UNIVERSIDADE DE SÃO PAULO ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA MECATRÔNICA E DE SISTEMAS MECÂNICOS

Robô Lançador de Bolas de Tênis de Mesa

Robson Kendy Watanabe
Rubens Augusto Amaro Junior

São Paulo
2009

UNIVERSIDADE DE SÃO PAULO ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA MECATRÔNICA E DE SISTEMAS MECÂNICOS

Robô Lançador de Bolas de Tênis de Mesa

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Graduação em Engenharia

Robson Kendy Watanabe
Rubens Augusto Amaro Junior

Orientador: Prof. Dr. Oswaldo Horikawa

Área de Concentração:
Engenharia Mecatrônica

São Paulo
2009

FICHA CATALOGRÁFICA

Watanabe, Robson Kendy
Robô lançador de bolas de tênis de mesa / R.K. Watanabe, R.A. Amaro Junior. -- São Paulo, 2009.
96 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1. Robôs (Controle) 2. Tênis de mesa I. Amaro Junior, R.A. II. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II. t.

AGRADECIMENTOS

Ao Prof. Dr. Oswaldo Horikawa, pela orientação, paciência, apoio e incentivo durante este trabalho.

A Vitor Finotto Cores, Rogério Issamu Yamamoto, Danilo Martins Vieira, Orlando Homen de Mello, Victor Sverzuti e Ivo da Paz Lopes pela paciência, apoio e incentivo durante todo o projeto realizado no laboratório.

A todos os colegas e funcionários do Departamento de Engenharia Mecatrônica e Sistemas Mecânicos da Escola Politécnica da Universidade de São Paulo, que contribuíram direta ou indiretamente para a realização deste trabalho.

A nossas famílias pela compreensão e apoio durante esta etapa de nossa vida.

RESUMO

No treino de Tênis de Mesa, diversas características do jogador podem ser trabalhadas de forma sistemática ou alternada dependendo do resultado que se pretende alcançar. Devido à facilidade de manuseio e ótima relação custo-benefício, muitas escolas de Tênis de Mesa utilizam robôs lançadores de bolas de Tênis de Mesa no treino de seus atletas. Estes robôs, além de apresentarem alta repetibilidade e precisão, podem ser programados de diversas maneiras modificando o treino desejado. Nosso objetivo neste trabalho é desenvolver um Robô Lançador de bolas de Tênis de Mesa utilizando materiais de baixo custo para sua estrutura e motores (CC e passo) para obtermos as movimentações necessárias do robô. Para programação de seus movimentos, deseja-se desenvolver uma interface homem - máquina (IHM) através de um microcomputador. Seu funcionamento será executado por um usuário frente a uma interface de um microcomputador. O usuário poderá escolher entre um modo manual em que ele poderá definir a cada instante a movimentação, velocidade de lançamento e lançamento da bola ou então pelo modo automático no qual o usuário estabelece uma rotina com os movimentos que ele deseja que o robô faça por um determinado tempo.

Palavras chave: Robôs (Controle), Tênis de Mesa.

ABSTRACT

In the training of table tennis, several features of the player can be worked systematically or alternating depending on the outcome to be achieved. Due to the ease of handling and excellent value for money, many table tennis schools use table tennis balls launcher robots in the training of their athletes. These robots, besides offer high repeatability and accuracy, can be programmed in various ways by modifying the training desired. Our objective in this work is to develop a table tennis balls launcher robot using low cost materials to their structure and motors (DC and step) to obtain the necessary movements of the robot. For programming their movements, it is intended to develop a human interface - machine (HIM) through a PC. Its operation will be run by a User Interface for PC. User can choose between a manual mode where he can set every time the movement, speed-up and throwing the ball or else the automatic mode in which the User provides a routine with the moves that he wants the robot to do for a given time.

Keywords: Robots (Control), Table Tennis.

ÍNDICE DE TABELAS

ÍNDICE DE FIGURAS

SUMÁRIO

ÍNDICE DE TABELAS

ÍNDICE DE FIGURAS

1 INTRODUÇÃO E MOTIVAÇÃO

Nosso objetivo neste trabalho é projetar e construir um robô (Figura 1) de baixo custo capaz de controlar a movimentação, velocidade e freqüência de um lançador de bolas de Tênis de Mesa utilizando motores de controle contínuo (CC) e motores de passo como atuadores e um microcontrolador juntamente com sensores para acionamento. Este robô auxilia o treinamento de jogadores de Tênis de Mesa podendo ser adotado em diferentes treinos específicos. À medida que mudamos sua posição, velocidade no lançamento ou a freqüência de lançamentos das bolas, muda-se o tipo de exercício treinado pelo jogador.

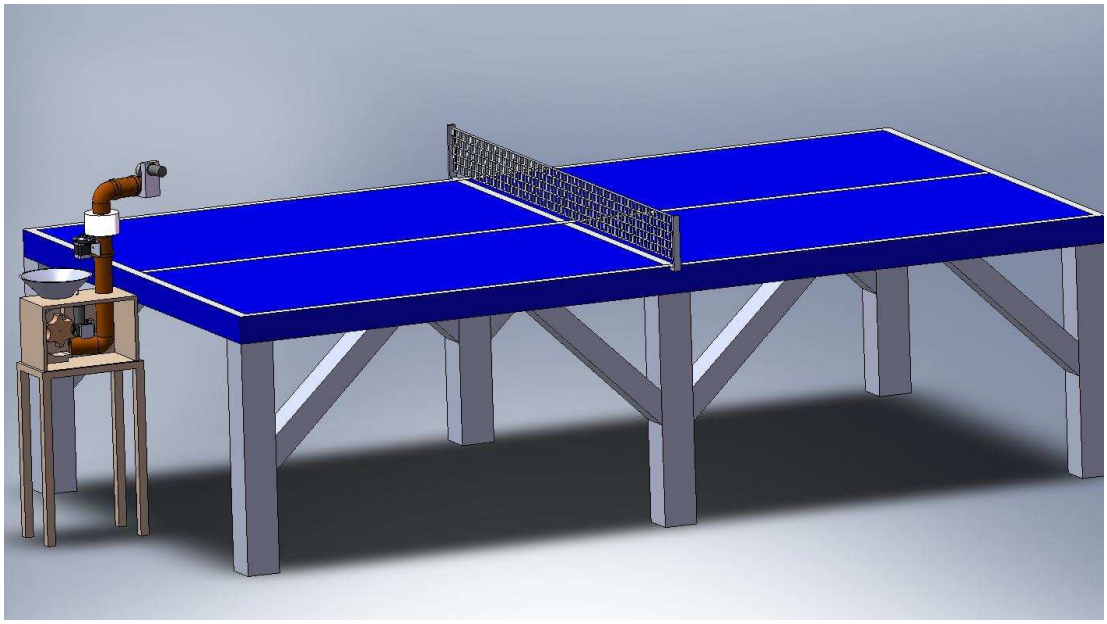


Figura 1 - Robô Lançador

Estes robôs já existem comercialmente e são muito empregados em escolas de Tênis de Mesa. Os principais robôs encontrados no mercado são: Amicus 3000 (Figura 2) da Butterfly [5], Robô Pong 2040 (Figura 3) da Newgy [3] e o SmartPong (Figura 4) fornecido pela SB Distribution [4]. Dependendo de suas características, o preço atual destes robôs varia entre $ 229,00 e $ 2.900,00.

Figura 2 – Butterfly Amicus 3000

Figura 3 – Robô Pong 2400

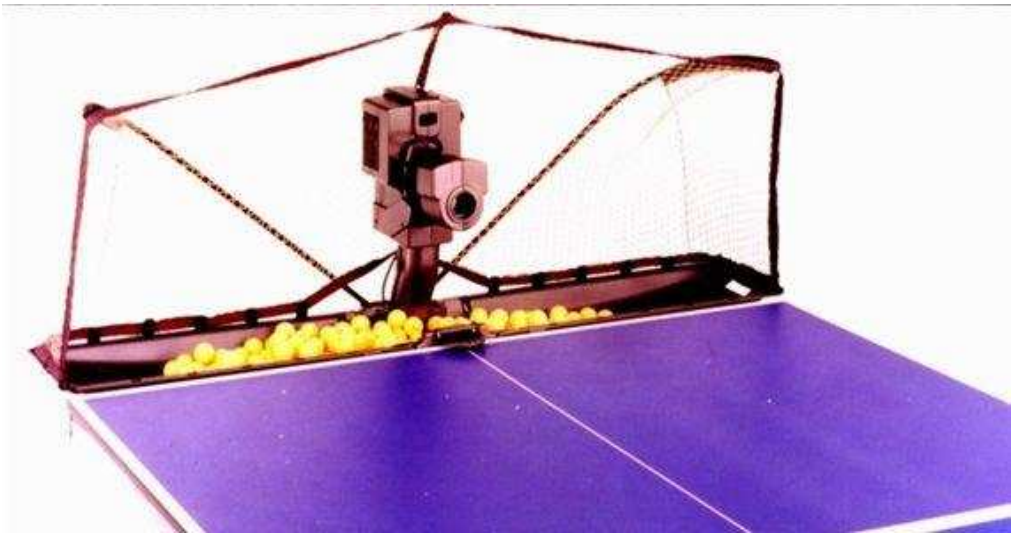


Figura 4 – Smart Pong

Devido ao preço e dificuldade de importação desse produto pelas escolas de Tênis de mesa ou próprios mesa-tenistas, nosso objetivo neste trabalho é o projeto e construção de um protótipo atendendo os requisitos básicos de um treino e que seja financeiramente possível de se adquirir. O estudo de viabilidade do nosso protótipo não fornecerá o custo de fabricação exato, devido à não inclusão de gastos com equipamentos para usinagem do robô, ou programação e gravação do microcontrolador utilizados na Escola Politécnica da Universidade de São Paulo, no entanto, permitirá uma boa estimativa do custo envolvido para uma futura produção em larga escala deste tipo de robô.

No treino de Tênis de Mesa, diversas características do jogador podem ser trabalhadas. Com a utilização de um robô lançador de bolas de Tênis de Mesa, podemos destacar as seguintes características:

- Posicionamento
- Equilíbrio
- Agilidade
- Concentração
- Rebatida

Alternando o lançamento das bolas entre as extremidades da mesa, o jogador pode trabalhar seu posicionamento e equilíbrio durante o treino. Essa troca, repetitiva, força o jogador a estabelecer uma passada característica e padronizada, melhorando seu desempenho na partida. Além de uma passada padronizada, o jogador melhora seu equilíbrio quando em movimento. O equilíbrio é fundamental para que o jogador possa rebater a bola com maior eficiência.

A freqüência de lançamento pode ser alterada, exercitando a agilidade do jogador. Aumentando esta freqüência e alternando o lançamento, o jogador é obrigado a se movimentar rapidamente durante o treino. Na maioria dos treinos, esta freqüência é gradativamente aumentada, à medida que o jogador adquire um nível melhor.

Movendo o lançador para diferentes pontos da mesa de forma aleatória pode-se treinar sua concentração. Este treino é o que mais se aproxima de um jogo real, simulando as ações do adversário durante a partida. O robô lançador pode ser programado para diferentes seqüências, variando o grau de dificuldade exigido.

Em conjunto com todas as características citadas anteriormente, o tipo de rebatida pode ser aprimorado e treinado com as funcionalidades do lançador. Lançando a bola em um único local, ou em locais alternados, o jogador pode treinar os efeitos que irá imprimir a bola durante a rebatida.

Uma rede atrás da mesa serve para recolher as bolas rebatidas pelo jogador e permitir a realimentação do robô, evitando dessa forma, que o jogador pare o treino para recolocar as bolas ou então, que mais outra pessoa seja necessária para que o robô opere sem interrupções.

Estas características do lançador apresentam estes e muitos outros benefícios utilizados nos treinos de Tênis de Mesa, cabendo aos técnicos selecionar as melhores opções para cada aluno.

Como base para nosso protótipo, já se encontra desenvolvido um primeiro protótipo deste robô desenvolvido pelo Prof. Dr. Oswaldo Horikawa (Escola Politécnica da Universidade de São Paulo – EPUSP). O protótipo dispõe de ajustes manuais para alterar a trajetória da bola e não dispõe de ajustes de velocidade de lançamento ou de freqüência com que as bolas são lançadas.

Para se obter a automatização do robô, um motor de passo irá rotacionar o lançador para direita ou para a esquerda permitindo desta forma, que a bola seja lançada em diversos pontos da mesa. Um motor de corrente contínua, com um disco em seu eixo, será utilizado para aplicar uma velocidade inicial à bola no momento do lançamento. Por fim, teremos um motor de passo para girar um disco dentado realimentando as bolas de Tênis de Mesa no lançador.

Pretende-se ainda que todo o sistema seja controlado por um microcontrolador do tipo PIC, muito acessível para compra e programação. A interface máquina-usuário será feita por um microcomputador. O PIC irá controlar os motores baseando-se em sinais emitidos por sensores e pelo microcomputador.

A estrutura mecânica do robô será feita, essencialmente, de materiais facilmente encontrados comercialmente e que apresentem baixo custo como PVC, nylon, madeira, alumínio, aço e acrílico.

2 METODOLOGIA

Inicialmente, foi realizada uma análise de um protótipo já desenvolvido pelo Prof. Dr. Oswaldo Horikawa (Escola Politécnica da Universidade de São Paulo - EPUSP). No protótipo, havia apenas uma estrutura mecânica sem qualquer tipo de controle sobre os atuadores, sendo todos os movimentos feitos manualmente.

Possíveis soluções para a automatização do protótipo foram avaliadas e o projeto do Robô Lançador foi dividido em duas áreas:

- Mecânica
- Eletrônica e Programação

A estrutura mecânica do protótipo foi avaliada e uma nova estrutura foi projetada. Foram definidos os movimentos de rotação do robô no plano horizontal (azimute), de realimentação das bolas de Tênis de Mesa e de lançamento das mesmas.

Partindo-se destas definições, foram projetados os sistemas de transmissão dos movimentos dos motores, biela-manivela para o movimento de rotação do lançador, engrenagem e correia dentada para o realimentador, e movimento direto do eixo do motor para um disco lançador.

O sistema biela-manivela foi desenvolvido permitindo o lançamento entre as extremidades da mesa. Com base nas dimensões de uma mesa de Tênis de Mesa (2.74 x 1.52 x 0.76), o deslocamento angular do lançador (azimute), considerando o deslocamento do lançador da borda da mesa, teve de ser de:

$$\hat{a}ngulo = arctg\left(\frac{0,7}{2,9}\right) \cong 13,6^{o} \quad \text{(Eq. 1)}$$

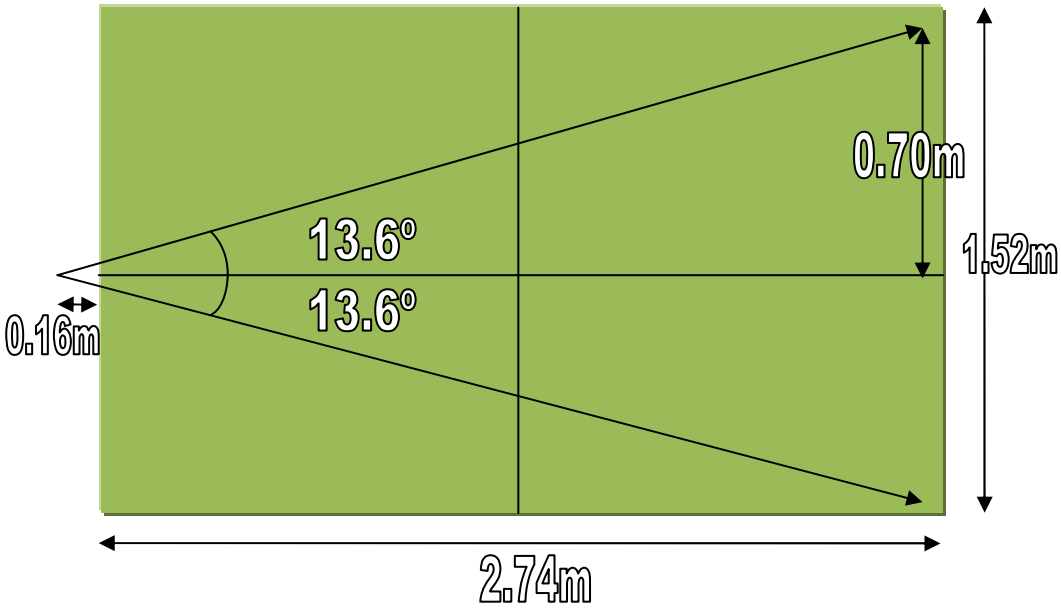


Figura 5 – Azimute máximo e mínimo

Com base nestes movimentos, a parte eletrônica pode ser projetada. Foram definidos os atuadores, motores de passo para os movimentos de rotação e realimentação, devido à necessidade de controle de posição, e motor de corrente contínua para o lançamento, devido à necessidade de controle de velocidade. Para o controle dos motores, a partir de sensores e comandos estabelecidos pelo usuário, um microcontrolador foi escolhido.

Baseado nos atuadores definidos para o Robô Lançador, foram projetados os circuitos de acionamento dos motores de passo e de corrente contínua, o circuito dos sensores, o circuito de isolação entre microcontrolador e o acionamento dos motores, prevenindo o primeiro em caso de sobre-corrente e o circuito de conversão RS232/TTL permitindo a comunicação entre microcontrolador e microcomputador.

Definidos os circuitos de acionamento e controle do robô, a programação pode ser estruturada. Inicialmente foi feita a programação para o acionamento dos motores. Uma interface para comunicação entre o usuário e o robô foi elaborada. Por fim, a programação permitindo a comunicação entre o microcomputador e o robô foi elaborada.

3 ATIVIDADES EXECUTADAS

3.1 Parte Mecânica

Todas as peças componentes do Robô Lançador de bolas de Tênis de Mesa foram desenhadas no software SolidWorks[1]. Levando-se em conta a distribuição das massas dos componentes e melhor posicionamento para movimentação das partes do robô, as seguintes atividades foram executadas:

- Projeto e construção da base para fixação das peças
- Projeto e construção do Sistema de Realimentação
- Projeto e construção do Sistema de Rotação do Lançador
- Projeto e construção do Lançador

3.1.1 Base

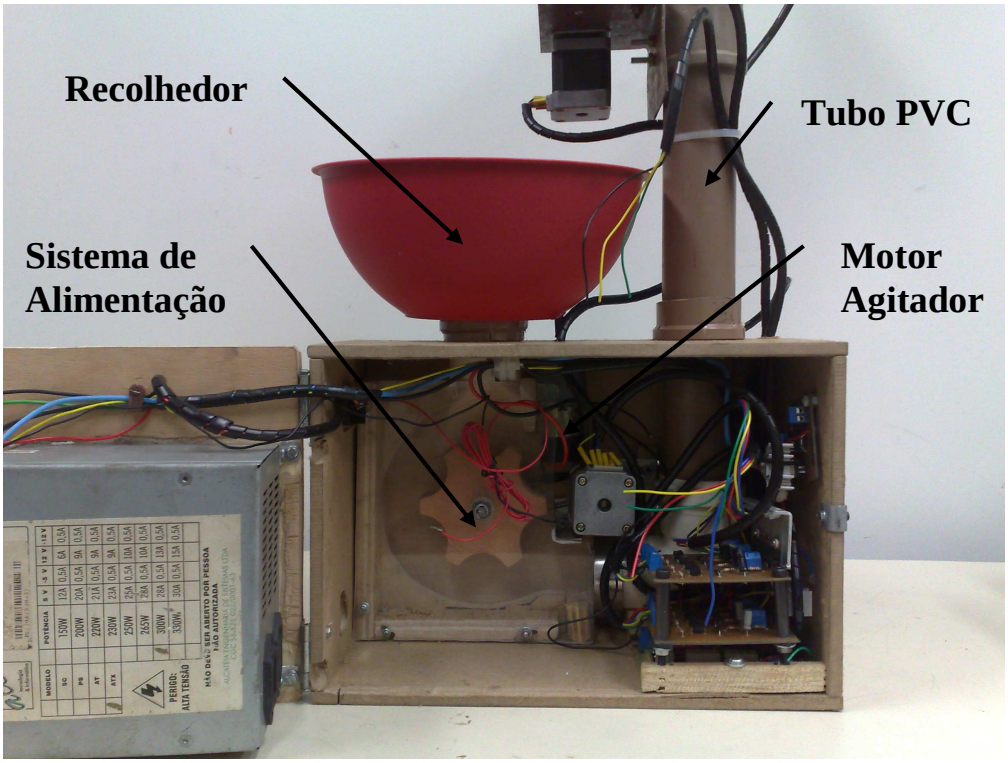


Figura 6 - Base do robô

A base do robô (Figura 6) serve de fixação para o recolhedor, o Sistema de Realimentação (roda dentada – correia – motor de passo), o tubo condutor (conduz as

[1] Software CAD para elaboração de formas tridimensionais a partir de formas geométricas elementares.

bolas até o lançador) e o motor agitador de bolas (motor de corrente contínua). A parte externa foi fabricada com MDF e os apoios internos com compensado de madeira. Canos de PVC foram utilizados como tubos condutores e uma cesta plástica como recolhedor.

