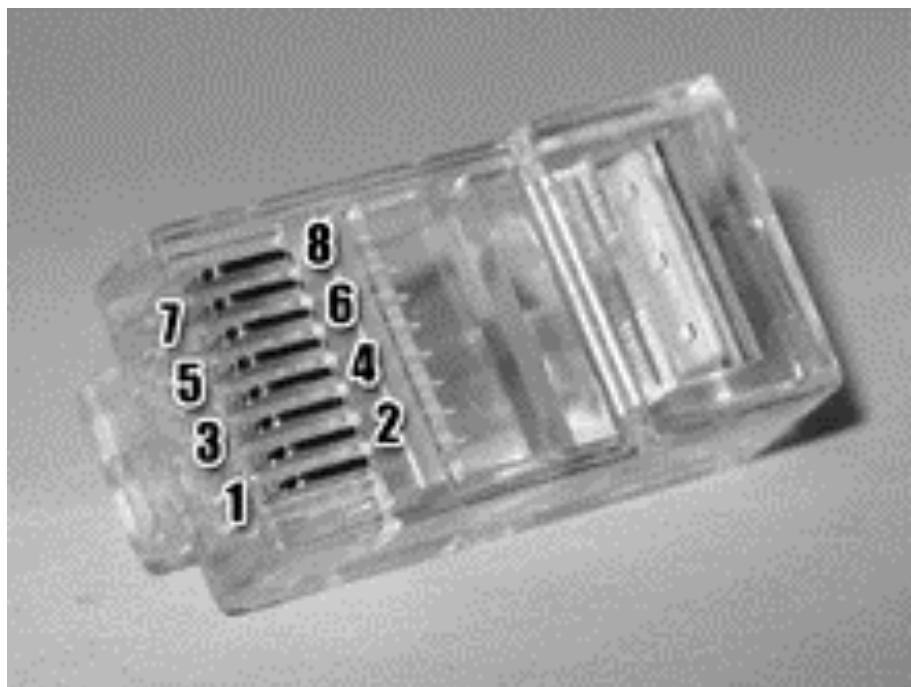
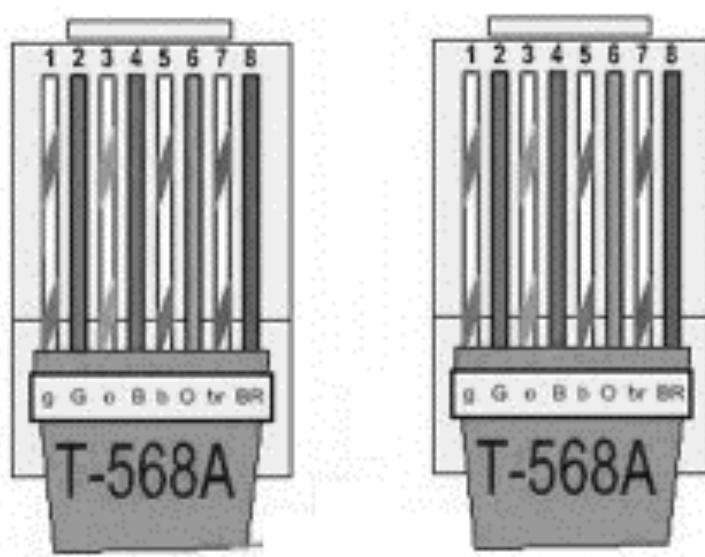


Figura 154- Visão do conector RJ45

Cabeamento RJ45

Os cabos de conectores RJ45 usados no nosso robô utilizam o padrão de cabo normal de rede (T-568A).



Seguindo a ordem:

g – Verde e branco

G – Verde

o – Laranja e branco

B – Azul

b – Azul e branco

O – Laranja

Tr – Roxo e branco

BR – Roxo

Figura 155- Padrão de conexão do cabo de rede

As conexões são realizadas para ligar as placas de controle dos servomotores.

Lista de Componentes

A lista de componentes apresenta os elementos necessários para confecção das placas. Por questão prática, apresentaremos os componentes e, em seguida, pelo nome, a lista dos componentes.

Conecotor RJ45 PCI Fêmea

São conectores que permitem acoplar o RJ45 Macho, permitindo a união elétrica de 8 pontos elétricos. Normalmente utilizado em rede de dados do PC.

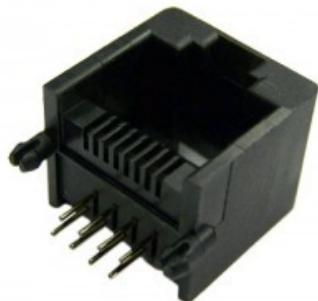


Figura 156 - Conecotor RJ45 para PCB

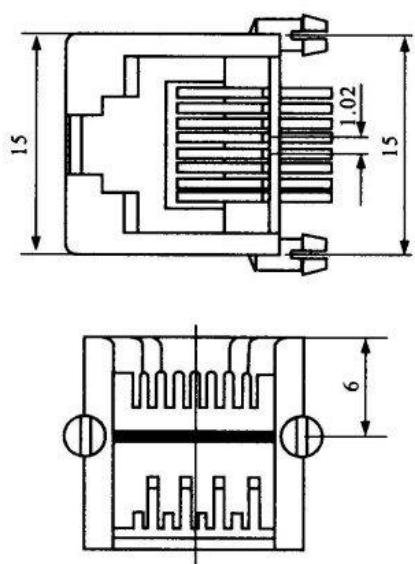


Figura 157 - Esquemático do RJ45 para PCB

Barra de Pinos Macho 180°

Tem função de elemento de ligação, entre os contatos da placa e diversos componentes, como jumpers, fios elétricos com conexão.



Figura 158- Barra de pinos macho

Seu uso está associado a montagem de placas de PCB.

Borne

O Borne é um elemento de fixação de fios na placa.



Figura 159 - Conector Borne

Existem vários modelos deste padrão. No caso, podemos escolher o que estiver mais acessível.

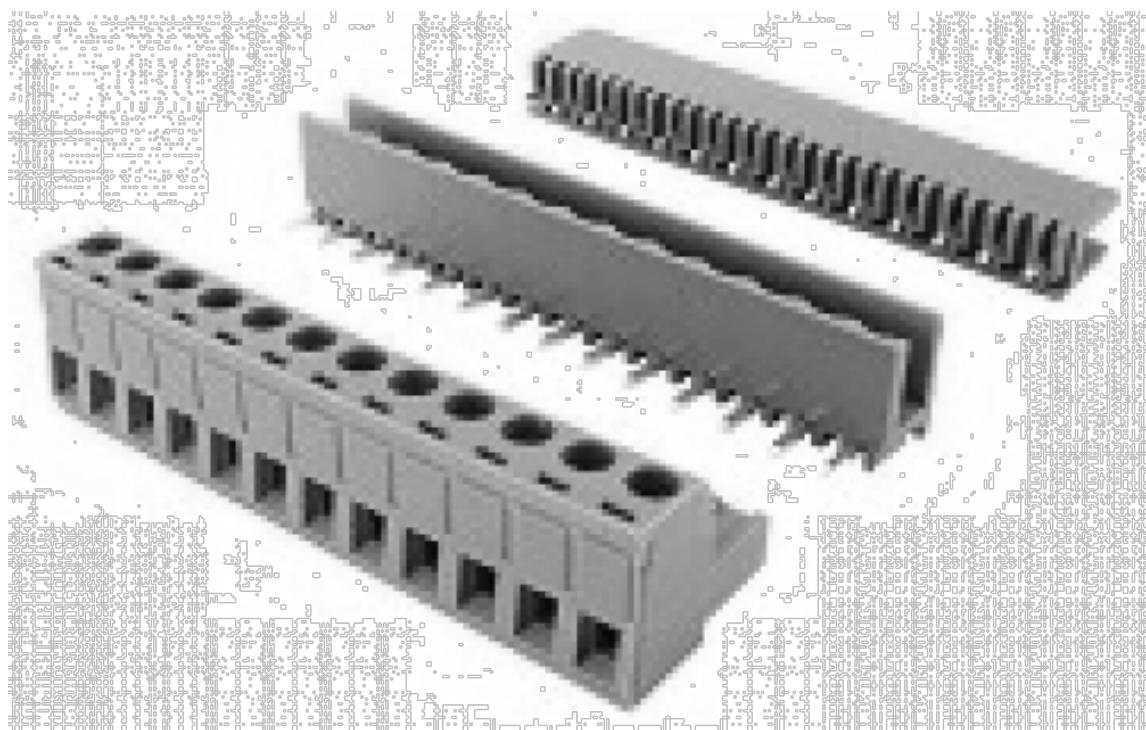


Figura 160 - Conector Borne com várias entradas

Resistores e LEDs

São elementos comuns que permitem o controle de corrente (resistor) e iluminação (led).

O Resistor trabalha conforme anunciado anteriormente sobre a formula $V = R \times i$

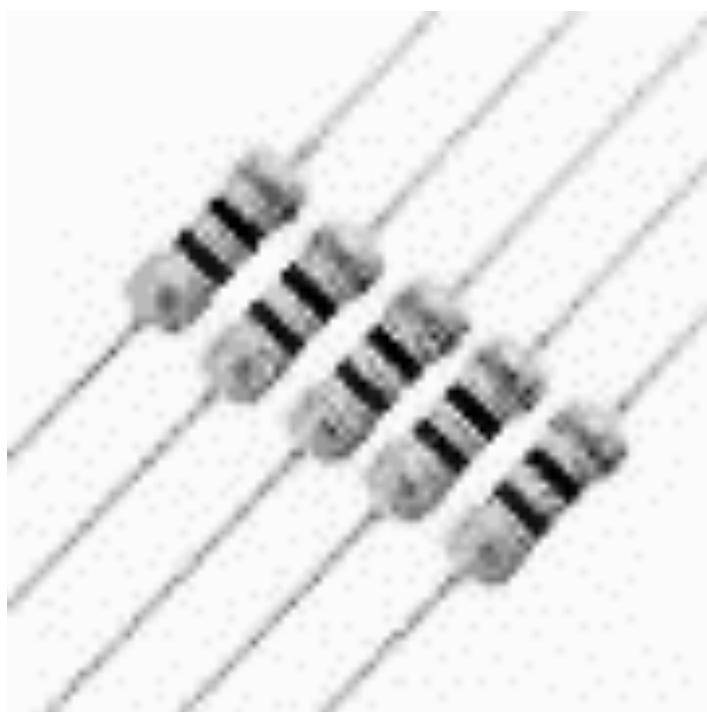


Figura 161 - Exemplo de Resistor

Tabela de Cores de Resistor:

Os resistores possuem uma tabela de cores.

Cor	Valor
Preto	0
Marron	1
Vermelho	2
Laranja	3
Amarelo	4
Verde	5
Azul	6

Violeta	7
Cinza	8
Branco	9

Onde deve-se substituir cada cor por seu respectivo número.

Um resistor possui 4 faixas de cores

Faixa 1	Faixa2	Faixa3	Faixa4

Onde a primeira e a segunda faixa formam um número, exemplo Vermelho e Cinza = 28.

A Faixa3 indica a potência de 10. No exemplo se tivermos a cor laranja na terceira faixa, será o número 10^3 , que será 1000, no caso nosso resistor hipotético será 28×1000 , ou $28K\Omega$.

A Faixa4 indica a tolerância.

Podendo apresentar duas cores:

Dourado = +/- 5% de tolerância, ou seja, possuir resistência 5% superior ou inferior neste mesmo valor.

Prateado = +/- 10% de tolerância, ou seja, possuir resistência 10% superior ou inferior neste mesmo valor.

Em nosso resistor hipotético caso, tenha faixa prateada, poderia ter valores maiores de $(28K\Omega + 2.8K\Omega) \Rightarrow 30.8K\Omega$. E Valores menores de $(28K\Omega - 2.8K\Omega) \Rightarrow 25.2K\Omega$

Desta forma, poderíamos medir no multímetro valores de $25.2K\Omega$ até $30.8K\Omega$.

O LED, é o elemento de iluminação, não entrarei em seu detalhamento técnico porque não acredito ser necessário.

LED é a sigla para Ligth Emitting Diode, que significa **Diodo Emissor de Luz**.

É um componente eletrônico semicondutor, e que somente gera luminosidade quando eletrificado em um sentido exato.



Figura 162 - Exemplo de Led

O Led Possui duas “Pernas”, a maior chamada Ânodo, deve ser ligado ao polo positivo.

A perna menor, chamada de cátodo, deve ser ligada no polo negativo.

Caso, seja trocado os polos o LED não produz luminosidade.

O LED de forma geral precisa ser associado a um resistor.

O LED necessita de uma corrente constante, nunca superior à máxima informada em seu catálogo.

Anteriormente no capítulo 1, abordamos como calcular o valor do resistor no tópico **Cálculo de Tensão/Corrente e Resistência.**

5.3 Software

Criação de Daemons

Os daemons são programas que rodam constantemente e sem a intervenção do usuário, realizando atividades repetitivas e de controle.

O daemon do robô permite que sejam lidas as requisições por TCP e redirecionadas para dois serviços.

Existem três daemons:

- TCP->Serial
- TCP->Falar
- Motion (câmera>web)

Já abordamos o processo de instalação do motion em capítulos anteriores. Desta forma, agora daremos prioridade ao processo de instalação dos demais módulos.

Já abordamos o processo da síntese de voz no capítulo anterior.

Bluetooth - Parte 2

No Capítulo 4 apresentamos algumas funcionalidades do software Bluetooth, porém o controle motor dos braços não foi apresentado por questões didáticas. Voltamos a utilizar o robotinics\tools\bluetooth para controlar nosso robô, desta vez controlando não o movimento de locomoção, mas o da cabeça e braços.

O programa Bluetooth possui uma aba Corpo, que permite a movimentação dos servos associados a cada eixo do braço.

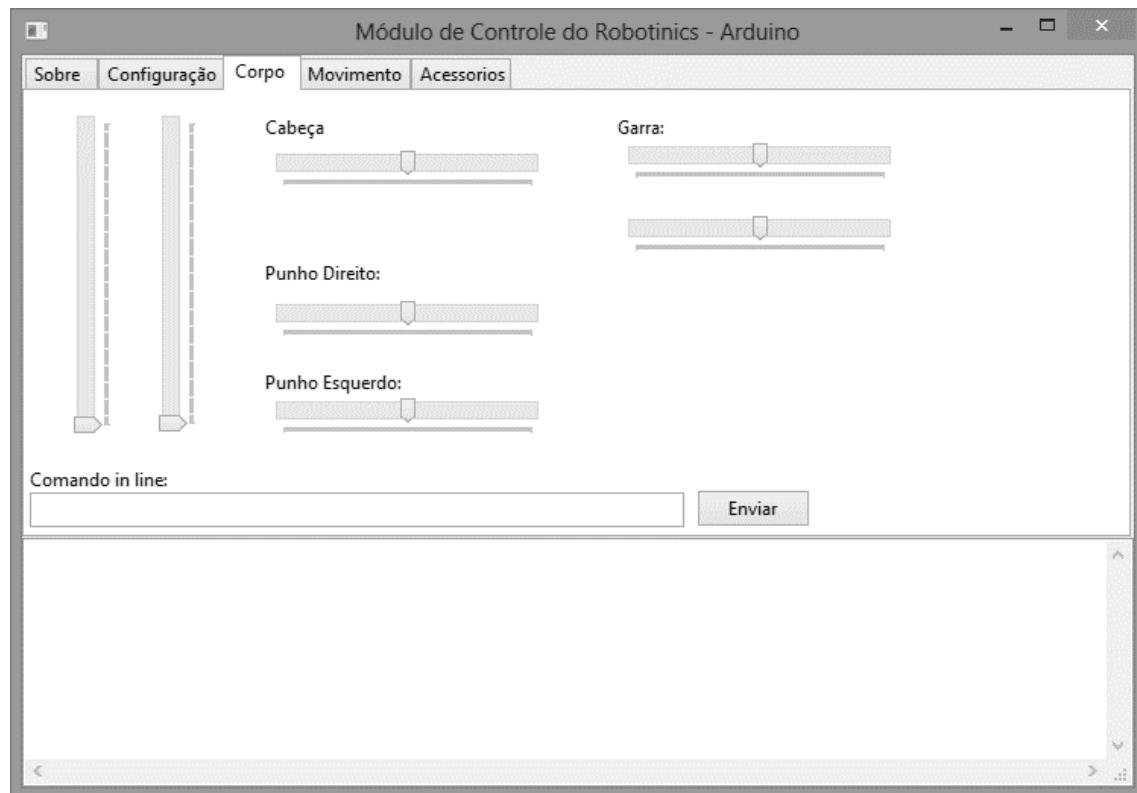


Figura 163 - Exemplo de aplicação de controle do Robô

Fala por TCP

O programa ToolFalar envia um comando de falar através da porta TCP (7091).

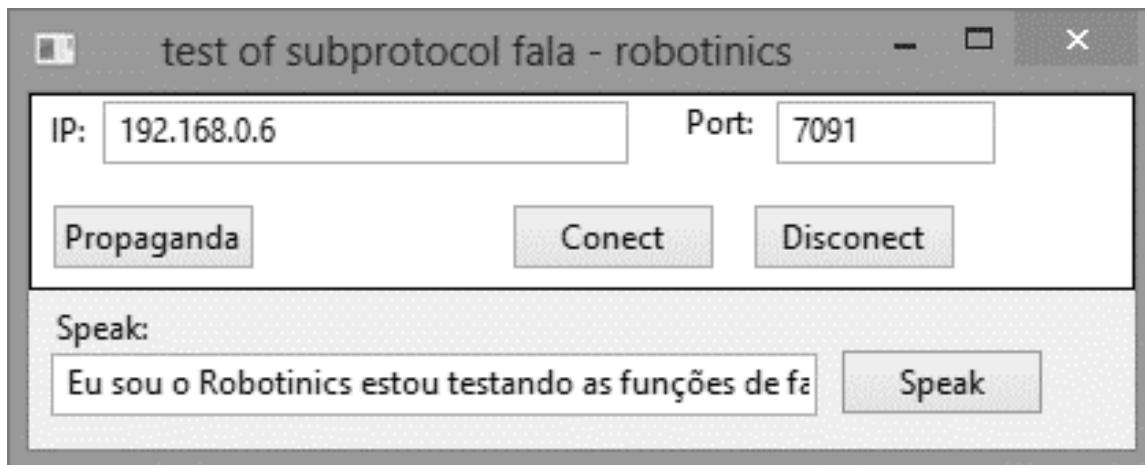


Figura 164- Programa de simulação de fala ToolFalar

Fonte: robotinics\tools\ToolFalar

Este programa foi desenvolvido em Lazarus e necessita da biblioteca **Inet** para funcionar. Roda em Windows e Linux.

