Centro Universitário da FEI

Projeto de Iniciação Científica

INICIAÇÃO CIENTÍFICA

IMPLEMENTAÇÃO DO SISTEMA ELETRÔNICO DE UM ROBÔ HUMANÓIDE

Relatório Final

Orientador: Reinaldo A. C. Bianchi

Candidato: Eduardo Mello Nottolini

Nº de Matrícula: 11.109.312-6

Data: 27/7/2012 12:38

Curso: Engenharia Elétrica

Projeto: Futebol de Robôs

AGRADECIMENTOS

Gostaria de agradecer primeiramente ao meu orientador, Reinaldo Bianchi, pelas muitas conversas e recomendações para sempre manter o foco nos trabalhos que seriam desenvolvidos a fim de alcançarmos os objetivos do projeto.

Também gostaria de agradecer ao amigo José Ângelo Gurzoni pelas horas dispendidas na explicação e esclarecimentos de conceitos de programação em C, a ajuda com a interface *Stop & Go*, além de muitas outras dicas em áreas como a eletrônica e o uso de microcontroladores.

RESUMO

Este Relatório Final apresenta todo o trabalho desenvolvido ao longo da Iniciação Científica com a finalidade da implementação do sistema eletrônico do robô humanoide.

O trabalho apresenta a continuação da Iniciação Científica anterior, o desenvolvimento da placa de Navegação proposta anteriormente, os testes desta placa e as conclusões sobre a utilização deste material.

Com a observação de alguns problemas, houve a necessidade de mudança do desenvolvimento do projeto, com a elaboração de uma placa para testes, a movimentação dos servos motores utilizados no robô e sua demonstração de funcionamento, a programação de uma interface para criação de *scripts* de movimento, a elaboração da placa de controle para a leitura dos sensores inerciais e acionamento dos motores e a programação do microcontrolador contido nesta placa. Por fim, foram inseridos os dados dos *scripts* criados na interface na placa de controle para sua execução, demonstrando o funcionamento de toda a eletrônica permitindo agora o desenvolvimento do controle do robô para sua establização.

Palavras-Chave: Eletrônica Humanóide, Implementação, RoboCup

Sumário

1. INTRODUÇÃO

A robótica, termo genérico dado a um conjunto de máquinas capazes de se inserir o controle por meios computadorizados, é, sem sombra de dúvida um assunto comentado à exaustão no século XXI. Dá-se aos robôs a glória de uma revolução na indústria sem precedentes, com a maximização da produção, melhoramento da qualidade do produto, diminuição do tempo produtivo, etc. Entretanto tais robôs são meramente máquinas capazes de realizar diversas tarefas na indústria, sejam elas programadas ou não, diferentemente do termo “robô” que o escritor Isaac Asimov coloca em seu livro de ficcção "I, Robot” (Eu, Robô) [1], de 1950 onde um robô era uma máquina criada à semelhança do ser humano.

A partir daí temos o início do desenvolvimento de máquinas com características mecânicas mais humanas, primariamente devido aos robôs não humanóides causarem um estranhamento ao ser humano na sua utilização, em aplicações onde há grande contato com o usuário final e pessoas não tão familiarizadas com o ambiente industrial, local onde há uma maior concentração de robôs para automação. Assim, mesmo sendo robôs de uma complexidade maior, devido a locomoção por pernas, presença de braços articulados o tronco e a cabeça e elementos capazes de interagir com o ser humano, como a emissão de sons, o reconhecimento da fala e a visão computacional, como o famoso humanóide da Honda, o ASIMO [2] ou o mais recente NAO [3], há um grande

desenvolvimento nestes tipos de robôs para aplicações como as domésticas e hospitalares, locais onde um robô convencional não seria aceito de forma natural.

Figura 1 – Honda ASIMO

Com o avanço das tecnologias de hardware e software, em 1992 nasceu a idéia de robôs jogando futebol no artigo "*On Seeing Robots*" elaborado pelo professor Alan Macworth[4] e em 1993 um grupo de pesquisadores Japoneses incluindo Minoru Asada, Yasuo Kuniyoshi, e Hiroaki Kitano, decidiram lançar uma competição para o estudo científico de robôs [5] onde em algumas modalidades temos justamente o jogo de futebol como plataforma para o desenvolvimento.

Exemplos de robôs humanóides participantes podem ser dados ao citar os quatro times que foram finalistas - em ordem de colocação - da *RoboCup 2010* em Singapura na

categoria *Kid Size*: *Darmstadt Dribblers* [6], *FUmanoids* [7], *CIT Brains Kid* [8] e *Team DARwIn* [9]. Comparando-se as tecnologias de tais robôs temos uma unanimidade na questão dos servo motores utilizados sendo normalmente utilizados os RX28 e RX64 da ROBOTIS com a linha Dynamixel [10] devido a sua simplicidade de utilização. No projeto do robô humanóide serão utilizados servo-motores RX28, juntamente com o uso do padrão de comunicação RS-485, também presente em todos os robôs mencionados. Em relação aos sensores, temos a presença constante do acelerômetro nos robôs, diferindo apenas no sensor de bússola e giroscópio – os times *Team DARwIn* e *FUmanoids* se utilizam de bússola, enquanto os times *Darmstadt Dribblers* e *CIT Brains Kid* se utilizam do giroscópio – o projeto proposto irá se utilizar de dois dos três mencionados, julgados mais importantes: acelerômetro e giroscópio.

Dando continuidade ao desenvolvimento do projeto de Iniciação Científica entitulado “Eletrônica do Humanoide” do autor Eduardo Garcia [11], é proposta a elaboração, verificação e conclusão da eletrônica do robô Humanóide do Centro Universitário da FEI.

1.1. Objetivo

O objetivo desta Iniciação Científica é a implementação do *Hardware* para o Robô Humanóide do Centro Universitário da FEI utilizando como estudo a Iniciação Científica do aluno já formado Eduardo Garcia, entitulada “Eletrônica do Humanóide” e realizando

o densenvolvimento de uma interface para a captura dos movimentos iniciais do robô humanoide, o desenvolvimento de uma placa eletrônica capaz de realizar o controle de "baixo nível" do robô como a aquisição de dados de sensores, controle dos motores, gerenciamento da energia das baterias, comunicação com o computador controlador e a programação do microcontrolador utilizado nesta placa com testes de demonstração dos periféricos e circuitos.

1.2. Justificativa

O projeto de Futebol de Robôs da FEI desde 2003 desenvolve robôs para as competições nacionais e internacionais da categoria *RoboCup Soccer* [12], sendo contruídos ao longo destes anos robôs de alto nível nas subcategorias *Very Small Size* [13] e *Small Size* [14]. Com a maturação do projeto de robôs da categoria *Small Size* em termos eletrônicos, com sua arquitetura de hardware bem definida e a grande estabilidade dos circuitos, junto do aprendizado e conhecimento gerados pela pesquisa destes robôs, houve a necessidade da evolução do projeto com o desenvolvimento de um robô humanóide para a categoria *Kid Size*.

Tal robô, devido a sua complexidade locomotora, com mais de 20 graus de liberdade para movimentação, o sensoriamento da aceleração linear e angular para o controle dos servo-motores, a programação de "baixo nível" para a locomoção e criação de jogadas, o desenvolvimento da visão de máquina no uso de uma câmera para o

reconhecimento do campo e da bola, o software de controle de "alto nível", com as estratégias, e comandos enviados pelo Juiz, etc, é com certeza um grande desafio integrando diferentes áreas de engenharia, sendo uma plataforma ativa para a pesquisa em Inteligência Artificial, Robótica, Aprendizado por Reforço, e muitas outras.

Com a finalização desta Iniciação Científica, o Centro Universiário da FEI contará com um robô de nível internacional para a competição *Robocup* além de ser capaz de servir de base a pesquisas posteriores.

1.3. Metodologia

A metodologia empregada nesta Iniciação Científica seguiu-se como definido no Cronograma do relatório Parcial, entregue após os 6 primeiros meses, consistindo de:

a) Estudo da Iniciação Científica "Eletrônica do Humanóide" e das referências constantes no projeto, como as regras da competição, robôs utilizados por outras equipes, etc
b) Verificação do material apresentado para o projeto, com a montagem da Placa de Navegação desenvolvida na Iniciação Científica anterior
c) Realização dos testes de funcionamento desta placa
d) Desenvolvimento de um circuito capaz da comunicação dos motores com um computador pessoal e a Placa de Controle USB/RS485

e) Desenvolvimento de algoritmos com os protocolos utilizados pelos motores, e leitura das posições com uso do *software Matlab*[15]

f) Programação para a Interface *Stop and Go* para a criação de movimentos dos motores

g) Realização de uma movimentação para o caminhar, mostrando as funcionalidades da Interface

h) Desenvolvimento da placa eletrônica e determinação dos periféricos necessários, e do microcontrolador utilizado

i) Programação do microcontrolador para leitura do acelerômetro e comunicação com os motores.

j) Inserção dos parâmetros de movimentação criados na Interface *Stop and Go* para o microcontrolador e sua execução

2. REVISÃO BIBLIOGRÁFICA

A revisão bibliográfica constituí-se do estudo da Iniciação Científica anteriormente desenvolvida, a fim de serem obtidas todas as informaçãoes referentes ao projeto, ou seja, os objetivos a serem perseguidos, as regras resumidas da competição, o comparativo de times, o protocolo dos Servo-Motores, a placa de navegação desenvolvida, etc.

2.1. Estudo da Iniciação Científica "Eletrônica do Humanóide"

Os itens mais focados no estudo da Iniciação Científica estarão organizados abaixo de forma resumida.

2.1.1. A RoboCup

Em 1992 um grupo de pesquisadores japoneses organizaram um *workshop* em Tóquio afim de discutir o uso do jogo de futebol como plataforma básica ao desenvolvimento de robôs com Inteligência Artificial (I.A.), sendo que já em 1993 alguns pesquisadores incluindo Minoru Asada, Yasuo Kuniyoshi e Hiroaki Kitano decidiram lançar uma competição justamente para este desenvolvimento, obtendo grande aceitação e entusiasmo de pesquisadores de todo o mundo. Tal competição recebeu o nome de *Robot World Cup Initiative*, ou mais conhecida como *RoboCup.*

A RoboCup possui muitas segmentações, sendo que a mais famosa é sem dúvida a de futebol de robôs, sendo subdivididos em várias categorias:

- *Small Size League;*
- *Middle Size League;* [16]
- *Humanoid League;* [17]
- *Standard Platform League;* [18]
- *Simulation League;* [19]

Para esta Inciação Científica será desenvolvido um robô para a *Humanoid League*.

2.1.2. A liga de robôs Humanóides

A competição de Robôs Humanóides possui 3 categorias, listadas a seguir:

- *Kid Size* – altura entre 30e 60cm
- *Teen Size* – altura entre 1 e 1,2m
- *Adult Size* –altura maior que 1,3m

Para o início da competição humanóide foi decidida a construção de um robô da categoria *Kid Size* devido a sua menor dificuldade em implementação, menor custo em materiais e dentre as categorias da liga de humanóides é a que possui maior número de participantes inscritos, o que proporciona uma maior troca de experiências entre os participantes e um maior número de artigos escritos.

2.1.3. Regras da Eletrônica

O resumo de algumas regras mais importantes à eletrônica estão relacionadas abaixo, sendo que todas as regras estão disponíveis no site da *RoboCup* [20].

- Não é permitido o uso de sensores ativos nos Robôs, como aqueles que emitem luz, ondas eletromagnéticas ou ultra-som. Sensores passivos como de toque, pressão, acelerômetros, giroscópios, bússolas são livres.
- Os Robôs devem agir de maneira autônoma com todo o processamento embarcado, seja da visão computacional bem como do controle de atuadores e sensoriamento.
- O número de câmeras para a visão do Robô é limitada a 2 com ângulo de visão máximo de 180º na horizotal e 270º (+/- 135º) na vertical.

2.2. A eletrônica desenvolvida

Em sua Iniciação Científica, Eduardo Garcia desenvolveu uma placa nomeada Placa de Navegação para a eletrônica do humanóide.

2.2.1. A Placa de Navegação

A placa de navegação foi projetada apenas para servir de suporte aos sensores e alguns circuitos necessários ao robô, como os reguladores de tensão de 3,3v e 5v , circuito de proteção contra descarga da bateria, amplificador de áudio, leds de indicação e conectores. A visão em CAD da placa de navegação está presente na figura 1.

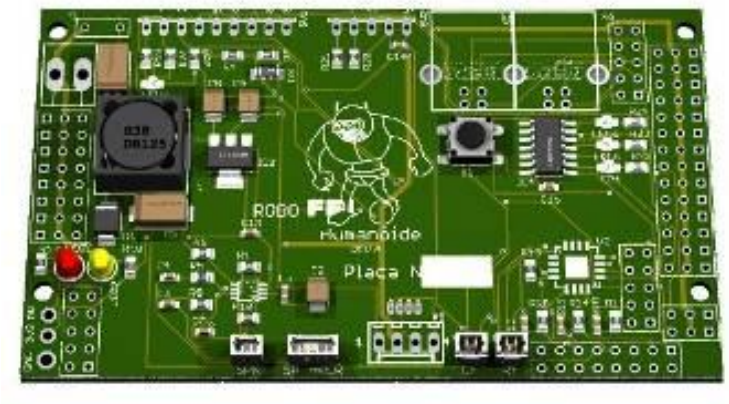


Figura 2 – Placa de Navegação

2.2.2. Os Servo-Motores

Os servo-motores escolhidos ao projeto foram os RX-28 série *Dynamixel* da *Robotis* [21], presentes na maioria dos robôs da liga humanóide categorias *Kid Size* e *Teen Size*. Tais servos possuem uma comunicação serial RS-485, sendo que todos os servo-motores se utilizam de um mesmo barramento para a comunicação, diminuindo em muito a quantidade de cabos presentes no robô. Outro detalhe presente são as medições internas de torque, tensão e temperatura, disponíveis ao controlador mestre. Os detalhes da escolha do servo motor podem ser conferidos na Iniciação Científica "Eletrônica do Humanóide".

As figuras abaixo mostram o servo RX-28 e uma simplificação de sua forma de comunicação.

Figura 3 – Servo-Motor RX-28

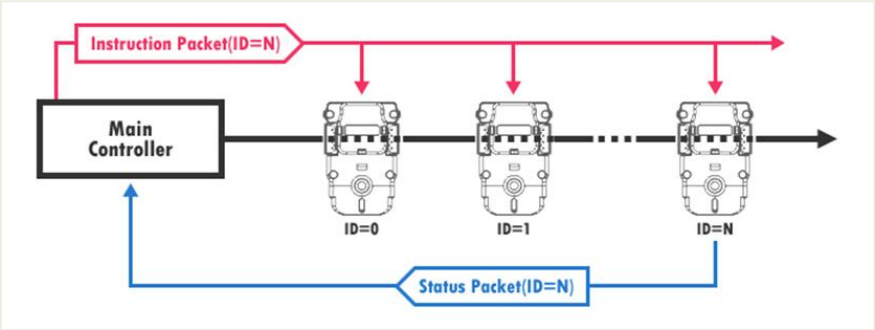


Figura 4 – Comunicação Serial em único barramento- RS-485

2.2.3. O protocolo de Comunicação dos Servo-Motores

O protocolo de comunicação é basicamente dividido em pacotes de instrução e pacotes de status. Os pacotes de instrução correspondem aos dados enviados pelo controlador mestre, sendo que estes dados são relativos a escrita de determinados comandos nos registradores do microcontrolador interno do motor. Os pacotes de status são os dados retornados do motor para o controlador mestre.

O datagrama dos pacotes trocados do dispositivo mestre para o servomotor segue o padrão abaixo, em heaxadecimal:

0xFF	0xFF	ID	LEN	INST	PARM1	PARM...	PARM(N)	CHK

O significado de cada campo é mostrado a seguir.

0xFF 0xFF: Caracteres de inicialização do pacote;

ID: É o número de identificação de cada um dos motores e possibilita seu endereçamento no barramento único da RS-485.

LEN: É o tamanho do pacote. O tamanho do pacote é calculado como o número de parâmetros (N) + 2.

INST: Define a instrução do respectivo pacote e pode assumir os valores da tabela a seguir:

Tabela 3 - Funções do parâmetro INST.

Valor	Nome	Função	Nº de Parâmetros
0x01	PING	Usado para ver se o controlador do servo está pronto para receber uma instrução.	0
0x02	READ DATA	Lê um dado do RX-28.	2
0x03	WRITE DATA	Escreve um dado no RX-28.	2 ou mais
0x04	REG WRITE	Similar ao WRITE DATA, mas não é executado até que o comando ACTION é enviado.	2 ou mais
0x05	ACTION	Executa movimentos enviados pelo comando REG WRITE.	0
0x06	RESET	Restaura o estado do RX-28 para o padrão de fábrica.	0
0x83	SYNC WRITE	Usado para controlar vários RX-28 ao mesmo tempo.	4 ou mais

PARM1..N: Usado quando a instrução requer um dado auxiliar.

CHK: É o checksum, ou seja, utilizado para verificar a integridade do pacote recebido. O campo CHK é calculado de acordo com a seguinte fórmula:

CHK = ~(ID+LEN+INST+PARM1+PARM...+PARM(N))

Os pacotes de status seguem o mesmo padrão do pacote de instruções:

0xFF	0xFF	ID	LEN	ERR	PARM1	PARM...	PARM(N)	CHK

A única diferença entre os dois pacotes é a substituição do campo INST pelo ERR. O campo ERR contém o código do estados de erro que o motor se encontra, adquirindo os valores presentes abaixo:

Tabela 4 – Código do campo ERR

Bit	Nome	Conteúdo
7	0	-
6	Instrução	Instrução enviada não pertence à lista prevista.
5	Sobrecarga	Corrente de carga não pode atuar no torque necessário.
4	Checksum	Soma verificadora do pacote recebido não está de acordo.
3	Range	Comando enviado está fora da tabela de uso.
2	Sobretemperatura	Temperatura interna do RX-28 excede o máximo seguro.
1	Limite de Ângulo	Posição solicitada excede os limites mecânicos do servo.
0	V de Entrada	Quanto à tensão aplicada está fora da região de operação.