3.1.2 Sistema de Realimentação

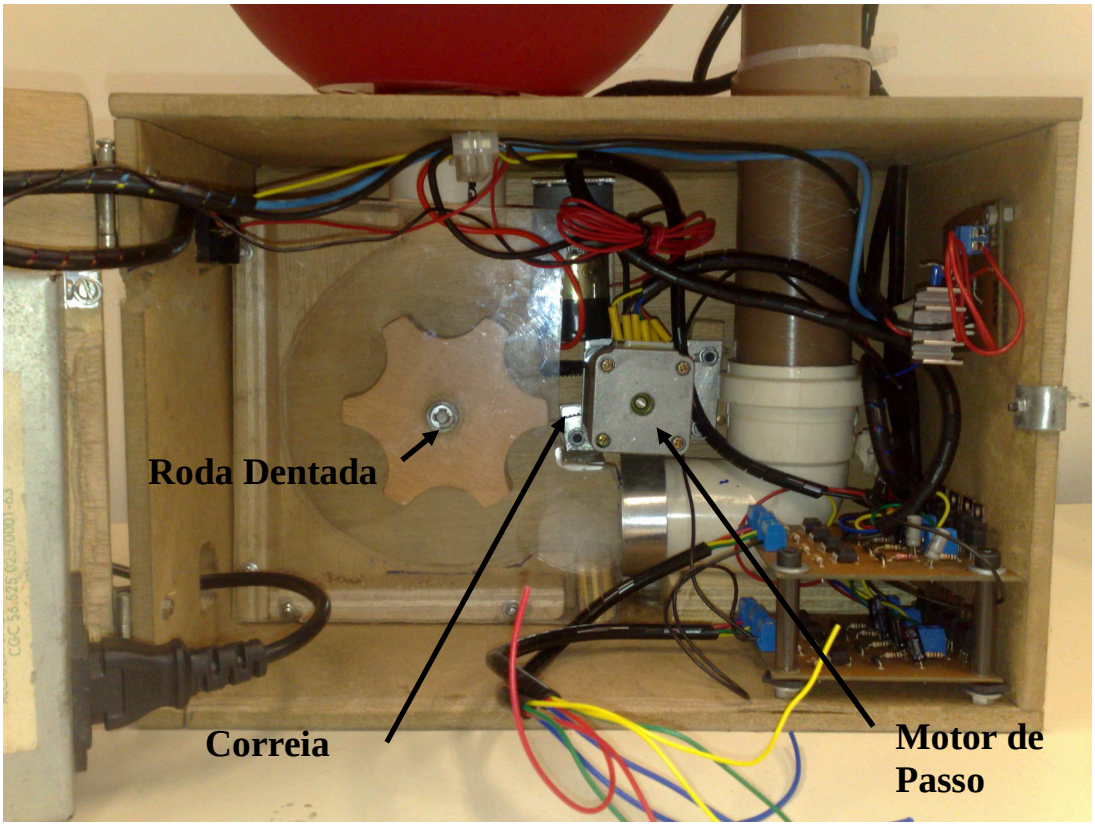


Figura 7 – Sistema de Realimentação

Visando o controle da freqüência de lançamento das bolas, o Sistema de Realimentação (Figura 7) consiste em um motor de passo, um apoio para o motor, uma roda dentada, uma correia e uma engrenagem. Para o apoio do motor foi fabricada uma peça de alumínio, permitindo a fixação do motor à base. A roda dentada foi fabricada com compensado de madeira. Uma correia de borracha e uma engrenagem plástica permitiram a transmissão do movimento do motor à roda dentada.

3.1.3 Sistema de Rotação do Lançador

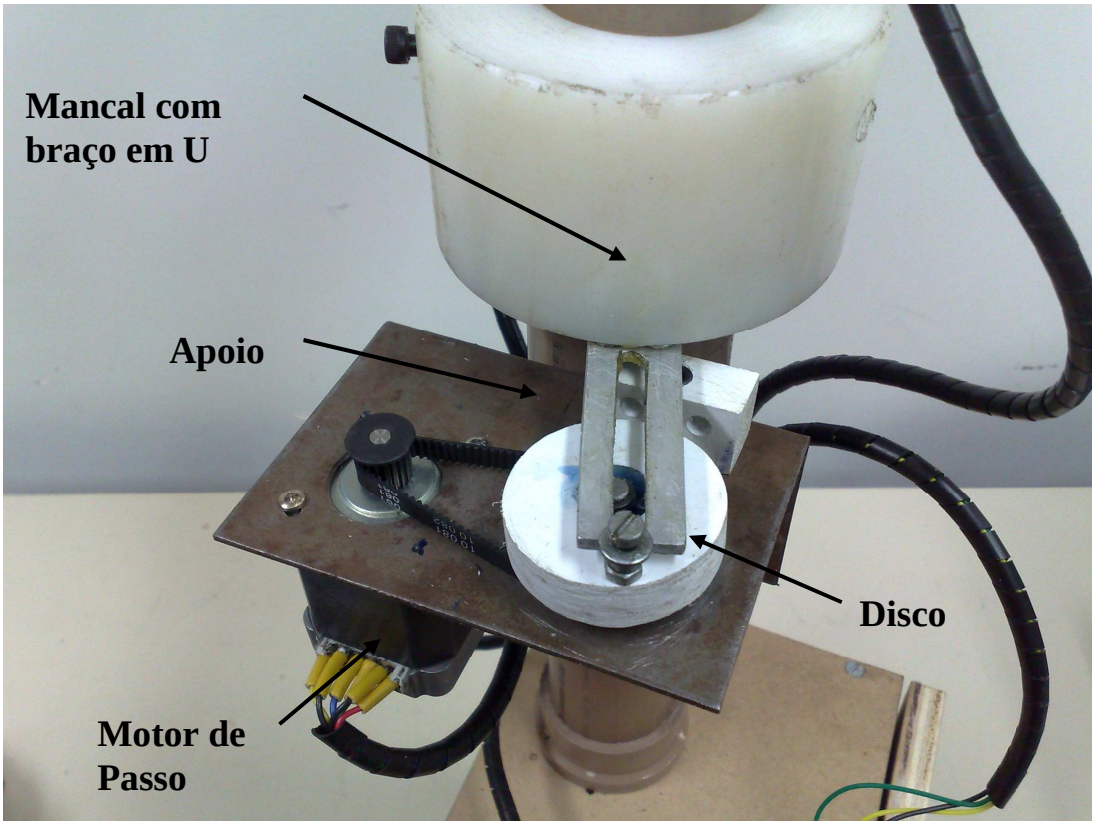


Figura 8 – Sistema de Rotação do Lançador

O Sistema de Rotação do Lançador (Figura 8) consiste em um mancal (união de um rolamento e o tubo de PVC), um braço em formato de U, um apoio para o motor, um motor de passo, uma correia, uma engrenagem, um rolamento e um disco. Para a fabricação do mancal foi usinado um tarugo de nylon. O braço em formato de U foi fabricado com uma chapa de alumínio. Um apoio de aço em formato de L foi fabricado para a fixação do motor ao tubo de PVC. Uma correia de borracha, uma engrenagem plástica e um disco de nylon foram utilizados para a transmissão do movimento do motor ao braço.

3.1.4 Lançador

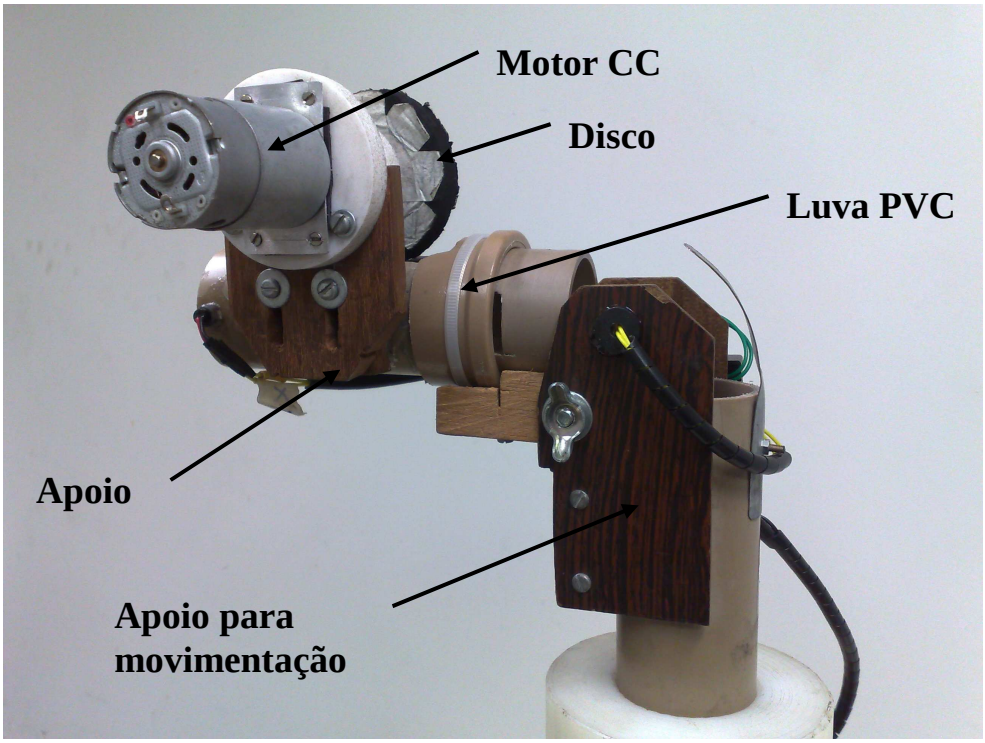


Figura 9 – Lançador

O Lançador (Figura 9) é composto por um motor de corrente contínua, um apoio para o motor, um disco emborrachado e uma luva de PVC. Um apoio de madeira e um mancal de nylon foram fabricados para fixação do motor ao tubo. Apoios de madeira permitem a movimentação vertical do lançador e uma fina chapa de aço serve de guia para as bolas seguirem até o lançador. Um disco de borracha foi fixado ao eixo do motor permitindo o lançamento das bolas.

O protótipo do Robô, unificando todas as partes apresentadas, pode ser visto na Figura 10.

Figura 10 - Protótipo

3.2 Parte Eletrônica e Programação

As seguintes atividades foram executadas:

- Projeto e construção do Sistema de Acionamento dos Motores
- Projeto e construção do Sistema de Sensores
- Projeto e construção da Conversão RS232/TTL
- Programação do Microcontrolador PIC 16F877A [6]
- Programação da Interface Homem-Máquina (IHM)

3.2.1 Sistema de Acionamento dos Motores

Os circuitos construídos para o acionamento dos motores seguiram os circuitos apresentados no relatório final da disciplina PMR2500 (Projeto de Conclusão de Curso I). O protótipo do Robô Lançador é composto por 2 motores de passo e 2 motores de corrente contínua.

3.2.1.1 Motores de Passo

Abaixo segue o circuito utilizado para o acionamento dos motores de passo:

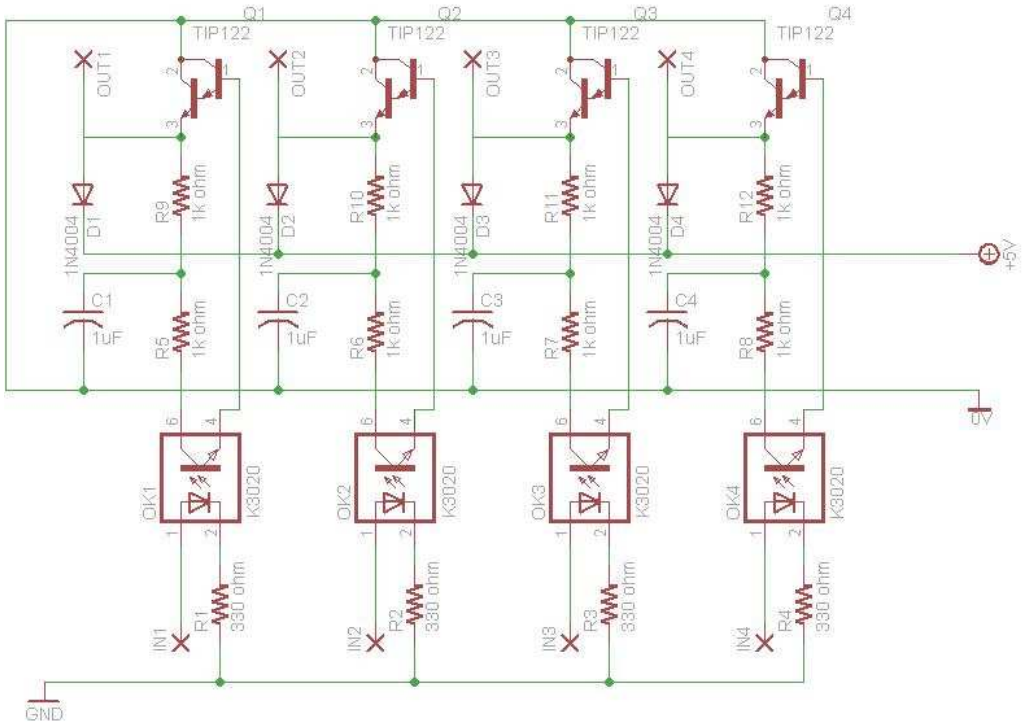


Figura 11 – Circuito de Acionamento do Motor de Passo

O circuito de acionamento do motor de passo (Figura 11) é composto por transistores, resistores, capacitores, optoacopladores e diodos retificadores. Aplicando um nível lógico alto à base do transistor, a junção coletor-emissor é conectada e a corrente fornecida pela alimentação do motor é liberada no enrolamento conectado ao transistor. À medida que o PIC envia a seqüência de sinais de acionamento, o motor de passo gira com uma velocidade controlada pela freqüência que os sinais são enviados. Os outros componentes possuem funções de proteção e filtro nesse circuito. Os resistores oferecem uma oposição à passagem de corrente elétrica. Os capacitores servem de filtro, evitando ruídos nos sinais enviados pelo PIC. Os diodos permitem a circulação da corrente enviada para os enrolamentos, mesmo quando os transistores estão desabilitados, evitando a queima destes componentes. O optoacoplador isola o circuito do PIC e circuito de acionamento.

3.2.1.2 Motores de Corrente Contínua

Abaixo segue o circuito utilizado no acionamento do motor de corrente contínua:

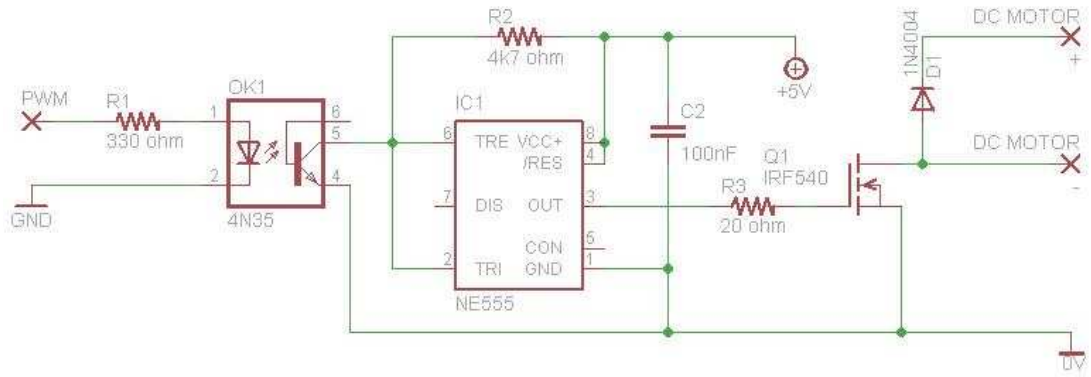


Figura 12 - Circuito de Acionamento do Motor de Corrente Contínua

O circuito de acionamento do motor de corrente contínua é composto por resistores, capacitores, diodos retificadores, um optoacoplador, um mosfet IRF540N e o circuito integrado 555. Mosfets de potência apresentam capacitância de gate relativamente alta. Para chaveá-los, é necessário carregar ou descarregar rapidamente estes capacitores. O foto-acoplador OK1, polarizado através de R1, forma um inversor lógico com o sinal nos pinos 6 e 2 do 555. Portanto, quando há sinal na entrada PWM, os pinos 6 e 2 apresentam 0V, e quando não há sinal apresentam a tensão da fonte de +5v. Já o 555, está configurado também como inversor lógico. Sua saída é aproveitada por possuir um buffer de potência de 200mA contínuos e até mais em surtos. O sinal enviado pela saída 3 controla, através do resistor R3, o gate do mosfet . Quando esta saída é alta, toda tensão de alimentação é aplicada no dreno do mosfet A entrada PWM, recebe o sinal proveniente de um microcontrolador PIC que é gerado conforme valor de entrada, havendo uma variação do sinal de 0 a 100% a razão cíclica (duty rate). Desta forma, razão cíclica nula (0) corresponde à velocidade nula do motor e razão cíclica 100% corresponde à velocidade máxima do motor.

Em todos os circuitos de acionamento, foi utilizado o optoacoplador (Figura 13).

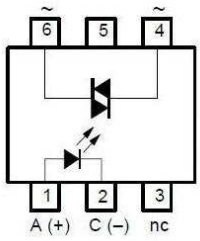


Figura 13 - Optoacoplador

Este componente consiste de um fototransistor óptico e um diodo emissor-infravermelho. Com o optoacopolador, o sinal de saída do PIC é isolado do sinal de entrada do acionamento dos motores, evitando a queima do PIC com picos de corrente durante funcionamento dos motores. Além dessa proteção contra queima, o optoacoplador auxilia na filtragem do sinal do PIC, contra os ruídos causados pelo campo magnético do motor.

3.2.2 Sensores

Três circuitos de sensores foram projetados para o robô:

- Sensores no sistema de rotação (Figura 14) indicando ao PIC um posicionamento inicial para o lançador, permitindo ao programa estabelecer as posições programadas.
- Sensores próximos ao lançador (Figura 15) indicando a presença de bolas, permitindo uma nova posição ao lançador.
- Sensores na saída do lançador (Figura 16) indicando que a bola foi lançada e o próximo comando pode ser executado.

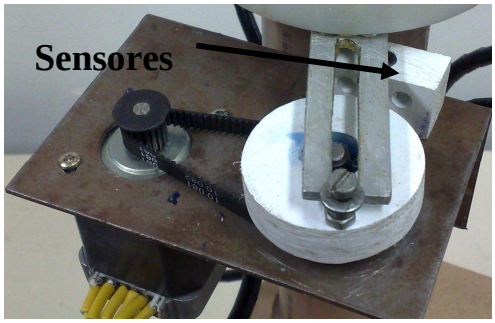


Figura 14 – Sensor Sistema de Rotação

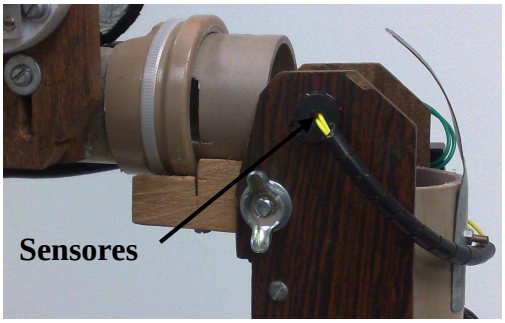


Figura 15 – Sensor Presença de bolas

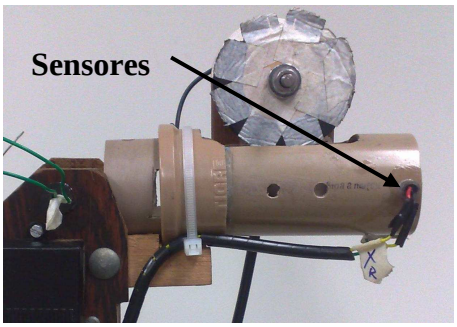


Figura 16 – Sensor Lançamento

Os três circuitos foram projetados da mesma forma (Figura 17) e são compostos por capacitores eletrolíticos, resistores, um fotodiodo TIL 32, um fototransistor TIL 78 e um transistor BC 548A. Os dois capacitores filtram sinais de alta e baixa freqüência. Os resistores oferecem uma oposição à passagem de corrente elétrica. O fotodiodo TIL 32 emite luz para o fototransistor TIL 78 e sempre que essa luz é obstruída, a corrente circula pela base do transistor BC 548A que envia um sinal ao PIC.

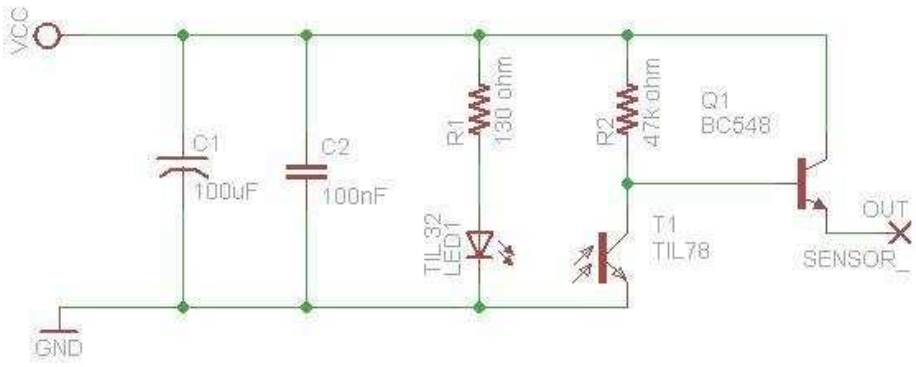


Figura 17 – Circuito dos Sensores

3.2.3 Conversão RS232/TTL

O circuito de conversão será utilizado devido à necessidade de se trabalhar em nível TTL nos sinais de entrada do microcontrolador. Dessa forma, o conversor de sinais RS232/TTL fará a conexão entre o microcontrolador e o computador (Figura 18).