O software de teste permite realizar a comunicação de fala entre o robô e outro computador. Para realização do teste de fala, siga os seguintes passos:

- 1) Conecte o robô na mesma rede wifi ou ethernet que o seu computador
- 2) Entre mRemoteNG no seu PC
- 3) Identifique o IP do robô rodando no robô ifconfig
- 4) Pelo mRemoteNG, chame o console do Linux, indo na pasta de projetos. Procure a pasta robotinics\linux\fala.
- 5) Na pasta robotinics\linux\fala, execute o comando make all e aguarde a compilação do programa no robô
- 6) Na pasta robotinics\linux\fala, rode o programa .\falar
- 7) No seu computador, abra a pasta robotinics\tools\ToolFalar e rode o executável, ou compile no Lazarus.
- 8) Digite o texto que se deseja falar na caixa speak:
- 9) Indique no campo IP: o endereço IP do robô, que obteve no passo 1
- 10) Pressione o botão Speak
- 11) Aguarde pela caixa de som do robô e verifique se ela emite sons

Reconhecimento de Imagem

O reconhecimento de padrões de imagem é uma técnica avançada da computação gráfica. Utilizaremos o OPENCV para desenvolvimento.

OPENCV

O OpenCV é uma biblioteca de manipulação gráfica que permite, entre outros recursos, o desenvolvimento de algoritmos de reconhecimento de padrões.

O conjunto de opções disponíveis do OpenCV é muito grande, fugindo do tema deste livro, que apenas apresentará uma visão introdutória.

Pré-requisitos:

```
sudo apt-get install cmake  
sudo apt-get install unzip  
sudo apt-get install libjpeg-turbo*  
sudo apt-get install cmake pkg-config libgtk2.0-dev libavformat-dev libswscale-dev  
sudo apt-get install cmake cmake-curses-gui  
sudo apt-get install default-jdk ant  
sudo apt-get install synaptic
```

Instalação do Opencv

A instalação do OpenCV pode ser encontrada em vários tutoriais. Recomendo, antes de instalá-lo, seguir alguns procedimentos:

1. Realizar a instalação em um equipamento com o mínimo de pacotes instalados, pois algumas atualizações dificultam a implementação do OpenCV, pois existem muitas dependências quanto a bibliotecas que são relacionadas a respectivas versões
2. Sempre realizar a instalação manual, e não do pacote. Apesar de ser mais simples a instalação utilizando o apt-get no Debian ou rpm no Fedora, o uso de pacotes mascara uma série de recursos, que ficam pulverizados em diversas pastas.
3. Ao instalar o OpenCV, sempre verificar a versão recomendada. Como o OpenCV se trata de uma biblioteca em desenvolvimento, existem versões que estão sendo desenvolvidas. Estas versões não estáveis são liberadas, porém podem apresentar vários problemas ainda não resolvidos. O uso de versões estáveis garante um menor esforço no aprendizado.

Procedimentos de Instalação:

Baixe em seu notebook ou PC o arquivo do repositório do OpenCV, localizado no repositório conforme link.

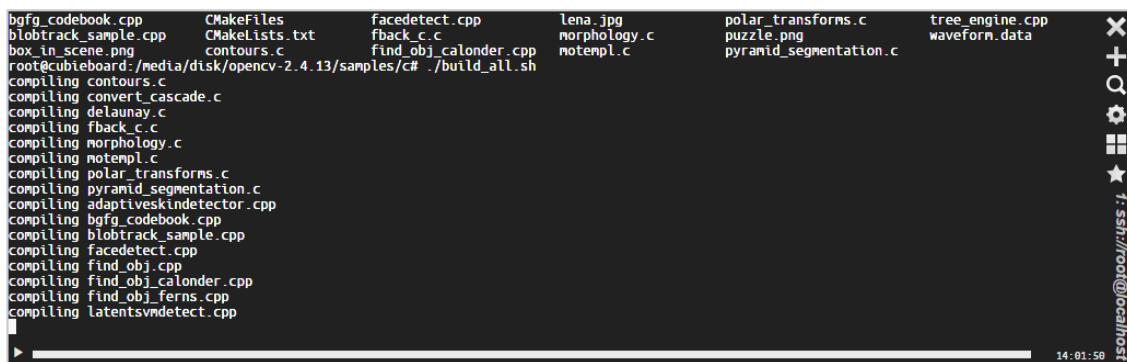
1. Link: <http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.9/opencv-2.4.9.zip>
2. Descompacte o arquivo em uma pasta do computador
3. Com o Filezilla, copie a pasta opencv-2.4.11 para a pasta do root
4. Entre na pasta do root, cd ~
5. Entre na pasta do OpenCV, cd opencv-2.4.9
6. Crie uma pasta build, mkdir build
7. Entre na pasta build, cd build
8. Crie as configurações do makefile executando cmake -DCMAKE_BUILD_TYPE=Release .
9. Rode o makefile executando make
10. Realize a instalação das bibliotecas executando make install
11. Remova os fontes, pois já terminou a instalação rm -rf opencv-2.4.9*

Compilação dos Exemplos apresentados

Ao instalar o OpenCV, diversos exemplos são incluídos para que os desenvolvedores possam estudar.

Para compilar os exemplos, siga os passos abaixo:

1. Entre na pasta do opencv-2.4.13
2. Entre na subpasta samples
3. Para exemplos em C, entre na pasta c
4. Execute o comando em lote de compilação ./build_all.sh



The screenshot shows a terminal window with a black background and white text. It displays the command `./build_all.sh` being run in a directory named `samples/c`. The output of the command is a list of files being compiled, including `bgfg_codebook.cpp`, `blobtrack_sample.cpp`, `box_in_scene.png`, `contours.c`, `lena.jpg`, `polar_transforms.c`, `tree_engine.cpp`, `blobtrack_sample.txt`, `CMakelists.txt`, `fbck_c.c`, `find_obj_calonder.cpp`, `morphology.c`, `puzzle.png`, `pyramid_segmentation.c`, `waveform.data`, `compiling contours.c`, `compiling convert_cascade.c`, `compiling delaunay.c`, `compiling fbck_c.c`, `compiling Morphology.c`, `compiling motempl.c`, `compiling polar_transforms.c`, `compiling pyramid_segmentation.c`, `compiling adaptiveskndetector.cpp`, `compiling bgfg_codebook.cpp`, `compiling blobtrack_sample.cpp`, `compiling facedetect.cpp`, `compiling find_obj.cpp`, `compiling find_obj_calonder.cpp`, `compiling find_obj_ferns.cpp`, and `compiling latentswdetect.cpp`. The terminal window has a dark theme with icons on the right side.

Figura 165- Compilando os exemplos do OpenCV

5. Aguarde a compilação dos exemplos, conforme apresentado na imagem acima.
6. Para executar um exemplo é necessário ter ambiente gráfico instalado. Porém, caso tenha, rode o comando `./facedetect`.

Criação de Aplicação Exemplo

O exemplo a seguir será utilizado em nosso robô para identificar uma face e lançará no banco de dados a posição (x,y) de onde encontrou o rosto.

O exemplo rodará a partir do motion, sendo iniciado sempre que o motion identificar movimento na tela.

Nosso sistema também não precisará capturar a imagem – o local da imagem será fornecido como parâmetro.

Reconhecimento de Voz no Raspberry

O Raspberry não possui entrada de áudio padrão em sua versão A. Desta forma, o reconhecimento de comandos de voz necessita de um hardware auxiliar (placa de som USB). Porém, o Cubieboard possui esta característica, e apresentaremos alguns detalhes sobre o procedimento de reconhecimento de voz.

Utilizaremos, para o reconhecimento de comandos de voz, o cmu sphinx, que tem repositório no site <http://cmusphinx.sourceforge.net/>.

CMU SPHINX

O CMU Sphinx é um conjunto de ferramentas criadas para o desenvolvimento de aplicações de reconhecimento de voz em desktops, aplicativos móveis, casas inteligentes, entre outros.

Entendendo a linguagem

A linguagem é um processo natural, e por este motivo, dificilmente avaliamos a complexidade de seu processo.

Apenas analisando o som podemos perceber que uma única vocalização, como A ou E, não é apenas uma frequência, mas um conjunto. Detalhes como entonação ou mesmo pronúncia mudam consideravelmente a forma como as vocalizações são feitas.

O Sphinx utiliza diferentes técnicas para realizar esta análise, que, de fato, não entra aqui no escopo deste livro, mas pode ser facilmente lida no tutorial do próprio site <http://cmusphinx.sourceforge.net/wiki/tutorialconcepts>.

O que é o Sphinx

O CMUSphinx é um conjunto de ferramentas de reconhecimento de voz que são utilizadas para construção de aplicações de voz.

O conjunto de ferramentas é dividido em aplicações, conforme seu uso:

Pocketsphinx - Biblioteca de reconhecimento de voz leve escrita em C

Sphinxbase - Biblioteca básica com funções utilizadas na pocketsphinx

Sphinx4 - Reconhecedor de fala escrito em JAVA

SphinxTrain - Conjunto de ferramentas para montagem de modelos acústicos

Aplicação prática do CMU Sphinx

Desta forma, utilizando o conjunto de ferramentas, podemos identificar as palavras ditas simplesmente criando uma comparação fonética com o que foi dito.

Nosso programa simplesmente faz uso desta técnica para dizer à aplicação o que escutou. O tratamento e o retorno da ação não serão executados pela aplicação que ouve, que apenas informa o que foi dito.

Desta maneira, fica vinculada a aplicação de IA (Inteligência Artificial) à tomada da ação em relação ao que foi dito. O uso desta técnica facilita o desenvolvimento do software, pois separa as aplicações em pequenos projetos, cada qual com sua complexidade.

Para instalar o Sphinx no Cubieboard ou outro sabor de hardware como o Raspberry, execute pela console os seguintes comandos.

Apesar de possuir distribuição de pacotes, recomendamos o uso do processo manual, sendo o processo por pacote fornecido apenas como referência neste processo.

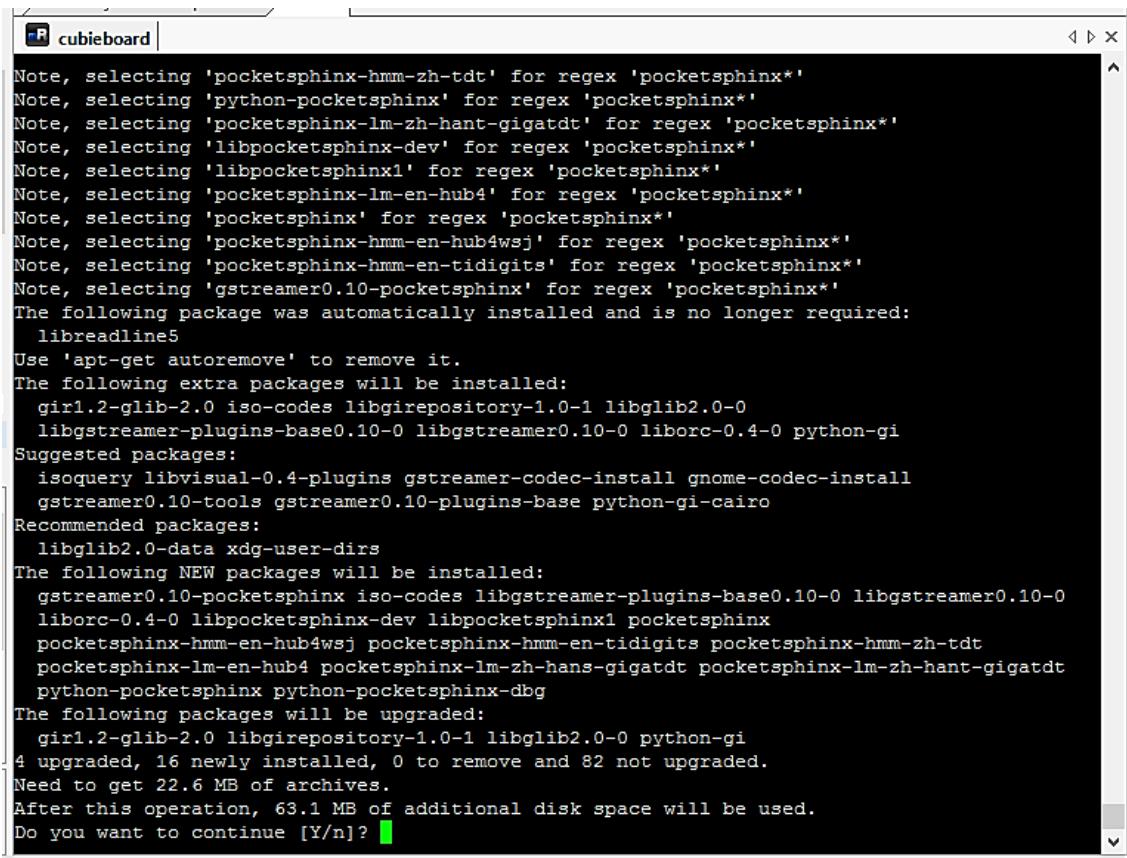
Pré requisitos

Para instalar o CMU Sphinx no Linux é necessária a instalação prévia dos seguintes comandos:

- apt-get install bison
- apt-get install libasound2-dev
- apt-get install swig
- apt-get install python

Instalação

```
apt-get install sphinxbase*
apt-get install pocketsphinx *
apt-get install bison
apt-get install libasound2-dev
```



```
Note, selecting 'pocketsphinx-hmm-zh-tdt' for regex 'pocketsphinx*'
Note, selecting 'python-pocketsphinx' for regex 'pocketsphinx*'
Note, selecting 'pocketsphinx-lm-zh-hant-gigatdt' for regex 'pocketsphinx*'
Note, selecting 'libpocketsphinx-dev' for regex 'pocketsphinx*'
Note, selecting 'libpocketsphinx1' for regex 'pocketsphinx*'
Note, selecting 'pocketsphinx-lm-en-hub4' for regex 'pocketsphinx*'
Note, selecting 'pocketsphinx' for regex 'pocketsphinx*'
Note, selecting 'pocketsphinx-hmm-en-hub4wsj' for regex 'pocketsphinx*'
Note, selecting 'pocketsphinx-hmm-en-tidigits' for regex 'pocketsphinx*'
Note, selecting 'gstreamer0.10-pocketsphinx' for regex 'pocketsphinx*'
The following package was automatically installed and is no longer required:
  libreadline5
Use 'apt-get autoremove' to remove it.
The following extra packages will be installed:
  gir1.2-glib-2.0 iso-codes libgirepository-1.0-1 libglib2.0-0
  libgstreamer-plugins-base0.10-0 libgstreamer0.10-0 liborc-0.4-0 python-gi
Suggested packages:
  isoquery libvisual-0.4-plugins gstreamer-codec-install gnome-codec-install
  gstreamer0.10-tools gstreamer0.10-plugins-base python-gi-cairo
Recommended packages:
  libglib2.0-data xdg-user-dirs
The following NEW packages will be installed:
  gstreamer0.10-pocketsphinx iso-codes libgstreamer-plugins-base0.10-0 libgstreamer0.10-0
  liborc-0.4-0 libpocketsphinx-dev libpocketsphinx1 pocketsphinx
  pocketsphinx-hmm-en-hub4wsj pocketsphinx-hmm-en-tidigits pocketsphinx-hmm-zh-tdt
  pocketsphinx-lm-en-hub4 pocketsphinx-lm-zh-hans-gigatdt pocketsphinx-lm-zh-hant-gigatdt
  python-pocketsphinx python-pocketsphinx-dbg
The following packages will be upgraded:
  gir1.2-glib-2.0 libgirepository-1.0-1 libglib2.0-0 python-gi
4 upgraded, 16 newly installed, 0 to remove and 82 not upgraded.
Need to get 22.6 MB of archives.
After this operation, 63.1 MB of additional disk space will be used.
Do you want to continue [Y/n]? █
```

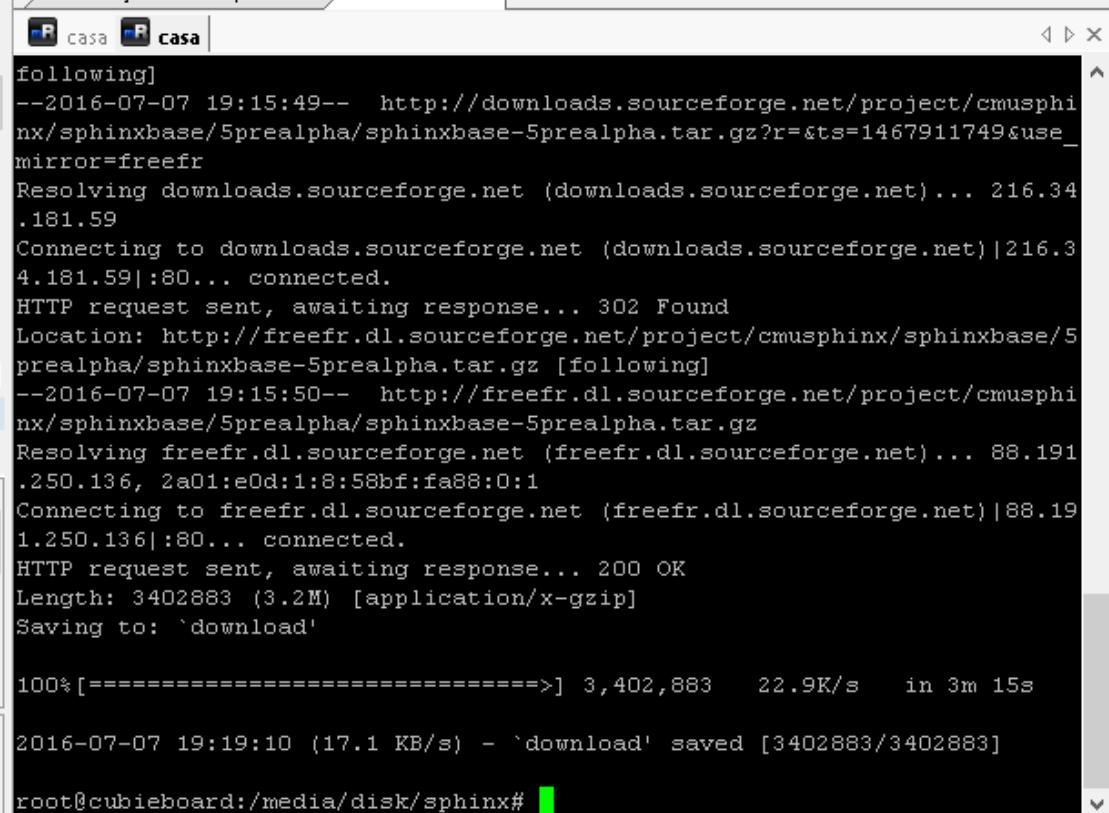
Figura 166- Instalação do Sphinx no Cubieboard

Instalação Manual

A instalação manual consiste em baixar os fontes das aplicações, compilar e testar. Eu recomendo fortemente o uso desta forma de instalação, pois garante que o usuário compreenda melhor como funciona a ferramenta.