3. O DESENVOLVIMENTO DA INICIAÇÃO ANTERIOR

3.1. Montagem da Placa de Navegação

Foi realizada a montagem da Placa de Navegação, seguindo os esquemáticos e listas de componentes presentes na Iniciação Científica anterior, utilizando os equipamentos e materiais presentes no Laboratório de Robótica e Inteligência Artificial. Para tanto foram utilizados componentes adquiridos anteriormente e a placa produzida em uma empresa especializada fabricação de placas de circuito impresso. O resultado da montagem é mostrado abaixo:

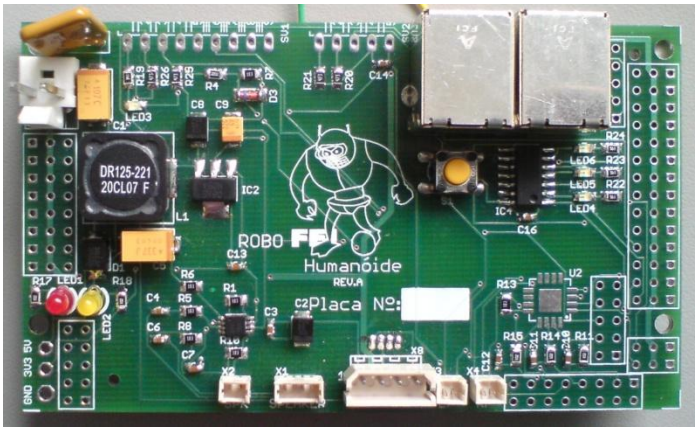


Figura 5 – A placa de Navegação Montada

3.2. O mini computador *Roboard*

Para todo o controle do Robô e dos servo-motores e periféricos, Eduardo Garcia escolheu o mini Computador *Roboard* [22], mostrado a seguir. Um do argumentos para

a sua utilização é a presença de uma porta RS485 integrada à placa, para a comunicação com os servo-motores diretamente através do barramento. Mais detalhes da escolha podem ser conferidas em sua Iniciação Científica.

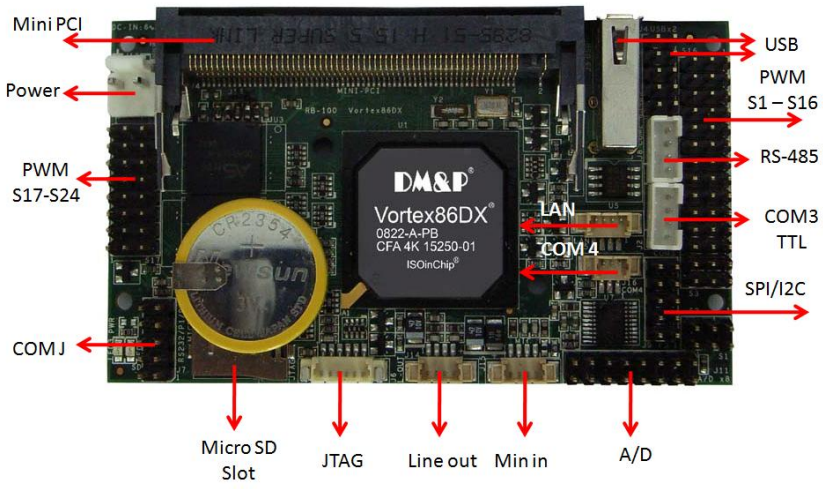


Figura 6 – Placa Roboard

3.3. Testes da placa de Navegação

Foram realizados os testes de todos os circuitos eletrônicos contidos na placa antes de sua utilização. Constatou-se as tensões corretas na saída dos reguladores de tensão, obtendo os 3,3v e os 5v. Para o teste da proteção da bateria foi utilizada a fonte variável do laboratório, variando a tensão até o ponto calculado para o desligamento da placa, constatando o correto desligamento.

Com os circuitos principais em funcionamento, começou-se os testes para o controle dos motores devido a necessidade de sua utilização em outra inciação científica "*Projeto Mecânico de um Robô Humanóide Futebol de Robôs – Humanóide League"* do aluno Milton Cortez [23].

Os testes iniciais foram a conexão da *RoBoard* a Placa de Navegação e por fim aos Servo-Motores. Utilizando o programa *Docklight* [24], instalado na própria *Roboard*, que é um terminal capaz de enviar caracteres pela porta serial do computador, foi enviado para a porta RS-485 caracteres de instrução relativos a movimentação do servo em conformidade com o protocolo dos motores. Mesmo com o envio dos caracteres corretos, cálculo do checksum, não houve qualquer resposta do servo motor. Tentou-se também o uso do programa feito exclusivamente para o configuração e teste dos servos-motores, o *Dynamixel Configurator*[25] também não obtendo êxito. Com o não funcionamento seguiu-se os seguintes procedimentos para verificar a integridade dos dados enviados e da construção da Placa de Navegação:

- Verificação da não inversão dos pinos de dados da porta RS-485 da RoBoard.
- Verificação da tensão de alimentação correta do Servo-Motor e sua corrente em repouso comparando-os com os dados fornecidos pelo fabricante.
- Verificação da Placa de Navegação pelo esquemático e trilhas da placa circuito impresso.

- Medição dos dados no barramento da RS-485 pelos níveis de tensão, utilizando diretamente o osciloscópio.
- Comparação com os níveis de tensão da comunicação RS-485.

Com todas as verificações de hardware estando corretas e mesmo com todos os dados sendo enviados aparentemente sem erros, pelas observações das medições do osciloscópio, não houve qualquer resposta do Servo-Motor, seja pelo retorno do pacote de status seja pela movimentação ou indicação do recebimento das instruções. Uma das justificativas para o não funcionamento é o uso de um Baud Rate fora de padrão pelo motor, de 57142 bps, sendo que o presente na porta do computador são exatos 57600bps, não permitindo a direta comunicação dos motores com a

Com isto houve a necessidade de se utilizar uma outra forma para o controle dos motores diretamente através da RoBoard, pois haveria necessidade em se utilizar um circuito externo para a geração de um Baud Rate não usual para a comunicação através de um chip conversor. Com isso observou-se também que a RoBoard não foi a melhor escolha para este Robô, pois possui muitas saídas que não seriam utilzadas, como pinos de PWM para servo-motores comuns, baixa capacidade de armazenamento em disco – apenas 2gb para todo o sistema operacional e programa para execução, pinos de GPIO não utilizados e sua maior vantagem, a presença da porta de RS485 integrada a placa não passível de funcionamento com os servo-motores escolhidos.

Devido a esta justificativa foi reformulado todo o cronograma que seria seguido ao longo da Iniciação Científica, pois o trabalho atual estava centrado inteiramente no trabalho anterior, que por sua vez não apresentou a sua implementação totalmente possível.

4. PROJETO DA PLACA USB/RS485

Com a necessidade do controle dos motores utilizando um computador pessoal, encontrou-se melhor meio de comunicação pela porta USB, sendo atualmente presente em todos os computadores, em contrapartida com a porta serial, não disponível facilmente em *Notebooks*. A Placa USB/RS485 será utilizada apenas no desenvolvimento inicial, sendo substituída por um microcontrolador na versão final do robô.

4.1. O padrão RS-485

O RS-485 ou EIA-485 [26] foi produzida pela *Electronics Industry Association*, a mesma criadora dos padrões RS-422 , RS-423 e o famoso padrão RS-232. O prefixo RS refere-se a uma recomendação de uso – *Recommended Standard* – do padrão, sendo que a RS-485 foi projetada para comunicação industrial de longas distâncias.

Algumas características deste padrão são listados a seguir:

- Comunicação a 2 fios em par diferencial.
- *Half-Duplex* – a 2 fios ou *Full-Duplex* – a 4 fios.
- Topologias de comunicação em: Ponto a Ponto, Multiponto e Barramento Único.
- Velocidade de transmissão de até 10 Mbits/s.
- Distâncias de transmissão de até 1200 metros.

No caso da comunicação com os Servo-Motores é utilizada a comunicação a 2 fios, no caso *Half-Duplex*. Por este motivo tanto os dados de transmissão quanto de recepção (TX-RX) utilizam o mesmo barramento e portanto devem possuir um controle do fluxo de dados.

4.2. O circuito integrado conversor de tensão

A conversão da RS-232 para a RS-485 um simples circuito integrado conversor de tensões como o MAX485 [27], produzido pela *Maxin*, ou o SN75176 [28], produzido pela *Texas Instruments*, no caso deste projeto, foi o utilizado o circuito integrado SN75176, mostrado na figura abaixo:

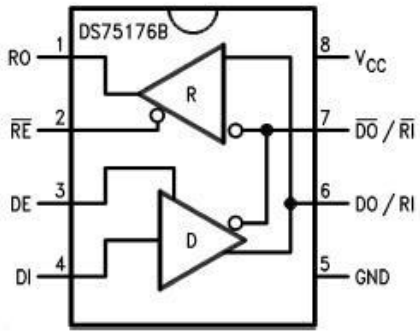


Figura 7 – SN75176

A descrição dos pinos é a seguinte:

RO – Receiver Output – Saída de dados do Receptor;

/RE –Receiver Enable –Habilitação do Receptor;

DE –Driver Enable - Habilitação do Transmissor;

DI –Driver Input –Entrada de dados no Transmissor;

VCC- Alimentação positiva do circuito integrado;

GND – Terra;

/DO - /RI – Driver Out, Receive In invertidos;

DO – RI –Driver Out, Receive In;

Nota-se, observando os pinos 6 e 7 que são complementares, o uso de um único barramento tanto para a transmissão quanto para a recepção de dados, sendo necessário o controle para a transmissão e recepção. Para algumas aplicações o controle do fluxo de dados é realizado por software, ou seja ativando os pinos /RE e DE ao mesmo tempo quando o programa necessitar enviar dados, entretanto para o projeto foi utilizado um controle por hardware, mais simples, que verifica a existência de nível de sinal de dados na entrada e ativa o pino de habilitação do driver e desablitação do receptor ao mesmo tempo.

4.3. O circuito projetado

O circuito foi projetado a partir de um artigo de Jan Axelson da *Circuit Cellar* com o título *"Designing RS-485 Circuits"* [29], onde é utilizado um outro circuito

integrado muito comum, o LM555 [30] na configuração monoastável com sua entrada ligada ao pino de recebimento de dados e a saída ligada a habilitação e desabilitação do driver. Resumindo, quando há dados as serem transmitidos, o pino de habilitação de driver será posto em nível lógico alto, caso contrário o circuito ficará em modo de recepção, esperando a chegada de dados.

O cálculo do tempo do monoastável para o circuito LM555 segue a fórmula 1:

$$T = C.R.ln(3) \qquad (1)$$

Onde:

T – tempo do monoastável

C – valor do capacitor em Farads

R – valor do resistor em Ohms

Para um baud rate padrão do servo-motor de 57143 baud, temos que o período dos pulsos é o inverso deste valor, ou seja:

$$T = 1/f \qquad (2)$$

$$T = 1/57143$$

$$T \sim 17{,}5\mu s$$

Para calacularmos o valor do tempo do monoastável, ou seja, o valor de R e C, fixamos o valor de C em 10 nF e calculamos o valor para meio período, ou apenas no tempo de nível lógico alto, assim temos que:

$$T/2 = C.R.ln(3)$$

$$8{,}74\mu s = 10nF.R.1{,}1$$

$$R = 795\ \Omega$$

O valor padrão mais próximo ao calculado é o de 820 Ω.

4.4. O conversor USB-Serial

Como meio de realizar a interface entre a porta USB e o conversor de tensões SN75176, foi utilizado um módulo adaptador *USB-UART* da Tato Equipamentos Eletrônicos [31], que basicamente é composto por um circuito integrado dedicado a esta tarefa, o FT232R da FTDI chip[32] , capaz de gerar *BaudRates* fracionários e não usuais para a comunicação com os servo-motores.

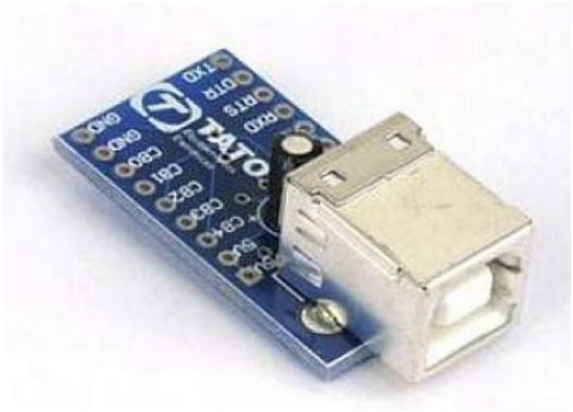


Figura 8 – Módulo USB-UART Tato

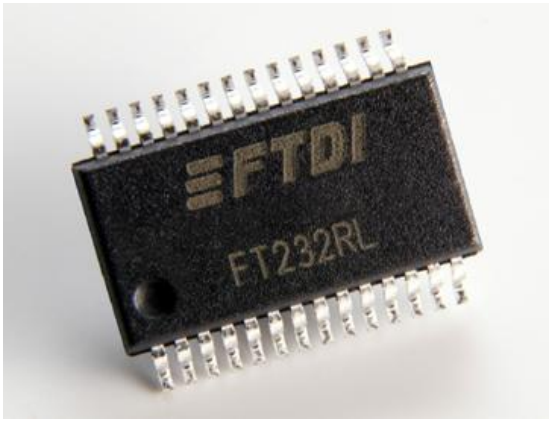


Figura 9 – FT232R

4.5. O circuito completo

Abaixo é mostrado o circuito completo inserido no robô junto do esquemático projetado, agora com o perfeito funcionamento e controle dos servo-motores. No Anexo C temos o esquemático em maior resolução.

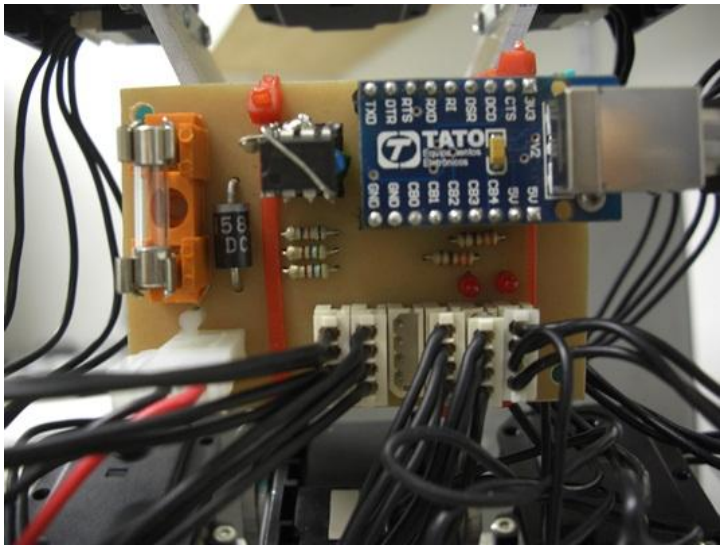


Figura 10 – A placa de controle inserida no Robô

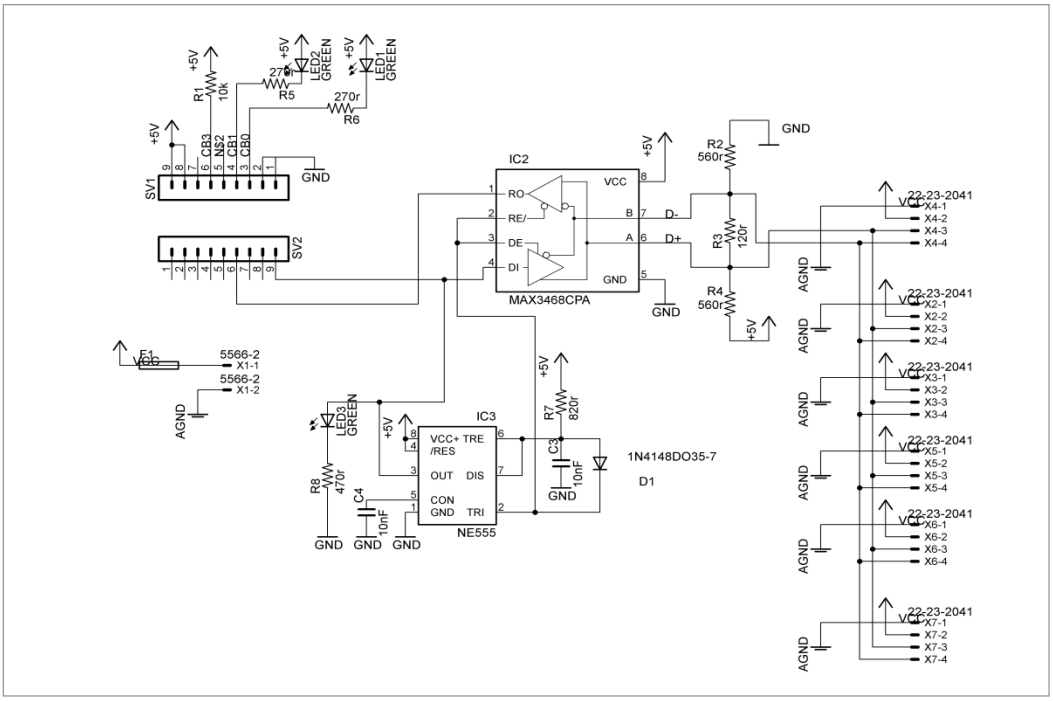


Figura 11 – O Esquemático do Circuito

Além dos circuitos integrados, foram colocados um diodo e um fusível de proteção para os motores além de dois leds indicando a transmissão e recepção de dados, junto de não apenas um conector, mas 6, cada conector responsável por um membro do corpo do robô.

Com a construção e montagem da placa de controle foi possível enfim o controle de todos os motores, deixando finalmente o robô na posição ereta e capaz de realizar os movimentos iniciais.

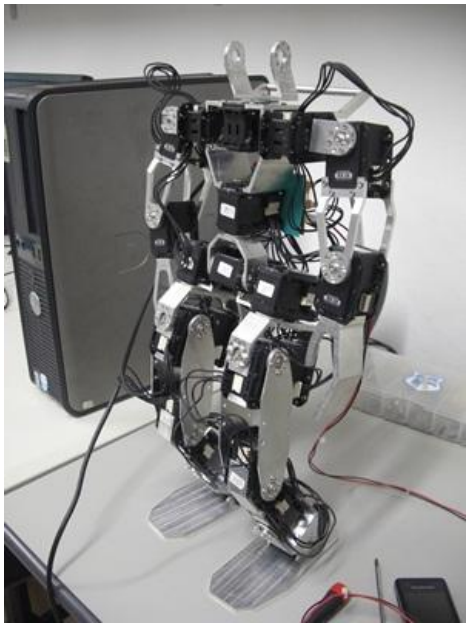


Figura 12 – O robô ereto

5. PROGRAMAÇÃO EM MATLAB

Para a realização dos testes necessários ao controle dos motores seria necessário a programação do protocolo de comunicação, criando os pacotes de envio, o cálculo do nº de parâmetros de envio, o *checksum*, e envio pela porta serial virtual criada pelo chip da FTDI.