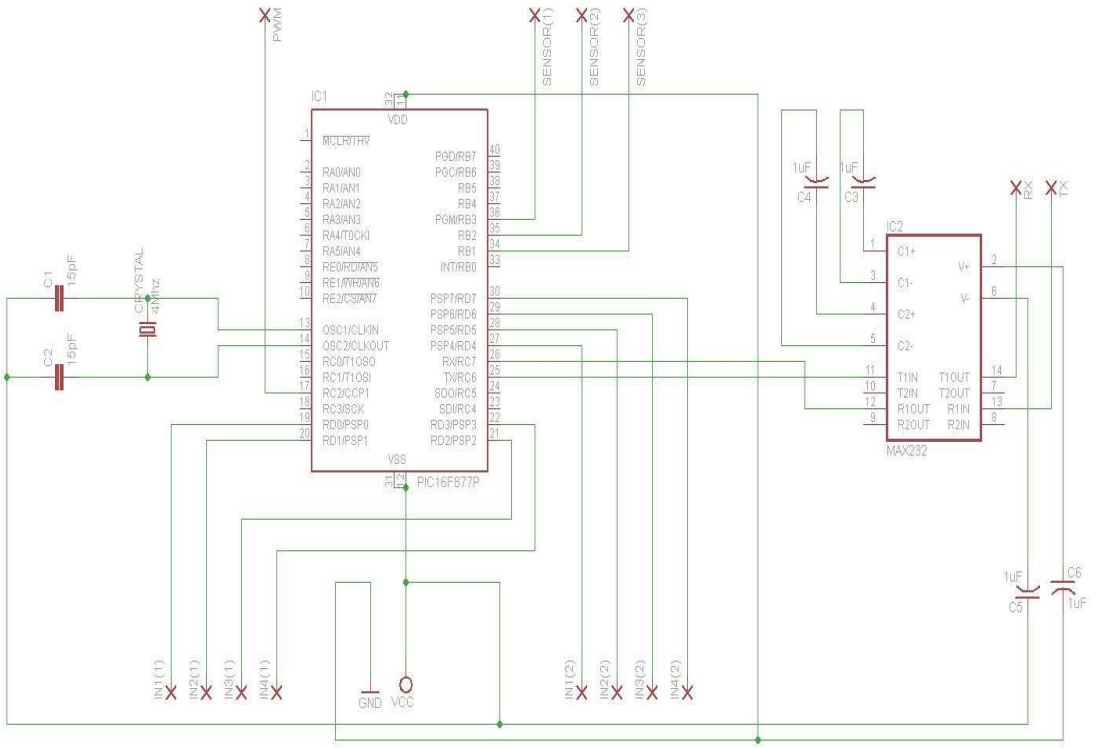


Figura 18 – Circuito do PIC e Conversão RS232/TTL

3.2.4 Programação do Microcontrolador PIC 16F877A

Para a etapa de Programação do Microcontrolador, utilizou-se o Kit de Desenvolvimento McMaster para a gravação dos programas, e o software MPLAB IDE v8. 15a [2] para a elaboração dos programas. Os programas foram desenvolvidos em Linguagem C e, então, gravados no Microcontrolador PIC16F877A.

No Desenvolvimento dos Programas, seguiram-se os seguintes passos:

Passo 1: Testes para familiarização com o microcontrolador e o Kit de Desenvolvimento. Nesta etapa, foram executados testes de acionamento dos Leds e

[2] Software contendo um conjunto de ferramentas que permite ao usuário escrever códigos fontes, compilar e testar microcontroladores PIC da Microchip.

leitura do teclado matricial para familiarização com os comandos básicos de programação do PIC utilizado.

Passo 2: Familiarização com os comandos estudados durante a disciplina PMR2500 e apresentados abaixo:

- Leitura dos sinais dos Sensores – Com base nos sinais lidos, o microcontrolador armazena variáveis que são enviadas por meio da comunicação serial ao Microcomputador. Esses dados são interpretados e apresentados na Interface Homem Máquina. Comandos associados a essa função:

  - input (A) – Lê o estado lógico do pino “A” do microcontrolador.

- Controle dos motores de passo – Com base nos sinais dos Sensores e nos comandos enviados pelo Usuário, o microcontrolador controla o movimento dos motores de passo, seja posicionamento ou velocidade. O controle é feito por meio de 4 portas de saída do microcontrolador que envia os pulsos necessários para movimentação do motor. Comandos associados a essa função:

  - set_timerX(A) – Modifica o conteúdo do timer interno “X” para “A”.
  - get_timerX() – Lê o Conteúdo do timer interno “X”.
  - setup_timer_X(modo) – Configura o Timer “X” conforme “modo”. O modo pode ser:
    - ✓ RTCC_INTERNAL: Timer “X” com clock interno;
    - ✓ RTCC_EXT_L_TO_H: Timer “X” com clock externo sensível à borda de subida;
    - ✓ RTC_EXT_H_TO_L: Timer “X” com clock externo sensível à borda de descida;

- ✓ RTCC_DIV_X: Prescaler dividido por “X”, podendo ser igual a 2, 4, 8, 16, 32, 64, 128 ou 256;
- ✓ RTCC_OFF: Timer “X” desligado;
- ✓ RTCC_8_BIT: Timer “X” em modo 8 bits.

- Enable_interrupts (valor) – Habilita uma interrupção em que “valor” pode ser:
  - ✓ GLOBAL: Habilitação global de interrupções;
  - ✓ INT_AD: Conversão A/D completa;
  - ✓ INT_ADOF: Overflow da conversão A/D;
  - ✓ INT_BUSCOL: Colisão de barramento (I2C);
  - ✓ INT_BUTTON:Pushbutton;
  - ✓ INT_CCP1: Módulo CCP1;
  - ✓ INT_CCP2: Módulo CCP2;
  - ✓ INT_COMP: Comparador Analógico;
  - ✓ INT_EEPROM: Escrita na EEPROM;
  - ✓ INT_EXT: Interrupção externa RB0/INT;
  - ✓ INT_EXT1: Interrupção externa INT1;
  - ✓ INT_EXT2: Interrupção externa INT2;
  - ✓ INT_LCD: Interrupção do módulo LCD;
  - ✓ INT_LOWVOLT: Detecção de baixa tensão;
  - ✓ INT_PSP: Chegada de dados no módulo PSP;
  - ✓ INT_RB: Mudança na porta B (RB4-RB7);
  - ✓ INT_RC: Mudança na porta C (RC4 - RC7);
  - ✓ INT_RDA: Recepção de dados na USART;
  - ✓ INT_RTCC: Estouro de contagem do timer 0;
  - ✓ INT_TBE: Buffer de transmissão vazio;

- ✓ INT_TIMER0: Estouro de contagem do timer0;
- ✓ INT_TIMER1: Estouro de contagem do timer1;
- ✓ INT_TIMER2: Estouro de contagem do timer2;
- ✓ INT_TIMER3: Estouro de contagem do timer3;

- Disable_interrupts(valor) – Desabilita interrupção. “valor” é o mesmo usado para habilitar.
- output_bit (A, B) – Coloca nível lógico “B” no pino “A”

- Controle do motor de corrente contínua – Assim como no caso dos motores de passo, o microcontrolador controla o movimento do motor de corrente contínua. No entanto, somente a velocidade do motor é controlada. Esse controle também será baseado nos sinais dos sensores e nos comandos do Usuário. Comandos associados a essa função:

- setup_ccpX(modo) – Configura o funcionamento do watchdog. O “modo” pode ser:
  - ✓ CCP_OFF: módulo CCP desligado;
  - ✓ CCP_CAPTURE_FE: captura na borda de descida do sinal;
  - ✓ CCP_CAPTURE_RE: captura na borda de subida do sinal;
  - ✓ CCP_CAPTURE_DIV_4: captura a cada 4ª borda de subida do sinal;
  - ✓ CCP_CAPTURE_DIV_16: captura a cada 16ª borda de subida do sinal;
  - ✓ CCP_COMPARE_SET_ON_MATCH: modo de comparação, seta pino CCPx;
  - ✓ CCP_COMPARE_INT: modo de comparação com geração de interrupção;

- ✓ CCP_COMPARE_RESET_TIMER: modo de comparação com reset do timer 1 ou timer 3;
- ✓ CCP_PWM: configura o modo de geração de sinal PWM.

- Set_pwmX_duty(valor) – configura o ciclo ativo do módulo CCP no PWM. “valor” é uma variável ou constante inteira de 8 ou 16 bits.

- Comunicação com Computador – o microcontrolador enviará e receberá dados ao Computador por meio da comunicação serial. Os dados enviados serão dados de status do sistema (sensores) e os dados recebidos serão comandos de controle do Lançador provenientes da Interface. A comunicação é baseada em interrupções ativadas conforme mudança nos buffers de leitura e de envio de dados.

Passo 3: Elaboração do controle dos motores, leitura dos sensores e envio de dados pela comunicação serial. Esta etapa será baseada nos dados recebidos da Interface Homem-Máquina pela comunicação serial. Esses dados serão lidos e decodificados de maneira a alterar variáveis de controle do robô como modos (manual e automático), freqüência de lançamento, período de lançamento, quantidade de bolas lançadas e posição do lançamento. Os sinais dos sensores serão lidos periodicamente baseados em interrupções do microcontrolador e, a cada mudança no nível do sinal, provocam alterações em variáveis e envio de dados do microncontrolador para a Interface. Na interface esses dados serão novamente decodificados e atualizarão a janela de status.

3.2.5 Interface Homem-Máquina

O projeto da Interface Homem Máquina (IHM) foi baseado na arquitetura MVC (Model-View-Controller) que consiste na divisão da estrutura em 3 camadas:

- Model – É o modelo da aplicação no qual são definidas as propriedades e atributos.

- View – É a camada de visualização da aplicação, onde são exibidas as informações ao usuário.

- Controller – É a camada na qual são processadas todas as requisições feitas por meio da camada view.

Tendo definida a arquitetura, seguiram-se os seguintes passos: levantamento dos casos de uso, levantamento de requisitos e definição de um padrão de comunicação. Cada etapa será discutida a seguir.

3.2.5.1 Casos de Uso

As funcionalidades necessárias foram discutidas e a partir delas construiu-se um diagrama geral e dois outros diagramas para detalhar as funções do modo automático (execução em passos) e do modo manual (operação manual), como ilustrado nas figuras abaixo:

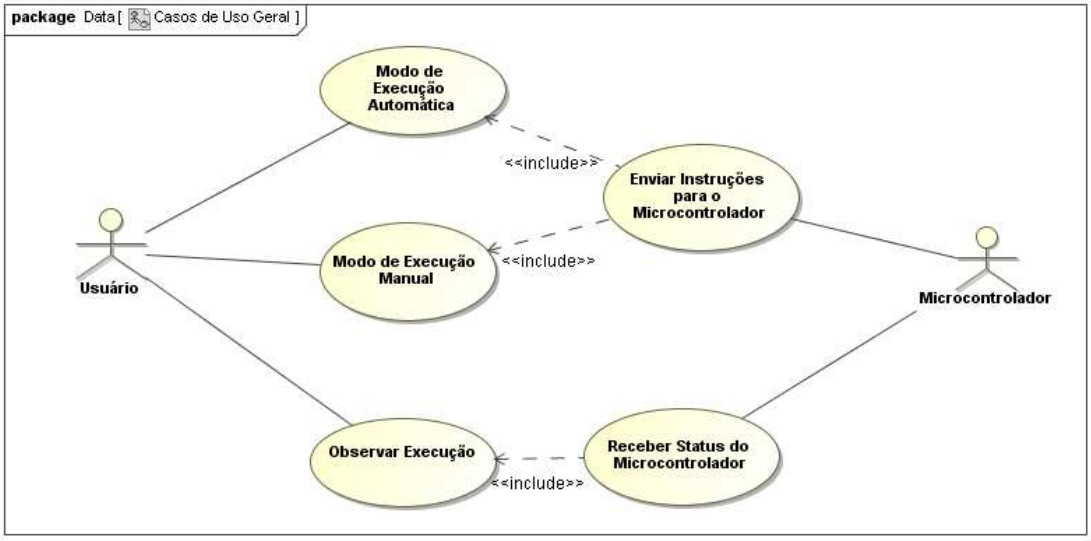


Figura 19 - Diagrama de Casos de Uso Geral

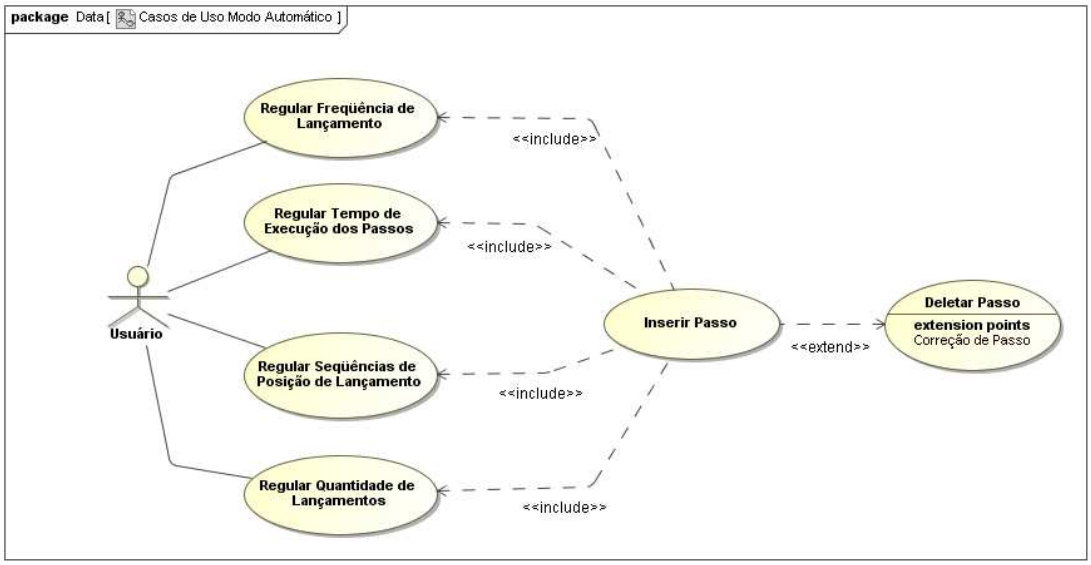


Figura 20 – Diagrama de Casos de Uso – Modo Automático

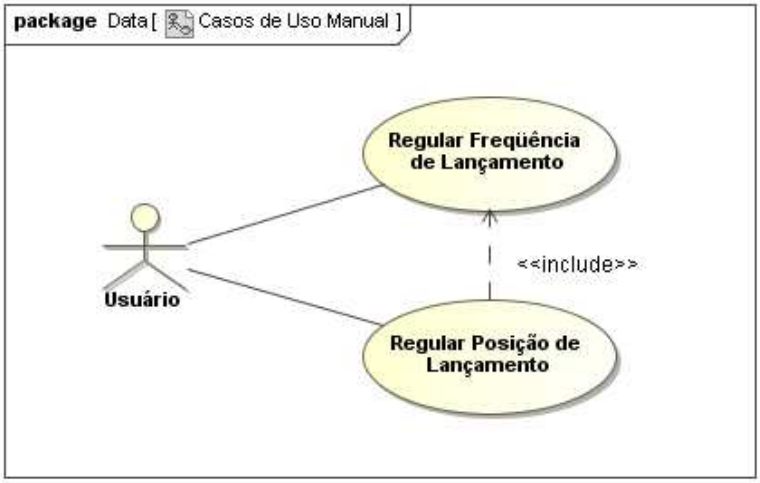


Figura 21 - Diagrama de Casos de Uso - Modo Manual

3.2.5.2 Requisitos

Tendo como base os casos de uso e o modo de funcionamento entre a programação da HMI e do microcontrolador, os seguintes requisitos foram levantados para o programa da HMI.

Tabela de Requisitos		
1.		O sistema deve armazenar valores das variáveis (posição, freqüência, tempo de execução, quantidade de passos) e enviar via serial para o microcontrolador.
2.		O sistema deverá operar em modo automático ou manual conforme escolha do usuário.
	2.1	No modo manual, o sistema permite ao usuário controlar diretamente o Lançador. Permitindo-se variar a posição em que a bola será lançada e a freqüência de lançamento.
	2.2	No modo automático, o usuário poderá pausar, continuar, abortar e criar passos de execução dos movimentos.
3.		O sistema recebe informações do microcontrolador, as traduz para linguagem humana e as exibe para o usuário. Essas informações conterão dados referentes á quantidade de bolas lançadas e status do realimentador.

Tabela 1 - Diagrama de Requisitos da Interface Homem-Máquina

3.2.5.3 Padrão de Comunicação

Para ser possível a interpretação dos dados transmitidos por parte do computador (Usuário) ou por parte do microcontrolador é necessário que haja uma padronização do formato da comunicação. Dessa forma, baseando-se no padrão desenvolvido no projeto do Torno CONTROLE da disciplina Projeto de Máquinas (PMR2450), segue o padrão para a comunicação do Lançador de Bolas de Tênis de Mesa:

Padrão de Comunicação				
CASO GENÉRICO				
SOH	**ID**	**COD**	**DATA**	**ETX**
SOH	Start of Header			
ID	Identificador da mensagem			
COD	Código da função			
DATA	dados		(opcional)	
ETX	End of Text			

Tabela 2 - Padrão de Comunicação

Nessa estrutura, o ID é um byte seqüencial de 0 a 255d. O COD é um byte que representa a função a ser executada (pausar, continuar, comando manual, inserir passo, etc). E caso seja alteração de variáveis como posição, freqüência de lançamento de bolas e passos, DATA informa o valor alterado.

Assim, por exemplo, um comando de “Start” pode ser feito da seguinte maneira:

Ex: Start					**Tradução**
Byte nº	**1**	**2**	**3**	**4**	
HMI	SOH	5d	253d	ETX	Start
Rabbit	SOH	5d	ACK	ETX	ok

Tabela 3 - Padrão de Comunicação

Obs: ACK é código ASCII para “Acknowledge”.

3.2.5.4 Interface Gerada

A interface foi desenvolvida em linguagem de programação JAVA com o uso do software NetBeans IDE 6.7.1 juntamente com o pacote de comunicação serial

javacomm. A seguir estão os modos Manual e Automático da Interface Homem Máquina (IHM) gerada:

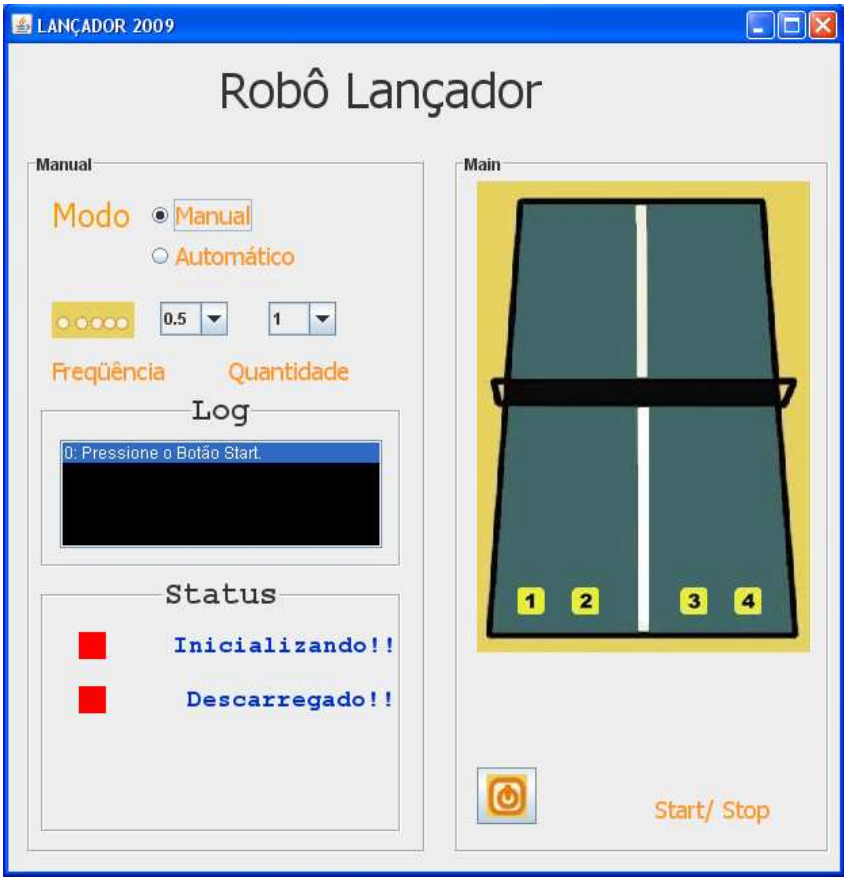


Figura 22 - Modo Manual da Interface Homem Máquina

No modo de execução manual é possível controlar a freqüência de lançamento e a quantidade das bolas e a posição na qual elas serão lançadas. A posição é definida pelos locais numerados de 1 a 4. Para iniciar ou interromper o lançamento deve-se pressionar o botão Start/Stop. Há uma janela de Log para comprovar o envio dos comandos da forma correta, indicando também as etapas a serem seguidas para o uso correto do Lançador. Além da Janela de Status, que apresenta o status do robô, indicando se está inicializado ou não e carregado ou não. Essas condições são obtidas por meio de envio do sinal dos sensores presentes na estrutura mecânica.