Procedimento Manual da Base

1. Baixe o fonte do Sphinxbase:



```
following]
--2016-07-07 19:15:49-- http://downloads.sourceforge.net/project/cmusphinx/sphinxbase/5prealpha/sphinxbase-5prealpha.tar.gz?r=&ts=1467911749&use_mirror=freefr
Resolving downloads.sourceforge.net (downloads.sourceforge.net)... 216.34.181.59
Connecting to downloads.sourceforge.net (downloads.sourceforge.net)|216.34.181.59|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://freefr.dl.sourceforge.net/project/cmusphinx/sphinxbase/5prealpha/sphinxbase-5prealpha.tar.gz [following]
--2016-07-07 19:15:50-- http://freefr.dl.sourceforge.net/project/cmusphinx/sphinxbase/5prealpha/sphinxbase-5prealpha.tar.gz
Resolving freefr.dl.sourceforge.net (freefr.dl.sourceforge.net)... 88.191.250.136, 2a01:e0d:1:8:58bf:fa88:0:1
Connecting to freefr.dl.sourceforge.net (freefr.dl.sourceforge.net)|88.191.250.136|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3402883 (3.2M) [application/x-gzip]
Saving to: `download'

100%[=====] 3,402,883   22.9K/s   in 3m 15s

2016-07-07 19:19:10 (17.1 KB/s) - `download' saved [3402883/3402883]

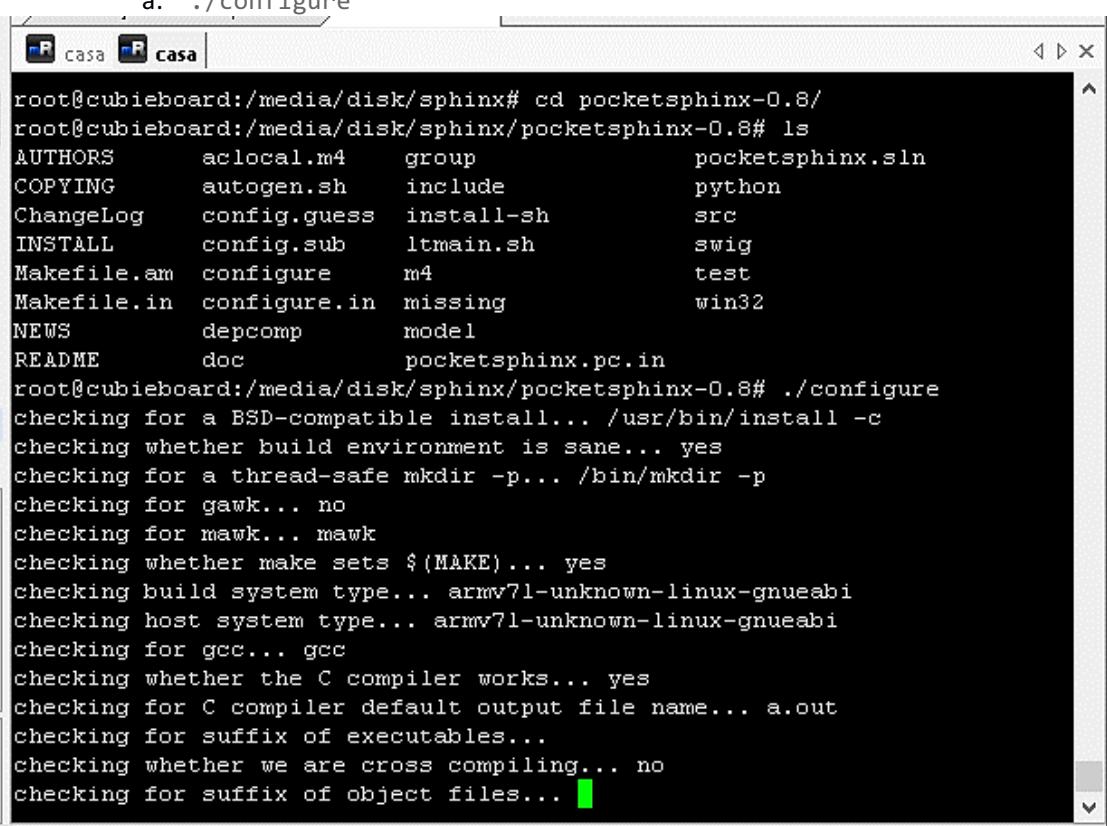
root@cubieboard:/media/disk/sphinx#
```

Figura 167- Baixando Sphinxbase do site

- a. wget <http://sourceforge.net/projects/cmusphinx/files/sphinxbase/0.8/sphinxbase-0.8.tar.gz/download>
2. Descompacte os fontes:
 - a. tar -xvf download
3. Entre na pasta do sphinxbase que criou, conforme a versão que escolheu:
 - a. cd sphinxbase-0.8
4. ./configure --enable-fixed
5. make clean all
6. make
7. make install

Procedimento Manual do Pocketsphinx

1. Baixe os fonts do pocketsphinx:
 - a. wget
<http://sourceforge.net/projects/cmusphinx/files/pocketsphinx/0.8/pocketsphinx-0.8.tar.gz/download>
2. Descompacte o arquivo, veja que o nome foi salvo como download.2, porém o padrão é download. No caso da imagem, use download.2 em vez de download.
 - a. tar -xzvf download
3. cd pocketsphinx-0.8/
4. Agora é necessário gerar a configuração do ambiente de compilação. Para tanto:
 - a. ./configure



```
root@cubieboard:/media/disk/sphinx# cd pocketsphinx-0.8/
root@cubieboard:/media/disk/sphinx/pocketsphinx-0.8# ls
AUTHORS      aclocal.m4      group          pocketsphinx.sln
COPYING      autogen.sh     include         python
ChangeLog    config.guess   install-sh    src
INSTALL      config.sub     ltmain.sh    swig
Makefile.am  configure     m4            test
Makefile.in  configure.in  missing        win32
NEWS         depcomp       model
README       doc           pocketsphinx.pc.in
root@cubieboard:/media/disk/sphinx/pocketsphinx-0.8# ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking build system type... armv7l-unknown-linux-gnueabi
checking host system type... armv7l-unknown-linux-gnueabi
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files...
```

Figura 168- Montando ambiente para compilação

5. Gere os binaries, compilando a aplicação:
 - a. sudo make
6. sudo make check
7. sudo make install
8. sudo make installcheck
9. O ldconfig é o conjunto de caminhos para permitir que o gcc reconheça as libs. Para tanto, é necessário rodar o ldconfig, que atualiza as referências o gcc (compilador c).
 - a. sudo ldconfig

```
cubieboard
-----
checking for Python.h... yes
checking for cython... no
checking for pkg-config... /usr/bin/pkg-config
checking pkg-config is at least version 0.9.0... yes
checking for GStreamer... no
checking for sphinxbase in /projetos/pocketsphinx-0.8/.../sphinxbase... no
checking for sphinxbase in /projetos/pocketsphinx-0.8/.../sphinxbase-5prealpha... yes
checking for /projetos/pocketsphinx-0.8/.../sphinxbase-5prealpha/include/sphinxbase/prim_type.h... yes
checking for /projetos/pocketsphinx-0.8/.../sphinxbase-5prealpha/src/libsphinxbase/libsphinxbase_la... yes
configure: creating ./config.status
config.status: creating pocketsphinx.pc
config.status: creating Makefile
config.status: creating include/Makefile
config.status: creating python/Makefile
config.status: creating python/setup.py
config.status: creating src/Makefile
config.status: creating src/libpocketsphinx/Makefile
config.status: creating src/programs/Makefile
config.status: creating src/gst-plugin/Makefile
config.status: creating doc/Makefile
config.status: creating doc/doxyfile
config.status: creating model/Makefile
config.status: creating model/hmm/Makefile
config.status: creating model/lm/Makefile
config.status: creating test/Makefile
config.status: creating test/testfuncs.sh
config.status: creating test/unit/Makefile
config.status: creating test/regression/Makefile
config.status: executing depfiles commands
config.status: executing libtool commands
root@cubieboard:/projetos/pocketsphinx-0.8#
```

Figura 169- Compilação terminada com pacotes instalados

E, para testar, basta rodar na pasta pocketSphinx:

[./pocketsphinx_continuous](#)

Se rodar, pronto você já superou a fase da instalação!

Um pouco mais de pocketsphinx_continuous
Escuta de forma continua o reconhecimento de voz

Sintaxe:

pocketsphinx_continuous [param01] ... [param99]

Descrição:

Este programa abre um dispositivo de áudio, e espera por uma voz programada. Ao verificar que existe algum vocábulo, ele reconhece o que foi dito.

Referência:

https://www.mankier.com/1/pocketsphinx_continuous

Parâmetros:

-adcdev – Nome do dispositivo de áudio que será utilizado para captura

-adcin – Arquivo de áudio tipo raw

-dict – Arquivo de dicionário de voz

-fdict – Arquivo de Pronunciação vocal

-hmm – Arquivos de Modelos Acústicos

-hyp – Arquivo de saída do que foi reconhecido

O Parâmetro hmm e dict são obrigatórios, e os parâmetros lm e fsg apenas se estiver usando um modelo de linguagem.

Exemplo de comando na console:

```
pocketsphinx_continuous -adcdev plughw:1,0 -kws_threshold 1e-20 -inmic yes
```

Lembrando que o parâmetro -adcdev plughw:1,0 depende do hardware que estiver usando, se não funcionar, tire ele!

Desenvolvimento e Integração com Aplicações em C

Agora, iremos construir um exemplo prático usando cmu sphinx.

Para compilar uma aplicação Sphinx, use a seguinte sintaxe:

Compilação Manual

Abaixo o arquivo Makefile

```
#HEADERS = program.h headers.h
CC= gcc
MYSQLCFLAGS= -I/usr/include/mysql -DBIG_JOINS=1 -fno-strict-aliasing -g
SPHINXFLAGS= `pkg-config --cflags --libs pocketsphinx sphinxbase`
MYSQLLIBS= -L/usr/lib/mysql -lmysqlclient -lpthread -lz -lm -lrt -ldl
SPHINXLIBS= -DMODELDIR=\"`pkg-config --variable=modeldir pocketsphinx`\"
default: compile
all: clean compile install run
compile:
    #gcc -o hello_ps hello_ps.c \
    -DMODELDIR=\"`pkg-config --variable=modeldir pocketsphinx`\\" \
`pkg-config --cflags --libs pocketsphinx sphinxbase` \
    $(CC) main.cpp -g -o srvOuve $(MYSQLCFLAGS) $(SPHINXFLAGS) $(MYSQLLIBS) $(SPHINXLIBS)
clean:
    rm -f srvOuve
    rm -f /usr/local/bin/srvOuve
run:
    /usr/local/bin/srvOuve
install:
    cp ./srvOuve /usr/local/bin/
```

Existem duas

fontes em c na pasta /robotinics/sphinx/srvOuve, o main.cpp tem a entrada de voz pelo microfone e foi obtido na integra por fóruns da internet.

O main.c, é um código-fonte mais simples e foi obtido como material do site da documentação, que passarei em breve.

Não irei apresentar o fonte em si neste livro, porém comentarei as principais funções, deixando no repositório para download o arquivo.

Iremos comentar o main.c a seguir:

Na linha primeira linha, vemos o chamado da biblioteca pocketsphinx.h.

Na linha abaixo, passamos os parâmetros, assim como fazemos no pocketsphinx_continuos,

```
config = cmd_ln_init(NULL, ps_args(), TRUE,
                     "-hmm", MODELDIR "/en-us/en-us",
                     "-lm", MODELDIR "/en-us/en-us.lm.bin",
                     "-dict", MODELDIR "/en-us/cmudict-en-us.dict",
                     NULL);
```

A mágica acontece neste segmento de código,

```
rv = ps_start_utt(ps);

    while (!feof(fh)) {
        size_t nsamp;
        nsamp = fread(buf, 2, 512, fh);
        rv = ps_process_raw(ps, buf, nsamp, FALSE, FALSE);
    }

    rv = ps_end_utt(ps);
    hyp = ps_get_hyp(ps, &score);
    printf("Recognized: %s\n", hyp);
```

que inicia o processo de captura, processando até encontrar algum padrão reconhecido, finalizando com sua respectiva apresentação.

Toda a API do Sphinx é documentada, e pode ser analisada através do site:

<https://cmusphinx.github.io/wiki/>

Modelo de linguagem

O Sphinx permite construir modelo de linguagem usando 3 padrões:

- Lista de Palavras-chave – Arquivo que contém uma lista de palavras chaves, que são pesquisadas.
- Modelo Gramatical – Utiliza uma estrutura (sintaxe) construtiva, que permite gerar comandos válidos. Como:
 - COMPUTADOR [LIGAR/DESLIGAR] [LUZ/IRRIGACAO/VALVULA]
- Modelo de Linguagem – Prevê a análise estatística do que é dito, a fim de determinar qual a probabilidade estatística de semelhança com um conjunto de palavras cadastradas.

Atualmente o uso de Modelo de linguagem tem se destacado. Porém sua concepção e desenvolvimento exige um volume grande de treinamento.

O Dicionário fonético faz parte integrante do modelo de linguagem.

O dicionário fonético fornece um conjunto de fonemas em sequência que identificam os sons.

Cada idioma possui seu dicionário fonético. Hoje estou utilizando o idioma fonético americano para testes, pois o dicionário para português, até o ponto que sei, é pouco desenvolvido.

Para desenvolver seu próprio dicionário, segue link de ferramenta que auxilia nesta tarefa.

<http://www.speech.cs.cmu.edu/tools/lmtool-new.html>

Infelizmente esse dicionário virá com a descrição fonética em inglês.

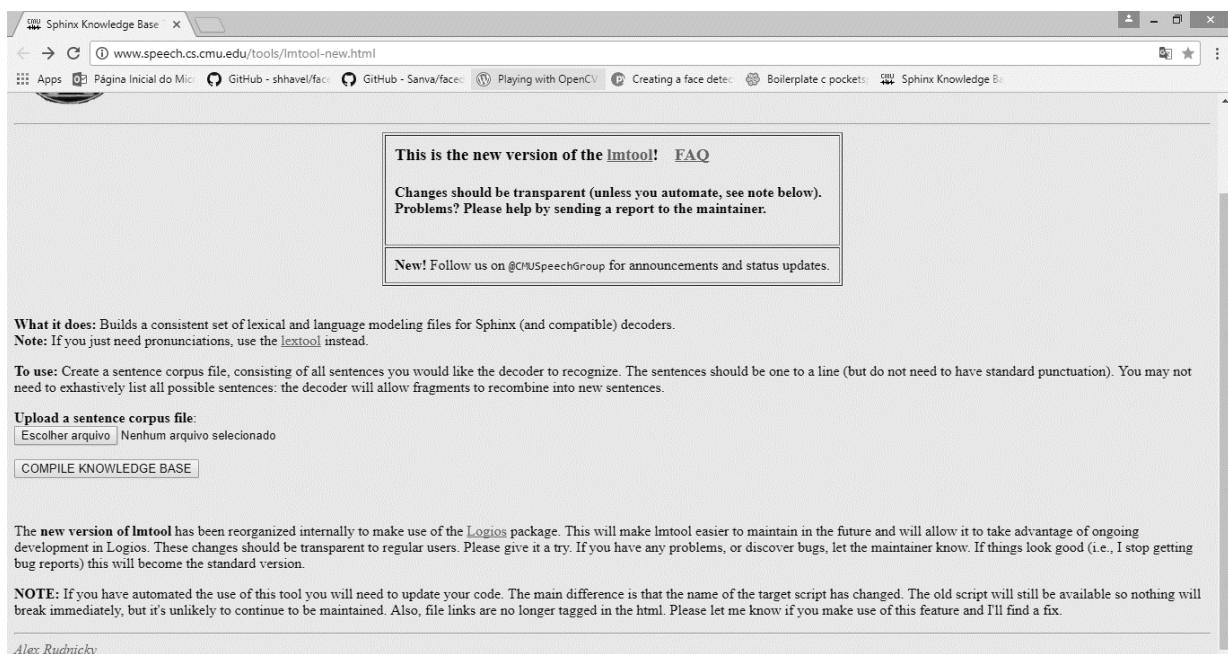


Figura 170 - Tela de criação de dicionário fonético

Resultado do site, já com comandos processados.

Your Sphinx knowledge base compilation has been successfully processed!

The base name for this set is **3433**. **TAR3433.tgz** is the compressed version.
Note that this set of files is internally consistent and is best used together.

IMPORTANT: Please download these files as soon as possible; they will be deleted in approximately a half hour.

Name	size	Description
3433.dic	479	Pronunciation Dictionary
3433.lm	3.0K	Language Model
3433.log_pronounce	380	Log File
3433.sent	322	Corpus (processed)
3433.vocab	138	Word List
TAR3433.tgz	1.6K	COMPRESSED TARBALL

Please include these messages in bug reports.

Apache/2.2.22 (Ubuntu) Server at www.speech.cs.cmu.edu Port 80

Figura 171 - Tela de retorno, permite baixar os arquivos gerados

Incluindo novo modelo de linguagem no cmu-sphinx

Por padrão o Sphinx inclui os modelos de linguagem na pasta:

/usr/local/share/pocketsphinx/model/

Estaremos incluindo em nosso pacote um modelo fonético criado por terceiro.

<https://twitter.com/pehlimaofficial>

O modelo fica na pasta /sphinx/model/pt-br/

Pedro Lima, obrigado pelo auxílio, seu vídeo foi muito instrutivo.

Para tanto basta copiar o arquivo em Linux/sphinx/pt-br.zip, e descompactar na pasta indicada a cima.

A referência no programa em C, deve ser como se segue:

```
config = cmd_ln_init(NULL, ps_args(), TRUE,
    "-hmm", MODELDIR "/pt-br/pt-br",
    "-lm", MODELDIR "/pt-br/pt-br.lm.bin",
    "-dict", MODELDIR "/pt-br/pt-br.dic",
    NULL);
```

Redes Neurais

As redes neurais são um tema muito complexo, e por este motivo apresentaremos apenas alguns conceitos desta tecnologia, aplicada exclusivamente à prática do uso da lib FANN.

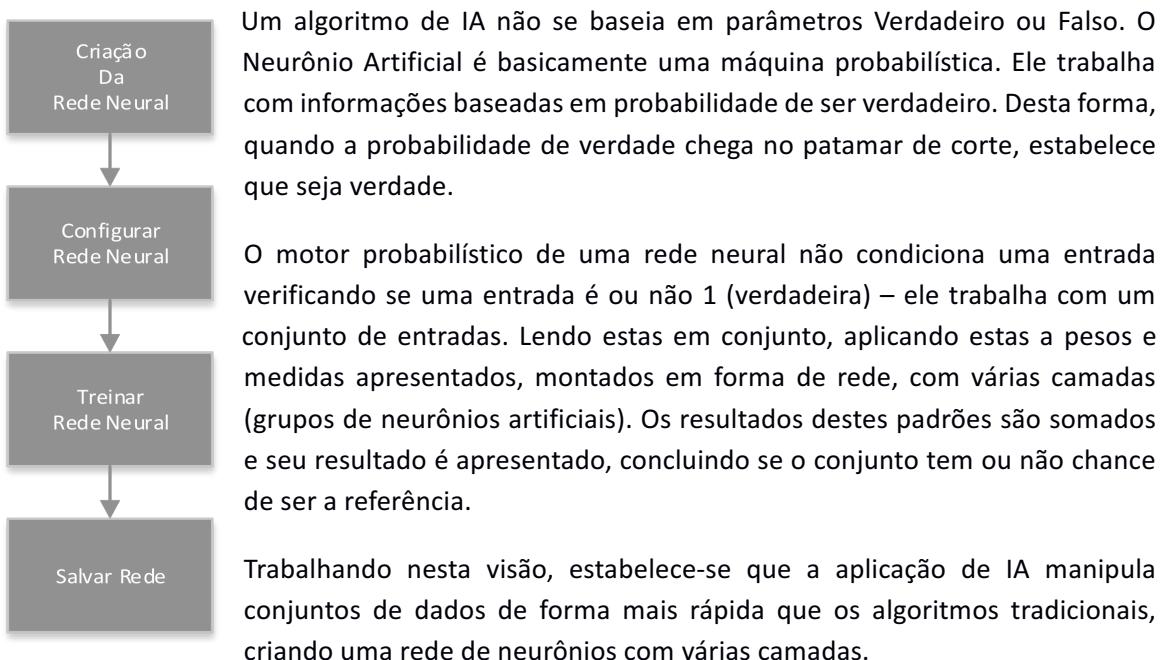
Os cientistas estudaram o funcionamento de nosso cérebro, vendo como as células (neurônios) funcionam. A partir deste estudo, os cientistas e projetistas de software criaram algoritmos que simulam o funcionamento e as respostas destes neurônios.