Para a programação das funções foi utilizado o *MATLAB* da *Mathworks* [15] que é uma poderosa ferramenta de resolução problemas em engenharia baseado no uso de matrizes. Tal programa é largamente utilizado para a resolução de problemas matemáticos, equações de controle, modelagem de sistemas complexos, aquisição de dados, programação, e incontáveis outras aplicações. Devido a necessidade posterior da utilização do *MATLAB* para a aplicação do controle ao robô e também a sua modelagem, foi utilizado o programa no controle dos servo-motores dos robôs para a sua familiarização e por sua rápida implementação da programação devido a sua simples sintaxe.

No Anexo A temos o código que foi utilizado para a criação dos pacotes e o envio destes.

5.1. Aquisição de dados de posicionamento e análise

No *MATLAB* foi realizado uma sequência de comandos visando a aquisição de dados dos servo-motores através da leitura dos registradores internos de posição,

demonstrando que todas as funcionalidades do Rx-28 estavam acessíveis ao projeto e o domínio do protocolo dos motores. Para isso foi escrita a sequência de comandos que está listada abaixo

a) Inicialização da Porta Serial Virtual e variáveis presentes no loop;
b) Travamento de todos os motores em uma dada posição, enviando o comando em broadcast;
c) Leitura de todas as posições dos motores;
d) Destravamento de todos os motores;
e) Início do próximo loop;

Tal sequência guarda em uma matriz qual a posição de cada um dos Servo-Motores em cada passo da sequência da movimentação para o caminhar. Com todos os dados obtidos pode-se ter uma idéia da movimentação de cada um dos motores e de como um movimento complexo, que se utiliza de 3 motores ao mesmo tempo, varia em função do tempo.

Abaixo é mostrada uma sequência de imagens tiradas da movimentação dos 3 dos motores do braço.

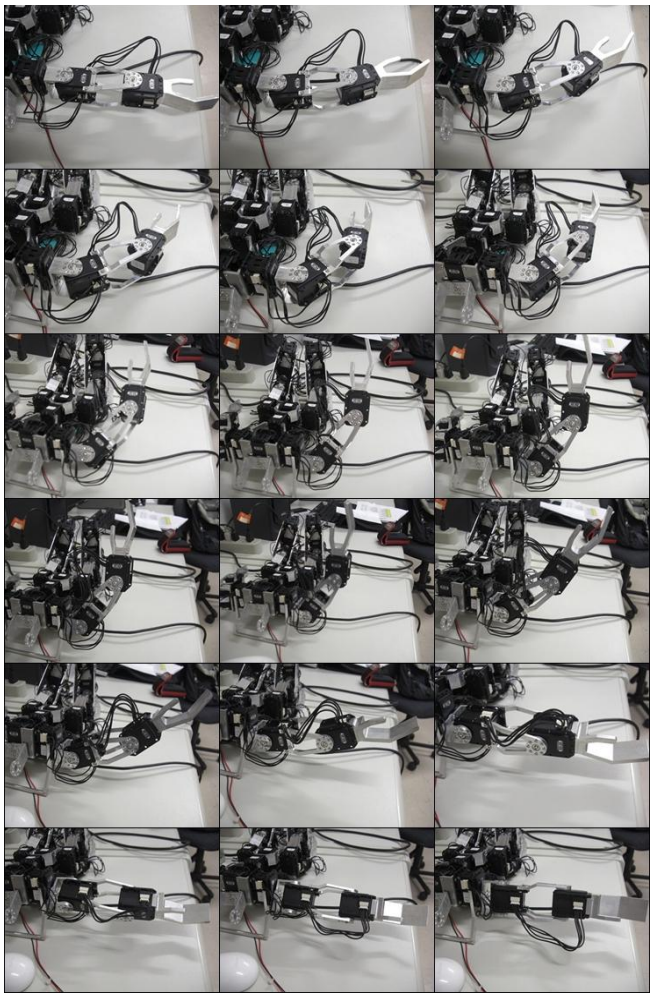


Figura 13 – Sequência de Movimentos do Braço Direito

Os dados de posição adquiridos foram inseridos em um gráfico mostrando a variação em graus a cada passo da sequência.

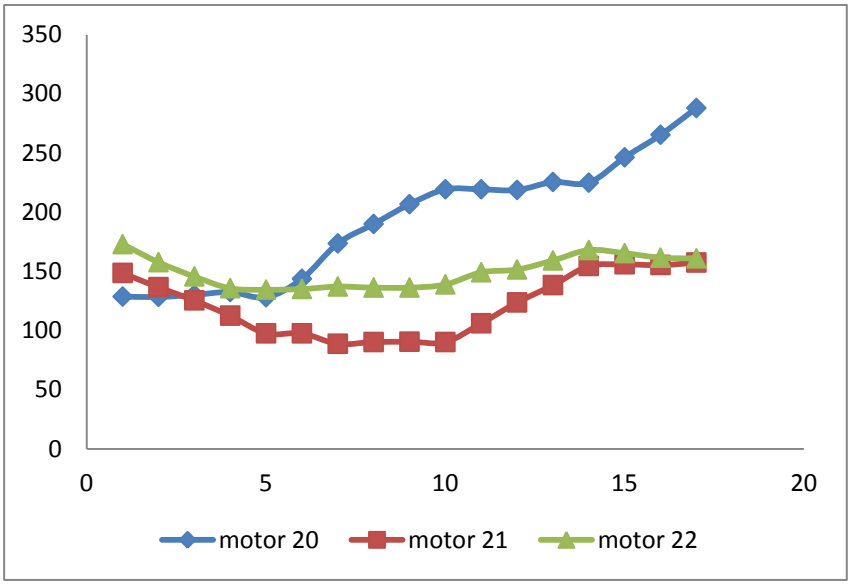


Gráfico 1 – Posição em graus a cada passo da sequência

É interessante notarmos a variação das leituras com a variação dos movimentos a partir deste gráfico, e de como ele é formado pelas movimentações dos 3 motores ao mesmo tempo. No Anexo B, temos o código exemplo do *MATLAB* comentado.

6. A INTERFACE E PROGRAMAÇÃO *STOP & GO*

Com os testes de envio de comandos para os motores estando validado, as funções de envio corretas, a aplicação do protocolo de comunicação dos motores e o recebimento de dados por parte deles, criou-se, a partir do software desenvolvido para a Iniciação Científica "Projeto Mecânico de um Robô Humanóide Futebol de Robôs – Humanóide League" do aluno Milton Cortez, uma interface *"Stop & Go"* , para a criação dos *scripts* necessários à movimentação. Tal método baseia-se em se ajustar a posição do robô, travar esta posição, adquirir os dados dos motores, gravá-los em um lista onde cada movimento ocorrerá em um instante específico e ao final, executar a movimentação ao pressionar o botão *Go.*

Utilizando-se esta interface podemos gerar *scripts*, ou seja, sequências de comandos pré-definidas dos servo-motores para a realização de tarefas como o levantar, o caminhar, o posicionamento do robô para o chute, a defesa do gol, etc, de uma forma mais simples e intuitiva.

6.1. A interface *Stop & Go*

Abaixo é mostrada a interface criada com respectiva explicação de cada um dos botões.

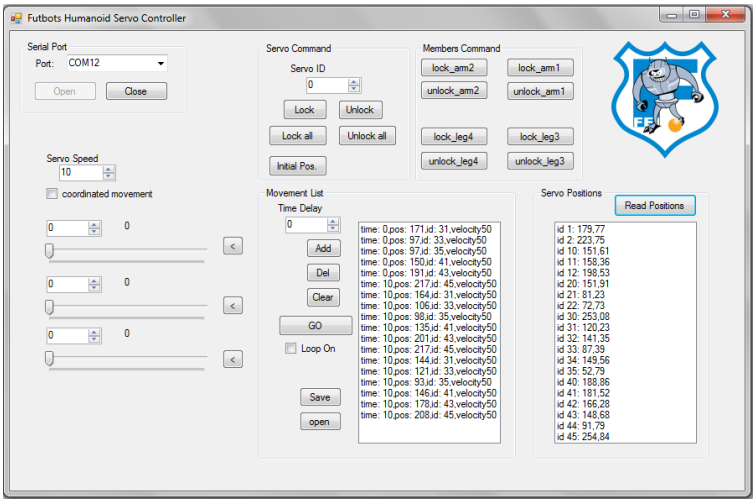


Figura 14 – Interface Stop & Go

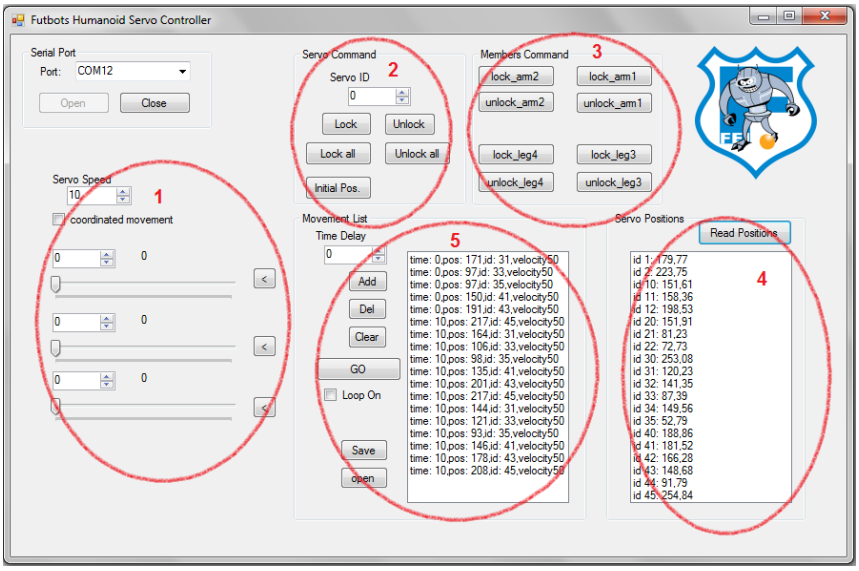


Figura 15 – Comandos da Interface

6.1.1. Coordinated Movement

Utilizado para coordenar os movimentos de 3 servo-motores ao mesmo tempo, ao deslocar apenas um dos *slider's* movimentamos todos os outros em relação a posição incial de referência.

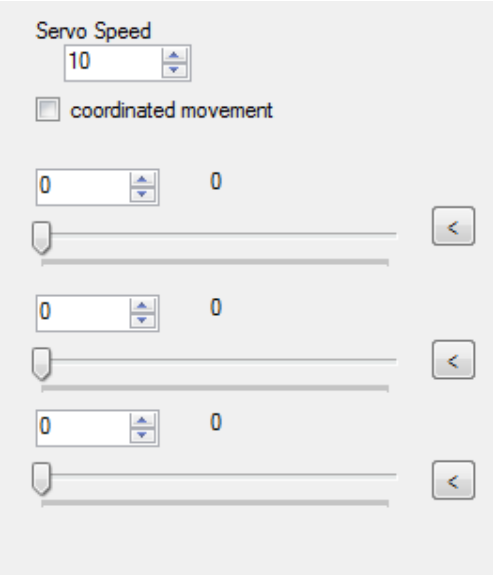


Figura 16 – Coordinated Movement

6.1.2. Servo Command

São funções relativas ao destravamento e travamento de todos os motores – envio do comando em *broadcast* – ou de apenas um dos motores, selecionado no Box 'Servo ID'. Também possui um botão de *initial position* para que o robô mantenha-se em sua posição inicial ereta.

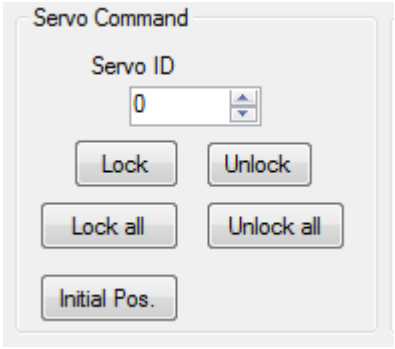


Figura 17 –Servo Command

6.1.3. Members Command

Tem como finalidade o destravamento e o travamento de apenas os membros do robô, como o braço esquerdo e direito e perna esquerda e direita. Os números 1,2,3 e 4 se referem as identificações de cada um dos motores em relação ao corpo do robô, sendo respectivamente o braço direito, braço esquerdo, perna direita e perna esquerda. Com isso a criação dos *scripts* de movimento são mais facilmente implementadas.

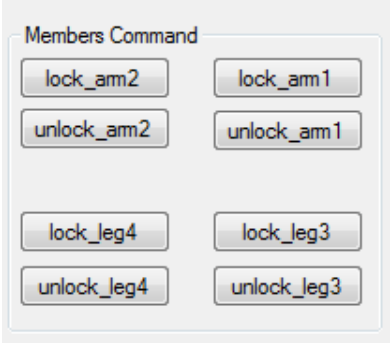


Figura 18 – Members Command

6.1.4. Read Positions

Utilizado para ler as posições atuais em graus de todos os 21 motores do humanoide.

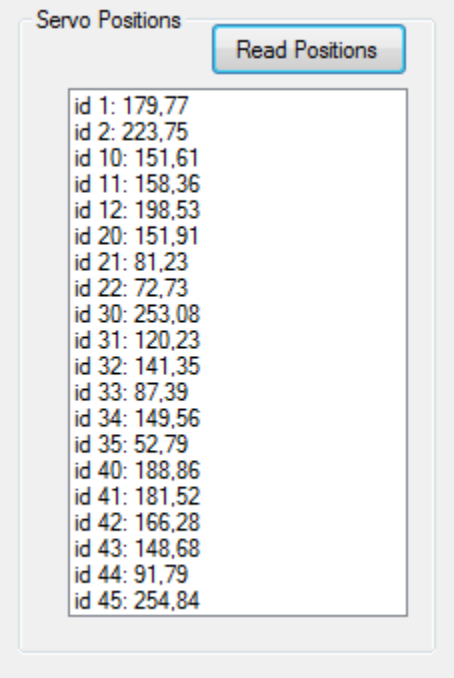


Figura 19 – Read Positions

6.1.5. Movement List

Cria os scripts de movimento, ao adicionar cada uma das posições de cada um dos motores, junto de um tempo para o acionamento e de uma velocidade. Com tal lista é possível criar os scripts necessários e a seguir pressionar o botão de *Go* para executar a movimentação que foi programada. Também é possível salvar este script e carregá-lo

depois. Foi adicionado um *Box* que liga o loop em que os movimentos descritos são executados indefinidamente.

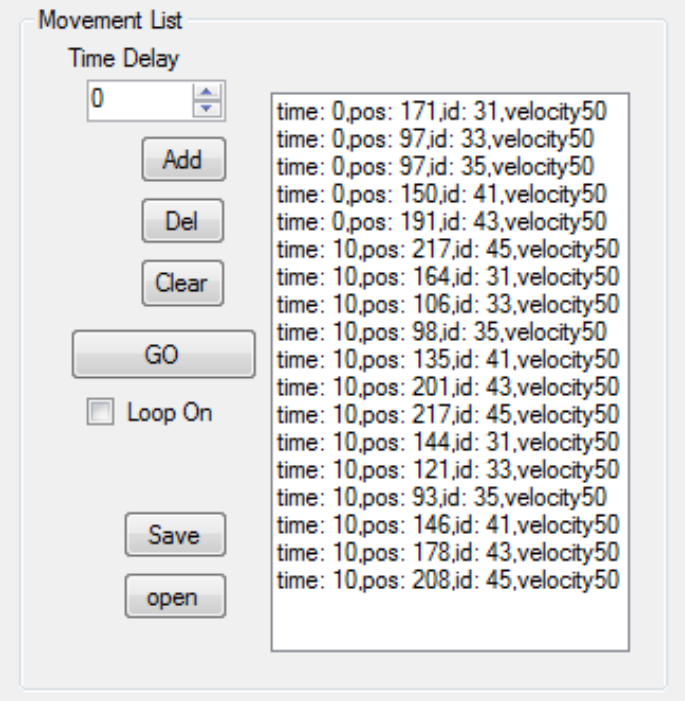


Figura 20 – Movement List

6.2. Observações e comentários

Com a utilização desta interface foi possível criar um movimento mais complexo que com o uso do *Matlab* - tornou-se mais intuitiva a programação dos *scripts* - e para demonstrar sua funcionalidade foi programado uma sequência de comandos com a finalidade do caminhar, estando disponível o vídeo no endereço: http://www.youtube.com/watch?v=IwsCAQOV7bc .

O robô ainda não caminha com naturalidade devido as bruscas transições entre os movimentos e sua malha de controle ainda estar “aberta” sem a realimentação dos sensores inerciais.

Um fato interessante a ser comentado sobre o vídeo de demonstração é o robô andar em círculos, isso se deve a baixa capacidade de transmissão de dados do ci SN75176 utilizado na placa de controle USB/RS485, devido ao seu *slew rate* de até 130ns, fazendo com que a maior taxa de transmissão de dados não passasse dos 115200 bits/s. Para se evitar perdas nos pacotes foi utilizado o baud rate padrão do motor de 57143 bits/s e delays propositais no software da interface entre o envio dos comandos. Se por um lado temos uma maior integridade dos pacotes agora, por outro temos um atraso em sua movimentação, fazendo com que o robô andasse de forma circular, como mostrado no vídeo. Todos os problemas observados serão elimidados com o uso do microcontrolador com velocidade de transmissão de 1Mbit/s para a comunicação entre os motores e o controlador e de um circuito integrado *transceiver* com maior taxa de transmissão de dados que o *transceiver* anterior, o MAX3485 [33] para 10 Mbits/s.

Um outro comentário sobre a interface é que mesmo com a sua utilização, o movimento do robô para o caminhar ainda é uma tarefa muito complexa que exigirá um trabalho exclusivo sobre o tema, envolvendo o detalhamento da malha de controle e a cinemática inversa do robô humanoide, bem como ajustes dos parâmetros de movimentação e critérios de estabilidade.