Figura 23 - Modo Automático da Interface Homem-Máquina

No modo de execução automático, é possível selecionar a sequência de lançamento, regular a freqüência de lançamento de cada passo e estabelecer a quantidade de repetições da sequência de passos. Para iniciar o modo automático, inicialmente deve-se definir a sequência de lançamento, freqüência de lançamento para cada posição da sequência e a quantidade de repetições. Um novo passo pode ser adicionado após a escolha das variáveis. Para iniciar ou interromper o lançamento deve-se pressionar o botão Start/Stop. À medida que o robô realiza a sequência, a janela de Log apresenta a execução de cada passo estabelecido. Toda movimentação, programada no passo, é executada a partir dos sinais enviados pelos sensores ao longo da estrutura do protótipo, evitando falhas quando o estoque de bolas não alimenta o lançador.

4 ANÁLISE DE PROJETO

Foram executadas as seguintes atividades:

- Projeto e construção da Estrutura Mecânica

- Projeto e construção do Sistema de Acionamento dos Motores

- Projeto e construção do Sistema de Sensores

- Elaboração da Comunicação Serial

- Programação do Microcontrolador

- Programação da Interface Homem-Máquina

Foram realizados diversos testes com os modos manual e automático. No primeiro modo, o robô apresentou bastante eficiência, retornando às posições definidas com precisão e rapidez. Uma faixa de freqüência de lançamento entre 20 e 240 bolas por minuto foi obtida com o Sistema de Realimentação do robô. Dispondo o lançador na posição vertical, e medindo a altura alcançada pela bola no lançamento, estimou-se a velocidade da bola no instante do lançamento pela Lei da Conservação da Energia:

Aceleração da gravidade (g) = 9,81 m/s^2

Altura alcançada (h) = 2 metros

$$Energia = cte = EnergiaCinética + EnergiaPotencial$$

$$Energia_{inicial} = Energia_{final} \Rightarrow \frac{mv^2}{2} = mgh$$

$$v^2 = 2 \times 9{,}81 \times 2 \qquad \text{(Eq. 2)}$$

$$v \cong 6{,}2 m/s$$

A estrutura mecânica foi inicialmente testada alimentando-se diretamente os motores por meio de uma fonte, e encontra-se operando bem. Problemas de folga e atrito foram eliminados, estando o robô em condições de ser controlado. Após a conclusão total da etapa de interface e parcial da etapa de programação do microcontrolador, foram possíveis diversos testes de controle dos motores. O sistema

mecânico apresentou alguns problemas relacionados à vibração, que foram logo solucionados.

Ambos os circuitos de acionamento dos motores de passo e corrente contínua foram testados nas condições fora da estrutura mecânica e junto à estrutura mecânica. Os resultados foram satisfatórios, operando com velocidades coerentes com o intuito do protótipo, treinamento de mesatenistas.

A interface Homem-Máquina está respondendo de forma correta os comandos selecionados pelo usuário. Respeitando o padrão de comunicação, todos os dados são enviados ao microcontrolador. Foi inserida uma janela de log, que apresenta os comandos enviados e os passos necessários para o uso da interface.

Com relação à programação do microcontrolador, há os dois modos de operação: manual e automático. Ambos os modos encontram-se operando bem, dadas as tarefas e condiçõ es estipuladas na interface. Foram realizados diversos testes e ajustes, como a alteração do cristal oscilador para aumentar a freqüência das interrupções responsáveis pelo controle e acionamento dos motores. Dessa forma, foi possível executar o programa com maior rapidez, além de se alterar os valores mínimo e máximo de freqüência de lançamento das bolas.

Além da programação e controle do robô, o custo final de todo protótipo foi avaliado.

Considerando apenas os materiais utilizados para a construção do robô, obtivemos a seguinte estimativa de custo:

Produto	Qte	Valor Unitário	Valor Total
Transístor TIP122	8	0,70	5,60
Optoacoplador k3020	8	0,90	7,20
Resistor 330 ohm 1/8 W	8	0,10	0,80
Resistor 1000 ohm 1/8 W	16	0,10	1,60
Capacitor eletrolítico 1microFx50v	8	0,20	1,60
Diodo 1N4007	8	0,10	0,80
Borne	12	0,40	4,80

Tabela 4 - Driver Motor de Passo

Produto	Qte	Valor Unitário	Valor Total
Transístor IRF540N	1	1,60	1,60
Transístor BC548a	1	0,20	0,20
Optoacoplador k3020	1	0,90	0,90
Resistor 4700 ohm 1/8 W	1	0,10	0,10
Resistor 20 ohm 1/8 W	2	0,10	0,20
Resistor 330 ohm 1/8 W	1	0,10	0,10
CI 555	1	0,50	0,50
Capacitor 100 nF	2	0,20	0,40
Diodo 1N4007	1	0,10	0,10
Borne	2	0,40	0,80
Soquete 8 terminais	1	0,30	0,30

Tabela 5 – Driver Motor de Corrente Contínua

Produto	Qte	Valor Unitário	Valor Total
Transistor BC548a	3	0,20	0,60
Capacitor eletrolítico 100microFx50V	3	0,20	0,60
Capacitor 100nF	3	0,20	0,60
Resistor 4700 ohm 1/8 W	3	0,10	0,30
Resistor 100 ohm 1/8 W	3	0,10	0,30
Diodo emissor TIL32	3	1,50	4,50
Diodo receptor TIL78	3	1,50	4,50
Borne	10	0,40	4,00

Tabela 6 – Circuito do Sensor

Produto	Qte	Valor Unitário	Valor Total
PIC 16F877A	1	15,00	15,00
Capacitor eletrolítico 1microFx50V	4	0,20	0,80
Capacitor 15 pF	2	0,20	0,40
MAX232	1	2,50	2,50
Cristal 1 MHz	1	1,00	1,00
Soquete 40 terminais	1	2,50	2,50
Soquete 16 terminais	1	1,50	1,50
Borne	8	0,40	3,20

Tabela 7 – Circuito PIC e Comunicação Serial

Produto	Qte	Valor Unitário	Valor Total
Parafusos, porcas, arruelas	70	0,10	7,00
Cotovelo 50mm x 90º	1	2,00	2,00
Luva 50 mm marrom	1	2,00	2,00
Tubo PVC 50 mm (metros)	1	6,00	6,00
Conectores	2	0,50	1,00
Fonte ATX	1	30,00	30,00
Motor DC	2	10,00	20,00
Placa de fenolite	1	5,00	5,00
Cabos	1	10,00	10,00
3 motores de passo, correias e engrenagens	1	60,00	60,00
Rolamento 50x80x16	1	24,15	24,15
Tarugo Nylon 90x10	1	12,00	12,00
Madeira	1	10,00	10,00
Alumínio	1	5,00	5,00
Acrílico	1	5,00	5,00
Tubo de caneta	1	0,50	0,50
Dobradiça	2	0,50	1,00
Conector DB9 PCI	1	1,40	1,40
Spiraduto (metros)	3	1,20	3,60
Cabo Serial	1	30,00	30,00

Tabela 8 – Componentes Diversos

Produto	Qte	Valor Unitário	Valor Total
Driver Motor de Passo	2	22,40	44,80
Driver Motor de Corrente Contínua	1	5,20	5,20
Circuito do Sensor	3	15,40	46,20
Circuito PIC e Comunicação Serial	1	26,90	26,90
Componentes Diversos	1	235,65	235,65

Tabela 9 – Protótipo

Somando o preço de todos os circuitos e componentes envolvidos no protótipo, o custo de fabricação do robô foi de *R$ 306,35*. Levando-se em conta que os componentes foram comprados em pequena quantidade e em diversos locais, nosso protótipo, produzido em larga escala, poderia ter um custo menor do que o estimado e estaria em plenas condições de competir com outros robôs comerciais de características semelhantes. Ainda que nosso protótipo tenha sido fabricado com

materiais de fácil usinabilidade (madeira, nylon, aço), pouco utilizados nos robôs comerciais, moldes plásticos, fabricados em grande quantidade de forma a reduzir seu custo de fabricação, poderiam ser utilizados na construção do robô, permitindo a comercialização do mesmo.

5 REFERÊNCIAS BIBLIOGRÁFICAS

[1]. **PEREIRA, F.** Microcontroladores PIC Programação em C. 6ª Edição. São Paulo: Érica, 2007. 358 p.

[2]. **SOUZA, D. J.; LAVINIA, N. C.** Conectando o PIC – Recursos Avançados. 4ª Edição. São Paulo: Érica, 2003. 384 p.

[3]. **NEWGY INDUSTRIES.** Fabricante e vendedora de equipamentos de Tênis de Mesa. Disponível em: < http://www.newgy.com/support/index.html >, acessado em 11/03/09.

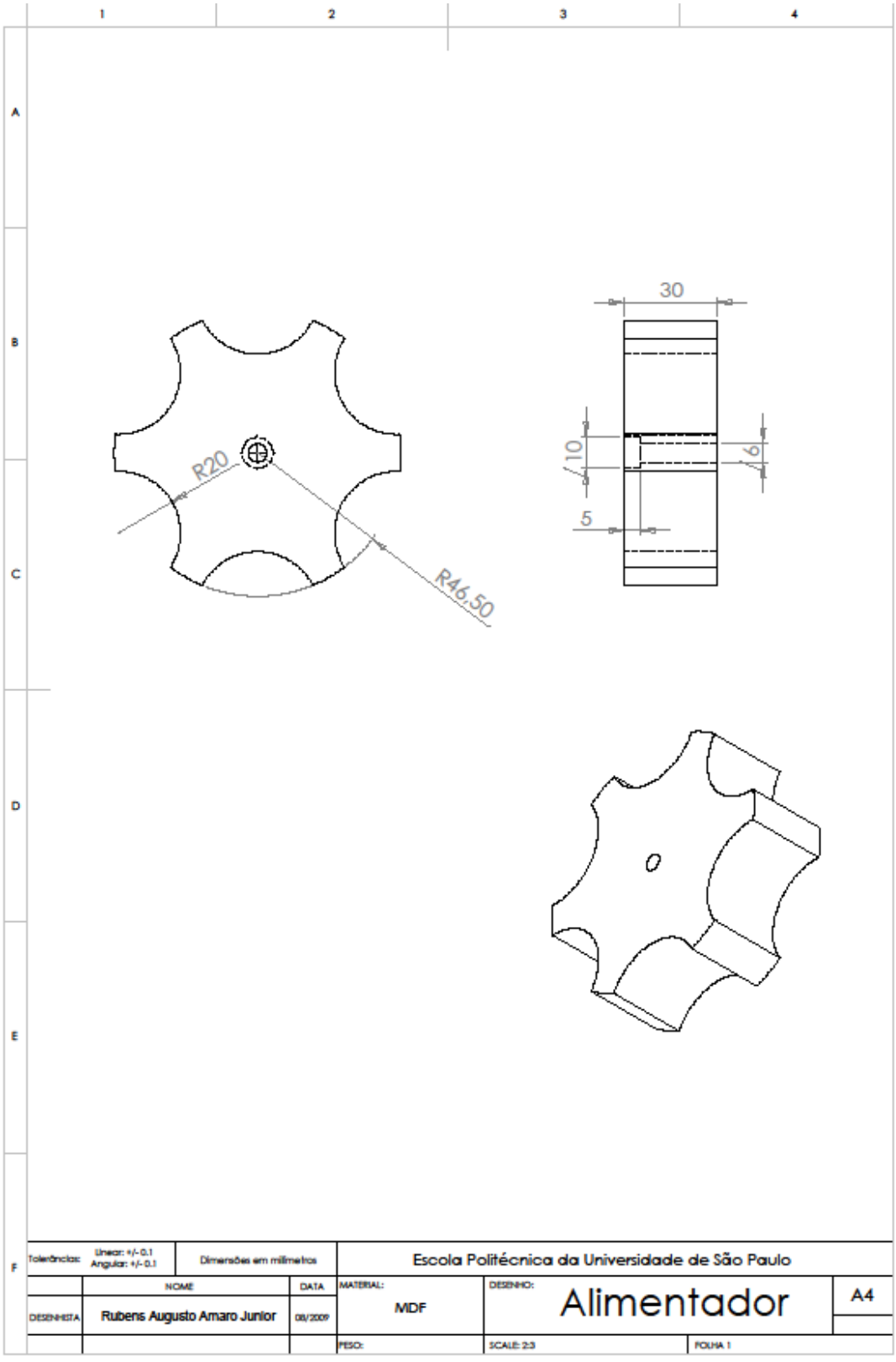
[4]. **SB DISTRIBUTION.** Distribuidora de equipamentos de Tênis de Mesa. Disponível em: < http: //www.smartpong.com /index.php >, acessado em 11/03/09.

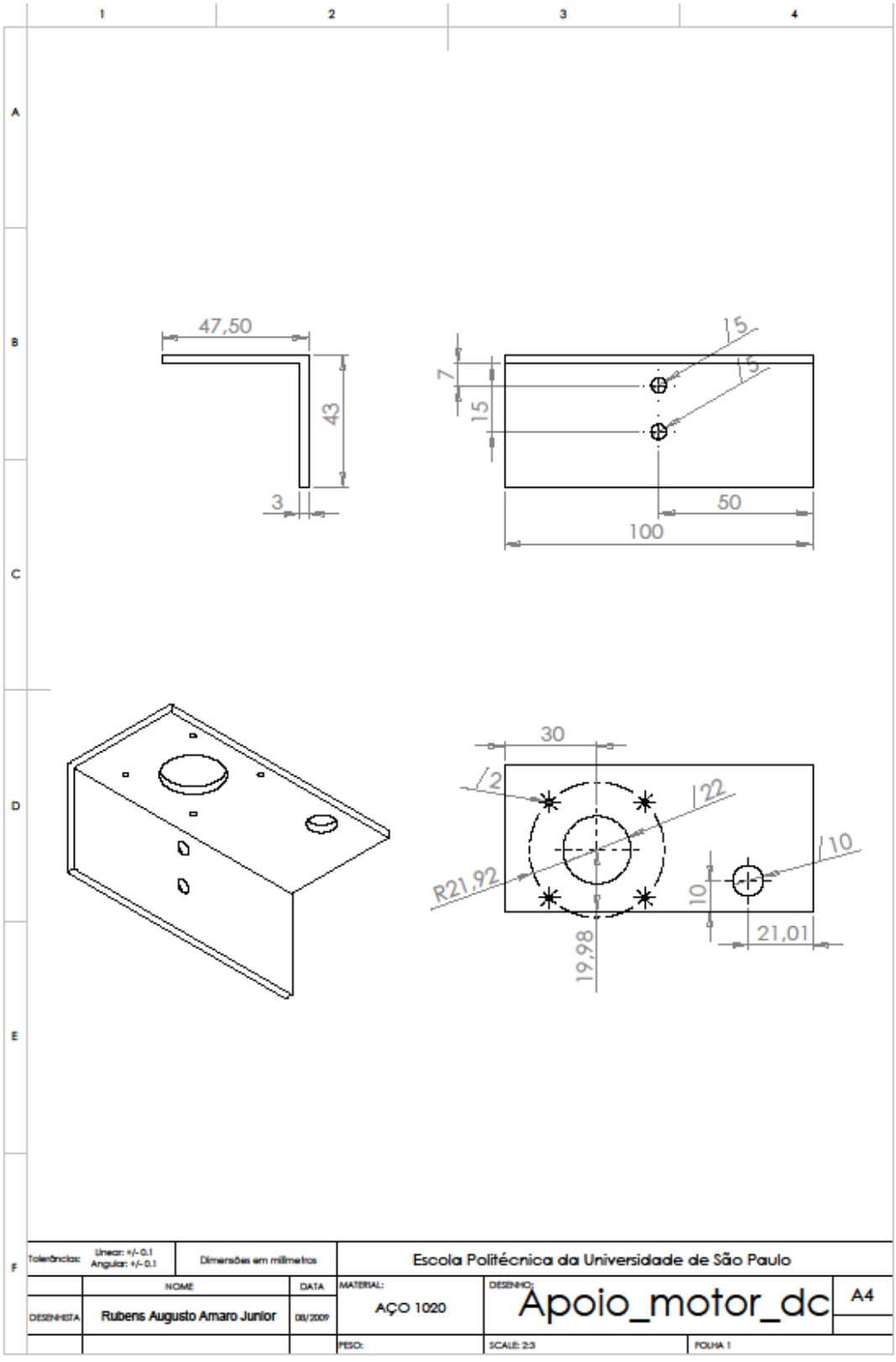
[5]. **BUTTERFLY.** Fabricante e vendedora de equipamentos de Tênis de Mesa. Disponível em: < http: //www.butterflyonline.com/robots.asp >, acessado em 11/03/09.

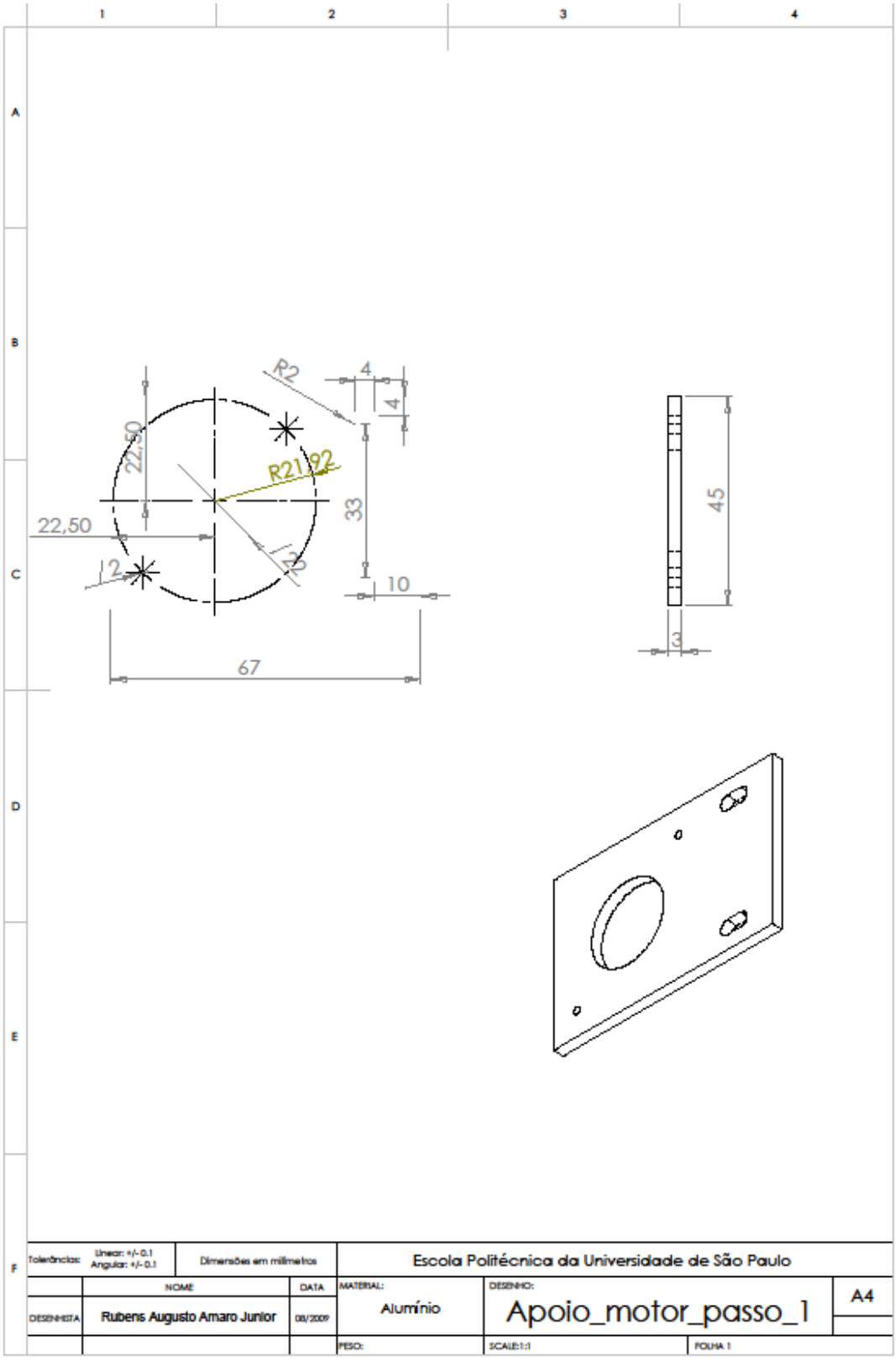
[6]. **MICROCHIP.** Fornecedora de Microcontroladores e Semicondutores Analógicos. Disponível em: <http://www.microchip.com/wwwproducts/Devices.aspx?dDocNam e=en010242>, acessado em 11/03/09.