Desta forma, abordaremos na prática como isso é feito.

O projeto de uma rede neural é bem diferente de um projeto de algoritmo tradicional.

Vamos abordar o seguinte exemplo de algoritmo tradicional.

No fluxo tradicional, o computador capture as entradas, e, baseado em uma operação lógica (verdadeiro ou falso), ele realiza o desvio para um processo verdade ou um processo falso. Assim, o computador fica amarrado a verdades absolutas, concretas e pouco flexíveis.



Concluindo esta visão simplista do conceito, podemos dizer que uma rede neural apresenta dois momentos:

- Treinamento – Criação de casos onde ocorre o caso, para calibrar a rede
- Aplicação – Usa-se a rede neural, a fim de verificar um caso.

O treinamento é a fase em que mostramos para nossa rede um conjunto de casos no aspecto que queremos aferir.

A qualidade de nossa rede depende da qualidade do treinamento que aplicamos.

Neste momento, criamos os pesos para que nossa rede possa reconhecer o padrão quando exposto a ele.

A aplicação é, como o próprio nome diz, o uso da rede, aferindo se um caso é ou não reconhecido como padrão.

Como se pode perceber, uma rede não diz 100% de certeza ou 100% de impossibilidade. Em vez disso, informa um percentual, em que o resultado aplicado a uma nota de corte diz se é verdade ou falso.

A rede neural carrega o arquivo contendo os pesos dos neurônios. Assim, torna-se possível aplicar a uma entrada (INPUTS) e obter o resultado (OUTPUT).

Instalação do FANN

Para instalar o FANN, siga os passos:

- 1) Crie uma pasta no Linux chamado FANN. Para efeitos de padronização, criei em
`/home/pi/projetos/fann`
- 2) Baixe o arquivo do FANN no site <http://sourceforge.net/projects/fann> ou execute o comando abaixo:
`wget http://sourceforge.net/projects/fann/files/fann/2.2.0/FANN-2.2.0-Source.tar.gz/download`
- 3) Descompacte o arquivo tar –xzf download
- 4) Vá na pasta do projeto, usando o comando cd `FANN-2.2.0-Source`

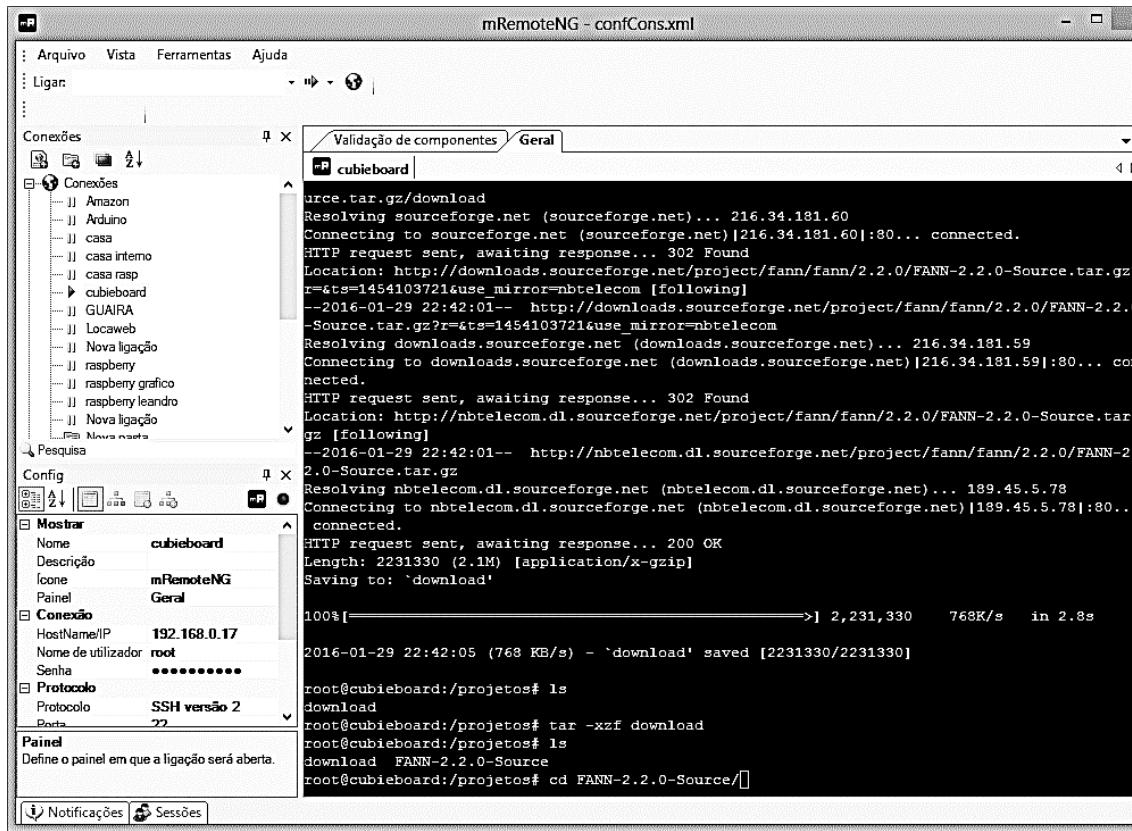


Figura 172 - Tela da console com retorno de dados

- 5) Instale os seguintes programas: apt-get install cmake make g++ gcc libgdal-dev libgs10-dev libarmadillo-dev
- 6) Na console, digite o seguinte comando:
cmake
- 7) Na console, digite o seguinte comando:
make all
- 8) Na console, digite o seguinte comando:
make install
- 9) Para mais informações digite
make help
- 10) Na console, digite o seguinte comando:
ldconfig

Projeto FANN

O projeto FANN, sigla de Fast Artificial Neural Network, é uma biblioteca destinada à implementação de algoritmos de inteligência artificial.

O arquivo pode ser baixado pelo link <http://sourceforge.net/projects/fann>.

No caso do nosso exemplo do tópico anterior, possuímos apenas duas entradas, ou seja, dois parâmetros na nossa lista (peso e altura). Porém, nossa lista pode possuir quantas entradas quisermos.

No fluxo ao lado, podemos ver quatro etapas para este processo:

A criação da rede consiste em montagem de uma estrutura de rede neural que permita a aplicação do treinamento.

A sua aplicação pode ser criada conforme o exemplo:

```
struct fann *ann = fann_create_standard(num_camadas, num_entradas,  
num_neuronios_ocultos, num_saidas);
```

Salvar rede – Ao gerar uma rede neural, os valores da matriz da rede podem ser salvas em arquivo para que sejam utilizadas em outros momentos. Desta forma, a rede neural não precisa ser sempre treinada – basta carregar o modelo salvo e aplicar ao que se pretende medir.

```
fann_save(ann, "robotinics.net");
```

Carregar uma rede pré existente – o comando `fann_create_from_file("robotinics.net")` – cria uma rede baseada nas definições da rede que treinamos e previamente salvamos.

Para aplicarmos os dados a uma rede, devemos utilizar o comando `fann_run(ann, input)`, que pega as informações de entrada (`input`) e o `ann` (ponteiro da rede criada), gerando, como resultado, um ponteiro de saída dos resultados esperados.

Modelo de Fonte de Treino de Rede Neural

```
#include "fann.h"  
  
int main()  
{  
    const unsigned int num_input = 2;  
    const unsigned int num_output = 1;  
    const unsigned int num_layers = 3;  
    const unsigned int num_neurons_hidden = 3;  
    const float desired_error = (const float) 0.001;
```

```
const unsigned int max_epochs = 500000;
const unsigned int epochs_between_reports = 1000;

struct fann *ann = fann_create_standard(num_layers, num_input,
                                         num_neurons_hidden, num_output);

fann_set_activation_function_hidden(ann, FANN_SIGMOID_SYMMETRIC);
fann_set_activation_function_output(ann, FANN_SIGMOID_SYMMETRIC);

fann_train_on_file(ann, "robotinics.data", max_epochs,
                   epochs_between_reports, desired_error);
/* Salvando arquivo de rede neural */
fann_save(ann, "robotinics.net");

fann_destroy(ann);

return 0;
}
```

Modelo de Análise de entrada em Rede Neural

```
#include <stdio.h>
#include "floatfann.h"

int main()
{
    fann_type *calc_out;
    fann_type input[2];

    struct fann *ann = fann_create_from_file("robotinics.net");
    /*Valores das entradas dos Sinais */
    input[0] = -1;
    input[1] = 1;
    calc_out = fann_run(ann, input);

    printf("Robotinics Resultado: (%f,%f) -> %f\n", input[0], input[1], calc_out[0]);

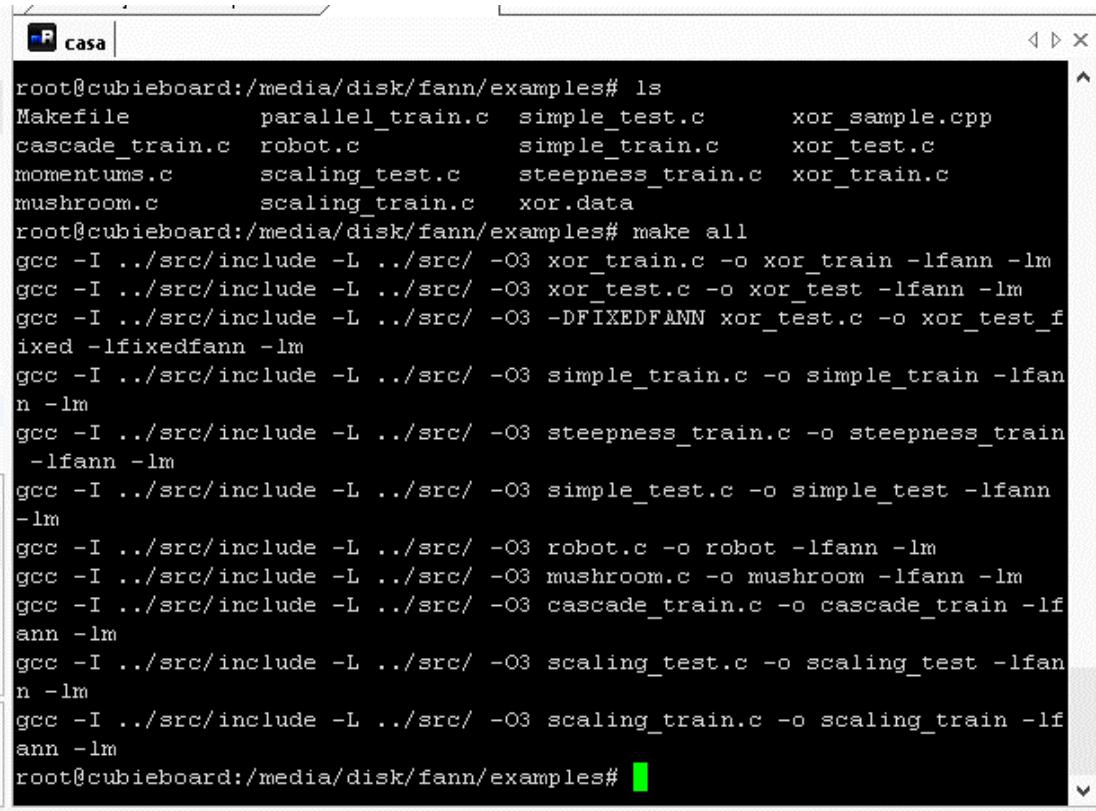
    fann_destroy(ann);
    return 0;
}
```

Ao carregar os padrões de valores dados no campo input, aplica-se os padrões à rede pelo comando fann_run. Desta forma, a rede neural retornará o resultado em calc_out.

Compilando Exemplos de Rede Neural

Siga os procedimentos descritos a seguir:

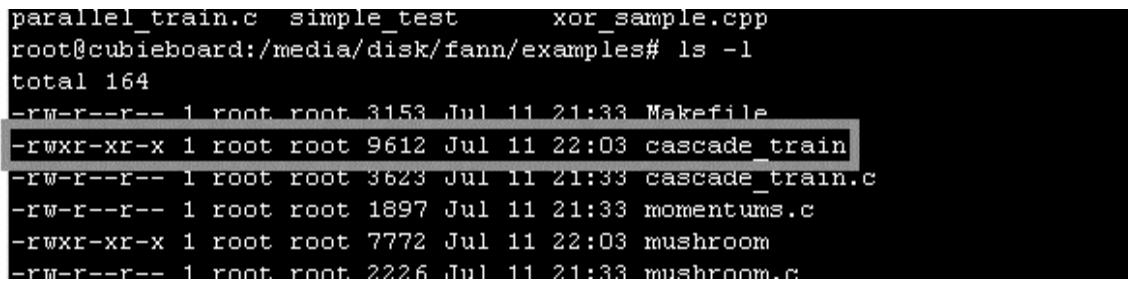
1. Entre na pasta FANN
2. Entre na subpasta examples usando o comando: cd exemples
3. Compile os códigos usando: make all



The screenshot shows a terminal window titled 'casa' with a black background and white text. It displays the command 'root@cubieboard:/media/disk/fann/examples# ls' followed by a list of files: Makefile, parallel_train.c, simple_test.c, xor_sample.cpp, cascade_train.c, robot.c, simple_train.c, xor_test.c, momentums.c, scaling_test.c, steepness_train.c, xor_train.c, mushroom.c, scaling_train.c, xor.data. Below this, the command 'make all' is run, followed by the output of multiple gcc compiler runs for each example. The terminal ends with 'root@cubieboard:/media/disk/fann/examples#'. A green cursor is visible at the bottom right.

Figura 173 - Compilação dos Exemplos

4. Liste os exemplos, usando: ls -l



The screenshot shows a terminal window with the same directory as Figure 173. The command 'ls -l' is run, showing the following file list:
parallel_train.c simple_test xor_sample.cpp
root@cubieboard:/media/disk/fann/examples# ls -l
total 164
-rw-r--r-- 1 root root 3153 Jul 11 21:33 Makefile
-rwxr-xr-x 1 root root 9612 Jul 11 22:03 cascade_train
-rw-r--r-- 1 root root 3623 Jul 11 21:33 cascade_train.c
-rw-r--r-- 1 root root 1897 Jul 11 21:33 momentums.c
-rwxr-xr-x 1 root root 7772 Jul 11 22:03 mushroom
-rw-r--r-- 1 root root 2226 Jul 11 21:33 mushroom.c

Figura 174 - Executando o programa exemplo

Todos os arquivos com x nas permissões são executáveis. No exemplo acima temos o **cascade_train**. Para rodar, digite: **./cascade_train**

5. Para formar a rede neural usamos dois arquivos e dois executáveis. O primeiro usa a informação de treinamento, que fica no código-fonte ls

6. Conforme linha abaixo:

```
train_data = fann_read_train_from_file("../datasets/parity8.train");
```

O arquivo **parity8.train** contém a relação de situações que representam a entrada e saída, ajustando o neurônio para calibrar a rede neural.

O resultado da calibração da rede neural é armazenado no arquivo **cascade_train2.net**, conforme código a seguir: `fann_save(ann, "cascade_train2.net");`

Como compilar um novo projeto

Um novo projeto de rede neural pode ser facilmente compilado usando a seguinte sintaxe:

```
gcc -O3 [fonte] -o [binário] -lfann -lm
```

6. Finalizando o Projeto

6.1 Mecânica – Cabeça do Robô

Finalmente estamos no último capítulo. Estamos acabando o processo de montagem do nosso robô. Pudemos vivenciar a longa jornada nesse novo mundo, onde pudemos acompanhar o passo a passo na montagem, verificando todos os detalhes e procedimentos.

Resumindo nosso projeto até o momento:

- 1 – Conceitos básicos
- 2 – Base do robô com rodas
- 3 – Finalização da base
- 4 – Montagem do corpo do robô
- 5 – Montagem dos braços

A cabeça será o último elemento do projeto mecânico.

Os elementos mecânicos e ligações necessárias serão avaliados até a montagem final.

Montagem da Cabeça

A cabeça é formada por três segmentos distintos, conforme apresentado na imagem:

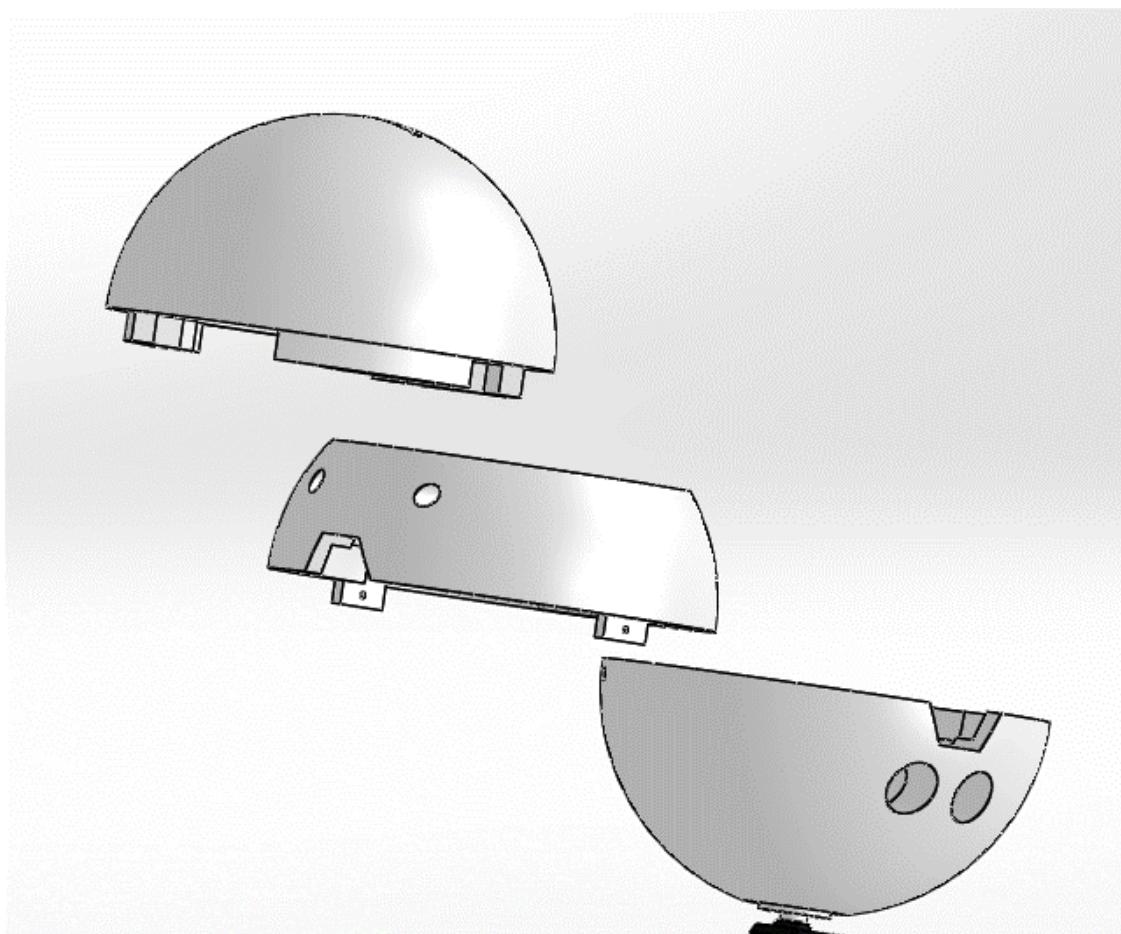


Figura 175 - Visão das partes da cabeça do robô

Seguindo de baixo para cima:

- Cabeça Inferior
- Cabeça Superior
- Chapéu

Cada segmento tem uma função específica no projeto.