Os testes apresentados não tiveram a finalidade em se realizar a movimentação precisa do robô humanoide, apenas como meio de demonstração das possibilidades reais do caminhar, verificando que os servo-motores, a mecânica e sua estrutura estão compatíveis ao desenvolvimento do controle e que a interface é de grande utilidade ao desenvolvimento de todo o robô.

7. O SISTEMA ELETRÔNICO DO ROBÔ HUMANÓIDE

7.1. Diagrama de Blocos do Sistema

Para uma melhor organização do sistema eletrônico, abaixo está indicado o diagrama de blocos representando as principais partes do sistema:

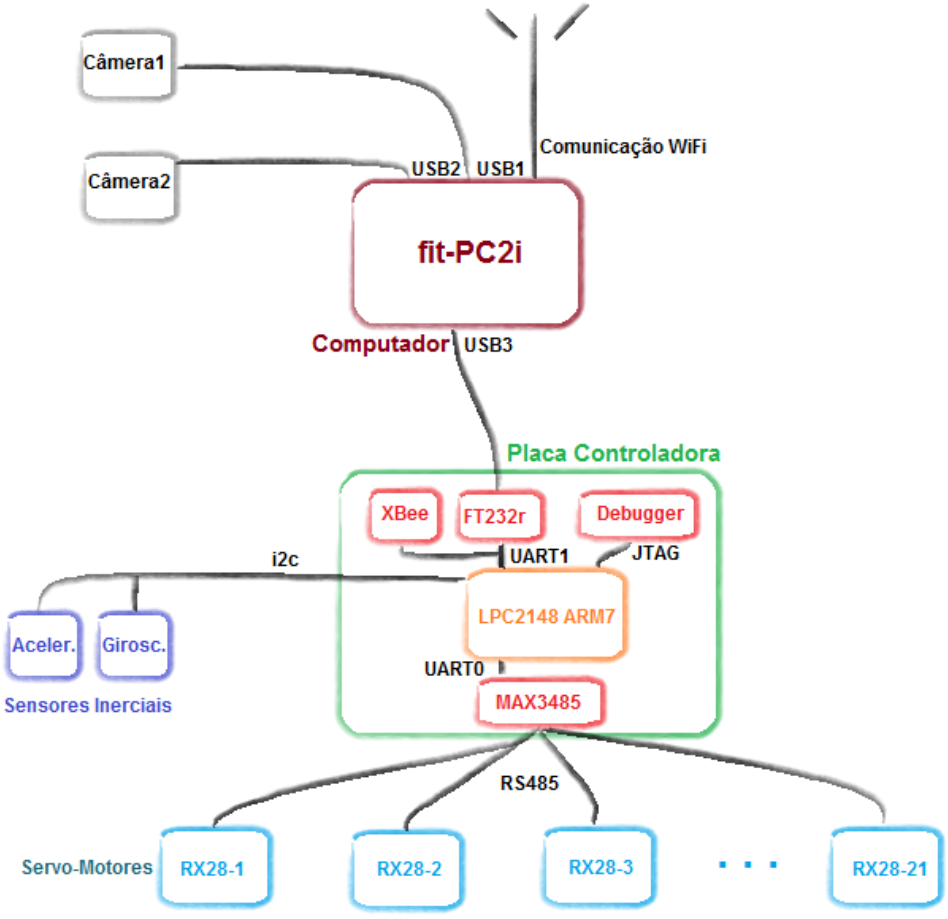


Figura 21 – Diagram de Blocos do Sistema

7.2. O computador fit-PC2i

Para o processamento das imagens provenientes das duas câmeras, o controle da máquina de estados do jogo de futebol, o recebimento do controle do jogo pelo juiz via WiFi e o controle do hardware de baixo nível, baixo consumo de energia e dimensões, é necessário um computador compatível com todas as funções que deverá exercer.

Algumas características do computador fit-PC2i[34] são:

- CPU - Intel Atom Z530 1.6GHz – comparável ao processamento de netbooks
- Chipset Intel US15W SCH
- Memória – 1GB DDR2-533
- Armazenamento – Solid State Disk 8GB
- Comunicação – WLAN - 802.11b/g/n; 2 x gigabit Ethernet
- Operação *"fanless"* – o próprio gabinete de alumínio é o dissipador do processador
- 4 portas USB 2.0
- Consumo de 8W @ 12vdc em processamento máximo
- Pequenas dimensões e massa - 101 x 115 x 27 mm, 370 gramas

Figura 22 – O Computador fit-PC2i

7.3. A Placa Controladora

A Placa Controladora desenvolvida tem por finalidade liberar o computador principal das tarefas de baixo nível, como a movimentação dos servo-motores e realizar a leitura do acelerômetro e giroscópio via barramento I²C, não encontrado em computadores, apenas em microcontroladores. Com o computador encarregado de tarefas mais complexas, como a parte da visão, a estratégia de jogo e a comunicação wireless, teríamos um atraso para a movimentação dos servo-motores devido ao fato observado com o uso direto com o computador: atrasos e perda de pacotes de dados, ocasionando movimentações imprecisas e diferenças nos tempos de execução de cada um dos servo-motores.

Com o computador enviaríamos comandos de alto nível para a placa controladora, como o “andar para frente” ou “andar de lado”, e estes comandos seriam interpretados

pelo microcontrolador e executados, sendo que a malha de controle estaria confinada no baixo nível, não teríamos necessidade de processar estes dados no computador principal.

A placa controladora contém a eletrônica necessária ao funcionamento do robô e possui essencialmente o microcontrolador, o conversor usb-uart que realiza a comunicação entre o fit-PC2i e o módulo microcontrolador, o circuito conversor de níveis de tensão padrão RS485 para a comunicação com os servo-motores, o barramento i2c para a leitura dos registradores dos sensores e o conector J-TAG para *debugger in circuit*.

Abaixo estão relacionados os circuitos principais da placa controladora.

7.3.1. O módulo microcontrolador

Para o humanoide foi utilizado um módulo da fabricante eSysTech[35] eLPC64 do tipo SOM – *Sytem on Module* – contendo um microcontrolador de 32 bits com seu núcleo baseado na arquitetura ARM7TDMI-S da fabricante NXP[36], o LPC2148[37]. Este módulo possui reguladores de tensão de 3,3vdc e 5vdc para o funcionamento do microcontrolador e para periféricos externos ao módulo, cristais de 32.768KHz para o RTC – *real time clock* – e de 16MHz para o núcleo – podendo chegar a 48MHz via PLL – e dois leds para indicação de funcionamento e do pino de *reset* do microcontrolador.

Abaixo temos uma imagem do módulo utilizado na placa controladora.

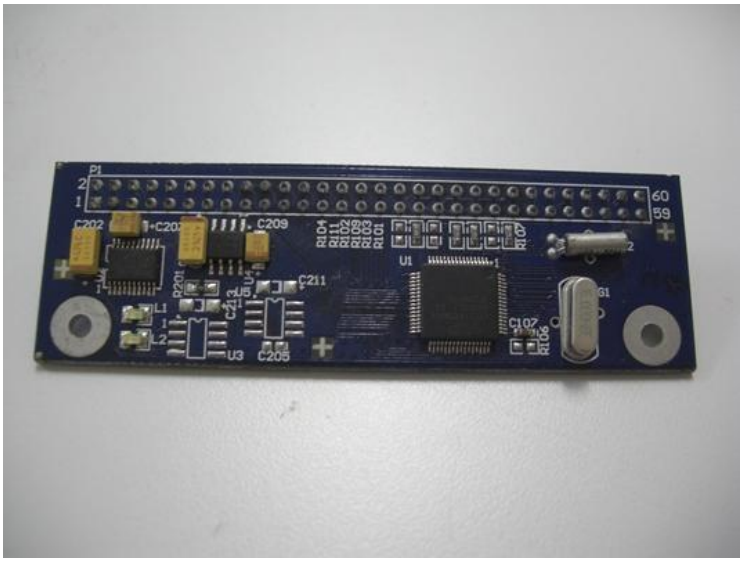


Figura 23 – Módulo eSysTech LPC2148

Com a utilização do módulo temos uma série de vantagens como a facilidade de implementação da placa de circuito impresso, pois o módulo possui barramento de pinos com largura padrão de 2,54 mm com acesso a todos os pinos do microcontrolador, a rapidez no desenvolvimento ao possuir os componentes necessários ao funcionamento e a saída de 5vdc e 3,3vdc regulada para a alimentação de circuitos externos.

7.3.2. Os sensores inerciais

Os sensores inerciais são de grande importância para robôs bípedes pois possibilitam a criação das malhas de controle para estabilização de um sistema que é naturalmente instável devido a geometria ereta do humanoide. O uso de um acelerômetro

de 3 eixos, em x, y e z, possibilita a medição da aceleração linear, enquanto o giroscópio possibilita a medição da velocidade angular, ou seja, a velocidade de rotação nestes 3 eixos, independente da aceleração gravitacional. Foram utilizados sensores em módulos *breakout* que possuem todos os pinos no padrão de 2,54 mm de espaçamento, já soldados na posição correta e com seus pinos facilmente acessíveis. Abaixo é mostrada a placa contruída para acomodar os dois sensores e o cabo com o conector para o encaixe na placa de controle.

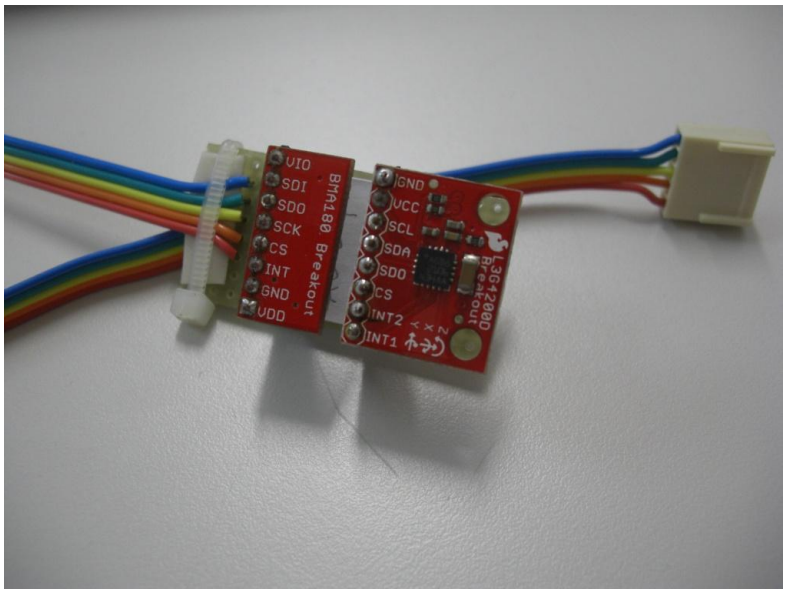


Figura 24 – Sensores BMA180 e L3G4200d

7.3.2.1. O barramento I²C

Ambos os sensores possuem comunicação SPI [38] – *Serial Peripheral Interface Bus* – ou I²C [39] – *Inter Integrated Circuit* – , que são os protocolos mais comumente encontradors na comunicação entre controladores e periféricos como sensores, conversores analógico-digitais, etc. Para a comunicação entre o microcontrolador LPC2148 e os sensores foi escolhido o protocolo I²C por este utilizar apenas 2 fios no barramento, enquanto no protocolo SPI é necessário uma linha de *slave select* para cada um dos sensores para ser realizada a comunicação. Abaixo a comparação entre as duas formas de comunicação e a necessidade de um barramento muito maior no protocolo SPI.

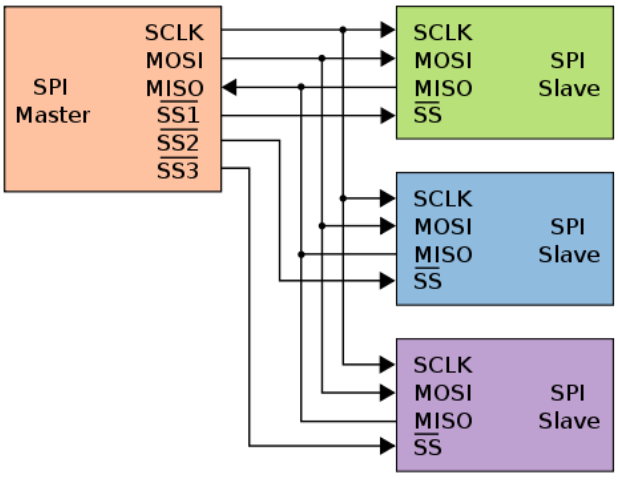
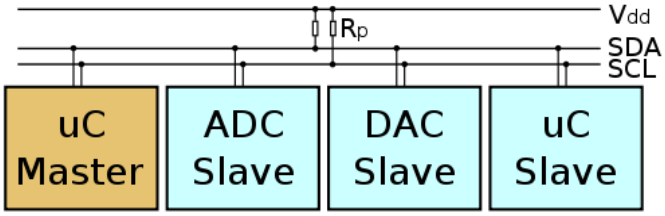


Figura 25 – Protocolo SPI



Fgura 26 – Protocolo I²C

O protocolo I²C foi desenvolvido pela *NXP Semiconductors* para a comunicação entre diferentes periféricos, como no caso da figura acima onde temos um ADC um DAC e microcontrolador escravo em um mesmo barramento, diminuindo em muito o número de linhas de transmissão de dados e facilitando a inserção de novos dispositivos em um barramento já existente.

A camada física envolve apenas duas linhas, SCL e SDA, a primeira consistindo do sinal de clock e a segunda do canal de dados serial, sendo ambos bidirecionais, do mestre para o escarvo e vice-versa. Ambas as linhas são do tipo “coletor aberto” onde o transistor de driver de saída não tem seu coletor conectado e devido a isso há a necessidade de resistores de *pull-up* para a polarização.

Para iniciarmos uma transmissão devemos realizar uma condição de *START* que consiste em se colocar a linha SDA de nível lógico 1 para 0 enquanto mantemos o SCL em 1. Do mesmo modo teremos uma condição de *STOP* quando a linha SDA para de 0 para 1 enquanto SCL está em 1. Abaixo a acarta de tempos ilustra as condições:

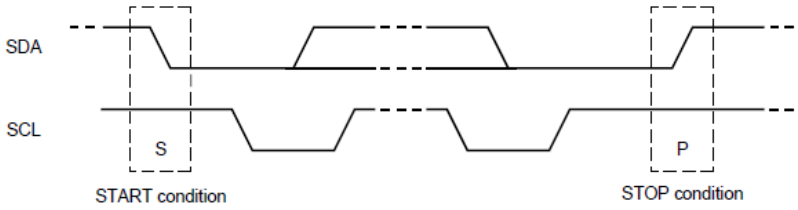


Figura 27 – Condições de Start e Stop do I²C

Após enviarmos uma condição de *START* pelo controlador mestre, necessitamos enviar o endereço do dispositivo escravo junto do bit de leitura/escrita. O endereço do escravo possui 7 ou 10 bits dependendo do formato de I²C utilizado, sendo que os sensores utilizados possuem endereço de 7 bits, com o 8º bit como indicador de leitura ou escrita. Com o fim do envio do endereço mais o bit indicador R/W, caso exista no BUS um escravo com este endereço, o escravo retornará um sinal de ACK – *acknowledgement*, reconhecimento – indicando ao mestre que está presente no barramento. Com isto poderemos então enviar dados relativos ao endereço do registrador, valores a serem escritos ou lidos, etc. A cada dado válido recebido, o dispositivo escravo retorna um ACK, indicando uma operação bem sucedida.

Abaixo mostramos a carta de tempos resumindo a operação completa de comunicação:

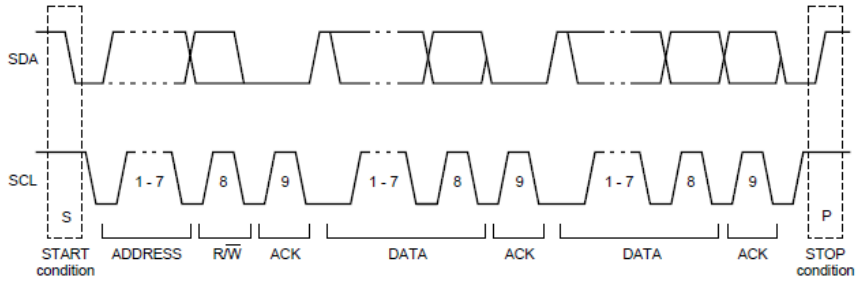


Figura 28 – Comunicação completa I²C

7.3.2.2. O módulo acelerômetro

O módulo utilizado é o da *Sparkfun Electronics*[40] que possui um acelerômetro de 3 eixos, Bosh BMA180[41] . Foi escolhido este, dentre muitos modelos existentes, por possuir grande versatilidade e funções que poderão ser usadas no humanóide:

- 3 eixos de medição de aceleração – x, y e z
- Escalas de medição de aceleração selecionável pelo usuário - +/- 1g, +/- 1,5g, +/- 2g, +/- 3g, +/- 4g, +/- 8g, +/- 16g.
- Resolução de 14 bits ou 12 bits
- Filtros digitais internos selecionáveis – 8 filtros passa-baixas com frequências de cortes diferentes, 1 passa-altas e 1 passa-banda
- Interrupções programáveis para o microcontrolador – Detecção de alta e baixa aceleração, *tap-sensing* e detecção de curva de aceleração
- 2 modos de operação – Baixo consumo ou baixo ruído

Diante de suas características, nota-se o grande número de funcionalidades que serão utilizadas no humanoide, como a escolha precisa da escala de aceleração necessária – de +/- 1g até +/-16g – os muitos filtros digitais selecionáveis pelo usuário, que permitirão obter medições capazes de atuar na malha de controle diretamente, interrupções que poderão detectar rapidamente a queda do robô e muitas outras funcionalidades.