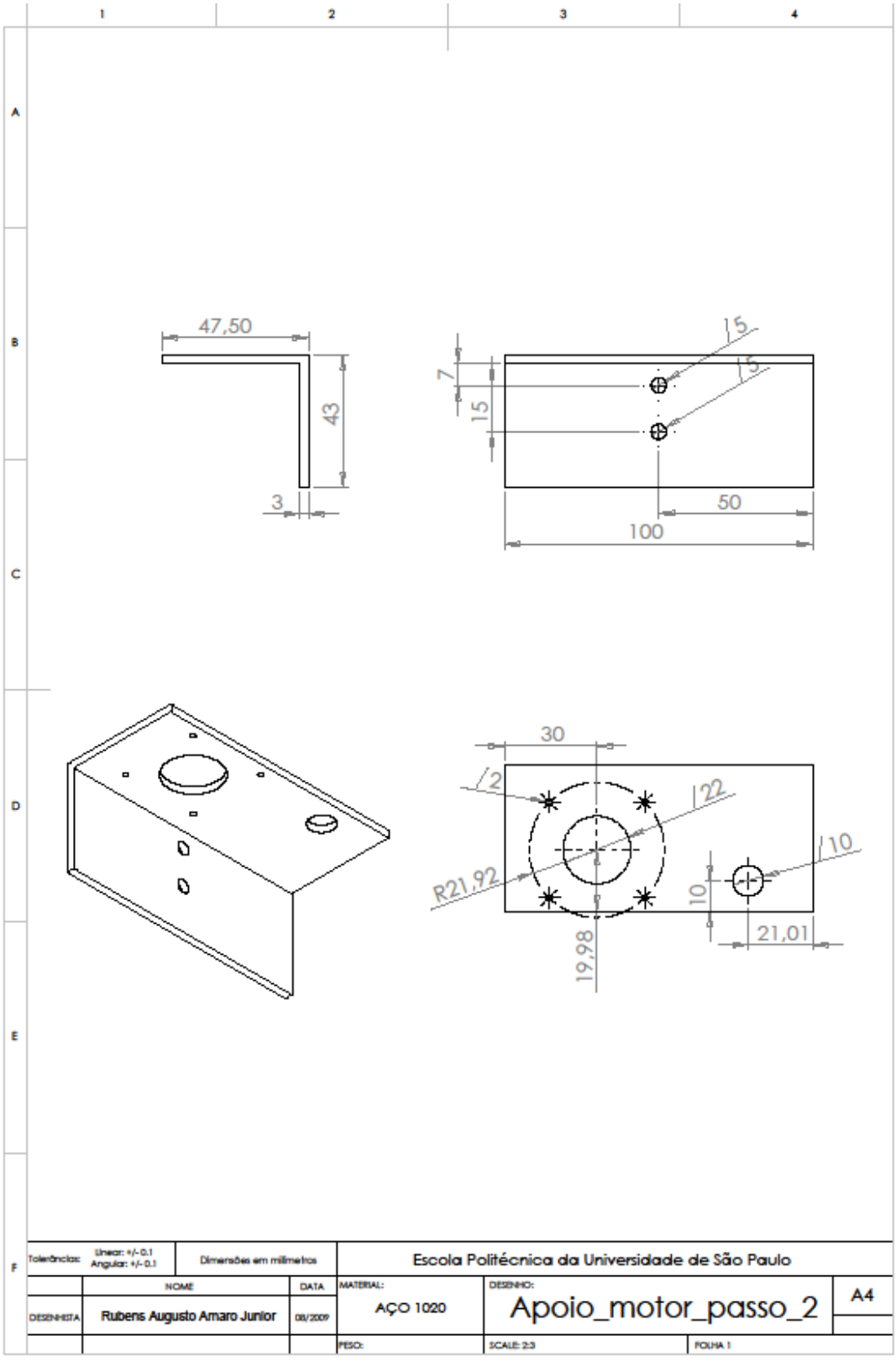
[7]. **LABTOOLS MOSAICO DIDACTIC DIVISION.** Divisão de Produtos Didáticos, Cursos, Livros, Kits, Componentes da Mosaico High Performance Solutions. Disponível em: < http://www.labtools.com.br/index.asp?area=21&subarea=b&idio ma=por& script=produtos&prod=319>, acessado em 11/03/09.

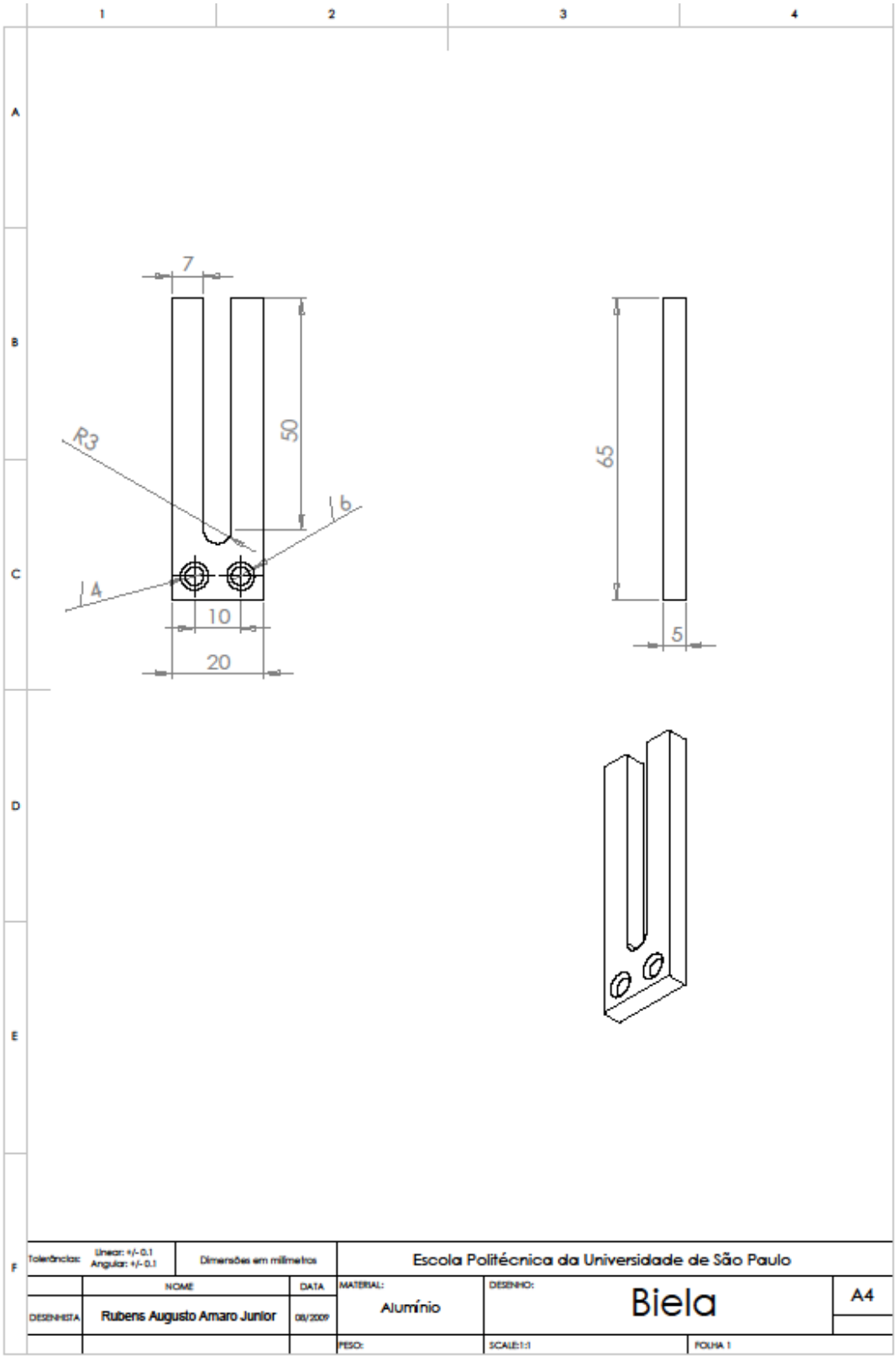
APÊNDICE A – DESENHOS DE FABRICAÇÃO DO ROBÔ

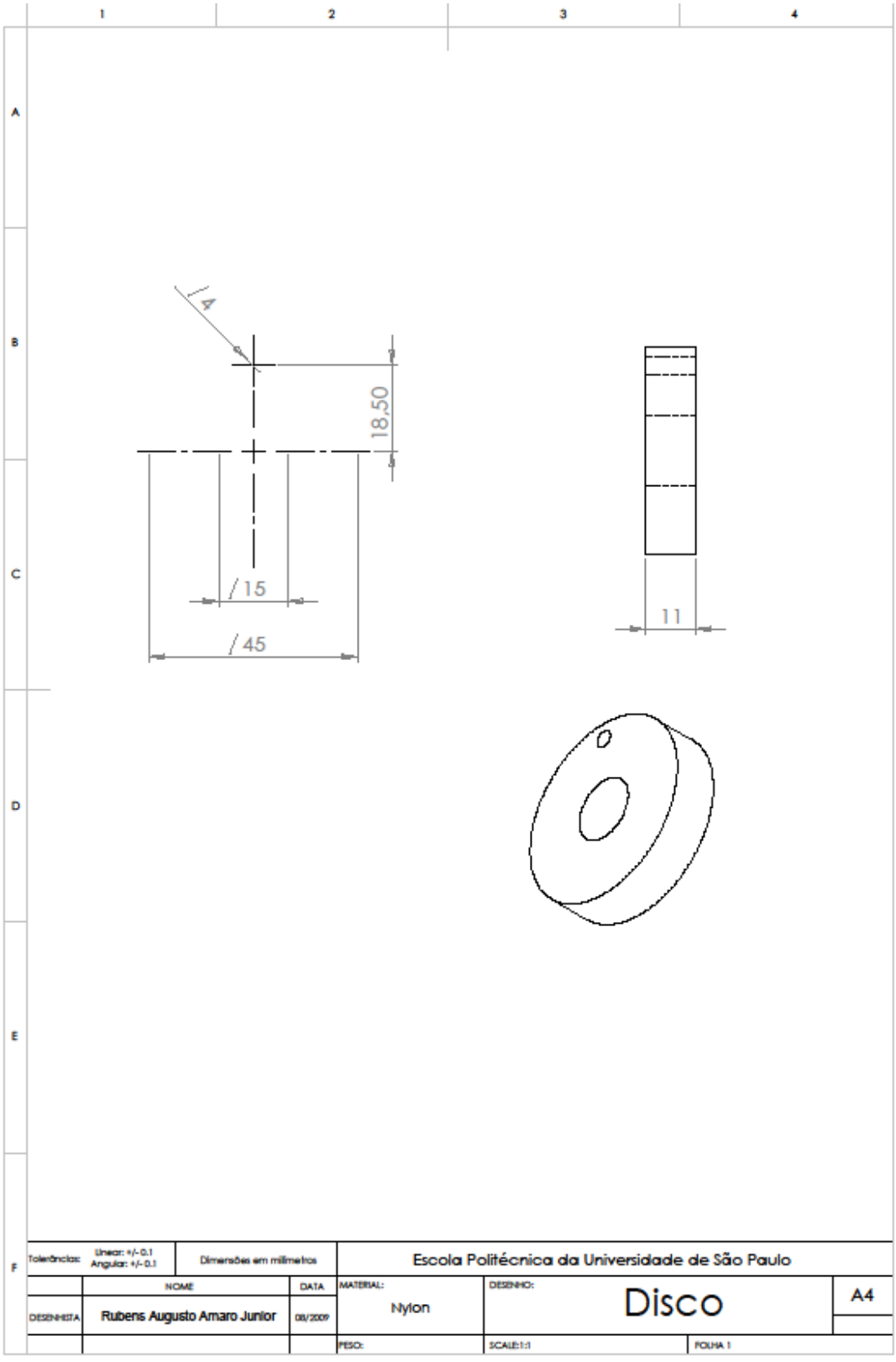

1
2
3
4
A
B
C
D
E
F
R20
R46.50
30
10
6
5
Tolerâncias:
Linear: +/- 0.1
Angular: +/- 0.1
Dimensões em milímetros
Escola Politécnica da Universidade de São Paulo
NOME
DATA
DESENHISTA
Rubens Augusto Amaro Junior
09/2009
MATERIAL:
MDF
DESENHO:
Alimentador
A4
PESO:
SCALE: 2:3
FOLHA 1

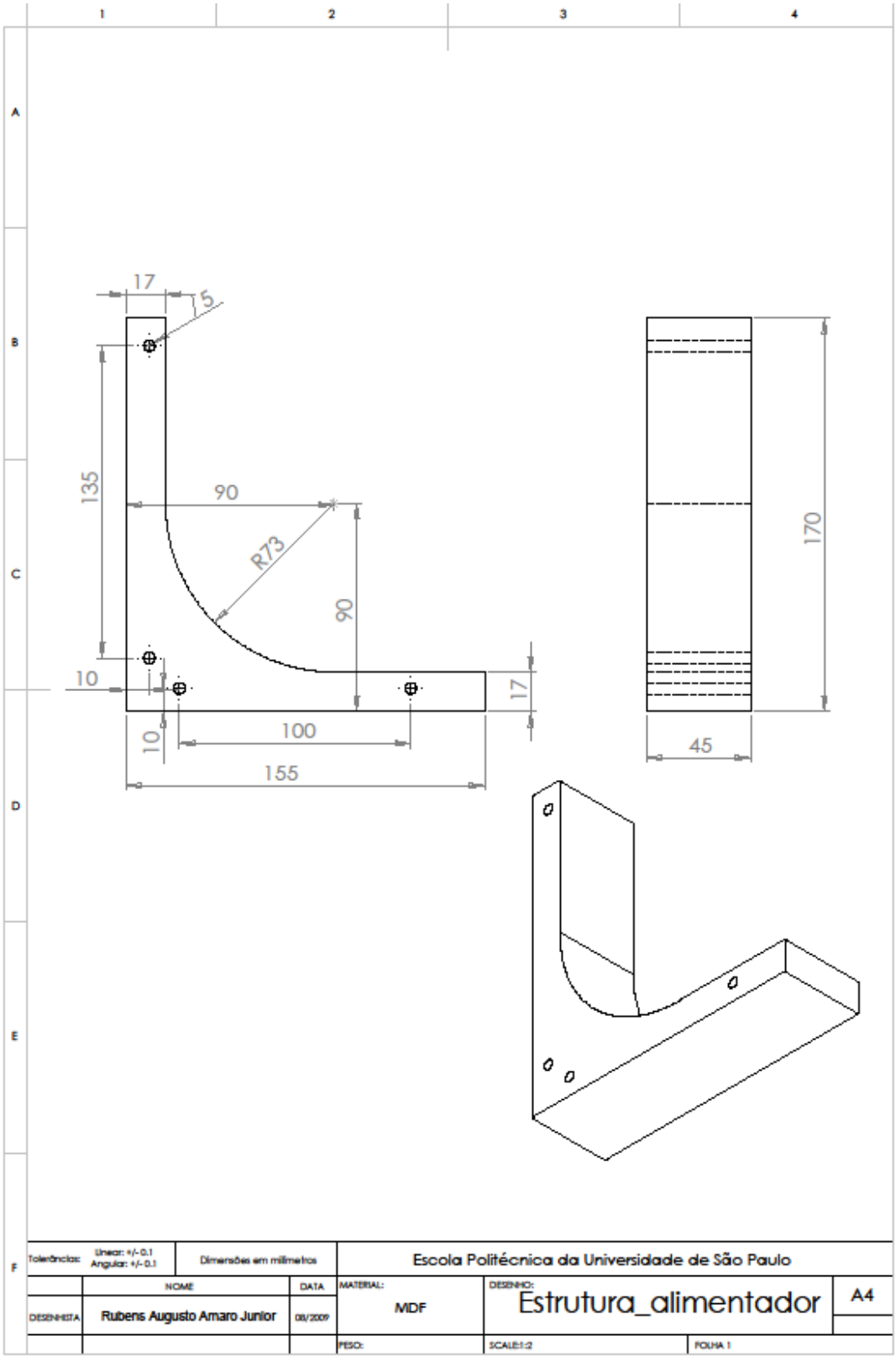

47,50
43
3
15
7
15
5
5
50
100
30
2
22
R21,92
10
10
19,98
21,01
Tolerâncias: Linear: +/- 0.1 Angular: +/- 0.1
Dimensões em milímetros
Escola Politécnica da Universidade de São Paulo
NOME
DATA
DESENHISTA
Rubens Augusto Amaro Junior
09/2009
MATERIAL:
AÇO 1020
DESENHO:
Apoio_motor_dc
A4
PESO:
SCALE: 2:3
FOLHA 1

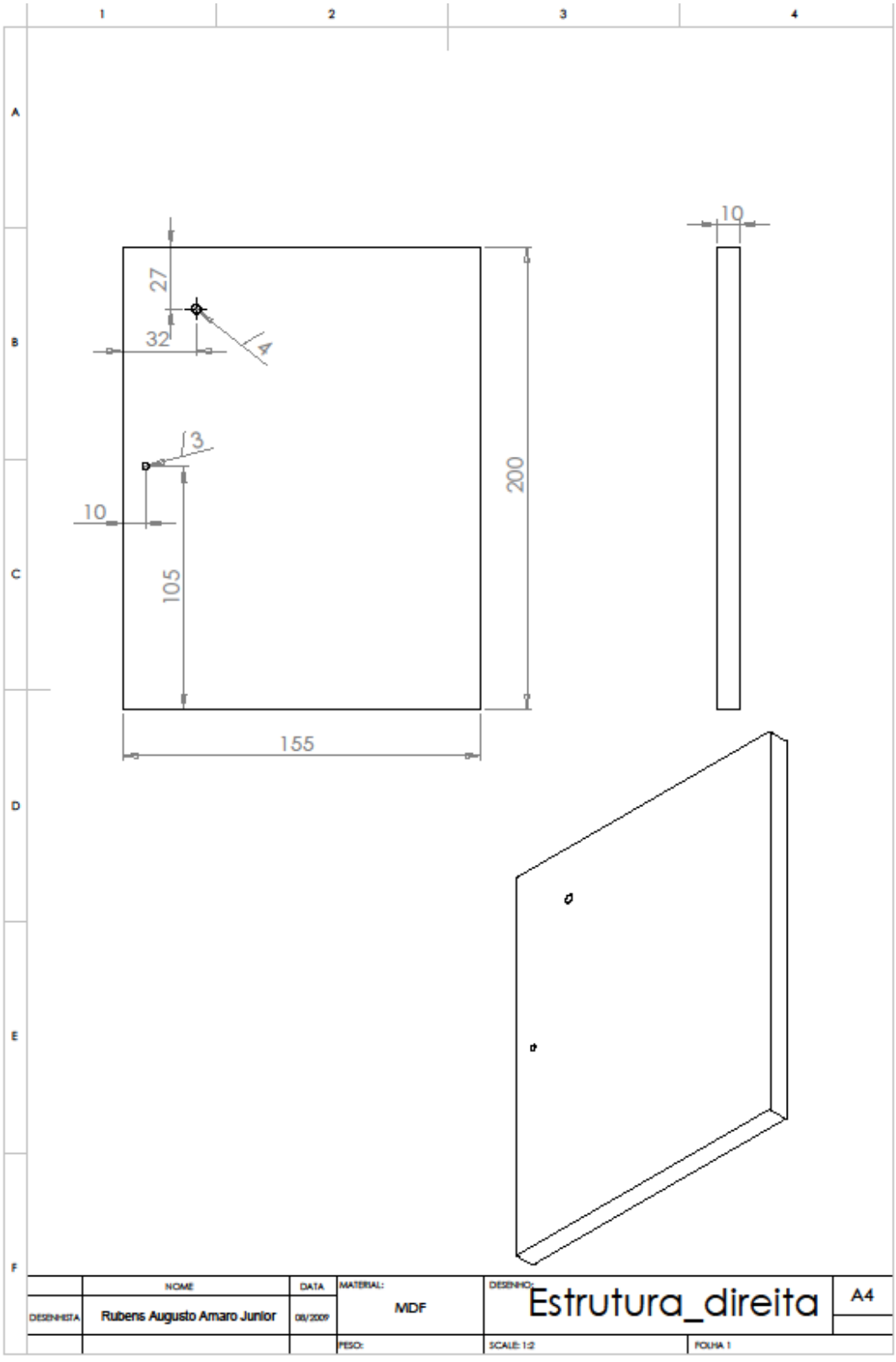

1
2
3
4
A
B
C
D
E
F
R2
4
4
22,50
R21,92
33
22,50
2
10
67
45
3
Tolerâncias: Linear: +/- 0.1 Angular: +/- 0.1
Dimensões em milímetros
Escola Politécnica da Universidade de São Paulo
NOME
DATA
MATERIAL:
DESENHO:
A4
DESENHISTA
Rubens Augusto Amaro Junior
09/2009
Alumínio
Apoio_motor_passo_1
PESO:
SCALE:1:1
FOLHA 1