A cabeça inferior, além de dar suporte ao restante da cabeça, pois possui os elementos de fixação ao corpo, também possui o sensor de ultrassom e a abertura da câmera.

A cabeça superior possui a fixação do Raspberry, que fica localizado em uma placa parafusada sobre a cabeça superior. A cabeça superior também possui conexão com servomotores para fixação do laser, que pode ser incluído para esquadrihar a área.

E, por último, o chapéu finaliza a cabeça, apresentando um aspecto iluminado, para que represente o humor do robozinho.

Iniciando a montagem da cabeça

Ao iniciarmos a montagem da cabeça, encontraremos os servomotores conectados ao corpo.

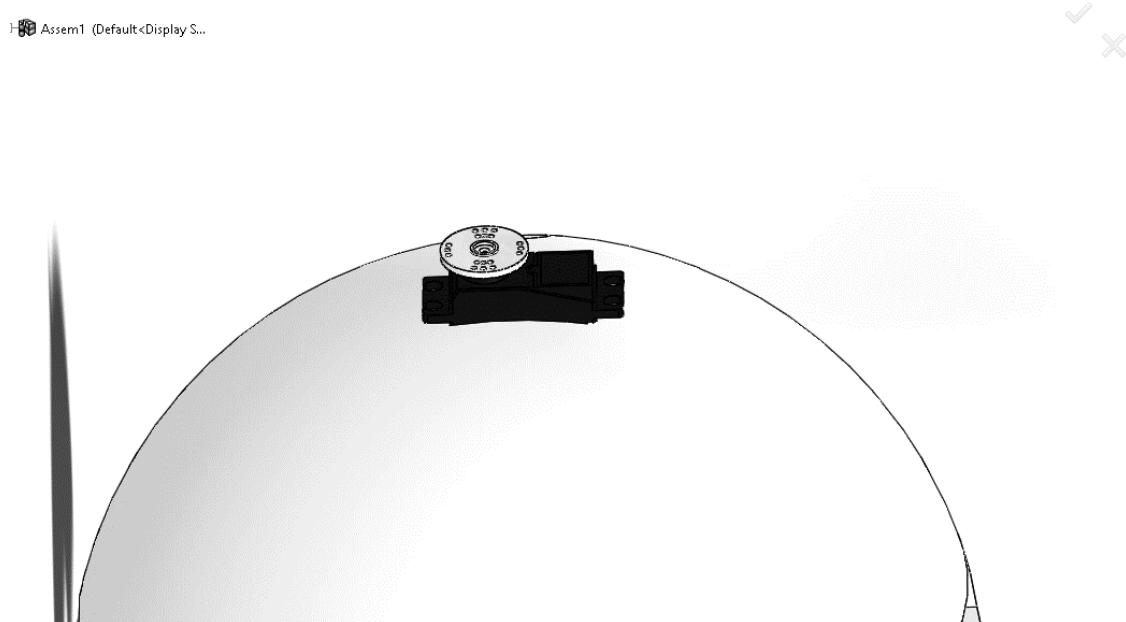


Figura 176- Corpo do robô com o servomotor

Desta forma, fixaremos a peça da cabeça inferior na base do servo, conforme indicado na figura a seguir:

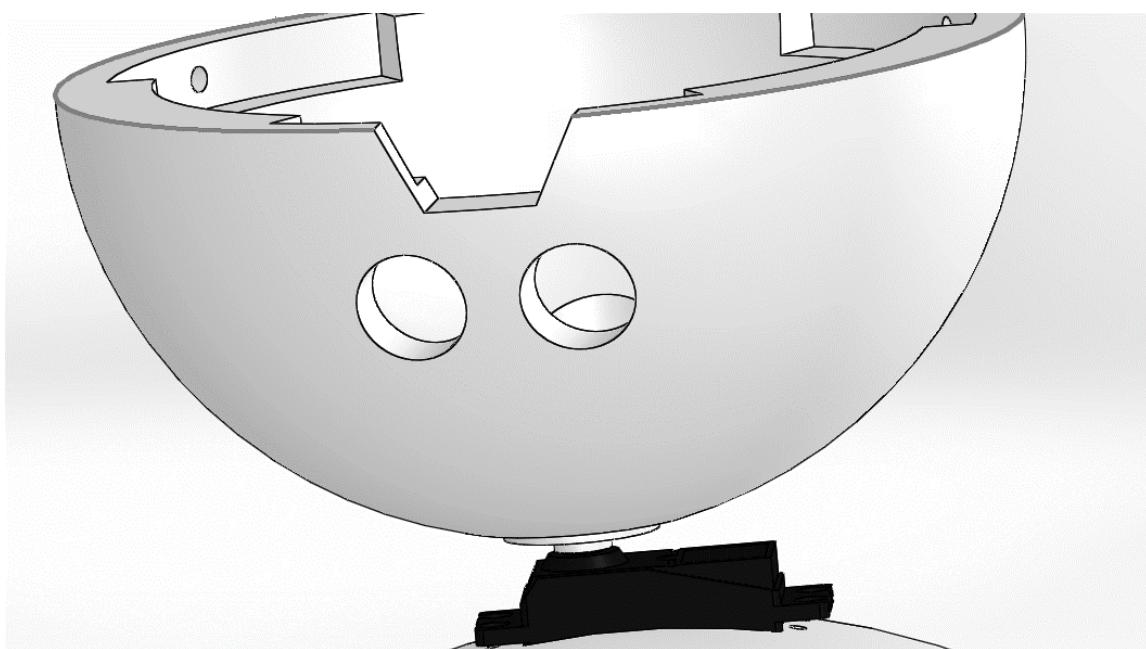


Figura 177 - Cabeça inferior conectada ao corpo

6.3 Software

PHPMYADMIN

O gerenciamento de um portal ou interface web que contenha dados em banco requer uma ferramenta de prospecção de dados.

O phpmyadmin fornece tal interface. Trataremos os detalhes básicos sobre esta ferramenta.

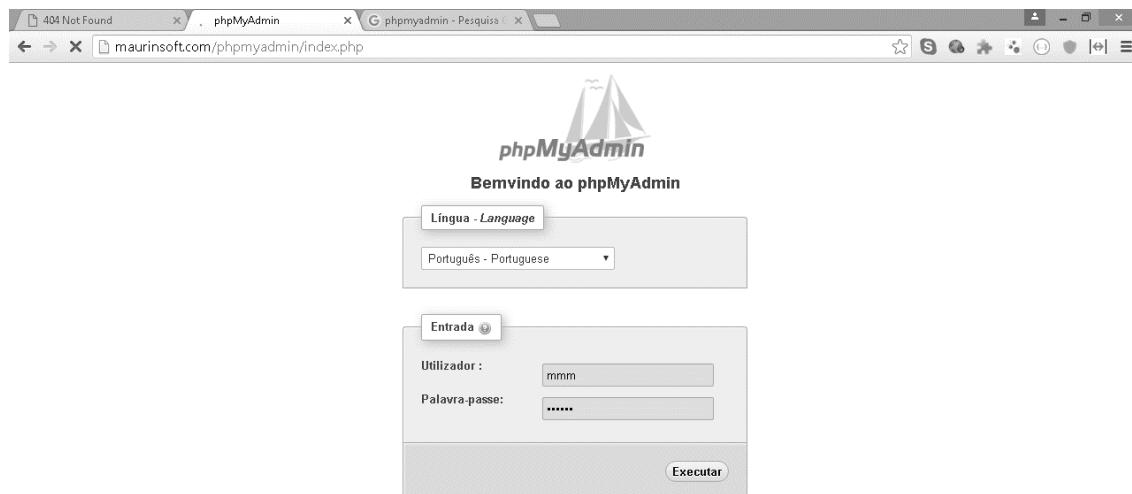


Figura 178 - Interface web phpmyadmin

Instalação do phpmyadmin

Para a instalação do phpmyadmin, digite o seguinte comando: `apt-get install phpmyadmin`

```
root@cubieboard:~# apt-get install phpmyadmin
Reading package lists... Done
Building dependency tree
Reading state information... Done
phpmyadmin is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 35 not upgraded.
26 not fully installed or removed.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Figura 179 - Instalando pacote phpmyadmin

Acessar o phpmyadmin

Para acessar o phpmyadmin, deve-se acessar o IP do servidor da seguinte maneira:

Meu IP/phpmyadmin

The screenshot shows the phpMyAdmin interface. On the left, there's a sidebar with a tree view of databases: 'Recente' (empty), 'Favoritos' (empty), 'New', 'information_schema', 'jomadadb', 'mysql', 'performance_schema', 'phpmyadmin', 'robotinicsdb' (selected, labeled 1), 'robofisica', 'users'. The main content area shows two tables: 'security' and 'users' (labeled 2). Below them is a 'Criar tabela' (Create Table) dialog box with fields 'Nome:' and 'Número de colunas:' set to 4 (labeled 3). The top navigation bar includes links for Estrutura, SQL, Pesquisar, Exportar, Importar, Operações, Privilégios, and Mais.

Figura 180 - Mostrando tabela do robotinics

Na imagem acima podemos perceber três áreas:

1. Menu de navegação – Permite visualizar as informações relativas aos bancos associados
2. Informações das estruturas (tabelas) – Associadas ao banco selecionado no item 1
3. Detalhamentos – Apresentam os detalhamentos associados às estruturas associadas

Entendendo o Projeto Web

Não temos aqui o objetivo de explicar programação em php, porém apresentar algumas informações necessárias para que qualquer programador com algum conhecimento em PHP possa desenvolver e manter alterações no portal.

Uma das primeiras ações que temos em um projeto web é a instalação e posteriormente o deploy (implantação) dos fontes.

Então vamos lá:

Instalando a interface web

As seguintes aplicações são associadas à web, sendo que muitas já foram mencionadas em capítulos anteriores, porém apenas mencionaremos seus instaladores.

- ✓ apt-get install apache2
- ✓ apt-get install mysql-server
- ✓ apt-get install libapache2-mod-php5 php5 php-pear php5-xcache php5-mysql php5-curl php5-gd

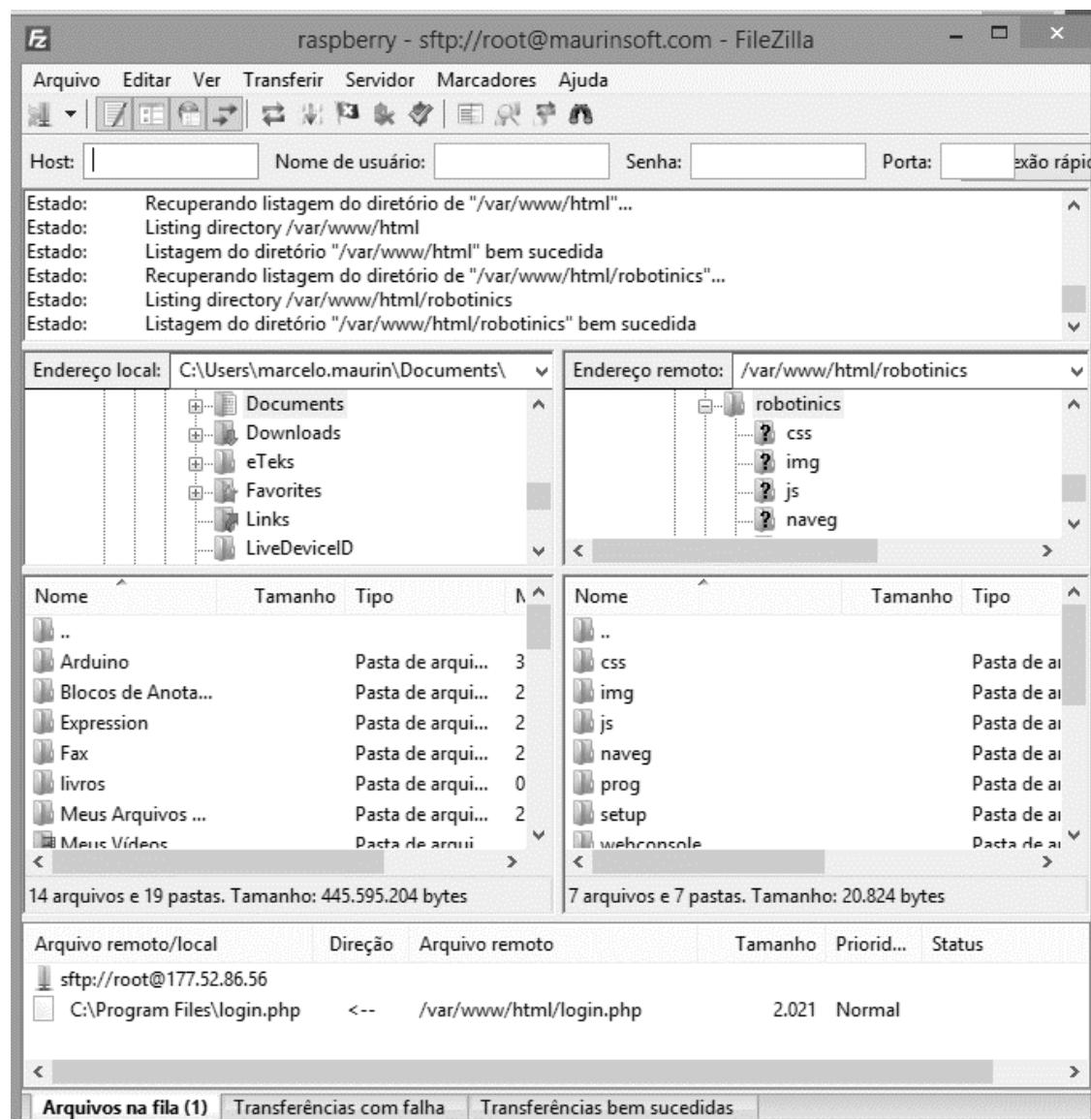


Figura 181 - Tela do Filezilla

Implantando os scripts

Os scripts da interface web ficam na pasta do projeto robotinics/linux/web e devem ser copiados para a pasta do robô, em /var/www/html.

Para realizar esta cópia, podemos utilizar o Filezilla.

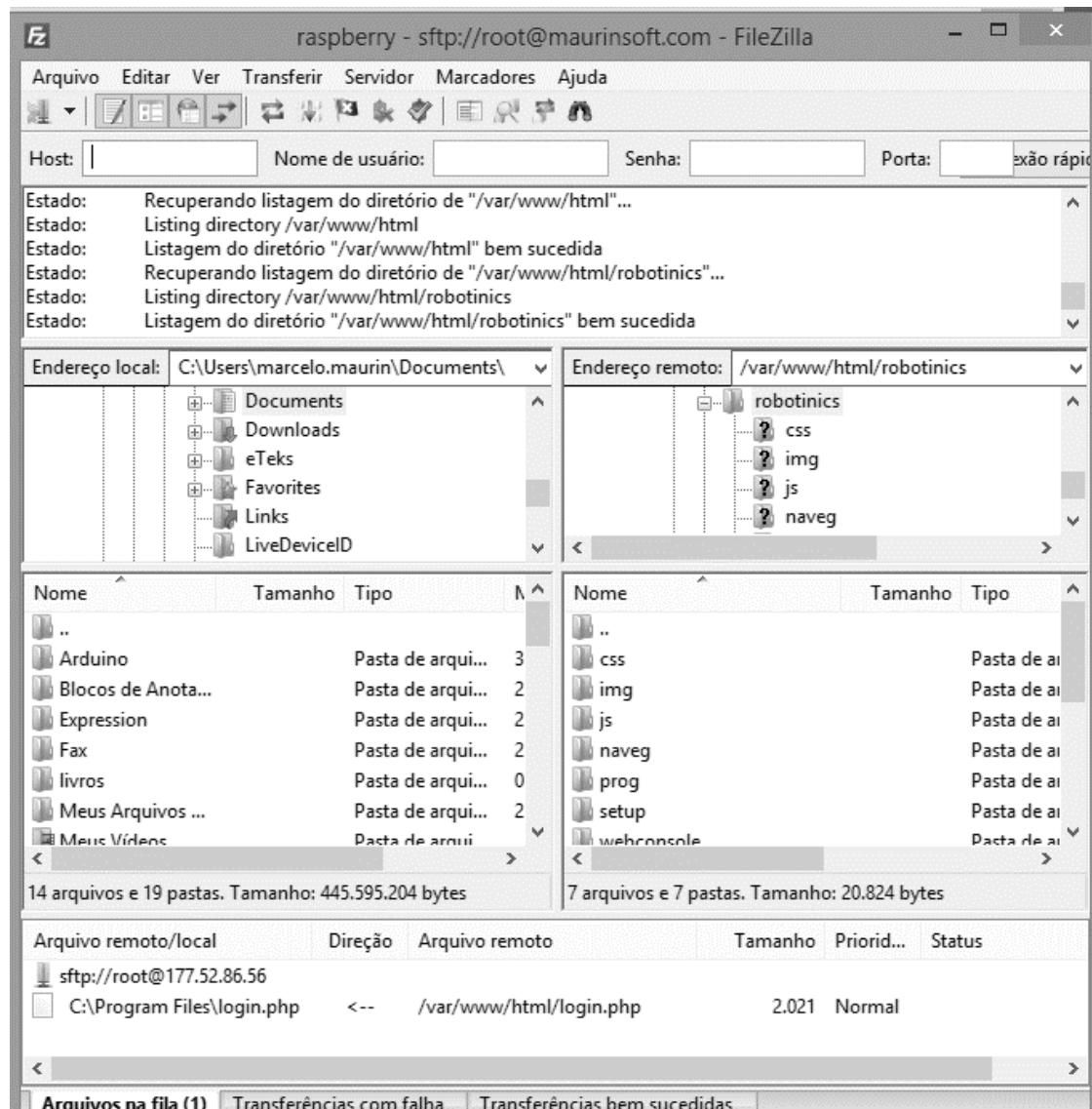


Figura 182- Cópia e configuração dos scripts da interface web do Robotinics

Criação e Configuração do Banco Web

Os scripts do banco de dados web ficam na pasta MySQL do projeto do Robotinics.

Para executar, siga os procedimentos descritos abaixo:

1. Entre no PHPMyAdmin digitando o [endereço ip]/phpmyadmin/index.php. Caso não tenha configurado o PHPMyAdmin, faça-o antes de executar esse procedimento.

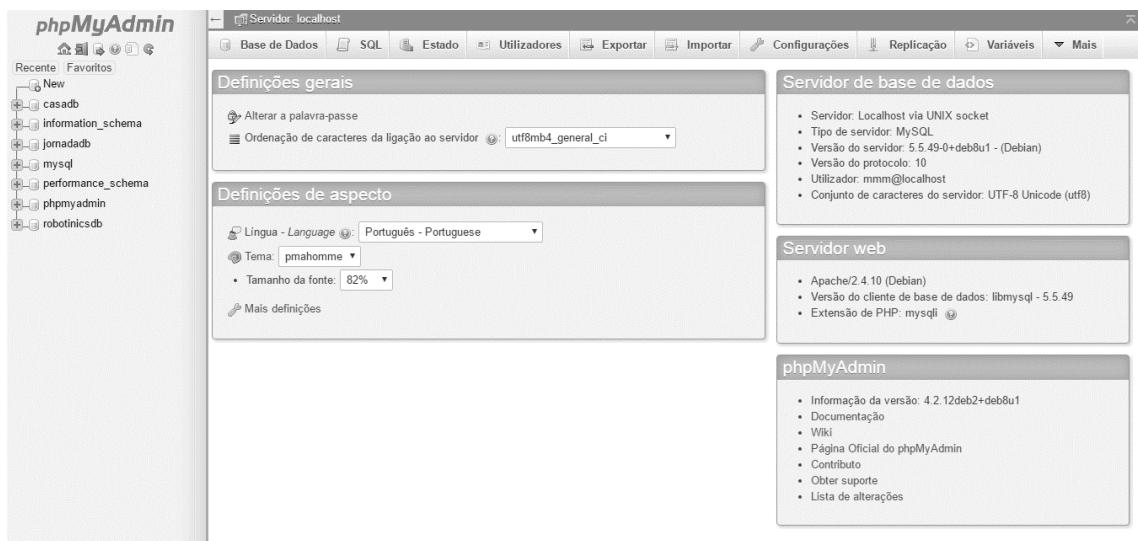


Figura 183 - Informações sobre o banco de dados

2. Crie um novo banco de dados usando a opção NEW, Criar base de Dados, incluindo o nome Robotinicsdb, com padrão latin1_swedish_ci para COLLATION.

Base de Dados

A screenshot of a 'Criar base de dados' (Create Database) form. It has a 'Nome da base de dados' (Database name) input field containing 'Robotinicsdb', an 'Agrupamento (Collation)' dropdown set to 'latin1_swedish_ci', and a 'Criar' (Create) button.

Figura 184- Procedimento para criar novo banco de dados

3. Clique no painel onde aparece o ícone de atualizar ..
4. Aparecerá o banco de dados Robotinicsdb, conforme criado no passo 1

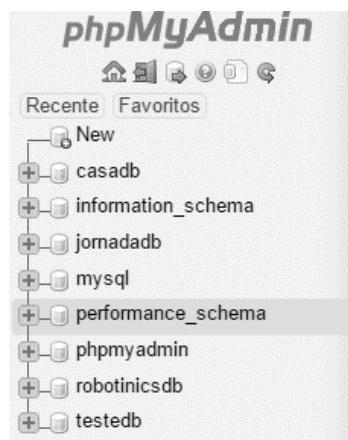


Figura 185- Visão dos bancos disponíveis no servidor

5. Clique no item NEW, abaixo do Robotinics

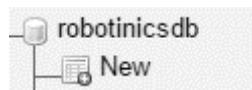


Figura 186- Opção Novo

6. Na aba direita surgirá um conjunto de abas, conforme apresentado a seguir. Selecione o item SQL.

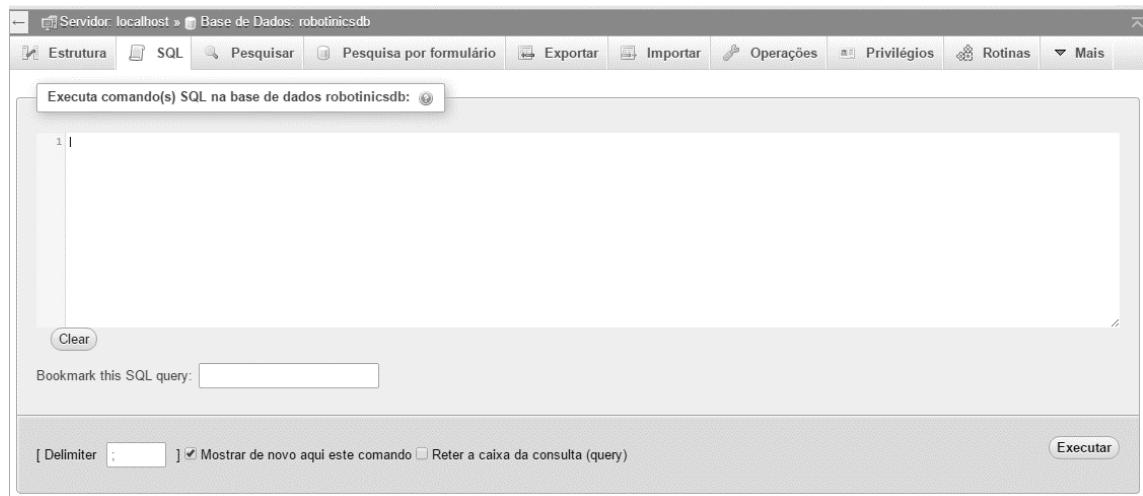


Figura 187- Inclusão de scripts no banco de dados

7. Copie os scripts da pasta de MySQL do livro, Users e Security.sql, e, se houver outros, repita todos os scripts.

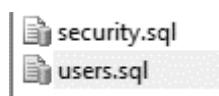


Figura 188 - Script a ser executado

8. Pronto: você criou as tabelas necessárias!

Terminal remoto via web

Muitas vezes a utilização de uma ferramenta web facilita muito o uso e operação do robô.

Fabricante:

<http://liftoffsoftware.com/Products/GateOne>

O GateOne é uma ferramenta dual, licenciado sobre AGPL v3 e Comercial. Seus fontes podem ser obtidos através do site Github:

Para baixar a ferramenta, siga os passos abaixo:

1. Entre na pasta do Robotinics
 2. Instale o python, apt-get install python
 3. Com usuário administrador, digite: git clone <http://liftoffsoftware.com/Products/GateOne>
 4. Aguarde os fontes serem baixados e entre na pasta gerada pelo git
 5. Para instalar, digite pid install stdeb
 6. Após instalado, basta digitar service gateone start