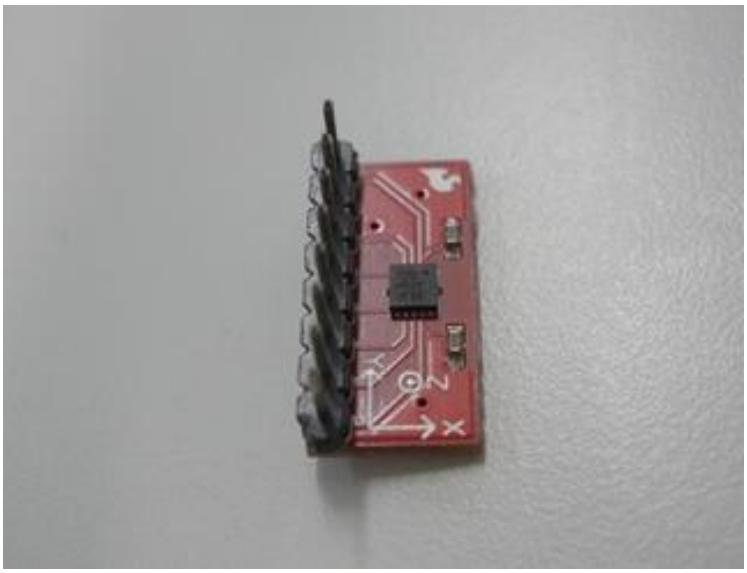


Figura 29 – Módulo Acelerômetro

7.3.2.3. O módulo giroscópio

O giroscópio também é um módulo da *Sparkfun Electronics*[42] possuindo o sensor L3G4200D[43] com rotação em torno dos eixos x,y,z.

Assim como o acelerômetro, foi escolhido este giroscópio por possuir algumas funções que serão de grande utilidade ao robô, podendo destacar as seguintes:

- 3 escalas de medição de velocidade angular: 250, 500 e 2000 graus/segundo
- 2 linhas digitais para interrupção e dados prontos para envio
- Filtros passa altas/baixas com banda definida pelo usuário
- Sensor de temperatura integrado
- Comunicação SPI/I²C

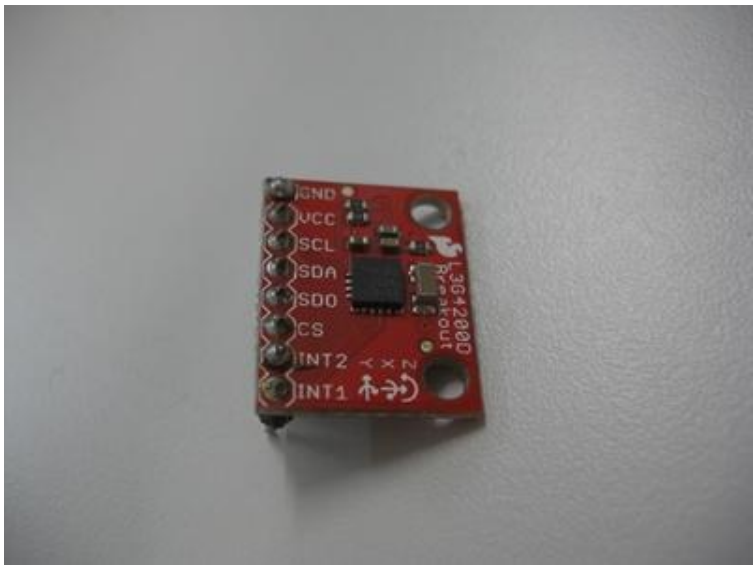


Figura 30 – Módulo Giroscópio

7.3.3. Os módulos e circuitos periféricos

Além do circuitos mencionados foram adicionados ao projeto da Placa de Controle um módulo de comunicação sem fio XBee series2 [44], com a capacidade de transmitir e receber dados via uart de um computador remoto. Sua inserção facilita o desenvolvimento do código sem a necessidade de cabos. Também foi adicionado um módulo conversor USB-Serial da *Sparkfun Elctronics*[45], semelhante ao módulo utilizado na placa USB-RS485.

Para a comunicação com os servo-motores foi utilizado o circuito integrado de transceiver RS485 MAX3485[33], que permite velocidades muito maiores de comunicação, até 10Mbps.

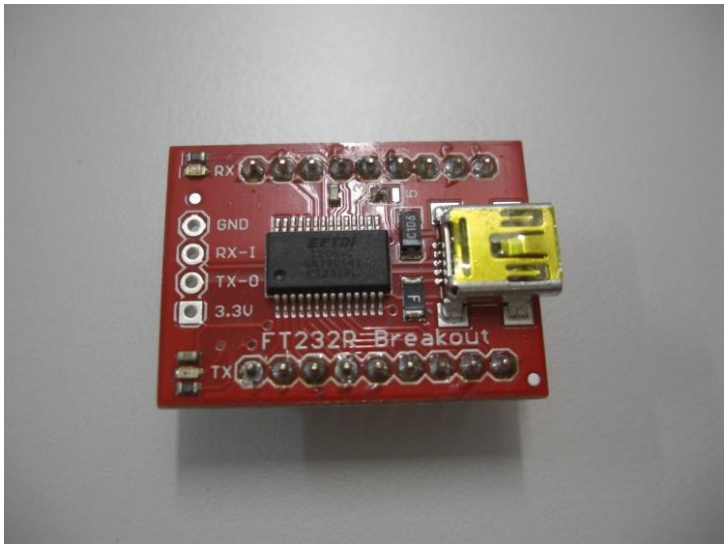


Figura 31 – Módulo com o chip FT232R

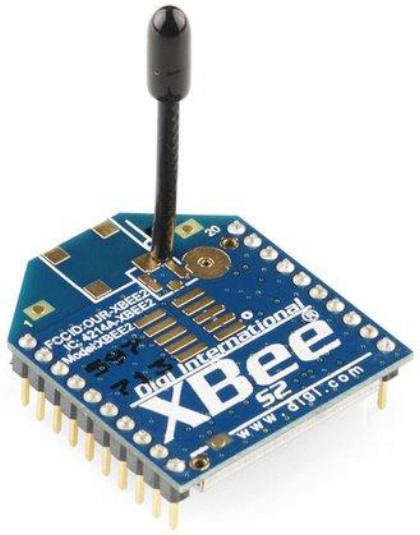


Figura 32 – Módulo XBee Series 2

Para a utilização do debugger, é necessário um conector do tipo header de 20 pinos, compatível com o padrão ARM JTAG[46].

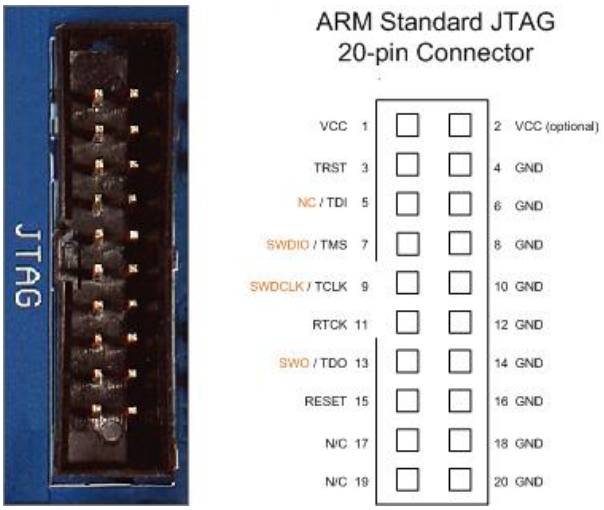


Figura 33 – Conector JTAG de 20 pino

7.3.4. A Placa de Controle prototipada

Para acomadarmos todos os periféricos foi desenvolvida uma placa de circuito impresso utilizando o software *Cadsoft Eagle*[47], para elaboração do esquemático e do layout da placa de circuito impresso.

Para o funcionamento da placa foi adicionado um regulador de tensão linear da LM7805[48] para a alimentação do módulo ARM pela bateria de 14,8v. Contudo, o uso deste tipo de regulador, que diminui a tensão ao dissipar calor ao ambiente deve ser evitado pois sendo o robô alimentado por baterias, há de se pensar o uso racional da energia disponível. Entretanto optou-se por sua utilização devido a grande facilidade de acomodação em termos de espaço quanto de implementação, além de que esta placa será usada somente durante o desenvolvimento. Na versão final será utilizado um regulador chaveado, de maior eficiência energética.

Por se tratar de um protótipo, a placa foi confeccionada pela CNC do Centro de Laboratórios Elétricos do Centro Universitário da FEI em placa de fenolite. No Anexo D é dado o seu esquemático.

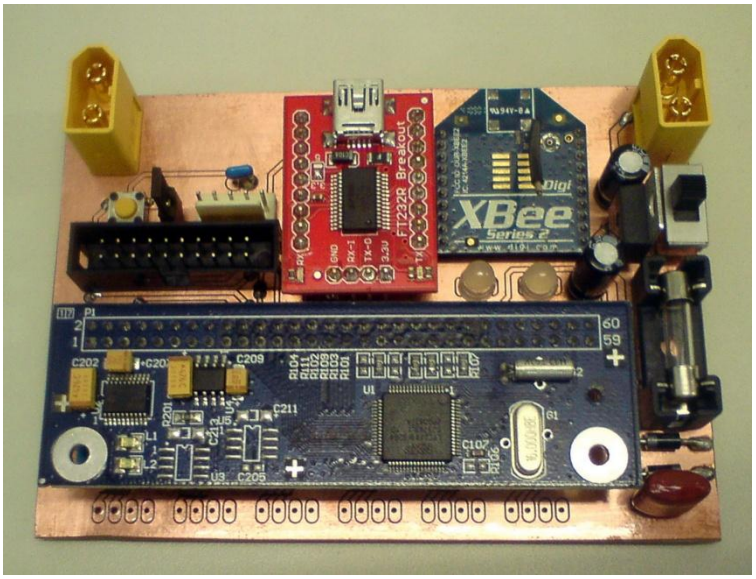


Figura 34 – Protótipo Placa de Controle

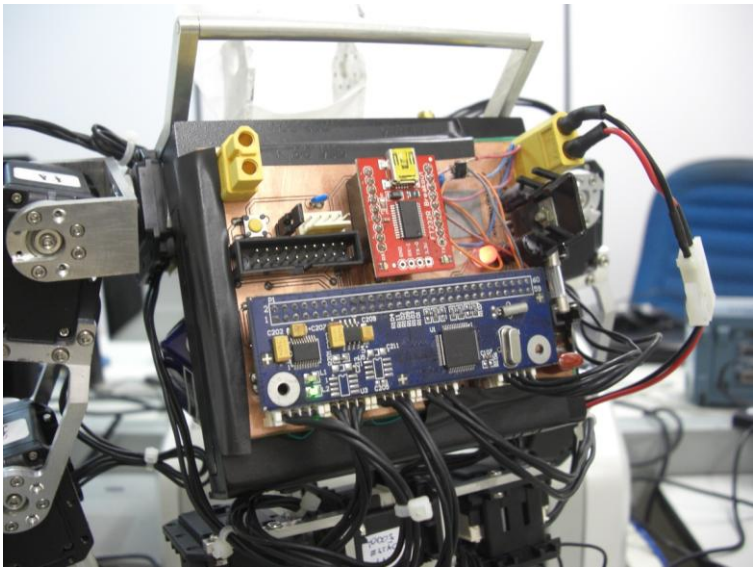


Figura 35 – Placa de Controle nas costas do robô

7.3.5. A Placa de Controle a ser produzida

Como projeto para a implementação da eletrônica do robô humanoide foi projetada a Placa de Controle em sua versão final. As maiores diferenças entre a placa prototipada e a placa a ser produzida industrialmente são listadas abaixo:

- Placa de fibra em FR4, muito mais resistente a choques mecânicos e umidade que o fenolite da placa prototipada
- Possibilidade da execução de trilhas com espessura muito menor que a resolução da CNC
- Máscara de solda e Silk com as numerações de componentes
- Possibilidade da utilização de componentes em encapsulamento SSOP, necessário ao chip FT232R
- Diminuição do tamanho geral da placa

Abaixo é mostrada a placa desenvolvida em uma visão 3D, tanto superior quanto inferior.

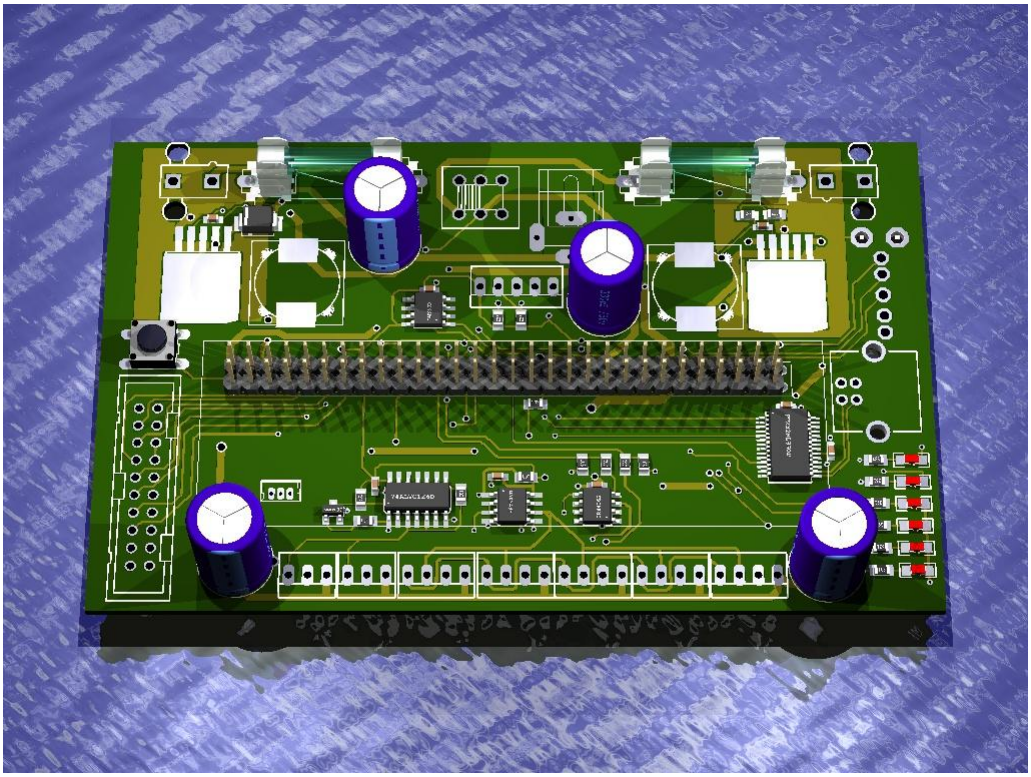


Figura 36 – Placa Controladora versão final superior

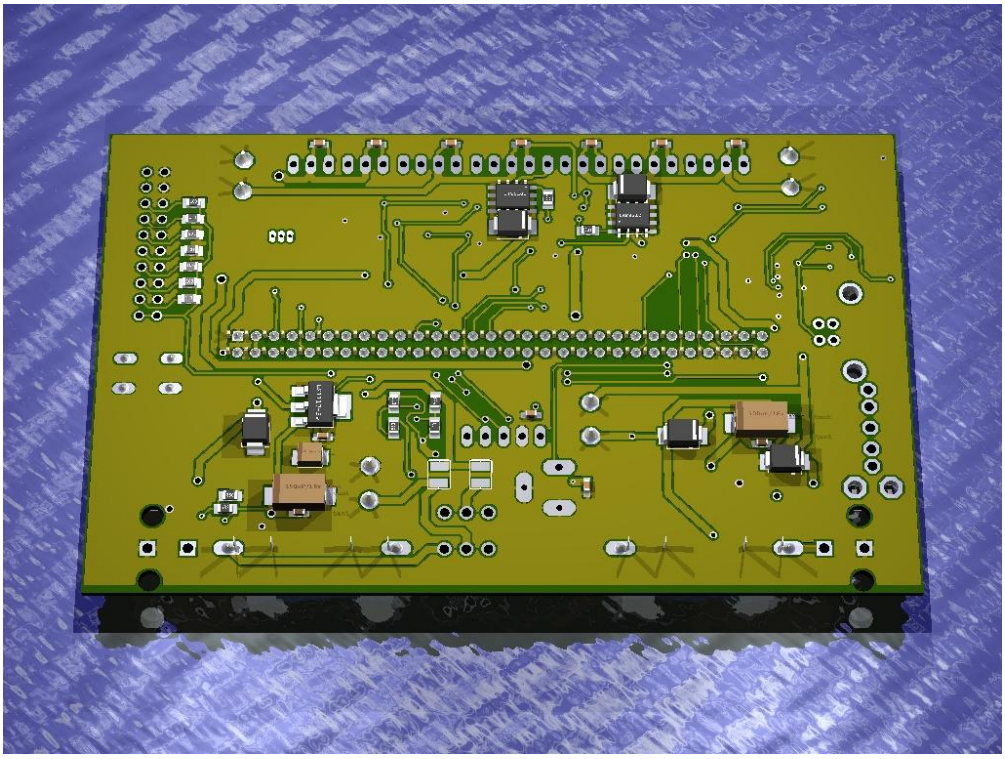


Figura 37 – Placa Controladora versão final inferior

Em relação ao projeto original desenvolvido esta placa possui algumas pequenas diferenças que durante a fase de testes da placa anterior julgou-se necessário.

Dentre as modificações, foi retirado o módulo xbee pois não será mais necessária a sua utilização na versão final todo o controle será feito através do computador principal, o módulo com o chip FT232R foi retirado, deixando apenas o componente principal a ser soldado na placa. Criou-se um circuito de *motor_enable* para a ativação da alimentação dos motores a partir do microcontrolador principal. Sua principal função é de ativar os

motores sequencialmente, diminuindo a corrente de pico ao ligarmos a placa. Neste circuito foi utilizado um MosFET em encapsulamento SMD SO8, o IRF9310 [49] de até 20A de corrente máxima suportada.

Outra modificação de grande impacto ao consumo foi a utilização de reguladores chaveados e um regulador linear LDO –*low dropout output* – para a alimentação do computador principal e de toda a placa. Foram escolhidos os LM2596 [50] , um regulador chaveado para até 3A e tensão máxima de entrada de 45v. O regulador linear LDO – *low drpout output* – escolhido foi o LM1117 [51].

Todos os circuitos e modificações podem ser conferidas no Anexo E, com o esquemático da versão final.

7.4. O *debugger*

A programação do microcontrolador e a depuração do código é realizada pelo *j-link edu* [52] da *Segger* um debugger já integrado a interface de programação *IAR Workbench*[53]. O uso de um debugger *in circuit* no projeto promove uma maior velocidade de desenvolvimento por ser possível realizar:

- Gravação no próprio circuito ao utilizar o conector do tipo *JTAG* com seus pinos conectados ao microcontrolador
- Depuração do código em tempo real

- Uso de *breakpoints*, pontos no programa onde pode-se pausar a sequência de instruções
- Capacidade de leitura e escrita de todos os registradores dos periféricos do microcontrolador durante a execução de um programa
- Conexão USB, sem a necessidade da utilização de uma porta paralela ou serial

Figura 38– Debugger j-link edu

8. PROGRAMAÇÃO DA PLACA DE CONTROLE

Após o projeto da Placa de Controle, iniciou-se o desenvolvimento da programação do microcontrolador para a leitura dos dados provenientes dos sensores e seu tratamento e a inserção do protocolo dos servo-motores para envio de comandos de controle, como a posição e a velocidade.