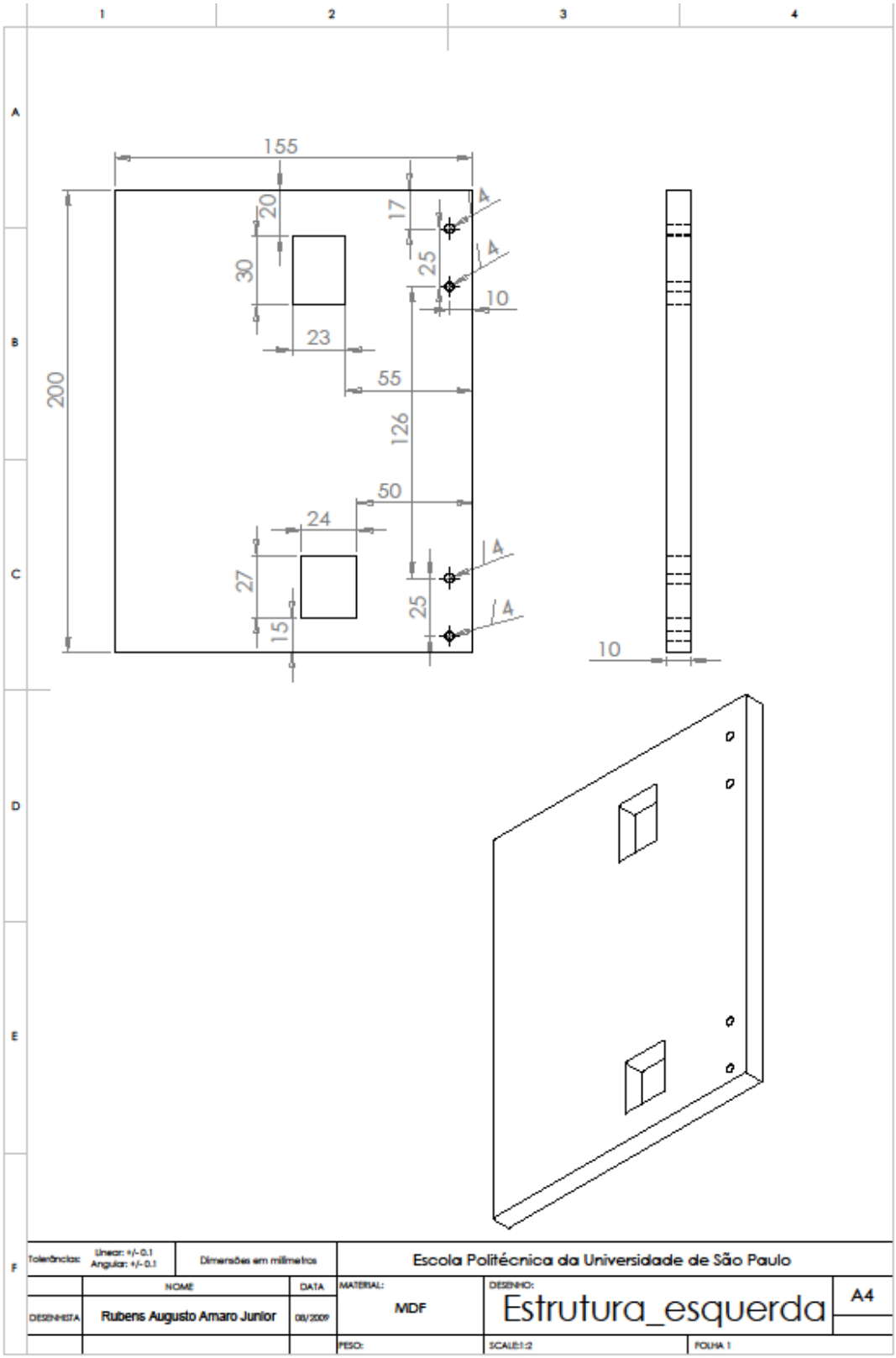

47,50
43
3
15
7
15
5
5
50
100
30
2
22
10
R21,92
10
21,01
19,98
Tolerâncias: Linear: +/- 0.1 Angular: +/- 0.1
Dimensões em milímetros
Escola Politécnica da Universidade de São Paulo
NOME
DATA
MATERIAL:
DESENHO:
A4
DESENHISTA
Rubens Augusto Amaro Junior
09/2009
AÇO 1020
Apoio_motor_passo_2
PESO:
SCALE: 2:3
FOLHA 1

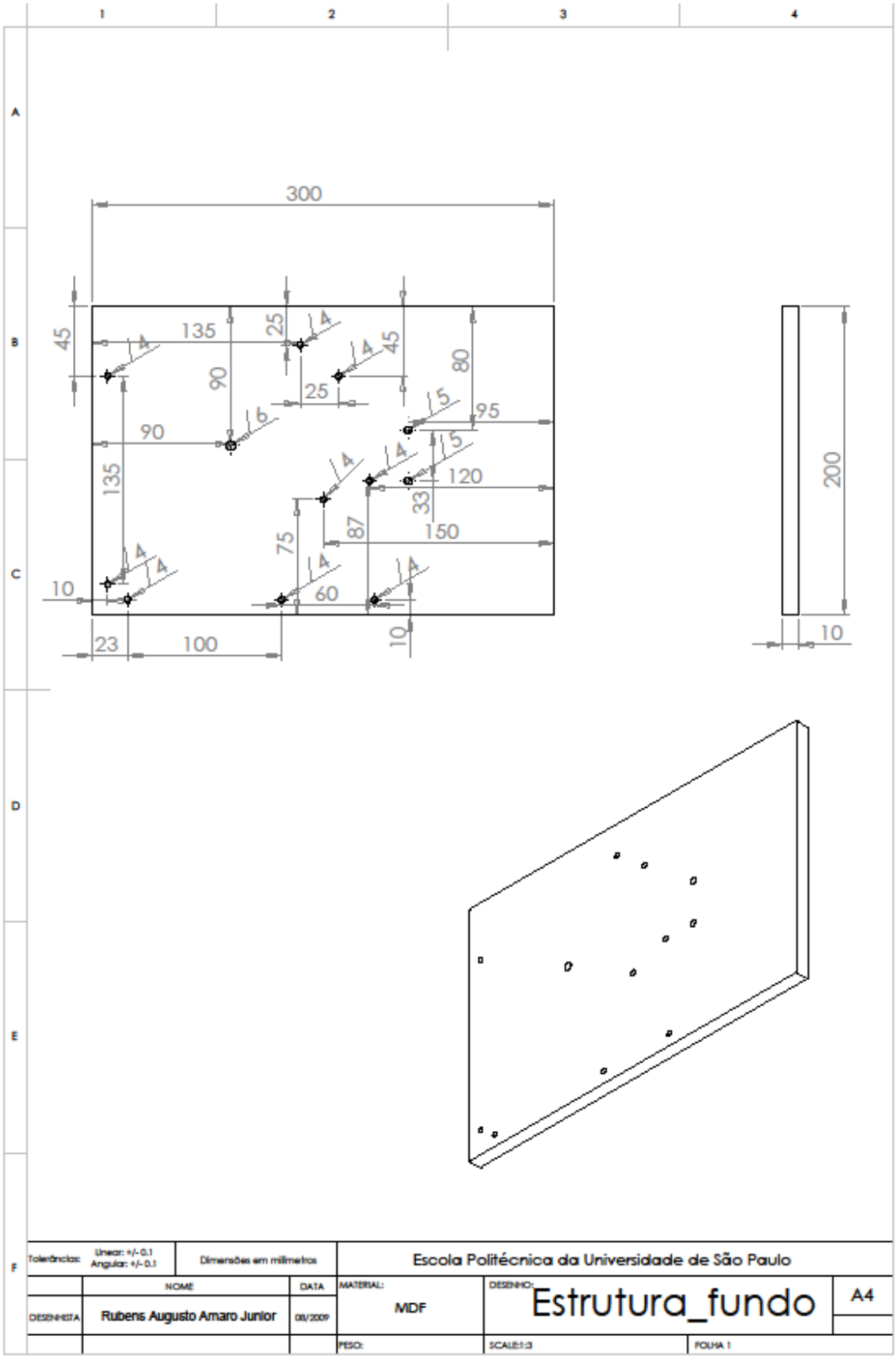

1
2
3
4
A
B
C
D
E
F
7
50
R3
6
4
10
20
65
5
Tolerâncias: Linear: +/- 0.1 Angular: +/- 0.1
Dimensões em milímetros
Escola Politécnica da Universidade de São Paulo
NOME
DATA
DESENHISTA
Rubens Augusto Amaro Junior
09/2009
MATERIAL:
Alumínio
DESENHO:
Biela
A4
PESO:
SCALE:1:1
FOLHA 1

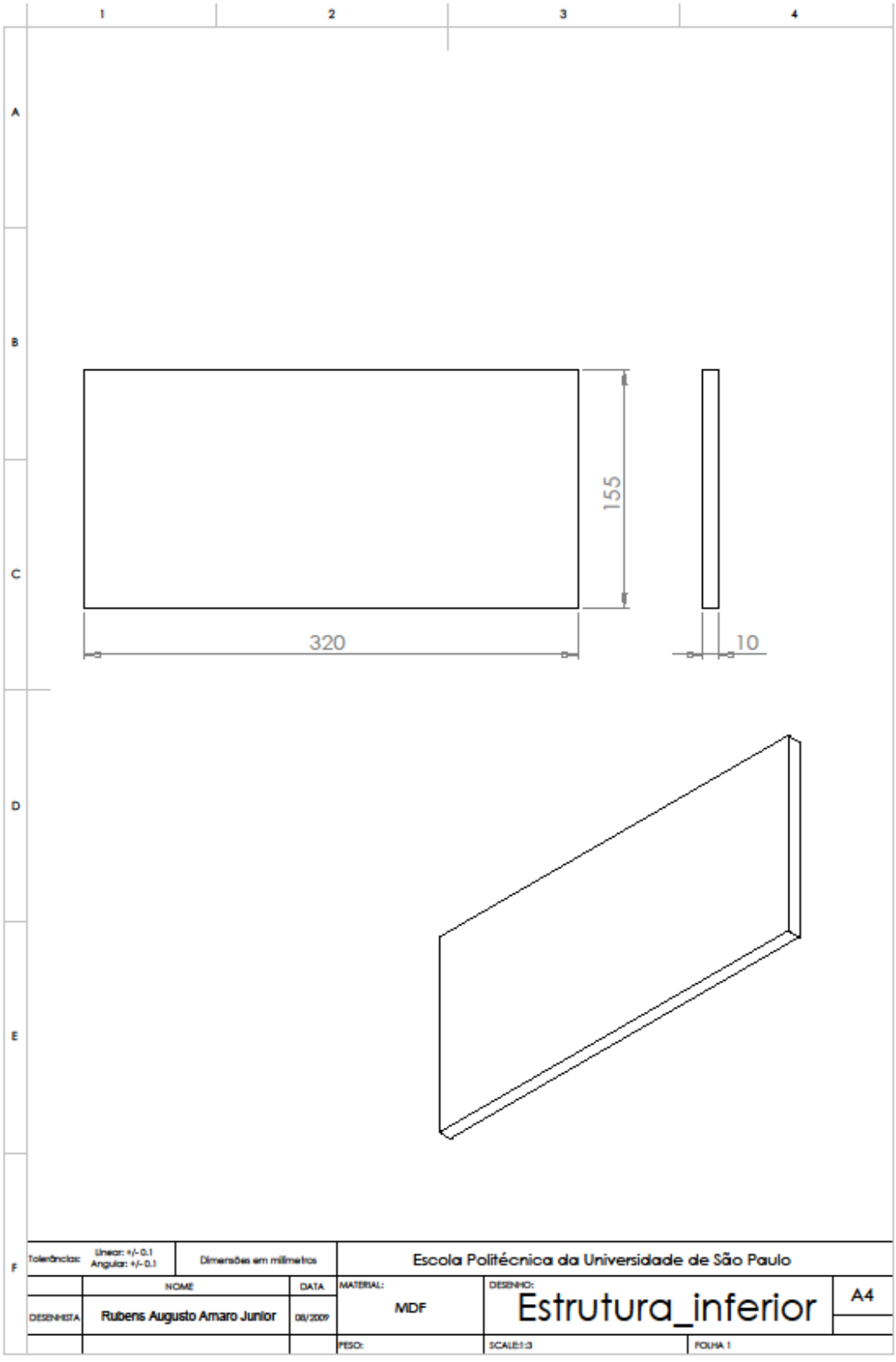

1
2
3
4
A
B
C
D
E
F
4
18,50
15
45
11
Tolerâncias:
Linear: +/- 0.1
Angular: +/- 0.1
Dimensões em milímetros
Escola Politécnica da Universidade de São Paulo
NOME
DATA
DESENHISTA
Rubens Augusto Amaro Junior
09/2009
MATERIAL:
Nylon
DESENHO:
Disco
A4
PESO:
SCALE:1:1
FOLHA 1

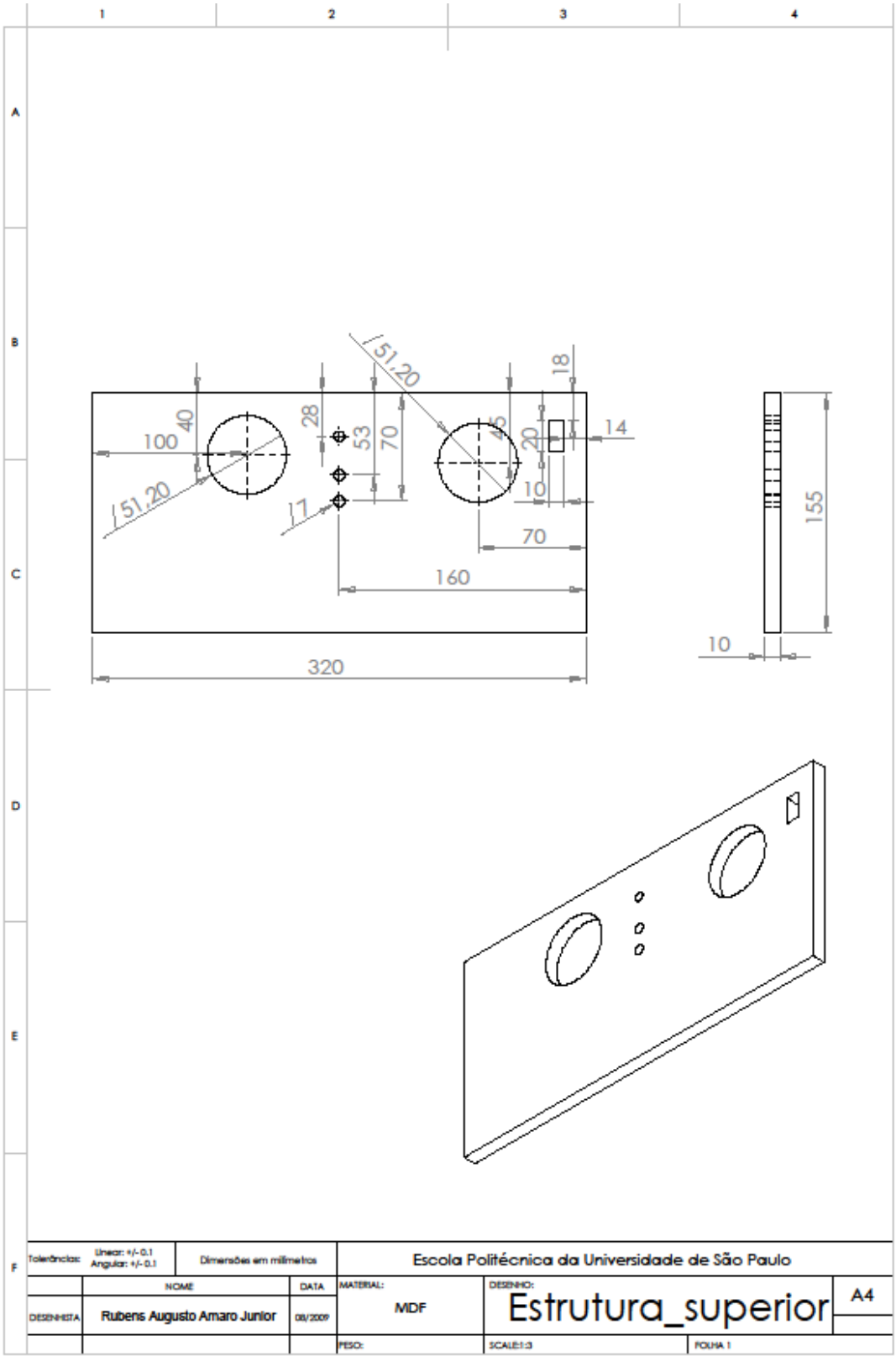

17
5
135
90
R73
90
10
17
10
100
155
170
45
Tolerâncias: Linear: +/- 0.1 Angular: +/- 0.1
Dimensões em milímetros
Escola Politécnica da Universidade de São Paulo
NOME
DATA
MATERIAL:
DESENHO:
A4
DESENHISTA
Rubens Augusto Amaro Junior
09/2009
MDF
Estrutura_alimentador
PESO:
SCALE:1:2
FOLHA 1

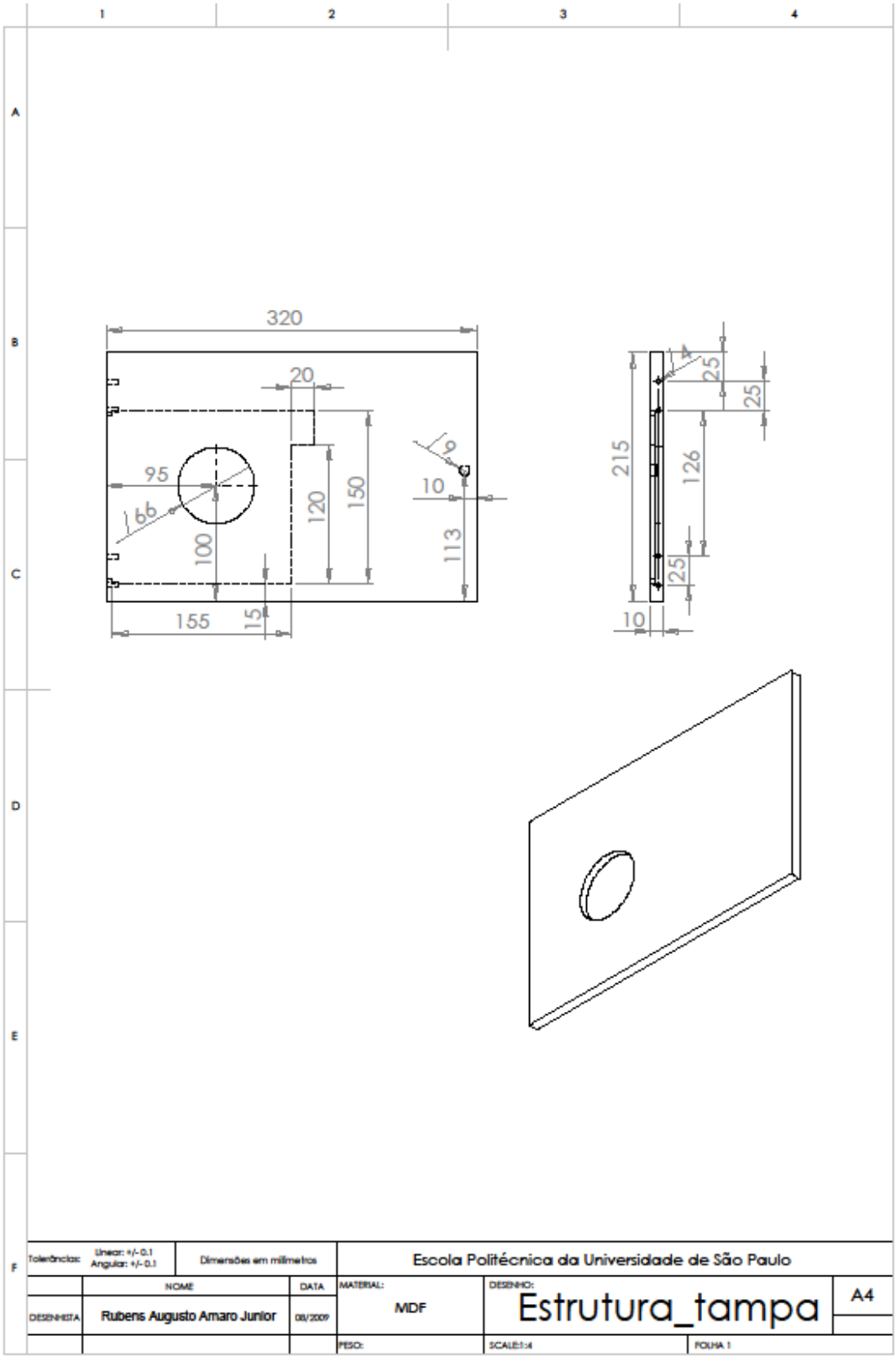

1
2
3
4
A
B
C
D
E
F
27
32
4
3
10
105
200
155
10
NOME
DATA
MATERIAL:
DESENHO:
DESENHISTA
Rubens Augusto Amaro Junior
08/2009
MDF
Estrutura_direita
A4
PESO:
SCALE: 1:2
FOLHA 1