```
Using username "root".  
[ ok ] Starting Gate One daemon: gateone.py.  
root@cubieboard:~# █
```

Figura 189- Start gateone

Servidor de Data e Hora

A configuração de Serviço de Data e Hora é muito importante para o robô, pois o Raspberry e o Cubieboard não possuem relógio interno.

Para configurar o relógio interno, basta seguir os seguintes passos:

- 1) Na console, instale o servidor de ntp, digitando na console `sudo apt-get install ntp`

```
root@cubieboard:/etc/motion# sudo apt-get install ntp
Reading package lists... Done
Building dependency tree
Reading state information... Done
ntp is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 35 not upgraded.
26 not fully installed or removed.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Figura 190- Instalando o ntp

- 2) Ainda na console, rode o comando `sudo dpkg-reconfigure tzdata`, escolhendo a região onde mora.

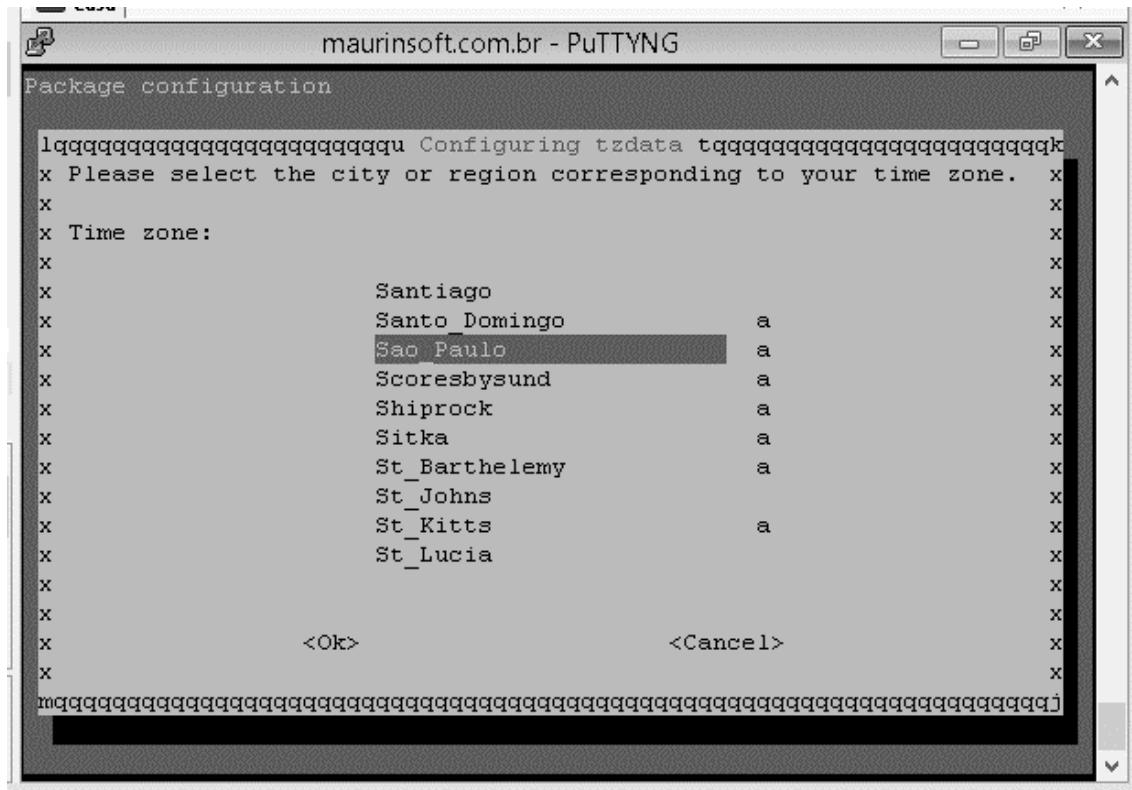


Figura 191 - Seleção do fuso horário por estado/país

- 3) Abra, com seu editor preferido, o arquivo de configuração do ntp (/etc/ntp.conf), realizando as seguintes modificações:
 - a. Server 127.127.1.0
 - b. Fudge 127.127.1.0 stratum 10
 - c. Restrict 192.168.1.0 mask 255.255.255.0 nomodify
- 4) Inclua os servidores de ntp da sua região, incluindo no caso do Brasil os itens abaixo, no arquivo /etc/ntp.conf
 - a. server 0.BR1 a.ntp.br 200.160.0.8
 - b. server 1.BR2 b.ntp.br 200.189.40.8
 - c. server 2.BR c.ntp.br 200.192.232.8
- 5) Reinicie o serviço pela console: sudo service ntp restart
- 6) Verifique se houve a mudança pela console, digitando sudo date

Para não dizer que não falei de GPIO

GPIO – É a sigla para General Purpose Input/Output, e são pinos disponíveis no raspberry e cubieboard que permitem a integração com hardwares adicionais, tais como fazemos no arduino.

O Desenvolvimento de GPIO foge do escopo deste livro, porém estarei apresentando uma solução baseada em C que utilizo.

O WiringPi é uma solução de controle do GPIO semelhante a utilizada no arduino, suas funções são muito próximas as escritas em arduino, o que permite uma curva de aprendizado muito rápida.

Para maiores informações entre no site:

<http://wiringpi.com/>

Instalação

Para instalar o WiringPi siga os passos a seguir:

- 1) Entre no terminal do seu robô

- 2) Na pasta dos projetos do Robo digite a seguinte linha:

```
git clone https://github.com/WiringPi/WiringPi.git
```

- 3) Aguarde baixar os fontes, após, surgirá uma pasta chamada WiringPi, Entre na pasta, com o comando cd WiringPi

- 4) Rode o comando para compilar o programa, com a seguinte sintaxe:

```
./build
```

- 5) Aguarde a compilação e pronto, você já tem o wiringPi no seu robô.

Primeiro Exemplo

Sempre que falamos de primeiro exemplo, em especial para hardware, pensamos no blink, que é uma piscadela de um led.

Desta forma podemos ver o controle mais simples possível de um pino do GPIO.

Na pasta do WiringPi, existe uma subpasta exemples: "WiringPi/examples"

O blink.c esta na pasta de exemples, e segue um fragmento.

```
#include <stdio.h>
#include <wiringPi.h>
// LED Pin - Em wiringPi pin 0 é BCM_GPIO 17.
#define LED    0

int main (void)
{
    printf ("Raspberry Pi blink\n") ;
    wiringPiSetup () ;
    pinMode (LED, OUTPUT) ;
    for (;;)
    {
        digitalWrite (LED, HIGH) ;      // On
        delay (500) ;                // mS
        digitalWrite (LED, LOW) ;     // Off
        delay (500) ;
    }
    return 0 ;
}
```

Segue breve explicação sobre sua implementação:

O include <wiringPi.h> é a chamada da biblioteca propriamente dita.

O define LED é o Pino que será utilizado, no caso 0 é o GPIO 17.

O wireingPiSetup inicializa a biblioteca assumindo o padrão de numeração do GPIO, onde 0 é o pino 17 e assim por diante.

`pinMode` atribui o pino LED, como **output** (saída), pois pinos do GPIO podem receber sinais (**INPUT**) ou mandar sinais (**output**).

`digitalWrite` muda o valor do pino LED, podendo ter estado **HIGH** (alto) ou **LOW** (baixo), no caso de **HIGH**, o led será ligado, e **LOW** ficará desligado.

Não entraremos aqui no detalhamento do esquema elétrico para implementação do led, pois já foi visto em capítulos anteriores.

Compilação

Para compilar um exemplo, basta digitar `make [nome do exemplo]`.

No nosso caso: **make blink**

Cada arquivo .c na pasta de exemplos possui sua contra partida no make.

Não entraremos em maiores detalhes sobre esta biblioteca, porém deixarei aqui o link da documentação dela:

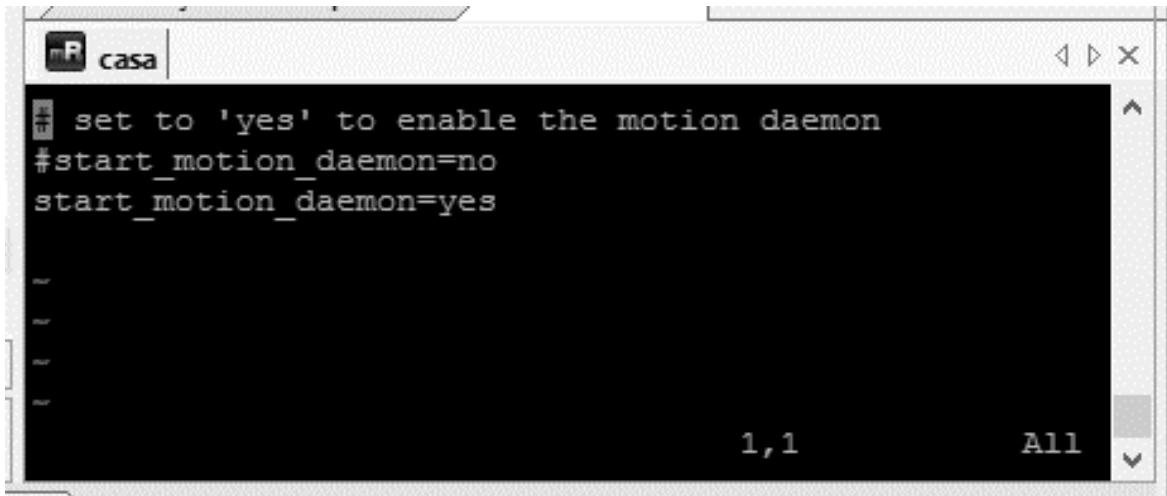
<http://wiringpi.com/reference/>

Visão Computacional

Configurando o motion como serviço de restart

Esta modificação torna possível rerestart do motion. Para tanto, siga os procedimentos a seguir:

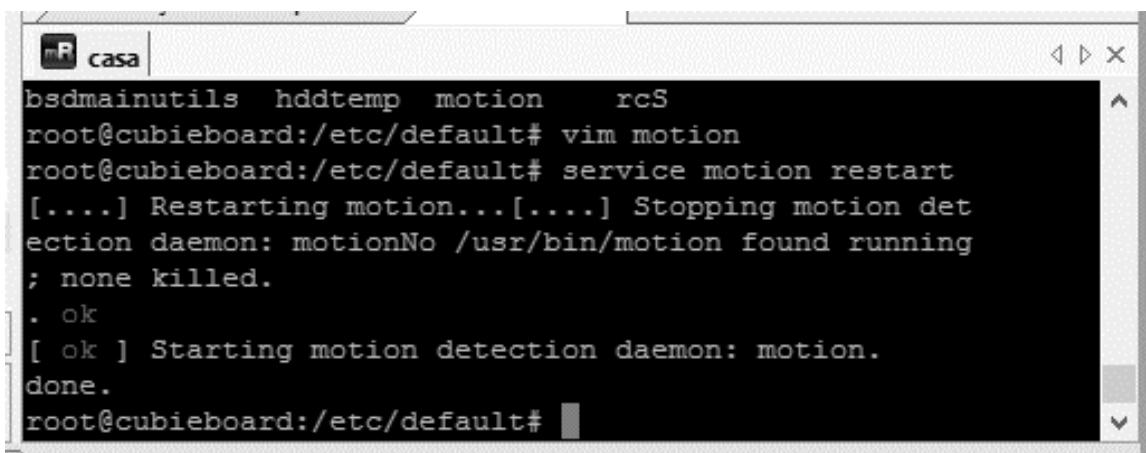
1. Entre no console e digite cd /etc/default
2. Entre no seu editor favorito. Usarei o VIM como referência: vim motion



```
# set to 'yes' to enable the motion daemon
#start_motion_daemon=no
start_motion_daemon=yes
```

Figura 192 - Opção do script para rodar em modo daemon

3. Digite na console: service motion restart



```
bsdmainutils hddtemp motion rcS
root@cubieboard:/etc/default# vim motion
root@cubieboard:/etc/default# service motion restart
[....] Restarting motion...[....] Stopping motion detection
daemon: motionNo /usr/bin/motion found running
; none killed.
. ok
[ ok ] Starting motion detection daemon: motion.
done.
root@cubieboard:/etc/default#
```

Figura 193 - Serviço do motion sendo iniciado

Configurando motion para captura de imagem

O motion captura as imagens detectando movimento. Quando percebe movimento, ele roda um script em shell script e salva em banco dados a referência da imagem ou vídeo.

Que pode realizar uma série de verificações. Entre elas, procurar rostos ou padrão na imagem.

Imaginemos o seguinte caso hipotético:

Ao encontrar uma mudança no motion, este roda um subprograma que pesquisa na foto se há um rosto. Como a foto possui registro no MySQL, vincula mais uma vez a imagem a um registro de rosto detectado.

Perceba que os processos ocorrem dessincronizados, ou seja, o programa que identifica face não é o mesmo que toma conta da ação.

A execução desta forma, torna os programas mais simples de serem criados, também permitindo melhorias sem modificar o que já está pronto.

Os algoritmos de Inteligência Artificial tratam destas entradas, lendo as entradas que foram registradas e, com base nelas, gerando saídas.