8.1. A *IDE IAR Workbench*

A programação do microcontrolador LPC2148 se deu através da IDE IAR Workbench para microcontroladores ARM. A versão utilizada é 6.30 a qual inclui suporte a inúmeros microcontroladores e exemplos de programas com bibliotecas abertas e funcionais.

Para se iniciar um novo projeto, é necessário criar um workspace, ou a área de trabalho que conterá todos os arquivos de programação. A linguagem de alto nível utilizada é o “C” que se permite criar programas relativamente complexos em poucas linhas de código.

A *IDE* possui algumas funcionalidades e seu ambiente é totalmente integrado ao debugger *j-link*. Pode-se destacar o uso dos breakpoints para depuração do código, os comandos para visualização dos registradores do microcontrolador e a visualização das variáveis utilizadas.

Abaixo é mostrado o ambiente de programação, e alguns breakpoints adicionados ao código, pontos vermelhos.

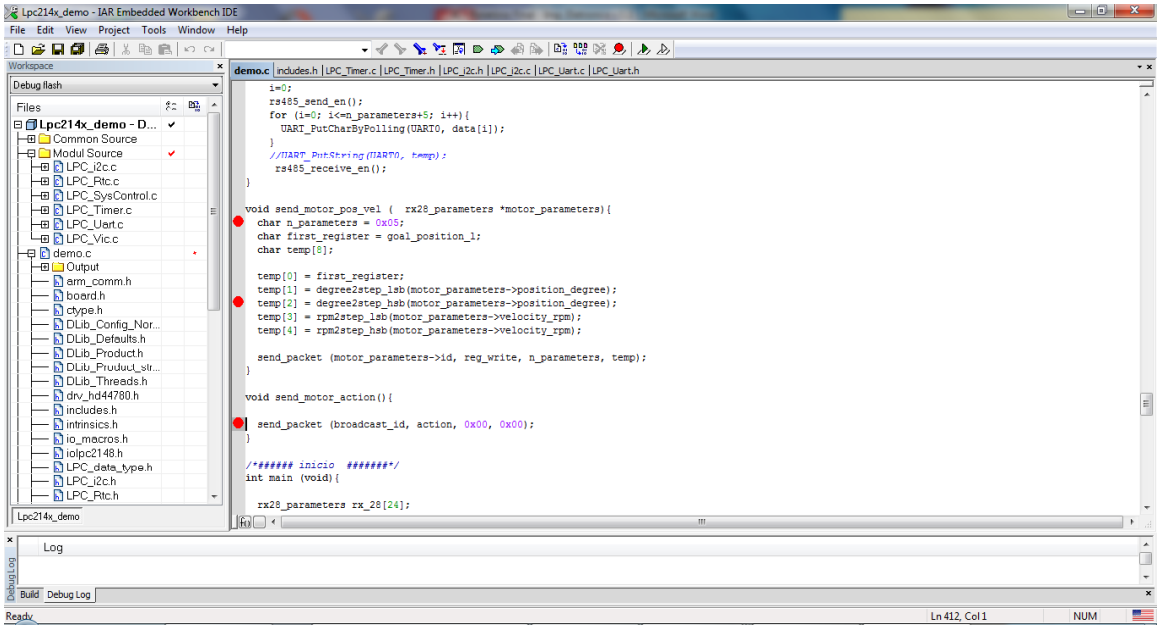


Figura 39– Interface IAR Workbench

8.2. Código para Leitura do acelerômetro

Para a demonstração das funcionalidades da placa foi programada as funções relativas a comunicação I²C, como o start, o envio de carcteres, etc, tal como explicitado no item **7.3.2.1,** e a função que faz a conversão dos dados lidos em complemento de dois para um valor inteiro.

Abaixo a parte relativa a leitura dos registradores de aceleração do sensor BMA 180.

```
//****inicio teste de leitura acelerometro****

    for (i=0; i<=5; i++){
        ledg1_on;
        reset_i2c0();
        start_i2c0();
        write_i2c0(acc_bus_id , acc_init_reg+i ); //envia comando do bus
mais o bit de escrita e o reg inicial
        repeat_start_i2c0();
        acc_data_return[i] = read_i2c0(acc_bus_id);
        master_ack_i2c0();
        stop_i2c0();
     }

     acc_x[j]= convert_acc2int(acc_data_return[1],acc_data_return[0]);
     acc_y[j]= convert_acc2int(acc_data_return[3],acc_data_return[2]);
     acc_z[j]= convert_acc2int(acc_data_return[5],acc_data_return[4]);

     j=j+1;
     if (j==100){
        ledr1_on;
        j=0;
     }

     if (k==2){
     ledg1_off;
     ledr1_off;
     k=0;
     }
     k=k+1;

//****fim do teste de leitura acelerometro*****


 //****funcao para conversao dos dados de complemento de dois para

inteiro****

int convert_acc2int ( char msb_data, char lsb_data){
    unsigned short int msb_data_int, lsb_data_int;
    short int acc_short_int;
    int acc_int;

    msb_data_int = msb_data<<6;
    lsb_data_int = lsb_data>>2;
    acc_short_int = ~(msb_data_int + lsb_data_int)+1;
    if (acc_short_int < -4096){
        acc_short_int = acc_short_int +16384;
```

```
    }
    acc_int = acc_short_int * acc_scale_constant;
    return acc_int;
}
```

Na primeira parte do código temos um loop com contador i que guarda no vetor **`acc_data_return[]`** os valores da aceleração lidos nos registradores. Neste loop estão inseridas as funções de acordo com o protocolo I²C, que não estão representadas neste enxerto de código, como a função de *start*, escrita, leitura, *master ack*, *repeat start*, etc, necessárias ao funcionamento. O valor de **`acc_init_reg`** é o valor onde começaremos a ler os dados, que pelo mapa de memória do sensor BMA180, página 21 do *datasheet*, inicia-se no registrador 02h. A cada ciclo temos um acréscimo da variável **`i`** o que promove a leitura dos dados dos 3 eixos. Ao final da leitura dos 6 valores – bytes mais significativos e menos significativos – o loop é encerrado.

Com isso utiliza-se a função que converte os dados na forma complemento de dois para inteiros. Com isso guardamos os dados de aceleração nos vetores **`acc_x`**, **`acc_y`** e **`acc_z`** que poderão ser utilizados em uma malha de controle para a estabilização.

Para demonstrar a aplicabilidade do uso do sensor de aceleração, abaixo temos o link com a demonstração da leitura acelerômetro, ao acender os leds da placa de controle:

http://www.youtube.com/watch?v=TmaDMmlkcWc&feature=youtu.be

8.3. Código para o comando dos servo-motores

Da mesma forma que a leitura da aceleração, também foi realizado o teste para comando dos servo-motores através da placa de controle. Para isso foi inserido o protocolo de comando, para controlar a velocidade e a posição para qual o servo irá se movimentar.

Abaixo é dado alguns trechos mais importantes do código para sua explicação:

-A *struct* dos servo motores:

```
typedef struct  {
   unsigned int position_degree;
   unsigned int velocity_rpm;
   char id;
} rx28_parameters;
```

Para uma melhor organização das variáveis presentes em cada um dos 21 servo-motores foi utilizado o conceito de struct, pois cada um dos servo motores deve possuir pelo menos 3 variáveis associadas, sendo uma o id, a posição em graus, e a velocidade com que este motor deverá ir para esta posição quando mandarmos o comando. Poderão ser adicionadas a essa struct variáveis relativas ao status atual do motor, como alarmes de torque ou temperatura.

-O vetor de *structs*:

```
rx28_parameters rx_28[24];
```

```
rx_28[0].id = 0;
rx_28[1].id = 1;
rx_28[2].id = 2;
rx_28[3].id = 10;
rx_28[4].id = 11;
rx_28[5].id = 12;
rx_28[6].id = 20;
rx_28[7].id = 21;
rx_28[8].id = 22;
rx_28[9].id = 30;
rx_28[10].id = 31;
rx_28[11].id = 32;
rx_28[12].id = 33;
rx_28[13].id = 34;
rx_28[14].id = 35;
rx_28[15].id = 40;
rx_28[16].id = 41;
rx_28[17].id = 42;
rx_28[18].id = 43;
rx_28[19].id = 44;
rx_28[20].id = 45;
```

Foi criado um vetor de 24 posições **`rx_28`** do tipo **`rx28_parameters`**, ou seja, um vetor que possui cada uma das variáveis necessárias ao comando dos motores e declarado abaixo temos o id relacionando cada um dos motores a uma posição do vetor **`rx_28`**.

-O loop para envio dos comandos:

```
i=0;
  for(i=0; i<Num_Servos; i++){
    rx_28[i].position_degree = init_position[i];
    rx_28[i].velocity_rpm = 5;
    send_motor_pos_vel(&rx_28[i]);
    Delay_us(5);
  }
  Delay_us(5);
  send_motor_action();
```

Neste loop temos de fato o envio dos comandos guardados na struct para os servo motores. A constante Num_Servos são o nº de servo motores presentes no robô. Foi criada um outro vetor com as posições iniciais de cada um dos servo motores, assim temos o envio da posição, velocidade e do id por meio da função **`send_motor_pos_vel()`**, onde passamos o ponteiro da struct relacionada. Para que todos os servos se movimentem ao mesmo tempo, foi utilizado o comando de SYNC WRITE, assim enviamos todos os dados para os motores e após o loop enviamos um comando para executarem os dados de posição e velocidade guardados na memória interna do servo motor. Essa função é a **`send_motor_action()`**.

-Funções para formatação dos dados

```
void send_motor_pos_vel (rx28_parameters *motor_parameters){
  char n_parameters = 0x05;
  char first_register = goal_position_l;
  char temp[8];

  temp[0] = first_register;
  temp[1] = degree2step_lsb(motor_parameters->position_degree);
  temp[2] = degree2step_hsb(motor_parameters->position_degree);
  temp[3] = rpm2step_lsb(motor_parameters->velocity_rpm);
  temp[4] = rpm2step_hsb(motor_parameters->velocity_rpm);

  send_packet (motor_parameters->id, reg_write, n_parameters, temp);
}

void send_motor_action(){

  send_packet (broadcast_id, action, 0x00, 0x00);
}
```

Tais funções são as utilizadas no código para formatar os dados a serem enviados pela função send_packet, uma função mais genérica e que de fato irá enviar os dados. Nestas funções temos mais duas funções que convertem os valores em Graus e RPM para o formato utilizado no motor. O vetor **`temp`** possui os parâmetros a serem enviados.

A função **`send_motor_action()`** envia o comando de action para todos os motores, que é o valor 254 no protocolo, contido na constante **`broadcast_id`**.

-Função para o envio de dados

```
void send_packet (char id, char instruction, char n_parameters, char
parameters[8]){
    char data[16] = {0x00};
    char i=0;
    char checksum;

    data[0] = 0xFF;
    data[1] = 0xFF;
    data[2] = id;
    data[3] = n_parameters +2; //n de parametros mais 2
    data[4] = instruction;

    if (n_parameters > 0x00){
      for (i=0; i<=n_parameters; i++){
        data[5+i] = parameters[i];
      }
    }
    i=0;
    checksum = 0;
    for (i=2; i<= 4 + n_parameters ; i++){
      checksum = checksum + data[i];
    }
    data[n_parameters+5] = ~checksum;

    i=0;
    rs485_send_en();
    for (i=0; i<=n_parameters+5; i++){
      UART_PutCharByPolling(UART0, data[i]);
```

```
    }
     rs485_receive_en();
}
```

Esta função é a que realmente envia os dados, recebendo os dados de id, instrução, o número de parâmetros e os parâmetros a serem escritos.

É criado o vetor de dados **data**, calculado o checksum, ativado o ci de transceiver de comunicação e enviado o vetor de dados pela uart do microcontrolador. Ao final, fechamos o link de comunicação.

Para demonstrar que todas as funções de envio são funcionais, foi realizado um teste de demonstração onde temos a leitura do sensor de aceleração e colocado esse valor dentro da struct de um motor, assim teremos a movimentação proporcional a posição do motor.

```
rx_28[3].position_degree = ((acc_x+100000)/10000)*15;
rx_28[3].velocity_rpm = 30;
send_motor_pos_vel(&rx_28[3]);
Delay_us(5);

send_motor_action();
```

Neste caso, como o valor de aceleração possui um valor alto devido a sua grande precisão, foi convertido o valor para a escala de 0 a 300º, utilizado pelo servo motor.

Abaixo temos um vídeo de demonstração:

http://www.youtube.com/watch?v=ukJl5p-HiR8&feature=youtu.be

8.4. A Placa de Controle executando um Script

Um ultimo teste é a inserção de dados adquiridos na interface *Stop & Go*, inseri-los na placa controladora para sua execução.

Os dados adquiridos foram para a prosição inicial do robô, e foram guardados em um vetor de **`init_position`**, com os seguintes valores de posição:

```
int init_position[] = {  0,172,235,                  //tronco
                         137,226,150,                //braco esquerdo
                         161,65,150,                 //braco direito
                         175,166,154,117,142,102,    //perna esquerda
                         171,137,142,181,142,207};   //perna direita
```

A seguir foram executadas essas posições da mesma maneira que o teste para a movimentação do motor pelo acelerômetro. Com isso demonstra-se a funcionalidade da interface, ao permitir criar movimentações e scripts e inseri-los na placa de controle.

Abaixo é mostrado um vídeo do robô levantando sem a utilização do computador, demonstrando o funcionamento da placa de controle para a movimentação de todos os servo-motores do robô.

http://www.youtube.com/watch?v=_x4hveZnUEo&feature=youtu.be

9. CONCLUSÃO

Com o fim da apresentção de todos os testes, circuitos desenvolvidos e programas, conclui-se a Iniciação Científica *Implementação da Eletrônica de um Robô Humanóide* com a demonstração de todas as funcionalidades agragadas ao robô com a inserção de sua eletrônica.

O projeto, necessita ainda a fabricação da Placa de Controle em sua versão final e todos os arquivos de desenho das placas de circuito impresso estarão anexadas junto desta Iniciação Científica. Junto dos arquivos de desenho serão também disponibilizados os arquivos de programação da placa de controle, a interface *Stop & Go* com seus respectivos códigos fonte e os vídeos com os testes e demonstrações dos experiementos realizados.

Com a fabricação da placa de controle o robô humanoide estará apto ao desenvolvimento do controle no baixo nível, com o estudo das malhas para sua estabilização na posição ereta – naturalmente instável – utilizando os sensores acelerômetro e giroscópio como realimentação desta malha, e a programação no microcontrolador.

Concomitantemente ao desenvolvimento do controle do robô, poderá ser iniciado o desenvolvimento do software de estratégia, utilizando simuladores e o computador já disponível para o humanoide, o fit PC2i, pois agora ele é desvinculado da eletrônica para

o funcionamento. Também poderá ser iniciado o desenvolvimento da visão do robô, com a definição das câmeras a serem utilizadas e os softwares para detecção da bola.

Mesmo concluindo-se esta Iniciação Científica, ainda há muito trabalho a ser realizado na parte de programação do baixo nível, cujo o foco não foi deste trabalho. Há a necessidade do teste do giroscópio para a sua inserção na malha de controle, testes intensivos no envio de comandos aos servo-motores pois é uma aplicação crítica, uma melhor organização da programação do microcontrolador ao separar os grandes blocos de funções, a criação de muitas outras funções necessárias ao funcionamento preciso do robô, etc. Entretanto, espera-se com este trabalho que o passo inicial tenha sido dado e as bases ao desenvolvimento do robô estejam mais definidas, ao disponibilizar um sistema totalmente funcional aos trabalhos futuros.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] **ASIMOV, Isaac. –** *Eu, Robô* ,EDIOURO-SINERGIA, 2ª Edição.

[2] "ASIMO | The Honda Humanoid Robot". Disponível em <http://world.honda.com/ASIMO/>. Acesso em 05/07/2011.

[3] "ALDEBARAN ROBOTICS | HUMANOIDS ROBOTS". "Nao's World". Disponível em <http://www.aldebaran-robotics.com/eng/Nao.php>. Acesso em 05/07/2011.

[4] **MACWORTH, Alan. -** *On Seeing Robots.* 1993. Artigo Científico - University of British Columbia, Vancouver, B.C. ,Canada.

[5] "*A brief history of RoboCup"*. Disponível em < http://www.robocup.org/about-robocup/a-brief-history-of-robocup/>. Acesso em 05/07/2011.

[6] "Darmstadt Dribblers ". Disponível em < http://www.dribblers.de/>. Acesso em 10/08/2011.

[7] "FUmanoids". Disponível em < http://www.fumanoids.de/>. Acesso em 10/08/2011.

[8] "CIT Brains Kid". Disponível em <http://sites.google.com/site/yasuohayashibara2/robocup>. Acesso em 10/08/2011.

[9] "Team DARwIn". Disponível em <http://www.me.vt.edu/Robocup/Site/Home.html>. Acesso em 10/08/2011.

[10] ROBOTIS- "*Dynamixel "*. Disponível em <http://www.robotis.com/xe/dynamixel_en> . Acesso em 15/08/2011.

[11] **GARCIA , Eduardo. -** *Eletrônica do Humanóide.* 2011. Projeto de Iniciação Científica – Centro Universitário da FEI, São Bernardo do Campo, São Paulo.

[12] *"RoboCup Soccer".* Disponível em <http://wiki.robocup.org/wiki/Main_Page#RoboCup_Soccer>. Acesso em 05/07/2011.

[13] "*Very Small Size".* Disponível em < http://portal.fei.edu.br/pt-BR/pesquisas_projetos/projetos_institucionais/Robo_FEI/informacoes_tecnicas/Paginas/robos.aspx> Seção Very Small Size. Acesso em 05/07/2011.

[14] *"Small Size".* Disponível em <http://wiki.robocup.org/wiki/Small_Size_League> . Acesso em 05/07/2011.

[15] *"MATLAB Solutions".* Disponível em <http://www.mathworks.com/solutions/?s_cid=global_nav>. Acesso em 21/11/2011.

[16] *"Middle Size League".* Disponível em <http://wiki.robocup.org/wiki/Middle_Size_League>. Acesso em 05/07/2011.