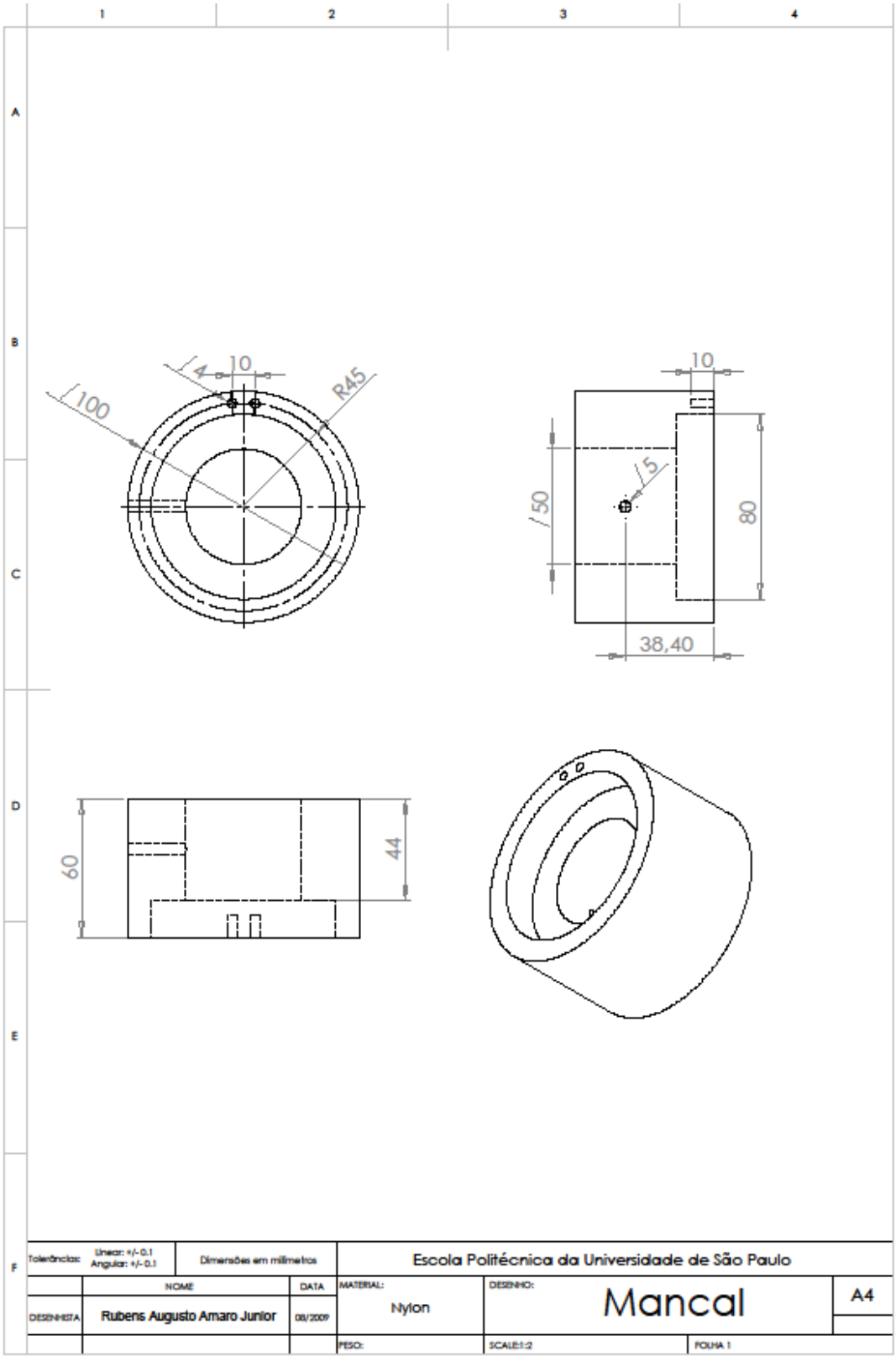

1
2
3
4
A
B
C
D
E
F
155
200
20
30
23
17
25
10
4
55
126
50
24
27
15
10
Tolerâncias: Linear: +/- 0.1 Angular: +/- 0.1
Dimensões em milímetros
Escola Politécnica da Universidade de São Paulo
NOME
DATA
DESENHISTA
Rubens Augusto Amaro Junior
09/2009
MATERIAL:
MDF
DESENHO:
Estrutura_esquerda
A4
PESO:
SCALE:1:2
FOLHA 1

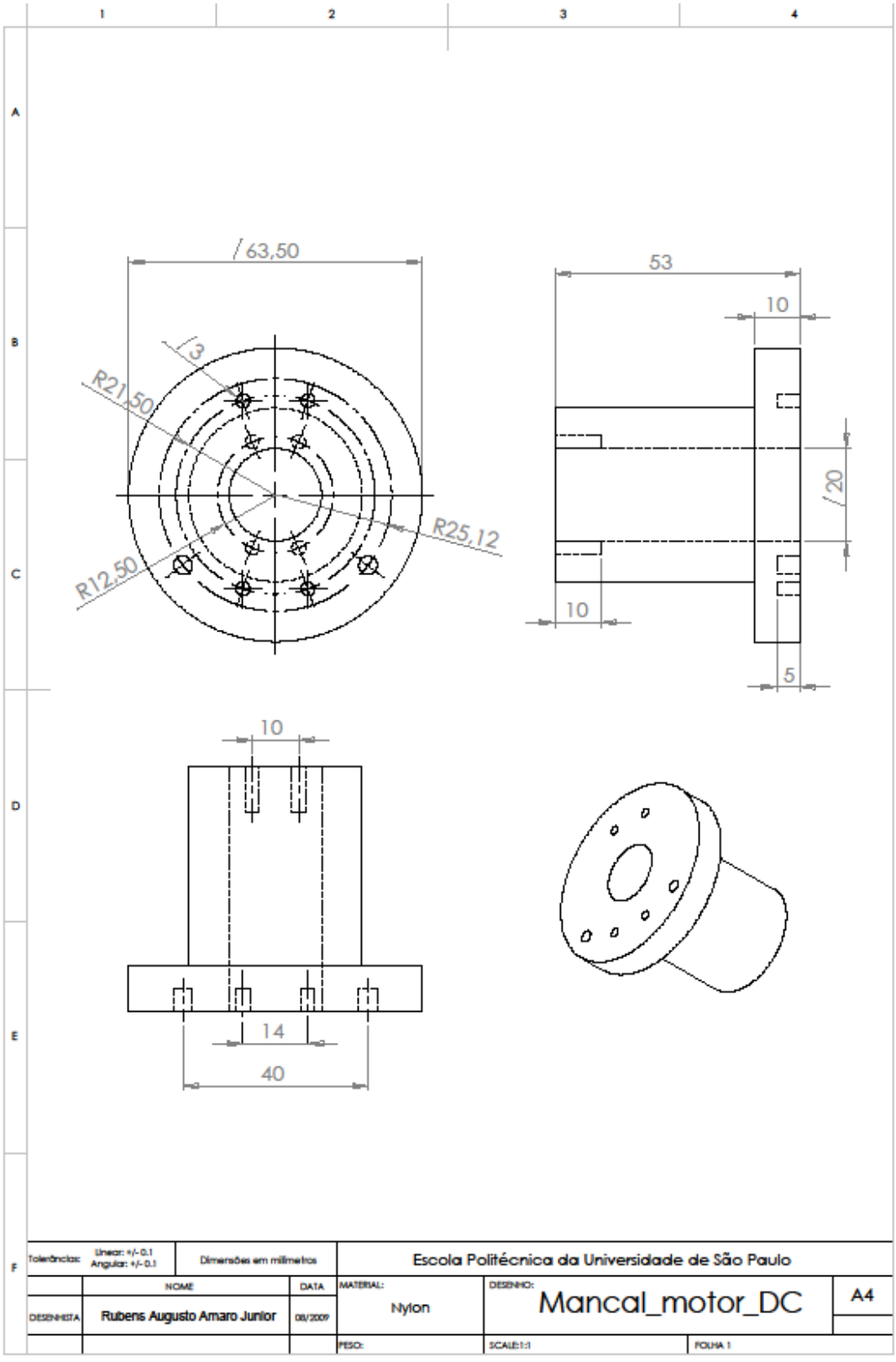

1
2
3
4
A
B
C
D
E
F
300
45
135
25
4
45
80
90
25
95
90
6
5
120
135
33
87
150
75
60
10
23
100
200
10
Tolerâncias: Linear: +/- 0.1 Angular: +/- 0.1
Dimensões em milímetros
Escola Politécnica da Universidade de São Paulo
NOME
DATA
MATERIAL:
DESENHO:
Estrutura_fundo
A4
DESENHISTA
Rubens Augusto Amaro Junior
09/2009
MDF
PESO:
SCALE:1:3
FOLHA 1


1
2
3
4
A
B
C
D
E
F
155
320
10
Tolerâncias:
Linear: +/- 0.1
Angular: +/- 0.1
Dimensões em milímetros
Escola Politécnica da Universidade de São Paulo
NOME
DATA
MATERIAL:
DESENHO:
A4
DESENHISTA
Rubens Augusto Amaro Junior
09/2009
MDF
Estrutura_inferior
PESO:
SCALE1:3
FOLHA 1


1
2
3
4
A
B
C
D
E
F
51,20
18
40
28
53
70
45
20
14
100
51,20
17
10
70
160
320
155
10
Tolerâncias: Linear: +/- 0.1 Angular: +/- 0.1
Dimensões em milímetros
Escola Politécnica da Universidade de São Paulo
NOME
DATA
MATERIAL:
DESENHO:
A4
DESENHISTA
Rubens Augusto Amaro Junior
09/2009
MDF
Estrutura_superior
PESO:
SCALE:1:3
FOLHA 1


1
2
3
4
A
B
C
D
E
F
320
20
95
66
100
120
150
9
10
113
155
15
215
25
25
126
25
10
Tolerâncias: Linear: +/- 0.1 Angular: +/- 0.1
Dimensões em milímetros
Escola Politécnica da Universidade de São Paulo
NOME
DATA
DESENHISTA
Rubens Augusto Amaro Junior
09/2009
MATERIAL:
MDF
DESENHO:
Estrutura_tampa
A4
PESO:
SCALE:1:4
FOLHA 1


1
2
3
4
A
B
C
D
E
F
4
10
R45
100
10
5
50
80
38,40
60
44
Tolerâncias: Linear: +/- 0.1 Angular: +/- 0.1
Dimensões em milímetros
Escola Politécnica da Universidade de São Paulo
NOME
DATA
DESENHISTA
Rubens Augusto Amaro Junior
09/2009
MATERIAL:
Nylon
DESENHO:
Mancal
A4
PESO:
SCALE1:2
FOLHA 1


1
2
3
4
A
B
C
D
E
F
⌀63,50
⌀3
R21,50
R25,12
R12,50
53
10
⌀20
10
5
10
14
40
Tolerâncias: Linear: +/- 0.1 Angular: +/- 0.1
Dimensões em milímetros
Escola Politécnica da Universidade de São Paulo
NOME
DATA
DESENHISTA
Rubens Augusto Amaro Junior
09/2009
MATERIAL:
Nylon
DESENHO:
Mancal_motor_DC
A4
PESO:
SCALE:1:1
FOLHA 1

APÊNDICE B – PROGRAMAÇÃO DO MICROCONTROLADOR

```
#include <16f877a.h>
#use delay(clock=4000000)
#fuses HS,NOWDT,PUT,NOLVP
#priority rda, timer0, timer2

/********************** Variáveis de Status ************************/
int valor = 3;                          // alteração da frequencia de interrupção
int stop = 0;                           // Condição máxima de Execução
int ini_carga = 0;                      // Condição de Carga
int ini_p0 = 0;                         // Condição de Posicionamento Inicial
int flag_ini = 0;
int flag_posigual = 0;

char stat_tempo [10] = "Atempo--B";             // Char de Status de Decremento
de tempo
char stat_quant [10] = "Aquant--B";     // Char de Status de Decremento de
quantidade
char stat_ncarga [10] = "AnchB";        // Char de Status de Sem Carga
char stat_ycarga [10] = "AychB";        // Char de Status de Com Carga

/********************** Variáveis Comunicação Serial
**********************/

int32 sensorcarga = 0;                  // Variável que conta tempo sem bolinha
char lido = 'z';                    // Caracter lido
int leitura = 0;                    // Situação de Leitura (0 = Não ler; 1 = Início da
Leitura; 2 = Fim da Leitura)
int ind_read = 0;                       // índice de leitura de posição
int okleitura = 1;                      // Habilita Leitura da Serial
int npos = 0;                           // Número Total de Posições

int quantauto = 0;                      // quantidade de passos
int tempo = 0;                          // tempo
int modotempo = 0;                      // define modo tempo ou quantidade
int32 tempo_int = 0;                    // contagem de interrupções para tempo

int m = 0;                              // flag - modo
int p = 0;                              // flag - posição
int fl = 0;                             // flag - frequencia de lançamento
int q = 0;                              // flag - quantidade de bolinhas
int t = 0;                              // flag - tempo

/********************** Variáveis de Acionamento dos Motores
**********************/

int realimentador = 1;                  // habilita movimento do realimentador
int pescoco = 1;                        // habilita movimento do pescoco
static boolean led;
```

```
int32 conta = 0;                        // Reduz Freqüência da Interrupção
float fint = 1000;                      // Período da Interrupção do Timer 0
int modo = 0;                           // inicia com modo manual
int32 nint = 0;                         // Número de Interrupções
int ind_pos = 0;                        // Índice da Posição a ser Movimentada
int posiguais = 1;                      // Posições Consecutivas iguais

// CC
long int vel_lan[10];                   // velocidade de lançamento

// Passo
int quantman = 0;                       // quantidade de bolinhas
float freq_lan[10];                     // frequencia de lançamento
int sentido[10];                        // Sentido de Rotação dos Motores
long int ang_lan[10];                   // Vetor de Posições desejadas
long int ang_lan_atual[10];             // Vetor de Posição atual
signed long int dif[10];                        // Diferença entre Posições
desejada e atual
int pulso[2];                           // Variável de Pulso aos Motores

/********************** Variáveis Auxiliares **********************/
int j = 0;

/*****************Configuração da Comunicação
Serial**********************/

struct rcsta_reg
  {
    int rx9d : 1;
    int oerr : 1;
    int ferr : 1;
    int aden : 1;
    int cren : 1;
    int sren : 1;
    int rx9  : 1;
    int spen : 1;
  } rcsta;

struct txsta_reg
  {
    int tx9d : 1;
    int trmt : 1;
    int brgh : 1;
    int xxx  : 1;
    int sync : 1;
    int txen : 1;
    int tx9  : 1;
    int csrc : 1;
```

```
   } txsta;

// define os endereços das variáveis
#locate rcsta = 0x18
#locate txreg = 0x19
#locate rxreg = 0x1a
#locate txsta = 0x98
#locate spbrg = 0x99
#byte r_pir1  = 0x0c   // define o registrador r_pir1
#bit flag_rc = r_pir1.5   // define o flag_rc
#byte r_pie1  = 0x8c   // define o registrador r_pie1
#bit flag_rcie = r_pie1.5   // define o flag_rcie
#byte intcon  = 0x0b   // define o registrador intcon
#bit flag_peie = intcon.6   // define o flag_peie
#bit flag_gie = intcon.7   // define o flag_gie
#bit flag_txie = r_pie1.4 // define o flag_txie

// definições utilizadas nas funções
void usart_inicializa ( int vel, boolean brgh )
{
        txsta.brgh = brgh;   // seleciona o modo do gerador de baud rate
        spbrg = vel;        // configura o gerador de baud rate

        // configura os pinos da USART como entradas !!!!!
        input (pin_c7);

        txsta.sync = 0;                 // seleciona o modo assíncrono
        rcsta.spen = 1;                 // habilita a USART
        txsta.tx9 = 0;                  // seleciona o modo de 8 bits
        txsta.txen = 1;                 // ativa o transmissor da USART
        rcsta.cren = 1;                 // modo de recepção contínua
        flag_peie = 1;                  // ativa dispositivos de interrupções
        flag_gie = 1;
        flag_rcie = 1;                  //ativa interrupção de recepção
        flag_txie = 1;                  //ativa interrupção de transmissão
} // usart_inicializa

/********************** Conversão char para int **********************/
char inttochar (int inteiro)
{
  switch (inteiro)
  {
    case 1: return '1';
    case 2: return '2';
    case 3: return '3';
    case 4: return '4';
    case 5: return '5';
    case 6: return '6';
```

```
      case 7: return '7';
      case 8: return '8';
      case 9: return '9';
      case 0: return '0';
    } // switch
} // inttochar

int chartoint (char charac)
{
  switch (charac)
  {
    case '1': return 1;
    case '2': return 2;
    case '3': return 3;
    case '4': return 4;
    case '5': return 5;
    case '6': return 6;
    case '7': return 7;
    case '8': return 8;
    case '9': return 9;
    case '0': return 0;
  } // switch
} // chartoint

/********************** Função de Reinicialização **********************/

void reinicializar () {
        stop = 1;                       // Pára execução
        ind_read = 0;                   // Inicializa índice de Armazenamento na
leitura
        npos = 0;
        ind_pos = 0;                    // Inicializa índice de leitura das
posições
        sensorcarga = 0;                // Zera o tempo da contagem sem
bolinha
        lido = 'z';                     // Variável de leitura da serial
        quantman = 0;                   // Zera quantidade de bolinhas a serem
lançadas
        okleitura = 0;                  // Encerra Leitura da Serial

/* Variáveis no modo tempo */
        tempo = 0;
        tempo_int = 0;
        modotempo = 0;

/*Variáveis no modo automático*/
        quantauto = 0;
```

```
	set_pwm1_duty (0);			// Zera velocidade de Lançamento
} // reinicializar

/********************** Leitura de Posição **********************/

void read_pos (char pos, int indice) {
	switch (pos){
		case '1':	ang_lan[indice] = 0;
				vel_lan[indice] = 0;
				break;
		case '2':	ang_lan[indice] = 100;
				vel_lan[indice] = 0;
				break;
		case '3':	ang_lan[indice] = 200;
				vel_lan[indice] = 0;
				break;
		case '4':	ang_lan[indice] = 300;
				vel_lan[indice] = 0;
				break;
		default:
				break;
	} // switch
} // read_pos

/********************** Interrupção de Envio **********************/
#int_tbe
void tbe_isr()
{
void usart_transmite (char *dados)
{
int i;
		for (i = 0; dados[i] != '\0';i++) {
			while (!txsta.trmt);	// aguarda o buffer de transmissão
esvaziar
				txreg = dados[i];	// coloca novo caractere para
transmissão
		}
}
}

/********************** Interrupção da Leitura **********************/

#int_rda
void rda_isr()
{
	if (rcsta.oerr) {
		rcsta.cren = 0;
		rcsta.cren = 1;
```

```
        } // if
        while (!flag_rc);
        lido = rxreg;
        if (lido == 'A') {
                leitura = 1;
                okleitura = 1;                          // Condição para Reinício de
Leitura
                nint = 0;                               // inicializa contagem da
interrupção
        } // if
        else if (lido == 'B') leitura = 2;
        if (leitura == 1 || leitura == 2 && okleitura == 1){
                switch (lido)
                {
                        case 'i' :      reinicializar();
                                        break;
                        case 'm' :      m = 1;                  // habilita leitura de modo
                                        break;
                case 'p' :  p = 1;                      // habilita leitura de posição
                            break;
                case 'f' :  fl = 1;                     // habilita leitura de freqüência de
lançamento
                            break;
                case 'q' :  q = 1;                      // habilita leitura de quantidade
de bolinhas/seqüências
                            break;
                case 't' :  t = 1;                      // habilita leitura de tempo
                            break;
                case 's' :  reinicializar();                            // STOP
                            break;
                case 'B':   npos = ind_read;
                            if (modo == 1) ang_lan [npos + 1] = ang_lan_atual [npos +
1];
                            break;
                default :    if (m == 1) {
                                m = 0;                  // desabilita leitura de
modo
                                stop = 0;               // desabilita STOP
                                ind_pos = 0;
                                modo = chartoint(lido); // leitura do modo
                                ind_read = 0;           // Inicializa ìndice de
leitura
                            } // if
                           else if (p == 1) {
                                p = 0;                  // desabilita leitura de
posição
                                if (modo == 0)          posiguais = 1; // Posições
Consecutivas Iguais
```

```
                        else if (modo == 1)    ang_lan_atual[ind_read+1] =
ang_lan[ind_read];              // Atualiza Vetor de Posição Atual
                        switch (lido){
                                case '1':       ang_lan[ind_read] = 0;
                                                vel_lan[ind_read] = 0;
                                                break;
                                case '2':       ang_lan[ind_read] = 100;
                                                vel_lan[ind_read] = 100;
                                                break;
                                case '3':       ang_lan[ind_read] = 200;
                                                vel_lan[ind_read] = 300;
                                                break;
                                case '4':       ang_lan[ind_read] = 300;
                                                vel_lan[ind_read] = 400;
                                                break;
                                default:        break;
                        } // switch
                        dif[ind_read] = ang_lan[ind_read] -
ang_lan_atual[ind_read]; // Calcula DELTA S desejado
                        if (dif[ind_read] > 0) {
                                sentido[ind_read] = 0;                  // Se
DELTA S > 0 Movimento no sentido 0
                        }// if
                        else if (dif[ind_read] < 0) {
                                sentido[ind_read] = 1;                  // Se
DELTA S < 0 Movimento no sentido 1
                        } // else if
                        set_pwm1_duty (vel_lan[0]);          // Velocidade de
Lançamento da 1ª Posição
                } // if
                else if (fl == 1) {
                        fl = 0;
                        switch (lido){          // Freqüências de Lançamento
                                case '1':       freq_lan[ind_read] = 0.5;
                                                break;
                                case '2':       freq_lan[ind_read] = 1;
                                                break;
                                case '3':       freq_lan[ind_read] = 2;
                                                break;
                                case '4':       freq_lan[ind_read] = 4;
                                                break;
                                default:        break;
                        } // switch
                        ind_read++;             // Após leitura da Freqüência de
Lançamento, incrementa índice de leitura
                } // else if
                else if (q == 1) {
                        q = 0;          // Desabilita leitura de quantidade
```

```
/* MODO TEMPO*/
                    modotempo = 0;                  // Modo de Quantidade
                    if (modo == 0) {                // MODO MANUAL
                        switch (lido){
                            case '1':   quantman = 1;
                                        break;
                            case '2':   quantman = 3;
                                        break;
                            case '3':   quantman = 5;
                                        break;
                            case '4':   quantman = 10;
                                        break;
                            default:    break;
                        } // switch
                    } // if
/* MODO AUTO*/
                    else if (modo == 1) {           // MODO
AUTOMÁTICO
                        quantman = 1;
                        switch (lido){
                            case '1':   quantauto = 1;
                                        break;
                            case '2':   quantauto = 2;
                                        break;
                            case '3':   quantauto = 3;
                                        break;
                            case '4':   quantauto = 4;
                                        break;
                            default:    break;
                        } // switch
                    } // else if
                } // if
/* MODO TEMPO */
                else if (t == 1) {
                    t = 0;
                    modotempo = 1;
                    tempo_int = 0;
                    switch (lido){
                        case '1':   tempo = 60 + 10;
                                    break;
                        case '2':   tempo = 120 + 10;
                                    break;
                        case '3':   tempo = 180 + 10;
                                    break;
                        case '4':   tempo = 240 + 10;
                                    break;
                        default:    tempo = 0 + 10;
                                    break;
```

```
                    }
                } // switch
        } // if leitura
        if (leitura == 2) leitura = 0;
} // rda_isr

/********************** Interrupção Timer 0 **********************/

#int_timer0
void trata_t0 ()
{
  // reinicia o timer 0 em 255
  set_timer0(230+get_timer0());
  conta++;                              // Redução na freqüência da Interrupção
  nint ++;
  // se já ocorreram as interrupções
  if (conta == valor)
  {
        conta = 0;
                if ((ang_lan[ind_pos] != ang_lan_atual[ind_pos]) || ini_p0 == 0) {
                        pescoco = 1;
                        flag_posigual = 0;
                }
                else {
                        pescoco = 0;
                        if (ang_lan [ind_pos] == ang_lan_atual [ind_pos] && ini_p0
== 1) flag_posigual = 1;
                }
                if ((quantman > 0 && nint > (fint - 800)/freq_lan[ind_pos] &&
posiguais == 1) || ini_carga == 0) realimentador = 1;
                else realimentador = 0;
                // MOVIMENTAÇÃO DO PESCOÇO_ CONDIÇÕES: DELTA S
PARA A POSIÇÃO DESEJADA OU POSICIONAMENTO INICIAL
                if (realimentador == 1) {
                        if (pulso[1] == 10) pulso[1] = 18;
pulso[1]--;
                        switch (pulso[1]) {
                case 11:
                                                output_bit (68,1);
                        output_bit (69,0);
                        output_bit (70,0);
                        output_bit (71,1);
                         break;
                 case 12:
                        output_bit (68,0);
                        output_bit (69,0);
                        output_bit (70,0);
                        output_bit (71,1);
```

```
        break;
case 13:
                                output_bit (68,0);
        output_bit (69,0);
        output_bit (70,1);
        output_bit (71,1);
        break;
case 14:
        output_bit (68,0);
        output_bit (69,0);
        output_bit (70,1);
        output_bit (71,0);
        break;
        case 15:
                output_bit (68,0);
        output_bit (69,1);
        output_bit (70,1);
        output_bit (71,0);
         break;
case 16:
        output_bit (68,0);
        output_bit (69,1);
        output_bit (70,0);
        output_bit (71,0);
        break;
case 17:
        output_bit (68,1);
        output_bit (69,1);
        output_bit (70,0);
        output_bit (71,0);
        break;
case 18:
        output_bit (68,1);
        output_bit (69,0);
        output_bit (70,0);
        output_bit (71,0);
        break;
default: break;
        } // switch
}  // Movimento do Realimentador
if (pescoco == 1){
        switch (sentido[ind_pos]) {
case 0:
        if (pulso[0] == 18) pulso[0] = 10;
        pulso[0]++;
                        if (ini_p0 == 1) ang_lan_atual[ind_pos]++;
        break;
case 1:
```

```
            if (pulso[0] == 11) pulso[0] = 19;
            pulso[0]--;
            if (ini_p0 == 1) ang_lan_atual[ind_pos]--;
            break;
default: break;
            } // sentido de movimento
            switch (pulso[0]) {
case 11:
            output_bit (64,1);
            output_bit (65,0);
            output_bit (66,0);
            output_bit (67,1);
             break;
 case 12:
            output_bit (64,0);
            output_bit (65,0);
            output_bit (66,0);
            output_bit (67,1);
            break;
case 13:
            output_bit (64,0);
            output_bit (65,0);
            output_bit (66,1);
            output_bit (67,1);
            break;
case 14:
            output_bit (64,0);
            output_bit (65,0);
            output_bit (66,1);
            output_bit (67,0);
            break;
case 15:
            output_bit (64,0);
            output_bit (65,1);
            output_bit (66,1);
            output_bit (67,0);
             break;
case 16:
            output_bit (64,0);
            output_bit (65,1);
            output_bit (66,0);
            output_bit (67,0);
            break;
case 17:
            output_bit (64,1);
            output_bit (65,1);
            output_bit (66,0);
            output_bit (67,0);
```

```
                        break;
                case 18:
                        output_bit (64,1);
                        output_bit (65,0);
                        output_bit (66,0);
                        output_bit (67,0);
                        break;
                default: break;
                        } // switch
                } //  Movimentação do pescoço
        } // if conta == valor
} // trata_t0

/***** TIMER 1 *****/

#int_timer1
void trata_t1 ()
{
  set_timer1(65200+get_timer1());
  if (flag_posigual == 1){
/* Modo Automático */
        if (modo == 1) {
                if (ind_pos == npos && leitura != 1){                    // Se as
posições se igualaram e não está lendo.
                        output_bit(pin_c0,1);
                        if (quantauto > 0) {
                                quantauto --;
                                usart_transmite (stat_quant);
                                ang_lan_atual[0] = ang_lan_atual[npos + 1];
                                dif[0] = ang_lan [0] - ang_lan_atual [0];
                                if (dif[0] > 0) sentido[0] = 0;
                                else sentido[0] = 1;
                                set_pwm1_duty (vel_lan[0]);
                                for (j = 1; j <= npos; j++) ang_lan_atual[j] = ang_lan
[j-1];
                        } // if
                        else {
/* MODO TEMPO */
                                if (modotempo == 0) reinicializar ();
                                else if (modotempo == 1) {
                                for (j = 1; j < 10; j++) ang_lan_atual[j] = ang_lan [j-1];
                        }// else if
                        npos = 0;
                } // else
                ind_pos = 0;
        } // if
        else if (ind_pos != npos && leitura != 1) {
                if (quantman == 0){
```

```
                        quantman = 1;
                        ind_pos++;
                        set_pwm1_duty (vel_lan[ind_pos]);
                } // if
        } // else
        output_bit(pin_c1,1);
} // if
}
}

/********************** Main **********************/

void main()
{
        char envio_inicio [10] = "AsokB";
// habilita as interrupções global, timer 0 e recepção
        enable_interrupts(global | int_timer0);
        enable_interrupts (int_timer1);
        enable_interrupts(int_rda);

// configura o timer 0 para clock interno e prescaler dividindo por 2
        setup_timer_0 ( RTCC_INTERNAL | RTCC_DIV_2 );
        set_timer0(230); // inicia o timer 0 em 230
        setup_timer_1 ( T1_INTERNAL | T1_DIV_BY_1 );
        set_timer1(65000); // inicia o timer 0 em 65000


/*********** Configuração do PWM ***********/
        setup_timer_2 (T2_DIV_BY_16, 248, 1);  // timer 2 = 20 khz
        setup_ccp1 (ccp_pwm);  // configura CCP1 para modo PWM
        set_pwm1_duty (0);  // configura o ciclo ativo em 0 (desligado)

/*********** Inicialização de Comunicação e de LCD ***********/
        usart_inicializa (25,1);

/*********** Inicialização de Variáveis ***********/
        sentido[0] = 0;
        sentido[1] = 1;
        pulso[0] = 10;
        pulso[1] = 18;
        vel_lan[0] = 0;
        ang_lan [0] = 0;
        ang_lan_atual[0] = 0;
        output_bit (pin_b7,1);

                while (true){
                /**** ENTRADAS****/
        if (input (pin_b3) == 1) {
```

```
            if (quantman > 0 && nint > 500)      {

                  quantman --;   // Decrementa quantidade
                  nint = 0;                  // Reinicia Contagem de Interrupções

                  usart_transmite (stat_quant);
            } // if
      } // Sensor de Lançamento

      // Leitura do Sensor de Carga de Bolinhas
      if (input (pin_b2) == 0){
            sensorcarga ++;
            if (sensorcarga > 3000) {
                  ini_carga = 0;           //acionar depois
                  usart_transmite (stat_ncarga);
            } // if
      } // Sensor de Carga
      else {
            ini_carga = 1;
            sensorcarga = 0;
            usart_transmite (stat_ycarga);
      }

      // Leitura do Sensor de Posição Inicial
      if (input(pin_b1) == 1) ini_p0 = 1;
      if ((ini_p0 == 1 && ini_carga == 1) && flag_ini == 0) {
            flag_ini = 1;
            ang_lan_atual[0] = 0;
            valor = 1;
            usart_transmite (envio_inicio);
      }
      }
}
```