O Banco de dados, auxilia no arquivamento das informações, sendo no caso a memória de longo prazo do robô.

Os detalhes como rosto, olhos, ou mesmo altura ou cor de pele, depende da capacidade do processador. No nosso caso, possui um processador modesto, ficando limitado.

Porém, já existem técnicas que permitem integrar o robô em nuvem (cloud) permitindo que análise de voz, imagem ou outros atributos, seja inteiramente realizado em nuvem, onde existe uma capacidade maior de processamento.

Configurando acesso ao Banco de Dados pelo Motion

O motion cria a visualização de sua câmera, gerando as imagens de saída quando detecta a mudança na imagem. Isso diminui muito o processamento necessário para a interpretação do robô.

Porém, é preciso que, ao ser gerada uma mudança, esta possa ser armazenada e processada.

Estaremos aqui apresentando a primeira etapa desse processo, em que geramos um registro em banco de dados.

Registrando Thread de Reconhecimento de Padrões

Para dar início ao processo de reconhecimento de padrões no motion, siga os passos abaixo:

1. Copie, no projeto do Robotinics, o arquivo motion.sh, localizado em \linux\motion, copiando para /etc/motion/
2. Atribua permissão pelo comando: *chmod 777 motion.sh*
3. Rode o serviço *service motion restart*

Criando a Tabela no MySQL necessária

Para estes procedimentos precisaremos criar uma tabela no MySQL, conforme passos abaixo:

1. Copie o arquivo do livro robotinics/mysql/security.sql para dentro do Cubieboard ou Raspberry
2. Entre no console e vá na pasta onde copiou o arquivo

```
root@cubieboard:~# cd projetos/
root@cubieboard:~/projetos$ cd robotinics/
root@cubieboard:~/projetos/robotinics$ ls
android delphi espeak fotos impressora3d livro mysql
arduino docs espeak.0.1.zip imagens linux material opencv
root@cubieboard:~/projetos/robotinics$ cd mysql/
root@cubieboard:~/projetos/robotinics/mysql$ ls
security.sql
root@cubieboard:~/projetos/robotinics/mysql$
```

Figura 194 - Script do MySQL

3. Digite o comando **mysql -u root -p**

```
root@cubieboard:~/projetos/robotinics/mysql$ mysql -u root -p  
Enter password: █
```

Figura 195 - Acessando MySQL pela console

4. Digite a senha do administrador do banco

5. Digite **use robotinicsdb;**

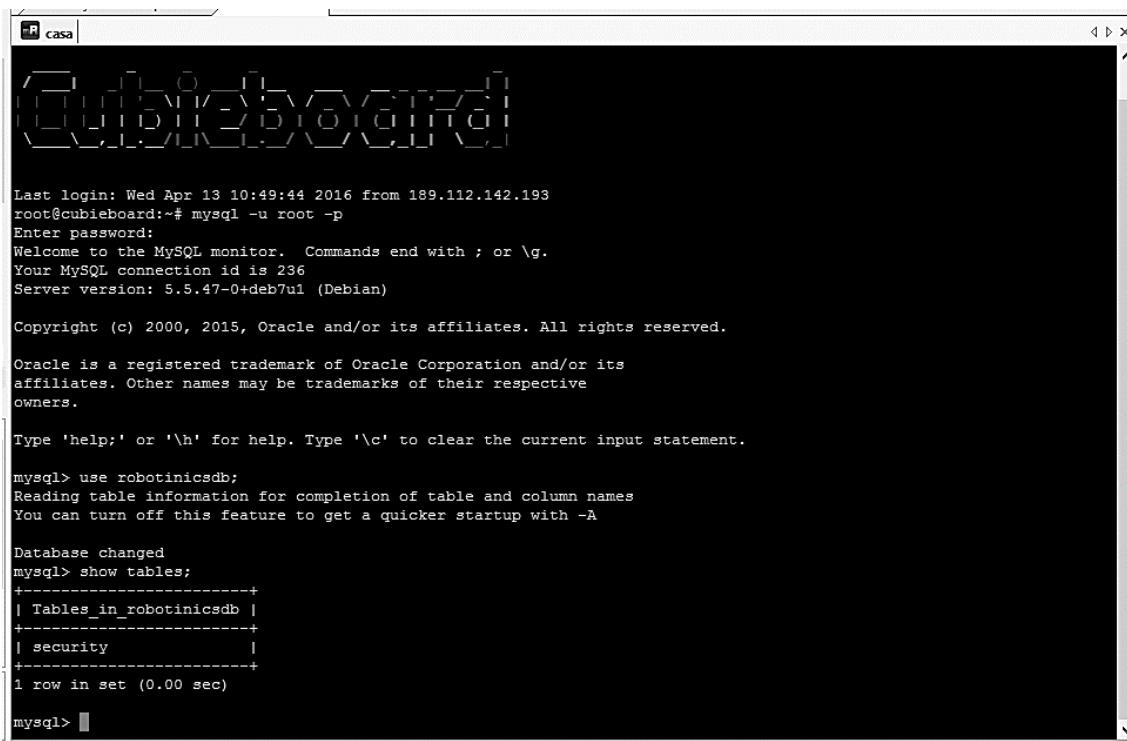
6. Digite **source security.sql;**

```
mysql>  
mysql> source security.sql;  
Database changed  
Query OK, 0 rows affected (0.01 sec)  
  
Query OK, 0 rows affected (0.03 sec)  
  
mysql> █
```

Figura 196 - Prompt do MySQL

7. Digite **show tables;**

8. Verifique se o resultado é conforme apresentado.



```
Last login: Wed Apr 13 10:49:44 2016 from 189.112.142.193
root@cubieboard:~# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 236
Server version: 5.5.47-0+deb7u1 (Debian)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use robotinicsdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_robotinicsdb |
+-----+
| security                |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figura 197 - Criação do script de registro do motion no MySQL

9. Para sair, digite **quit**;

Exemplo de arquivo motion com acesso ao MySQL e processamento de Script do OpenCV.

Registrando o banco MySQL no motion

Para realizar esta operação, siga os procedimentos:

1. Entre na console como administrador, digite cd /etc/motion
2. Digite, usando seu editor preferido, nano ou vim. Usarei o vim como exemplo: vim motion.cfg
3. Procure pelo seguinte bloco no arquivo

```
#####
# Database Options
#####

# database type : mysql, postgresql, sqlite3 (default : not defined)
```

```

; database_type value
database_type mysql

# database to log to (default: not defined)
; database_dbname value
database_dbname robotinicsdb

# The host on which the database is located (default: localhost)
Vim ; database_host value
database_host localhost

# User account name for database (default: not defined)
; database_user value
database_user root

# User password for database (default: not defined)
; database_password value
database_password (Digite sua senha do banco)

# Port on which the database is located
# mysql 3306 , postgresql 5432 (default: not defined)
; database_port value
database_port 3306

```

Procure pelo seguinte bloco no arquivo, realizando a modificação abaixo:

```

#####
# Database Options
#####
# database type : mysql, postgresql, sqlite3 (default : not defined)
; database_type value
database_type mysql

```

4. Salve o arquivo com **ESC + :wq**
5. Na console, digite **service motion restart**

6. Verifique se criou os registros, realizando o acesso ao MySQL apenas **digitando select * from security;**
7. Saia do MySQL com quit;

Finalização do projeto

Apresentamos aqui o final do livro.

Se seguiu todos os passos do livro, você tem um protótipo funcional do robô.

A finalização do livro, não encerra as diversas expansões que o robô pode sofrer.

A finalização do robô, depende de sua aplicação, bem como das futuras mudanças neste projeto.

Cabe ao leitor, continuar o projeto, participando ativamente da melhora contínua deste protótipo, que ao final é aberto e público.

O Próximo livro sobre a construção deste robô pode ser seu.

Sua evolução não trata de uma tarefa de um único homem, mas permite que qualquer um contribua e construa dispositivos IoT de forma fácil e dinâmica.

Eu (o autor) construí as bases para que o leitor crie seu próprio modelo de projeto. Dando apenas os elementos mais básicos e comuns ao projeto.

Segue aqui meu muito obrigado.

Apêndices

Baixando o projeto Robotinics

Todos os arquivos do projeto Robotinics podem ser baixados no meu site
<http://maurinsoft.com.br/projetos/robotinics/>

Porém, existe um repositório no site:

<http://sourceforge.net/projects/robotinics/?source=directory>

Os arquivos contidos no sourceforge são exatamente os apresentados no livro, possuindo referência exata dos projetos criados pelo autor no momento da publicação desta obra.

O meu site será modificado em função da evolução do projeto pela comunidade Open Source. Este, entretanto, poderá destoar do livro, pois não há, de fato, compatibilidade entre o repositório do meu site e o livro, pois o meu site sofrerá modificação com o tempo.

O autor recomenda que sejam utilizados os fontes do source forge, porem na ausência deste, poderá ser utilizada o repositório do meu site.

Simulação de Software

O Robotronics possui um ambiente virtual para testes de homologação. Para utilizar o ambiente de homologação, siga os passos:

- 1) Entre no site do repositório do projeto

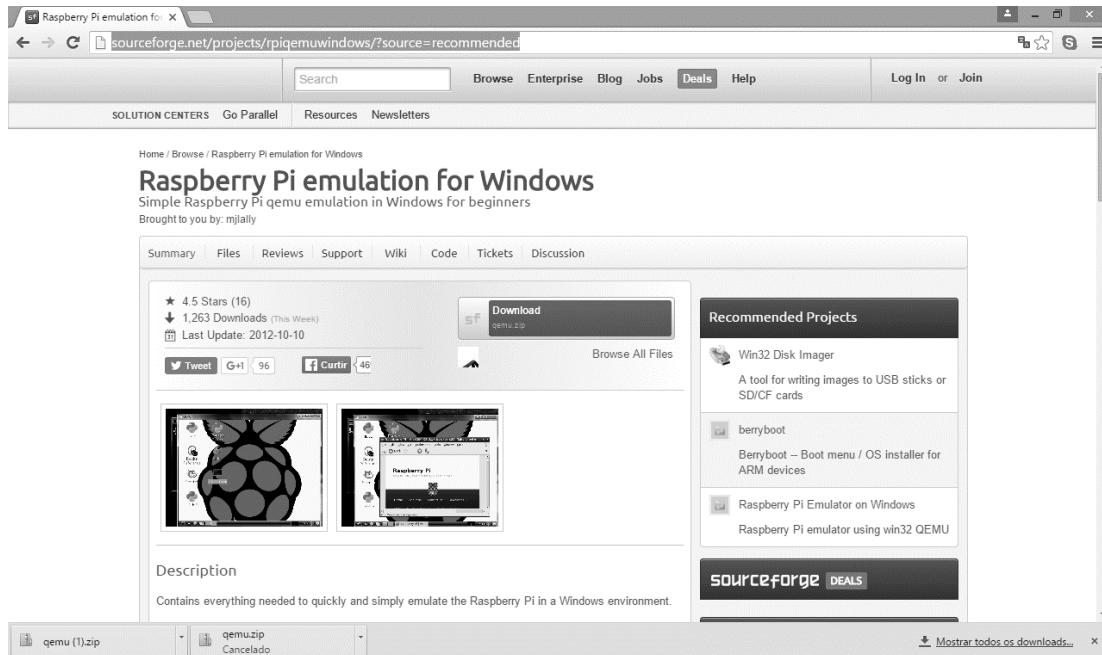


Figura 198 - Site sourceforge do projeto QEMU para Raspberry

Para baixar o simulador, entre no link:

<http://sourceforge.net/projects/rpiqemuwindows/?source=recommended>

- 2) Baixe o arquivo zipado qemu.zip

3) Copie o arquivo para a pasta c:/qemu

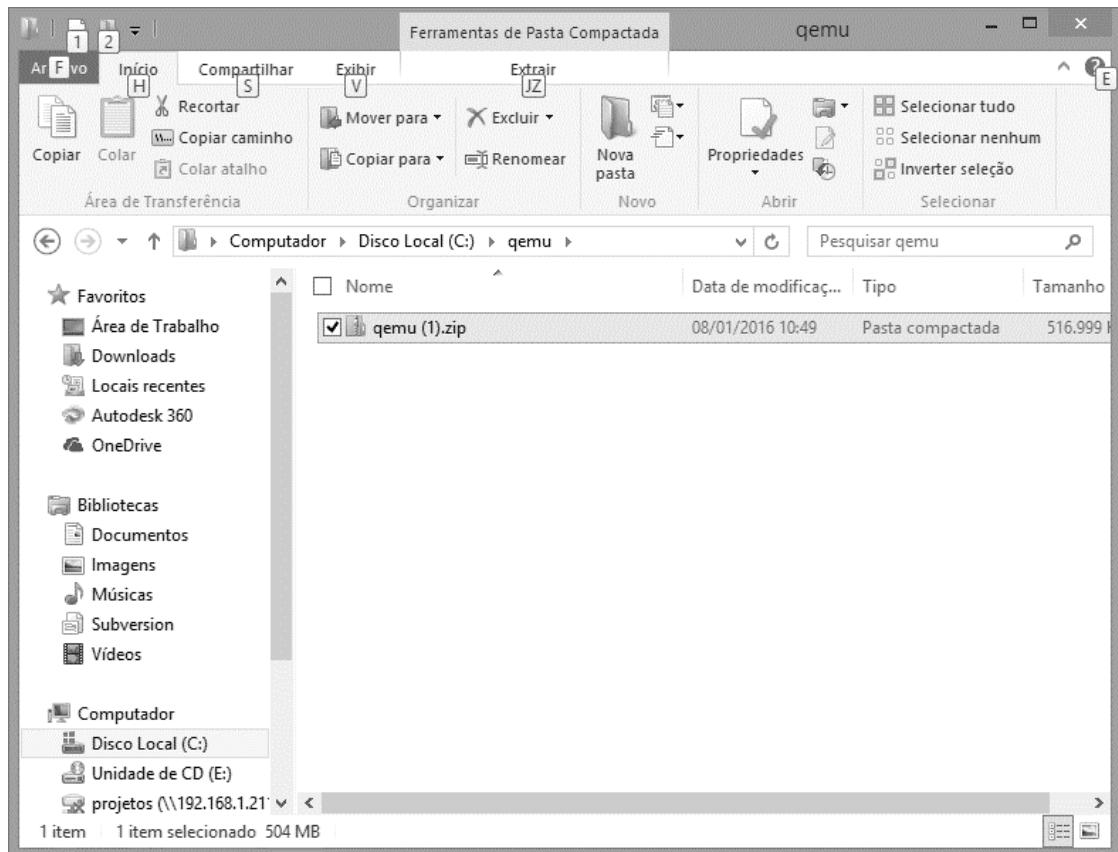


Figura 199 - Pacote zipado do qemu para Raspberry

4) Descompacte o arquivo com winzip ou winrar em uma pasta no diretório c:, recomendamos o c:/qemu

5) Rode o arquivo run.bat

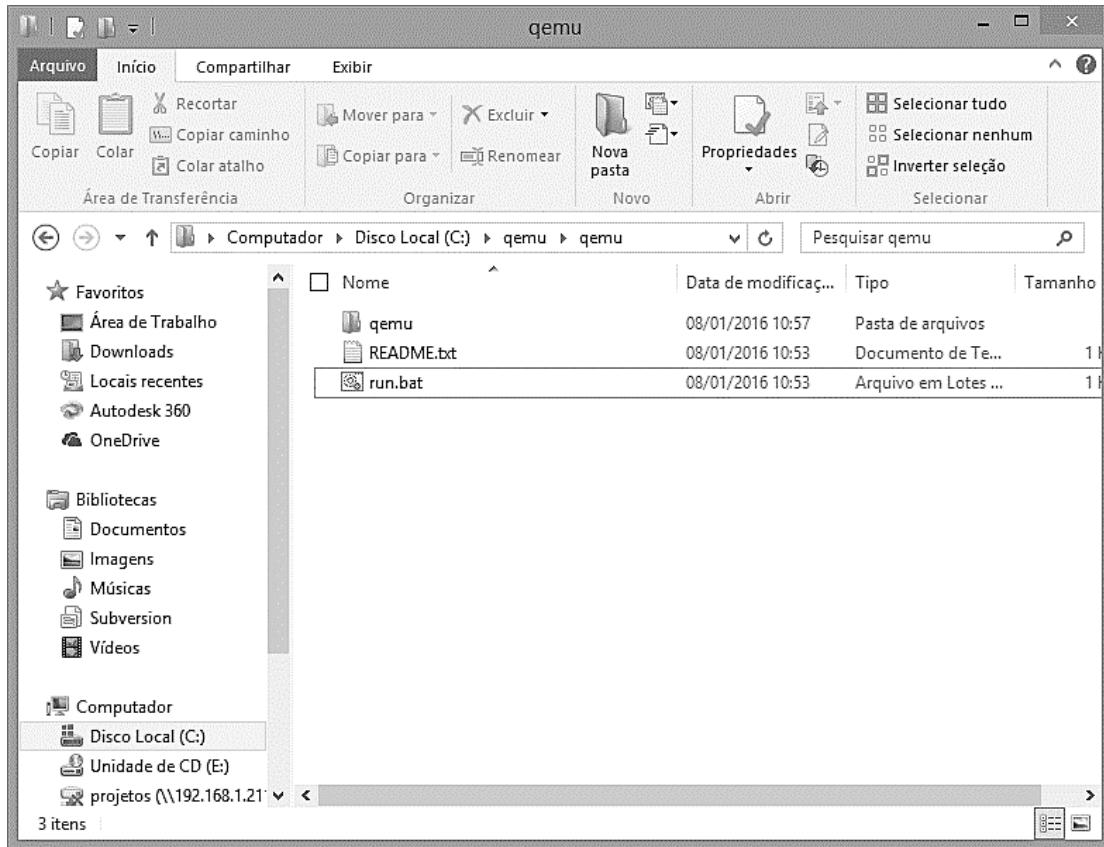


Figura 200- Exemplo da pasta QEMU

6) Aguarde a inicialização da VM

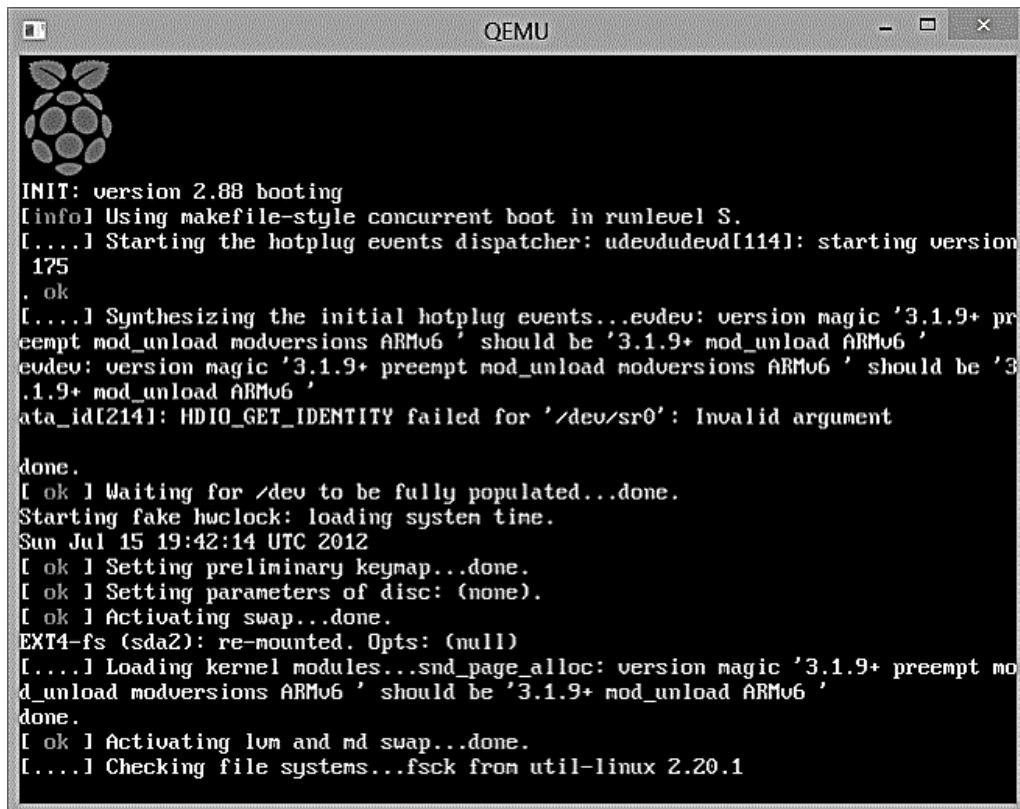


Figura 201- Emulador QEMU rodando

- 7) Configure sua VM conforme sua necessidade ou realize apenas o passo 8.
- 8) Para iniciar a vm já com os arquivos do projeto Robotinics, pegue o zip `vmrobotinics.zip` e copie na pasta do qemu

- 9) Edite o arquivo run.bat, mudando o nome da imagem que está sendo rodada para robotinics.img

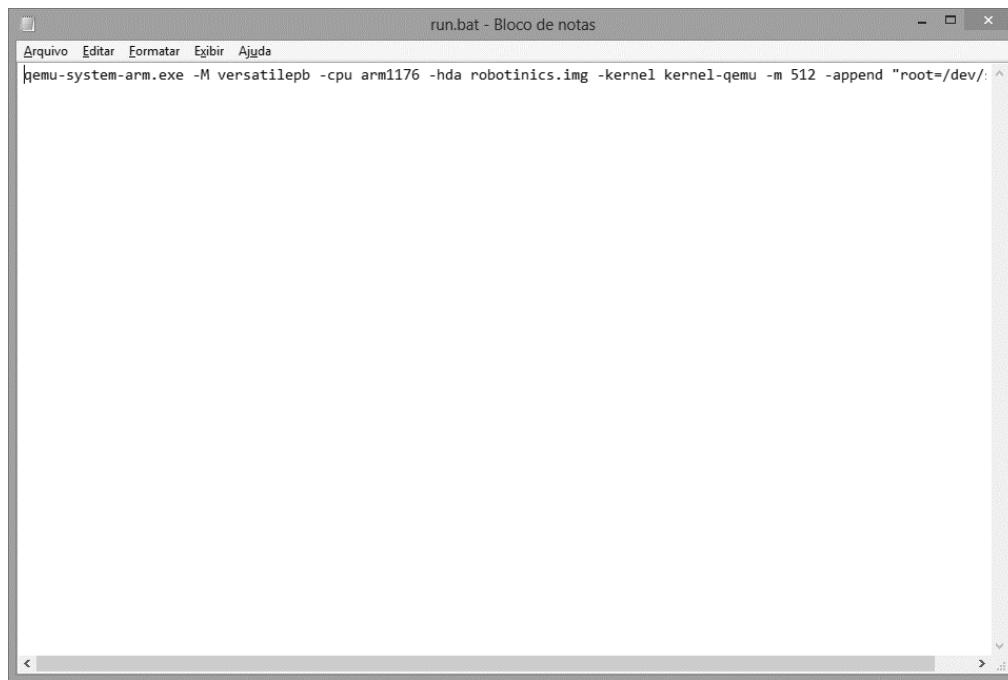


Figura 202- Mandando script do qemu

Gerando um espelho do SD do Robotinics para equipamento real

Este procedimento visa criar um espelho em um arquivo SD para ser utilizado em uma máquina real.

Para realizar este procedimento será necessário instalar o Win32DiskImager, conforme passado no Capítulo 1.

Siga os passos conforme anunciado:



Figura 203- Win32DiskImager

- 1) Inclua o SD que deseja ser espelhado
- 2) Selecione o arquivo espelho (arquivo com extensão .IMG)
- 3) Selecione o device (no nosso exemplo, o volume D:)
- 4) Pressione o botão Write (para começar a gravar o SD)
- 5) Aguarde até o SD ter sido gravado

Pronto: você já tem uma imagem no SD!

Bibliografia

- Alves, M. M. (s.d.). *Sockets Linux*. Brasport.
- Ciarcia. (s.d.). *Construa seu próprio Microcomputador Z80*. MacGraw-Hill.
- Dias, J. (2011). *Curso de Especialização de Engenharia Automotiva - Modulo de Transmissão*. Universidade Tecnológica Federal do Paraná.
- Fialho, E. (s.d.). *SolidWorks Premium 2013*. Erica.
- Joshua, E. (s.d.). *Arduino em Ação*. Novatec.
- McRoberts, M. (2011). *Arduino Básico*. Novatec.
- Raimondi, F. (s.d.). <http://www.rmnd.net/>.
- Richard C. Dorf, J. (s.d.). *Introdução aos Circuitos Elétricos*. LTC.
- Silveira, F. L. (2011). Potência de tração de um veículo automotor. *Revista Brasileira de Ensino de Física*.
- Souza, J. (s.d.). *Aderência Pneu-Pavimento em Revestimentos asfálticos aeroportuários*.
- Tateyama, W. (s.d.). *TCC Carregador de Bateriais de íon lítio através de microcontroladores*. Universidade de São Paulo, Escola de Engenharia de São Carlos.
- Wikipédia. (s.d.). <http://www.wikipedia.org>. Fonte: Wikipedia: <http://www.wikipedia.org>

Índice de Ilustrações

Figura 1 - Globo de isopor	18
Figura 2 -Tela do Solidworks	26
Figura 3 - Opção de abertura de arquivo no SolidWorks.....	27
Figura 4 - Parte que se deseja imprimir aberta no SolidWorks	27
Figura 5 - Opção de salvamento do arquivo	28
Figura 6 - Confirmação de salvamento STL.....	28
Figura 7 - Impressora 3D da Vince XYZPrinting.....	29
Figura 8 - Software XYZWare preparado para imprimir uma peça	30
Figura 9 - Opções para finalização da impressão.....	31
Figura 10 - Cartucho de filamento da XYZPrinting.....	31
Figura 11 - Forças relacionadas ao movimento do robô	33
Figura 12 - Ilustração que mostra efeito de força trativa	34
Figura 13 - Motor DC com caixa de Redução + Roda	35
Figura 14 - Layout da base do robô com quatro rodas motoras.....	36
Figura 15 - Robô subindo uma rampa	38
Figura 16 - Representação física de um braço e suas forças mecânicas.....	43
Figura 17 - Visualização dos segmentos do braço do robô.....	44
Figura 18 - Visão Geral de Consumo de Corrente.....	47
Figura 19 - Conexão led x resistor.....	49
Figura 20 - Exemplo de uso de capacitor.....	51
Figura 21 - Bateria NK 18650	56
Figura 22 - Baterias em série	58
Figura 23 - Ligação em paralelo.....	58
Figura 24 - Película de Dry Film facilita muito a montagem de PCBs	61
Figura 25 - Imagem de negativo impresso para confecção de placa PCB	61
Figura 26 - Exemplo de placa PCB	62
Figura 27 - Software Eagle, da CadSoft.....	63
Figura 28 - Visão do Schematic do Eagle	64
Figura 29 - Visão da Board.....	64
Figura 30 - Ligação entre entradas e saídas.....	65
Figura 31 - Arduino Mega	68
Figura 32 - Exemplo da IDE do Arduino	70
Figura 33 - Seleção da Versão do Hardware	71
Figura 34- Seleção da Porta de Comunicação USB/Serial.....	72
Figura 35 - Foto do Raspberry Pi 2	73
Figura 36 - Dispositivos do Raspberry	74
Figura 37 - Ferramenta de transferência de arquivos.....	76
Figura 38 - Exemplo de conexão feita facilmente pelo mRemoteNG	77
Figura 39- Tipos de conexões existentes no mRemoteNG	77
Figura 40 - Exemplo de uso do VIM para edição	78
Figura 41 - Código Makefile para compilação.....	79
Figura 42 - Resultado final do programa	80
Figura 43 - Interface do Robotinics – Módulo de Comunicação de Fala	83
Figura 44 - Desenho em visão da base inferior	85

Figura 45 - Visão da base superior criada no SolidWorks.....	86
Figura 46 - Especificações técnicas da base reta superior	86
Figura 47 - Rodas do robô	87
Figura 48 - Dimensionamento da caixa de redução	88
Figura 49 - Suporte e fixação para motor e caixa de redução	89
Figura 50 -Elementos de fixação do motor.....	89
Figura 51- Vista dos prolongadores	91
Figura 52 - Projeto de uma fonte básica	94
Figura 53 - Ripple de tensão após ser filtrada e o ripple na saída da ponte de diodos	95
Figura 54 - Step down XL4005	99
Figura 55 - Diagrama de conexão do carregador de bateria	100
Figura 56 - Pilha de Li-ion utilizada em nosso projeto	101
Figura 57 - Suporte Porta Pilhas	102
Figura 58 - Sensor de tensão	103
Figura 59 - Esquemático de ligação com Arduino.....	103
Figura 60 - Conexões elétricas do relê.....	106
Figura 61 - Diagrama esquemático de recarga da bateria	108
Figura 62- Visão da placa pronta	109
Figura 63 - Visão do Schematic.....	110
Figura 64 - Step down com regulagem de tensão e corrente	111
Figura 65 - Visão geral dos blocos principais para programação do robô	114
Figura 66 - Fluxo de start do Arduino	116
Figura 67 - Fluxograma de execução detalhado	117
Figura 68 - Fluxograma de execução no Linux	119
Figura 69 - Hierarquia de execução das aplicações	121
Figura 70- Instalação do software de sintetização de voz	126
Figura 71 - Opção de criação de uma nova montagem das peças no SolidWorks 2014.....	133
Figura 72 - Seleção do arquivo de visão das placas	134
Figura 73 - Visão das placas aplicadas à base	134
Figura 74 - Diagrama geral de alimentação do robô	136
Figura 75 - Esquema elétrico do sensor de corrente	137
Figura 76 - Imagem dos pinos da placa de Controle DC	140
Figura 77 Esquemático de ligação elétrica no Arduino.....	141
Figura 78 - Componente de regulagem de tensão	145
Figura 79 - Layout da placa	146
Figura 80 - Visão das trilhas.....	147
Figura 81 - Placa de expansão Sensor Shied.....	148
Figura 82 - Pinagem do Arduino Shield	149
Figura 83 - Visão do Sensor de Ultrassom	150
Figura 84 - Exemplo de funcionamento do ultrassom.....	151
Figura 85 - Esquemático de ligação do ultrassom	152
Figura 86 - Programa Arduino	157
Figura 87 - Selecionando fonte do projeto	158
Figura 88 - Ambiente de Edição do programa do arduino.....	159
Figura 89 - Compilação realizada com sucesso	160

Figura 90 - Carga de firmware realizada com sucesso	161
Figura 91 - Debugando retorno da USB.....	162
Figura 92 - Módulo de Controle do Robotinics - Arduino.....	170
Figura 93 - Módulo de controle, configuração do dispositivo	171
Figura 94 - Comandos de movimentação do robô	172
Figura 95 - Visualização do comando realizado.....	173
Figura 96 - Visão das peças já montadas no robô	174
Figura 97 - Visão da peça no SolidWorks.....	176
Figura 98 - Visão do corpo inferior, incluindo chanfro da tomada	178
Figura 99 - Visão do Suporte com a base de fixação do arduino	179
Figura 100 - Corpo superior.....	180
Figura 101 - Visão do robô montado até o final deste capítulo	181
Figura 102 - Visão das ligações elétricas.....	182
Figura 103 - - Imagem de um Bluetooth.....	183
Figura 104 - Visão do LCD 16x2	184
Figura 105 - Visão da traseira do LCD I2C	184
Figura 106 - Esquema elétrico do Arduino com LCD	185
Figura 107 - Visão de perfil de um servomotor	187
Figura 108 - Exemplo de Sinal PWM.....	187
Figura 109 - Servo motor de 15 kg	189
Figura 110 - Servomotor de 10 kg	191
Figura 111 - - Esquemático da placa controladora de Servo motores	193
Figura 112 - Visão do layout da placa.....	194
Figura 113 - Vista da placa de controle dos Servo motores.....	195
Figura 114 - Tabela de corrente de LEDs	196
Figura 115 - Placa LED	197
Figura 116- Visão da placa LED consolidada no SolidWorks	198
Figura 117 - Monitor LCD touch screen	199
Figura 118 - Diagrama da placa controladora LCD	200
Figura 119- Software Eagle, da Cadsoft.....	202
Figura 120- Gerando Exportação para SolidWorks	203
Figura 121 - Editor de placa Eagle Board	203
Figura 122 - Opções para geração de Exportação IDF	204
Figura 123 - Processando exportação do arquivo	205
Figura 124 - Finalização da exportação para IDF	205
Figura 125- Local onde o arquivo foi salvo	206
Figura 126 - Opções para adicionar Add Ins	207
Figura 127- Menu com CircuitWorks	208
Figura 128- Open ECAD File.....	208
Figura 129- Arquivo placaled.emn.....	209
Figura 130- Visão da placa no CircuitWorks	209
Figura 131 - Menu de opções de associação, com botão direito do mouse.....	210
Figura 132- Opções de componentes associados.....	211
Figura 133 - Notificação de associação de componente	211
Figura 134- Menu de opções do solidworks	212

Figura 135 - Confirmação do Solidworks	212
Figura 136- Visão da placa.....	213
Figura 137 - Salvando placa LEDs	213
Figura 138 - Exemplo de movimento de rotação do servomotor.....	225
Figura 139- Vista da peça sendo impressa no software	226
Figura 140- Visão do braço base.....	227
Figura 141- Visão da impressão do braço base	227
Figura 142- Montagem do braço	228
Figura 143- Visão do braço	229
Figura 144- Visão do braço montado	230
Figura 145- Servomotor e conexões	231
Figura 146 - Esquemático de vista da peça 1.....	232
Figura 147 - Esquemático de vista peça 2	233
Figura 148- Visão do braço no SolidWorks	234
Figura 149- Conexão do braço com servomotor visto pelo SolidWorks	234
Figura 150- Representação Denavit-Hatenberg.....	235
Figura 151- Fio paralelo vermelho e preto	238
Figura 152 - Par de fio elétrico P&V	239
Figura 153- Cabo Manga de oito vias	240
Figura 154- Visão do conector RJ45	241
Figura 155- Padrão de conexão do cabo de rede	241
Figura 156 - Conector RJ45 para PCB	242
Figura 157 - Esquemático do RJ45 para PCB	242
Figura 158- Barra de pinos macho.....	243
Figura 159 - Conector Borne	244
Figura 160 - Conector Borne com várias entradas	245
Figura 161 - Exemplo de Resistor	246
Figura 162 - Exemplo de Led	248
Figura 163 - Exemplo de aplicação de controle do Robô.....	250
Figura 164- Programa de simulação de fala ToolFalar	251
Figura 165- Compilando os exemplos do OpenCV	254
Figura 166- Instalação do Sphinx no Cubieboard	258
Figura 167- Baixando Sphinxbase do site	260
Figura 168- Montando ambiente para compilação	261
Figura 169- Compilação terminada com pacotes instalados	262
Figura 170 - Tela de criação de dicionário fonético	266
Figura 171 - Tela de retorno, permite baixar os arquivos gerados	267
Figura 172 - Tela da console com retorno de dados.....	270
Figura 173 - Compilação dos Exemplos	274
Figura 174 - Executando o programa exemplo	274
Figura 175 - Visão das partes da cabeça do robô	277
Figura 176- Corpo do robô com o servomotor.....	278
Figura 177 - Cabeça inferior conectada ao corpo	279
Figura 178 - Interface web phpmyadmin.....	280
Figura 179 - Instalando pacote phpmyadmin	280

Figura 180 - Mostrando tabela do robotinics	281
Figura 181 - Tela do Filezilla.....	283
Figura 182- Cópia e configuração dos scripts da interface web do Robotinics.....	284
Figura 183 - Informações sobre o banco de dados.....	285
Figura 184- Procedimento para criar novo banco de dados.....	285
Figura 185- Visão dos bancos disponíveis no servidor	286
Figura 186- Opção Novo	286
Figura 187- Inclusão de scripts no banco de dados.....	286
Figura 188 - Script a ser executado	286
Figura 189- Start gateone	287
Figura 190- Instalando o ntp	288
Figura 191 - Seleção do fuso horário por estado/país.....	288
Figura 192 - Opção do script para rodar em modo daemon	293
Figura 193 - Serviço do motion sendo iniciado	293
Figura 194 - Script do MySQL	295
Figura 195 - Acessando MySQL pela console	296
Figura 196 - Prompt do MySQL.....	296
Figura 197 - Criação do script de registro do motion no MySQL.....	297
Figura 198 - Site sourceforge do projeto QEMU para Raspberry	302
Figura 199 - Pacote zipado do qemu para Raspberry.....	303
Figura 200- Exemplo da pasta QEMU	304
Figura 201- Emulador QEMU rodando	305
Figura 202- Mandando script do qemu	306
Figura 203- Win32DiskImager	307

Quarta Capa

Atualmente tem-se apregoado que a nova revolução é a IOT (Internet das Coisas), em que todos os objetos são conectados à grande rede (Internet) e informações são promovidas para aprimorar nosso cotidiano, criando máquinas mais inteligentes.

Este livro apresenta o passo a passo para realização destas atividades, e apesar da obra ensinar a construir um robô, ela pode ser facilmente aplicada a qualquer projeto IOT, sendo o Robotinics um exemplo para o desenvolvimento e construção de projetos IOT.

A quem destina-se este livro

O Livro apresenta a junção de mundos.

Aos que possuem vivencia em eletrônica, apresenta como desenvolver seus projetos, criando caixas plásticas impressas em 3D, também a programação em Linux, e técnicas avançadas para programação IoT.

Aos Programadores, apresenta uma visão avançada de programação em embarcados e no mundo Linux. Com desenvolvimento de socket cliente e servidor, programação GPIO, USB, Integração com dispositivos I/O. Banco de dados e muito código em C e Free Pascal. Também falamos de redes neurais, reconhecimento de voz e sintetização de fala, sem falar lógico em reconhecimento de imagem.

Aos desenvolvedores de startups, que sonham em lançar seus produtos, damos os fundamentos e técnicas avançadas para que você realize seu sonho.

Aos Estudantes de Engenharia permite aprimorar técnico com desenvolvimento de projetos multimídia com recursos nunca antes sonhados.

Temas abordados

- Prototipagem rápida
- Eletrônica Digital
- Reconhecimento de Imagem
- Reconhecimento de Voz
- Sintetização de Voz
- Linguagem C para Linux
- Linguagem Free Pascal para Windows e Linux (Lazarus)
- Redes Neurais
- Arduino e Raspberry