[17] "*Humanoid League".*Disponível em <http://wiki.robocup.org/wiki/Humanoid_League>. Acesso em 05/07/2011.

[18] "*Standard Platform League"*. Disponível em <http://wiki.robocup.org/wiki/Standard_Platform_League>. Acesso em 05/07/2011.

[19] "*Simulation League"*. Disponível em <http://wiki.robocup.org/wiki/Soccer_Simulation_League>. Acesso em 05/07/2011.

[20] *"Humanoid League Rules".* Disponível em <http://www.tzi.de/humanoid/bin/view/Website/Downloads>. Acesso em 05/07/2011.

[21] *"Robotis RX28 e-Manual"*. Disponível em <http://support.robotis.com/en/>, Caminho <Home > Product Information > Dynamixel > RX Series > RX-28>. Acesso em 15/08/2011.

[22] *"RoBoard RB-100".* Disponível em < http://www.roboard.com/RB-100.htm>. Acesso em 15/08/2011.

[23] **CORTEZ, Milton.** *–Projeto Mecânico de um Robô Humanóide Futebol de Robôs – Humanóide League.* 2011. Projeto de Iniciação Científica - Centro Universitário da FEI, São Bernardo do Campo, São Paulo.

[24] "Docklight – RS-232 Terminal, Monitor". Download em <http://www.docklight.de/>. Acesso em 15/08/2011.

[25] *"Dynamixel Configurator".* Download em <http://www.besttechnology.co.jp/modules/knowledge/?%E3%82%BD%E3%83%95%E3%83%88%E3%82%A6%E3%82%A7%E3%82%A2>. Acesso em 21/08/2011.

[26] *"RS-485/ EIA-485".* Disponível em < http://en.wikipedia.org/wiki/RS-485>. Acesso em 22/10/2011.

[27] Maxin IC – *MAX485 - Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers.* Disponível em <http://www.maxim-ic.com/datasheet/index.mvp/id/1111> Acesso em 22/10/2011.

[28] Texas Instruments – *SN751756A - Differential Bus Transceiver.* Disponível em <http://www.ti.com/product/sn75176a>. Acesso em 22/10/2011.

[29] **AXELSON, Jan.** – *Designing RS-485 Circuits.* 1999. Artigo da Revista Circuit Cellar.

[30] National Semiconductor – *LM555 – Timer.* Disponível em <http://www.national.com/mpf/LM/LM555.html#Overview>. Acesso em 28/10/2011.

[31] TATO Equipamentos Eletrônicos – *TATO USB2 - Conversor USB-Serial nível TTL.* Disponível em <http://www.tato.ind.br/detalhe_produto.php?codigo_chave=72>. Acesso em 22/10/2011.

[32] FTDI Chip – *FT232R – USB UART IC.* Disponível em <http://www.ftdichip.com/Products/ICs/FT232R.htm>. Acesso em 28/10/2011.

[33] Maxin IC – *MAX3485* - 3.3V Powered, 10Mbps and Slew-Rate Limited, True RS-485/RS-422 Transceivers. Disponível em < http://datasheets.maxim-ic.com/en/ds/MAX3483-MAX3491.pdf> Acesso em 26/07/2012.

[34] CompuLab - *fit-PC2i specification.* Disponível em < http://www.fit-pc.com/web/fit-pc/fit-pc2i-specifications/> Acesso em 26/07/2012.

[35] eSysTech – *eLPC64 SOM* – Módulo ARM7 LPC2148. Disponível em < http://www.esystech.com.br/produtos/eLPC/Manual%20eLPC64%20-%20PR-ESYS-eLPC64-101.pdf> Acesso em 26/07/2012.

[36] NXP Semiconductor. Disponível em < http://www.nxp.com/> Acesso em 26/07/2012.

[37] NXP – *LPC214x – LPC2148 User Manual.* Disponível em < http://www.nxp.com/documents/user_manual/UM10139.pdf> Acesso em 26/07/2012.

[38] SPI Specification – *The SPI Interface.* Disponível em <http://www.muratamems.fi/sites/default/files/documents/tn15_spi_interface_specification.pdf> Acesso em 26/07/2012.

[39] I²C Specifications - *I2C-bus specification and user manual* - Disponível em <http://www.nxp.com/documents/user_manual/UM10204.pdf> Acesso em 26/07/2012.

[40] Sparkfun Elctronics – *BMA180 – Triple Axis Accelerometer Breakout.* Disponível em <https://www.sparkfun.com/products/9723> Acesso em 26/07/2012.

[41] Bosch-Sensortec –*BMA180 – Digital triaxial acceleration sensor.* Disponível em <http://www.bosch-sensortec.com/content/language1/downloads/BST-BMA180-FL000-03.pdf> Acesso em 26/07/2012.

[42] Sparkfun Elctronics - *L3G4200D - Tri-Axis Gyro Breakout.* Disponível em < https://www.sparkfun.com/products/10612> Acesso em 26/07/2012.

[43] ST Electronics - *L3G4200D Datasheet - ultra-stable three-axis digital output gyroscope.* Disponível em < http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00265057.pdf> Acesso em 26/07/2012.

[44] Digi International – *Xbee Series 2 Datasheet - XBee Series 2 OEM RF Modules.* Disponível em <ftp://ftp1.digi.com/support/documentation/90000866_A.pdf> Acesso em 26/07/2012.

[45] Sparkfun Elctronics – *FT232RL - Breakout Board USB to Serial* - Disponível em < https://www.sparkfun.com/products/718> Acesso em 26/07/2012.

[46] ARM Jtag - *ARM Standard JTAG Connector.* Disponível em <http://www.keil.com/coresight/connectors.asp> Acesso em 26/07/2012.

[47] "CadSoft Eagle 6.0". Download em < http://www.cadsoftusa.com/>. Acesso em 21/11/2011.

[48] National Semiconductor – *LM78xx – LM78XX/LM78XXA 3-Terminal 1A Positive Voltage Regulator.* Disponível em < http://www.fairchildsemi.com/ds/LM/LM7805.pdf> Acesso em 26/07/2012.

[49] International Rectifier – *IRF9310 – HEXFET Power MOSFET* . Disponível em <http://www.irf.com/product-info/datasheets/data/irf9310pbf.pdf>. Acesso em 26/07/2012.

[50] Texas Instruments – *LM2596 - SIMPLE SWITCHER Power Converter 150 kHz3A Step-Down Voltage Regulator*. Disponível em <http://www.ti.com/lit/ds/symlink/lm2596.pdf> Acesso em 26/07/2012.

[51] Texas Instruments –*LM1117- 800mA Low-Dropout Linear Regulator*. Disponível em <http://www.ti.com/lit/ds/symlink/lm1117-n.pdf>. Acesso em 26/07/2012.

[52] Segger - *J-Link EDU - J-Link for educational purpose*. Disponível em < http://www.segger.com/j-link-edu.html>. Acesso em 26/07/2012.

[53] IAR Systems - *IAR Embedded Workbench for ARM - INTEGRATED DEVELOPMENT ENVIRONMENT AND OPTIMIZING C/C++ COMPILER FOR ARM*. Disponível em <http://www.iar.com/en/Products/IAR-Embedded-Workbench/ARM/>. Acesso em 26/07/2012.

ANEXO A – Código *Matlab* 1

Abaixo temos o código utilizado para o envio de comandos aos motores, com a inicialização da porta serial e o loop simples de envio de dados para o motor indicado, estando comentadas as linhas de programação. Nota-se o uso de uma simplex sintaxe para a realização de funções complexas, como a inicialização da porta serial, a escrita de dados, conversão de caracteres, etc.

```
% Codigo Exemplo para envio de dados aos motores
%parametros para configuracao da porta serial
clc;
disp('BEGIN')
Ser_com = serial('COM8');
set(Ser_com,'BaudRate',57142); %% no motor: 57142
set(Ser_com,'DataBits',8);
set(Ser_com,'Parity','none');
set(Ser_com,'StopBits',1);
set(Ser_com,'FlowControl','none');
set(Ser_com, 'InputBufferSize', 4096);
set(Ser_com, 'TimeOut', 0.1)
fopen(Ser_com);

% dados a serem enviados
id_hex = '00' % vai de 0 a 0xFD broadcast 0xFE, identificacao de cada
motor
instruction_hex = '01' % 0x01 ping, 0x02 read data, 0x03 write data,
0x04 reg. write, 0x05 action, 0x06 reset
parameters_hex = char('', '') % colocar os dados a serem enviados

%conversao de hexadecimal para decimal
id_send= hex2dec(id_hex);
instruction_send= hex2dec(instruction_hex);
parameters_send= hex2dec(parameters_hex);
n_parameters= length(parameters_send); %n de parametros
length = n_parameters + 2;

% criando o pacote de dados em um vetor
data_send(1)= 255; % 0xFF caracteres de inicializacao
data_send(2)= 255;
data_send(3)= id_send;
data_send(4)= length;
```

```
data_send(5)= instruction_send;
i=1
for i=1:1:n_parameters
    data_send(5+i) = parameters_send(i);
end

%calculo do checksum
check_sum = 0;
for i=3:1:length+3
    check_sum= check_sum + data_send(i);
end
while check_sum > 255 %check_sum maior
    check_sum = check_sum - 256; % pega apenas os dois ultimos digitos
em hex
end
data_send(length+4) =  255 - check_sum; % inverte

%escrevendo os dados na porta serial
fwrite(Ser_com, data_send, 'char')
```

ANEXO B – Código Matlab 2

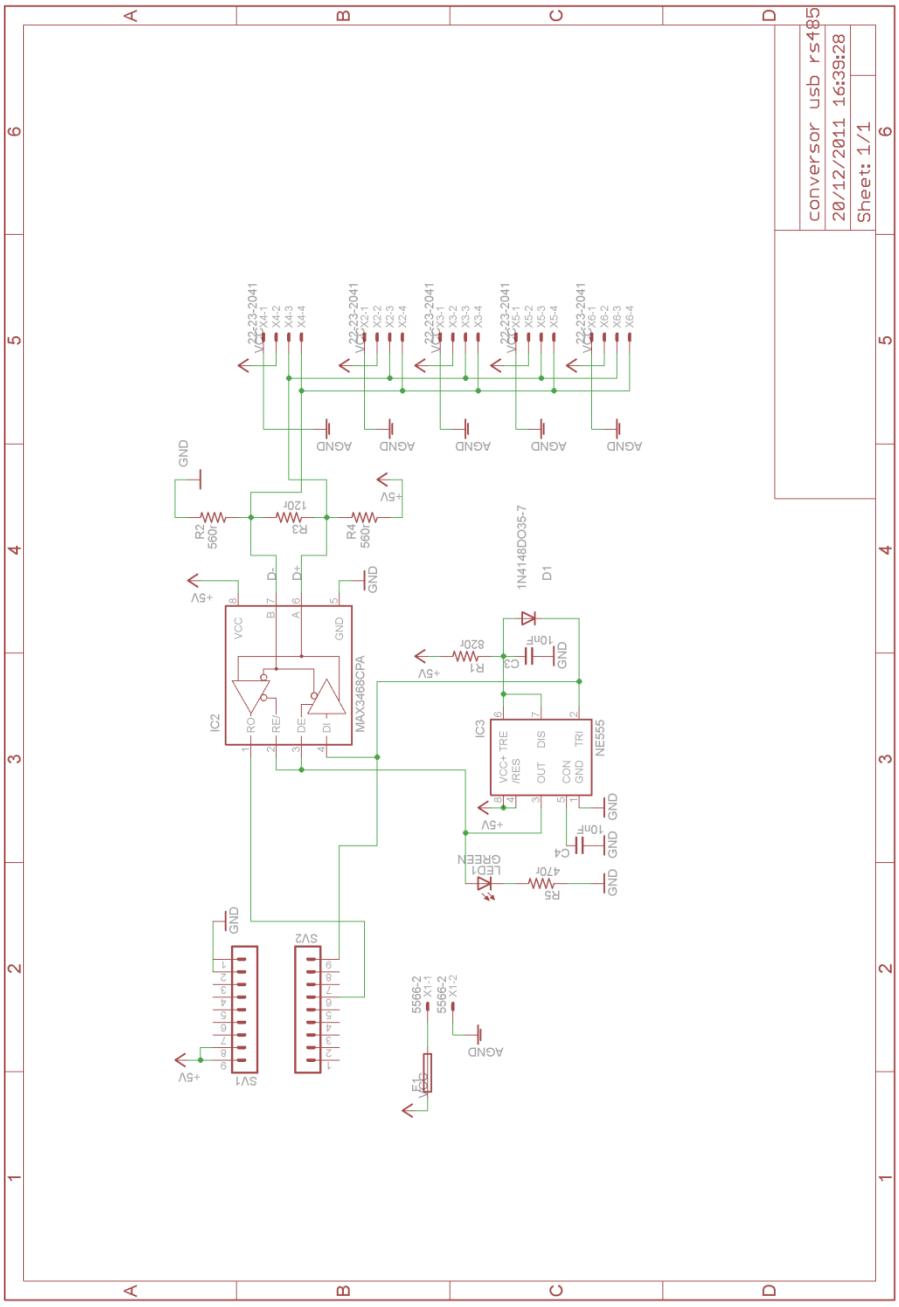
Abaixo é inserido o código de testes utilizado na aquisição de dados, utilizando também o *MATLAB*.

```
% cria o obj. porta serial
clc;
disp('BEGIN')
Ser_com = serial('COM8');
set(Ser_com,'BaudRate',57142); %% no motor: 57142
set(Ser_com,'DataBits',8);
set(Ser_com,'Parity','none');
set(Ser_com,'StopBits',1);
set(Ser_com,'FlowControl','none');
set(Ser_com, 'InputBufferSize', 4096);
set(Ser_com, 'TimeOut', 0.05)
fopen(Ser_com);

reg_read= [36, 02]; % parametros do registrador a serem lidos no loop
motor_id = [1,2,10,11,20,21,22,30,31,32,33,34,35,40,41,42,43,44,45] % id
dos motores a serem lidos
num_motors = length(motor_id);  %quantidade de motores para a leitura
param_lenght = length(reg_read); % tamanho dos parametros
instruction_sendr = 02; % instrucao para ler do registrador
instruction_sendw=  03; % instrucao para escrever no registrador
n=1; %% contador do loop

%funcionamento: todos os motores soltos, todos os motores travados
leitura dos dados

while(1)
    pause % pausa onde pressionado o enter para dar continuidade a
execucao dos comandos
    %%%%% ciclo de dados a serem enviados para travar o motor em
broadcast, o k serve como uma vari'l para travar e destravar o motor (0
e 1)%%%%
    for k=0:1:1
        id_send= 254; %broadcast 0xFE
        parameters_send= [24 , k]
        %%n_parameters= length(parameters_send); %"lengh" nao funciona
pois existe uma variavel k usa o "size"
        [z,n_parameters]= size(parameters_send); %n de parametros
        length_write = n_parameters + 2;
        data_send(1)= 255; %0xFF
```

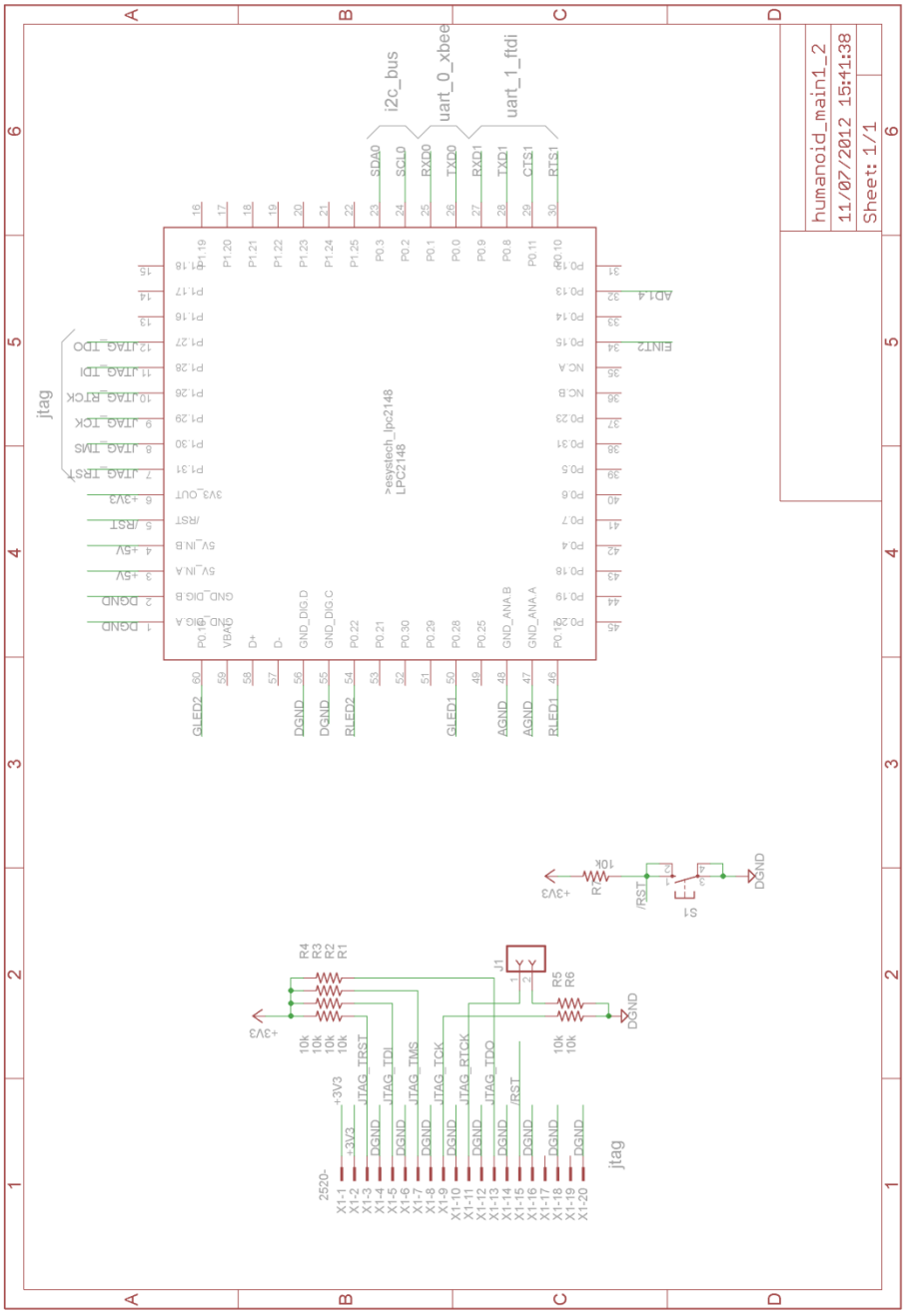
```
        data_send(2)= 255;
        data_send(3)= id_send;
        data_send(4)= length_write;
        data_send(5)= instruction_sendw;
            for i=1:1:n_parameters
                data_send(5+i) = parameters_send(i);
            end
            check_sum = 0;
            for i=3:1:length_write+3
                check_sum= check_sum + data_send(i);
            end
            while check_sum > 255 %check_sum maior
                check_sum = check_sum - 256; % pega apenas os dois
ultimos digitos em hex
            end
        data_send(length_write+4) =  255 - check_sum; % inverte
        %escreve os dados na porta serial
        fwrite(Ser_com, data_send, 'char')
        pause
    end

    %%%%ciclo para ler os dados dos motores%%%%
    for j=1:1:num_motors
        id_send_read = motor_id(j);
        parameters_read = reg_read; %parametros são todos os bytes de
reg_read

        %monta o pacote
         %n de parametros
        length_read = param_lenght + 2;
        data_send_read(1)= 255; % 0xFF
        data_send_read(2)= 255;
        data_send_read(3)= id_send_read;
        data_send_read(4)= length_read;
        data_send_read(5)= instruction_sendr;
        for i=1:1:param_lenght
            data_send_read(5+i) = parameters_read(i);
        end
        check_sum = 0;
        for i=3:1:length_read+3
            check_sum= check_sum + data_send_read(i);
        end
        while check_sum > 255 %check_sum maior
            check_sum = check_sum - 256; % pega apenas os dois ultimos
digitos em hex
        end
        data_send_read(length_read+4) =  255 - check_sum; % inverte
```

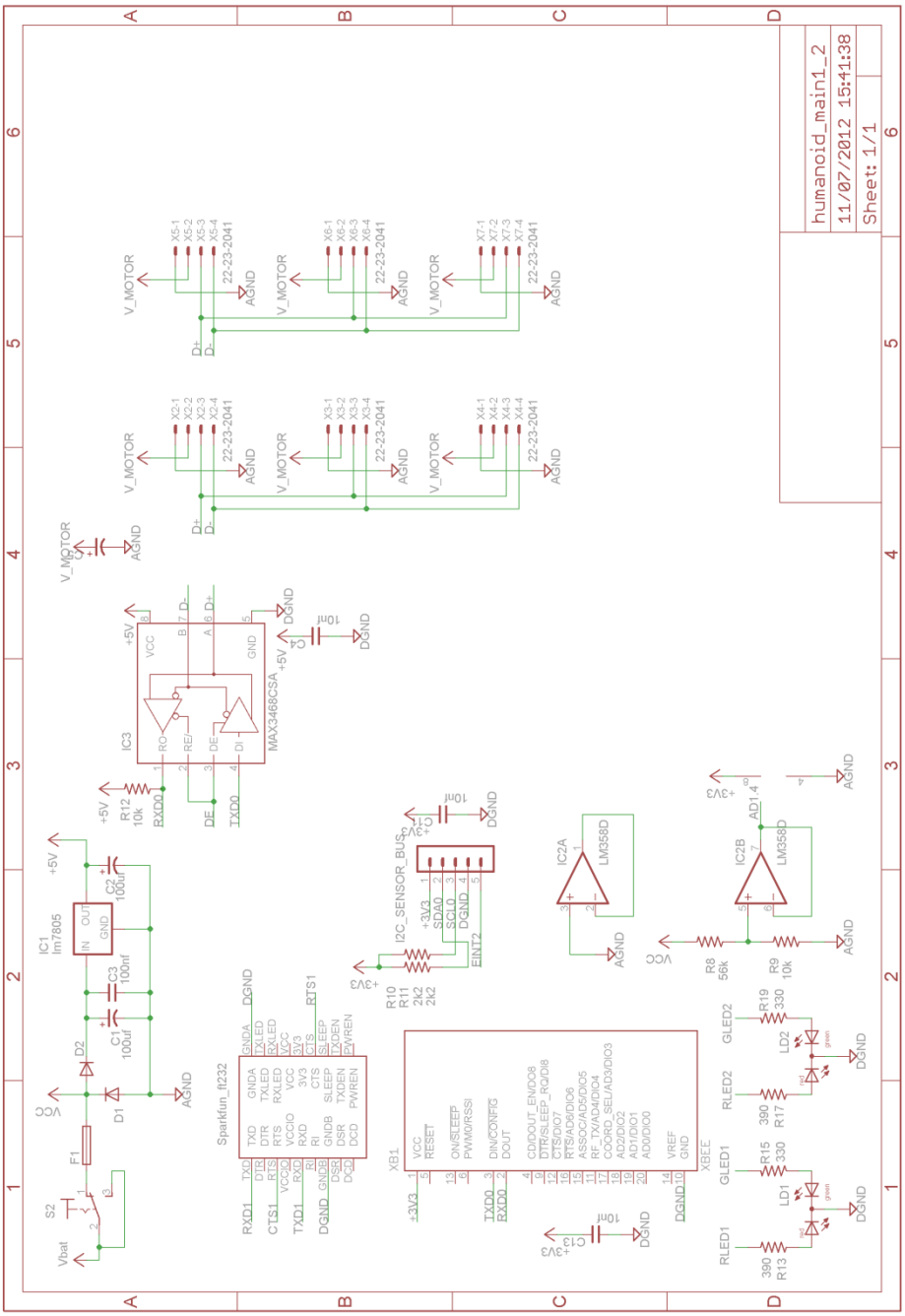
```
        fwrite(Ser_com, data_send_read, 'char') % envia o comando de
leitura
        %data_return(j,:) = fread(Ser_com); com o cabecalho
        data_return = fread(Ser_com);
        position_motor(n,j) = 256*data_return(7) + data_return(6); %high
byte + low byte
    end
    n=n+1
    position_motor
end
```

ANEXO C – Esquemático conversor USB-RS485

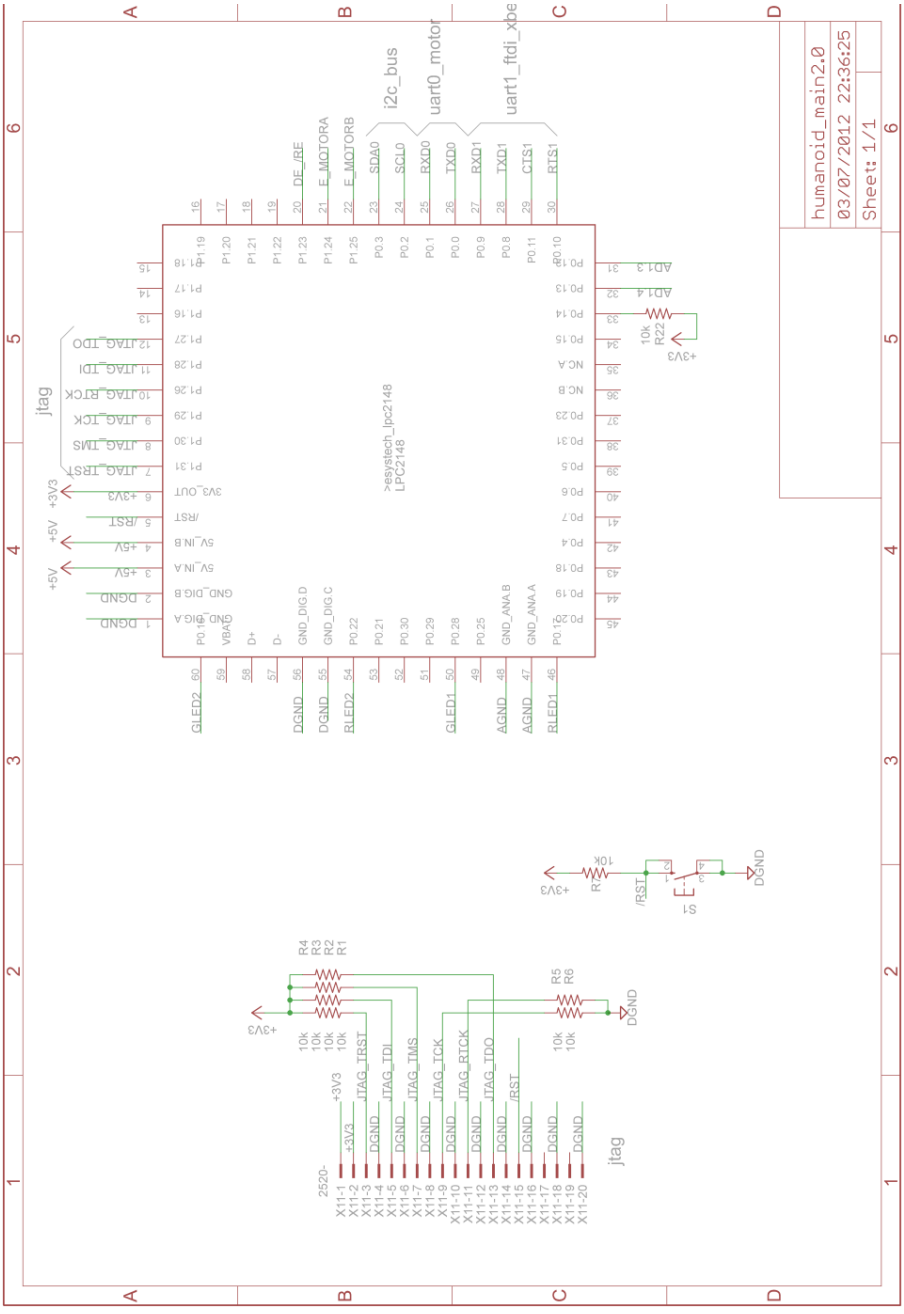


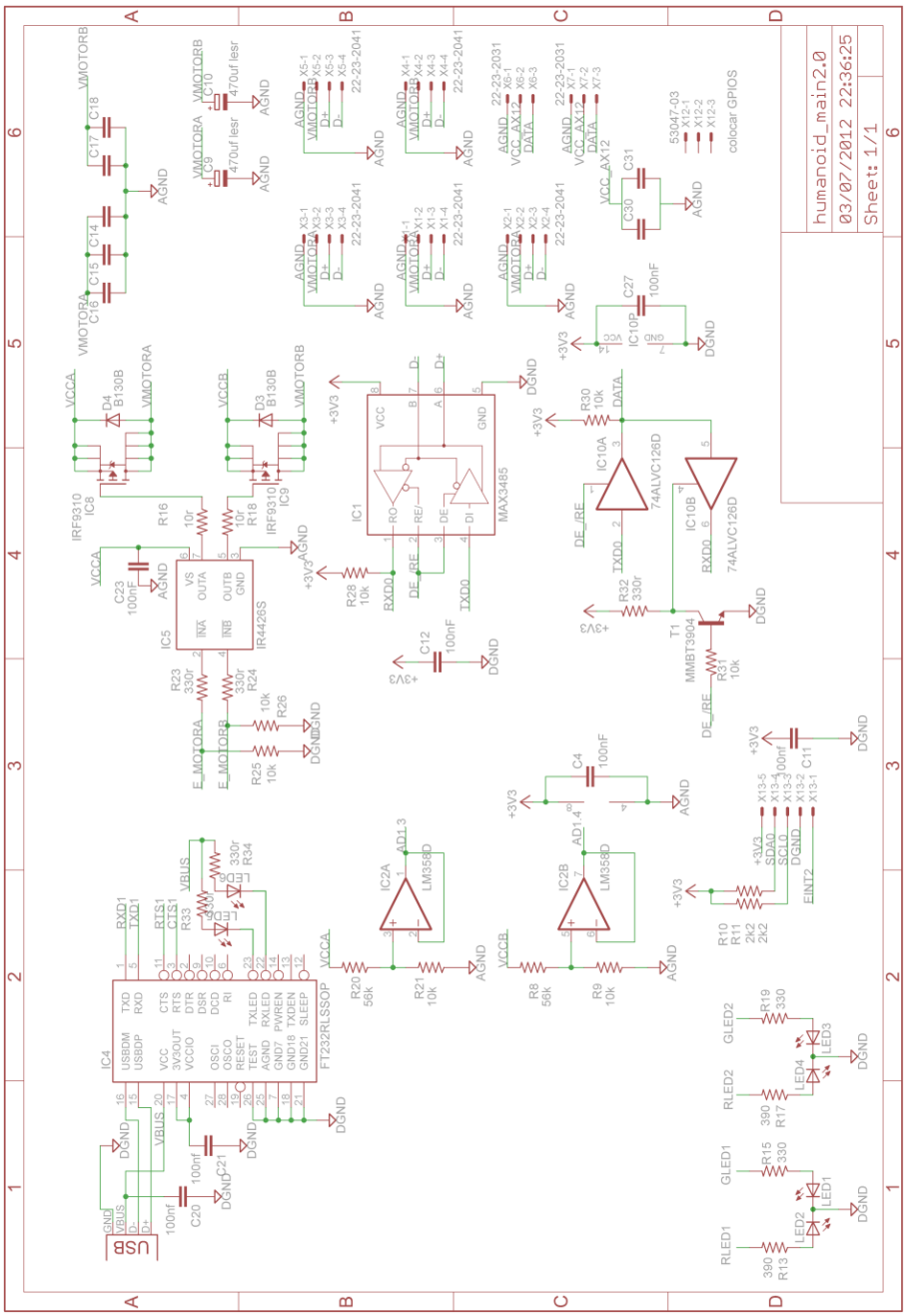
ANEXO D – Esquemático Placa Controladora

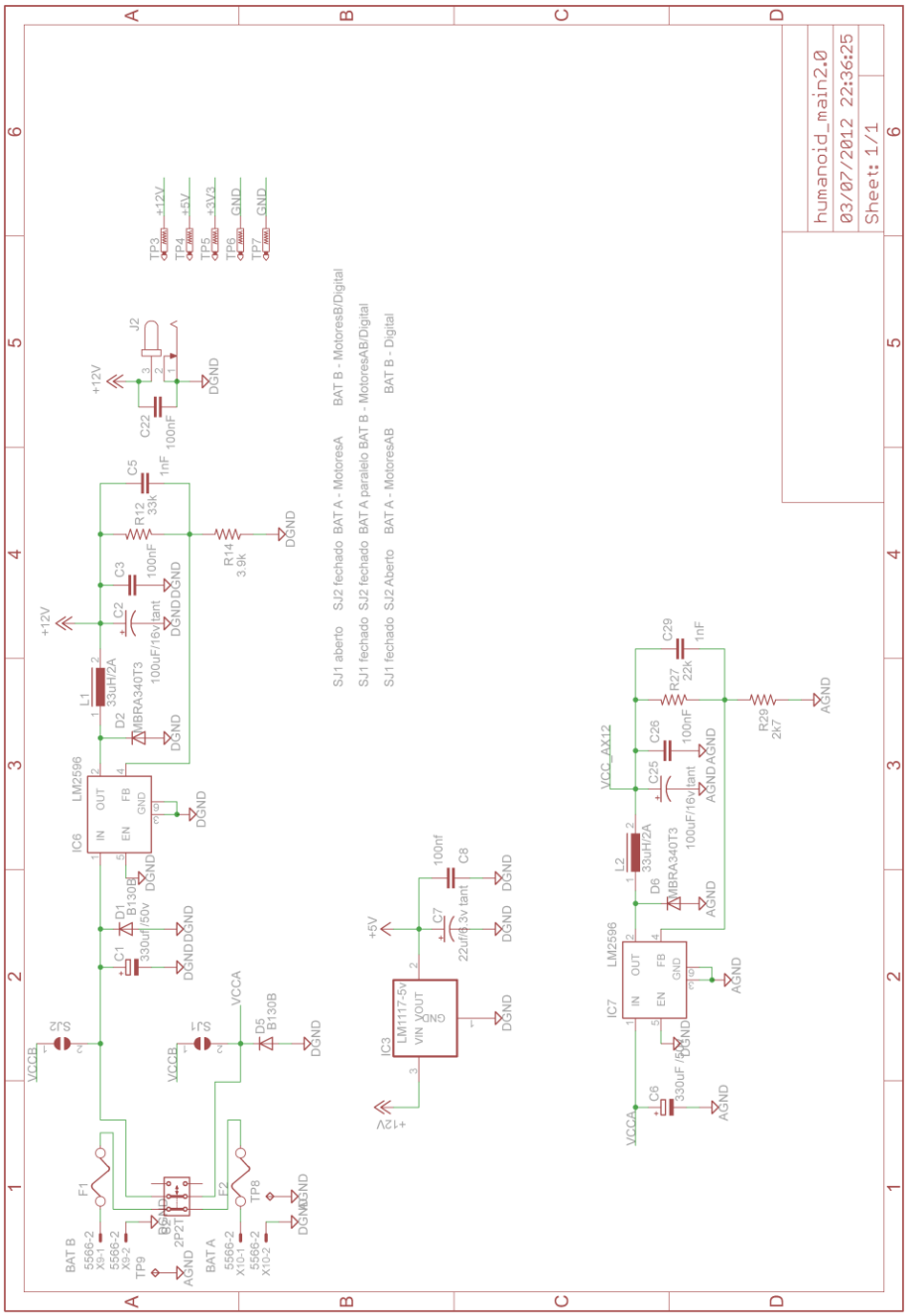



humanoid_main1_2
11/07/2012 15:41:38
Sheet: 1/1
V_MOTOR
AGND
22-23-2041
MAX3468CSA
LM358D
Sparkfun_ll232
I2C_SENSOR_BUS
XB1
DGND
+5V
+3V3

ANEXO E - Esquemático Placa Controladora versão final




humanoid_main2.0
03/07/2012 22:36:25
Sheet: 1/1
colocar GPIOS
MAX3485
74ALVC126D
LM358D
FT232RLSSOP
IR4428S
IRF9310
B130B
MMBT3904
53047-03
22-23-2041
22-23-2031
USB


humanoid_main2.0
03/07/2012 22:36:25
Sheet: 1/1
LM2596
LM1117-5v
MBRA340T3
B130B
SJ1 aberto SJ2 fechado BAT A - MotoresA BAT B - MotoresB/Digital
SJ1 fechado SJ2 fechado BAT A paralelo BAT B - MotoresAB/Digital
SJ1 fechado SJ2 Aberto BAT A - MotoresAB BAT B - Digital