APÊNDICE C – PROGRAMAÇÃO DA INTERFACE

```
/*
 * HMI.java
 *
 * Created on 30 de Maio de 2009, 15:40
 */

package controle;

import controle.*;
import java.awt.event.KeyEvent;
import java.io.IOException;
import javax.swing.plaf.FileChooserUI;

/**
 *
 * @author  Robson
 */
public class HMI extends javax.swing.JFrame {

    /** Creates new form HMI */
    public HMI() {
        c=new Controle(this);
        comm=new Comunicacao("COM4",c);
        c.setComm(comm);
        comm.start();
        initComponents();
        AutoPanel.setVisible(false);
        quantautoComboBox.setVisible (false);
        yinicialLabel.setVisible (false);
        ycargaLabel.setVisible (false);
        statquantLabel.setVisible(false);
        valorquantLabel.setVisible(false);
        stattempoLabel.setVisible (false);
        valortempoLabel.setVisible (false);
        inicialLabel.setText("Inicializando!!");
        cargaLabel.setText("Descarregado!!");

        Mesa.add(P1);
        P1.setBounds(30, 300, 20, 20);
        Mesa.add(P2);
        P2.setBounds(70, 300, 20, 20);
        Mesa.add(P3);
        P3.setBounds(150, 300, 20, 20);
        Mesa.add(P4);
        P4.setBounds(190, 300, 20, 20);
        this.atualizaLog("Pressione o Botão Start.");
    }
```

```
void resetar (){
   c.ind_seq = 0;
   c.ComandoFinal = "";
   c.envio = 0;
   c.addPasso = 0;
   c.nPasso = 0;
   c.seqpas = "";
   c.seqfreq = "";

   for (int i = 0; i < 11; i++){
      c.codauto[i] = "";
   }
   proglist.removeAll();
   loglist.removeAll();
}

void atualizaLog(String string) {
   loglist.add(loglist.getItemCount()+": "+string);
   loglist.select(loglist.getItemCount()-1);
}

void atualizaProg (int topico) {
   proglist.removeAll();
   proglist.add("Passo "+c.nPasso);
   proglist.add("Sequência: "+ c.seqpas);
   proglist.add("Frequência de Lançamento: "+ c.seqfreq);
   if (c.quant.equals("-")){
      proglist.add("Tempo: " + c.tempo);
   }
   else {
      proglist.add("Quantidade: " + c.quant);
   }
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

   fileChooser = new javax.swing.JFileChooser();
   buttonGroup1 = new javax.swing.ButtonGroup();
   MainPanel = new javax.swing.JPanel();
   P1 = new javax.swing.JButton();
   P2 = new javax.swing.JButton();
   P4 = new javax.swing.JButton();
```

```
        P3 = new javax.swing.JButton();
        Mesa = new javax.swing.JLabel();
        StartStop = new javax.swing.JLabel();
        startstopToggleButton = new javax.swing.JToggleButton();
        manualPanel = new javax.swing.JPanel();
        jLabel3 = new javax.swing.JLabel();
        freqComboBox = new javax.swing.JComboBox();
        bolinhaLabel = new javax.swing.JLabel();
        autoRadioButton = new javax.swing.JRadioButton();
        manualRadioButton = new javax.swing.JRadioButton();
        jLabel2 = new javax.swing.JLabel();
        jPanel5 = new javax.swing.JPanel();
        statquantLabel = new javax.swing.JLabel();
        valorquantLabel = new javax.swing.JLabel();
        ninicializaLabel = new javax.swing.JLabel();
        yinicialLabel = new javax.swing.JLabel();
        inicialLabel = new javax.swing.JLabel();
        ncargaLabel = new javax.swing.JLabel();
        ycargaLabel = new javax.swing.JLabel();
        cargaLabel = new javax.swing.JLabel();
        stattempoLabel = new javax.swing.JLabel();
        valortempoLabel = new javax.swing.JLabel();
        jPanel3 = new javax.swing.JPanel();
        loglist = new java.awt.List();
        quantLabel = new javax.swing.JLabel();
        quantmanComboBox = new javax.swing.JComboBox();
        quantautoComboBox = new javax.swing.JComboBox();
        AutoPanel = new javax.swing.JPanel();
        inserirpassoButton = new javax.swing.JButton();
        repetirButton = new javax.swing.JButton();
        jLabel11 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();
        jPanel4 = new javax.swing.JPanel();
        proglist = new java.awt.List();
        jLabel13 = new javax.swing.JLabel();
        tempoComboBox = new javax.swing.JComboBox();
        resetarButton = new javax.swing.JButton();
        jLabel4 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("LANÇADOR 2009");
        setBounds(new java.awt.Rectangle(0, 0, 500, 500));
        setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        setForeground(java.awt.Color.white);


MainPanel.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.Bo
rderFactory.createEtchedBorder(), "Main"));
```

```
        MainPanel.setAlignmentX(0.0F);
        MainPanel.setAlignmentY(0.0F);
        MainPanel.setMinimumSize(new java.awt.Dimension(280, 450));
        MainPanel.setPreferredSize(new java.awt.Dimension(267, 470));

        P1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/controle/icones/1.JPG"))); //
NOI18N
        P1.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1, 1, 1));
        P1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                P1MouseClicked(evt);
            }
        });

        P2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/controle/icones/2.JPG"))); //
NOI18N
        P2.setBorder(javax.swing.BorderFactory.createCompoundBorder());
        P2.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                P2MouseClicked(evt);
            }
        });

        P4.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/controle/icones/4.JPG"))); //
NOI18N
        P4.setBorder(javax.swing.BorderFactory.createCompoundBorder());
        P4.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                P4MouseClicked(evt);
            }
        });

        P3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/controle/icones/3.JPG"))); //
NOI18N
        P3.setBorder(javax.swing.BorderFactory.createCompoundBorder());
        P3.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                P3MouseClicked(evt);
            }
        });

        Mesa.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/controle/icones/Mesa_Posições.JP
G"))); // NOI18N
```

```
        StartStop.setFont(new java.awt.Font("Tahoma", 0, 18));
        StartStop.setForeground(new java.awt.Color(255, 153, 51));
        StartStop.setText("Start/ Stop");

        startstopToggleButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/controle/icones/start_stop.JPG")));
// NOI18N
        startstopToggleButton.addMouseListener(new java.awt.event.MouseAdapter()
{
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                startstopToggleButtonMouseClicked(evt);
            }
        });

        org.jdesktop.layout.GroupLayout MainPanelLayout = new
org.jdesktop.layout.GroupLayout(MainPanel);
        MainPanel.setLayout(MainPanelLayout);
        MainPanelLayout.setHorizontalGroup(

MainPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(MainPanelLayout.createSequentialGroup()
                .addContainerGap()

.add(MainPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEAD
ING)
                    .add(Mesa)
                    .add(MainPanelLayout.createSequentialGroup()

.add(MainPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAI
LING)
                            .add(org.jdesktop.layout.GroupLayout.LEADING,
MainPanelLayout.createSequentialGroup()
                                .add(P1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
20, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                                .add(34, 34, 34)
                                .add(P2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
20, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                            .add(startstopToggleButton,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 44,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))

.add(MainPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEAD
ING)
                            .add(MainPanelLayout.createSequentialGroup()
                                .add(40, 40, 40)
                                .add(P3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
20, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
```

```
                        .add(45, 45, 45)
                        .add(P4, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
20, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                    .add(org.jdesktop.layout.GroupLayout.TRAILING,
MainPanelLayout.createSequentialGroup()
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED,
57, Short.MAX_VALUE)
                        .add(StartStop)
                        .add(30, 30, 30)))))
            .addContainerGap(11, Short.MAX_VALUE))
        );
        MainPanelLayout.setVerticalGroup(

MainPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING,
MainPanelLayout.createSequentialGroup()
                .add(Mesa)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 34,
Short.MAX_VALUE)

.add(MainPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEAD
ING)
                    .add(P1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 20,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(P2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 20,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(P3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 20,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(P4, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 20,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 24,
Short.MAX_VALUE)

.add(MainPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAI
LING)
                    .add(startstopToggleButton)
                    .add(StartStop))
                .add(17, 17, 17))
        );


manualPanel.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.
BorderFactory.createEtchedBorder(), "Manual"));

        jLabel3.setFont(new java.awt.Font("Tahoma", 0, 18));
        jLabel3.setForeground(new java.awt.Color(255, 153, 51));
        jLabel3.setText("Freqüência");
```

```
        freqComboBox.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "0.5", "1", "2", "4" }));

freqComboBox.setBorder(javax.swing.BorderFactory.createCompoundBorder());
        freqComboBox.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                freqComboBoxActionPerformed(evt);
            }
        });

        bolinhaLabel.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/controle/icones/bolinhas.JPG"))); //
NOI18N

        buttonGroup1.add(autoRadioButton);
        autoRadioButton.setFont(new java.awt.Font("Tahoma", 0, 18));
        autoRadioButton.setForeground(new java.awt.Color(255, 153, 51));
        autoRadioButton.setText("Automático");
        autoRadioButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                autoRadioButtonActionPerformed(evt);
            }
        });

        buttonGroup1.add(manualRadioButton);
        manualRadioButton.setFont(new java.awt.Font("Tahoma", 0, 18));
        manualRadioButton.setForeground(new java.awt.Color(255, 153, 51));
        manualRadioButton.setSelected(true);
        manualRadioButton.setText("Manual");
        manualRadioButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                manualRadioButtonActionPerformed(evt);
            }
        });

        jLabel2.setFont(new java.awt.Font("Tahoma", 0, 24));
        jLabel2.setForeground(new java.awt.Color(255, 153, 0));
        jLabel2.setText("Modo");


jPanel5.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.Borde
rFactory.createEtchedBorder(), "Status", javax.swing.border.TitledBorder.CENTER,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Courier New", 1, 24))); // NOI18N
        jPanel5.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        statquantLabel.setFont(new java.awt.Font("Courier New", 1, 18)); // NOI18N
        statquantLabel.setForeground(new java.awt.Color(0, 51, 204));
```

```
        statquantLabel.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
        statquantLabel.setText("Quantidade");
        jPanel5.add(statquantLabel, new
org.netbeans.lib.awtextra.AbsoluteConstraints(30, 120, -1, -1));

        valorquantLabel.setFont(new java.awt.Font("Courier New", 1, 18));
        valorquantLabel.setForeground(new java.awt.Color(0, 51, 204));

valorquantLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        valorquantLabel.setText("10");
        jPanel5.add(valorquantLabel, new
org.netbeans.lib.awtextra.AbsoluteConstraints(170, 120, 50, 20));

        ninicializaLabel.setBackground(new java.awt.Color(255, 0, 0));
        ninicializaLabel.setForeground(new java.awt.Color(255, 0, 0));
        ninicializaLabel.setText("gfsg");
        ninicializaLabel.setOpaque(true);
        jPanel5.add(ninicializaLabel, new
org.netbeans.lib.awtextra.AbsoluteConstraints(30, 40, 20, 20));

        yinicialLabel.setBackground(new java.awt.Color(0, 255, 0));
        yinicialLabel.setOpaque(true);
        jPanel5.add(yinicialLabel, new
org.netbeans.lib.awtextra.AbsoluteConstraints(60, 40, 20, 20));

        inicialLabel.setFont(new java.awt.Font("Courier New", 1, 18));
        inicialLabel.setForeground(new java.awt.Color(0, 51, 204));
        inicialLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        inicialLabel.setText("Inicializando!!");
        jPanel5.add(inicialLabel, new
org.netbeans.lib.awtextra.AbsoluteConstraints(100, 40, -1, -1));

        ncargaLabel.setBackground(new java.awt.Color(255, 0, 0));
        ncargaLabel.setForeground(new java.awt.Color(255, 0, 0));
        ncargaLabel.setText("gfsg");
        ncargaLabel.setOpaque(true);
        jPanel5.add(ncargaLabel, new
org.netbeans.lib.awtextra.AbsoluteConstraints(30, 80, 20, 20));

        ycargaLabel.setBackground(new java.awt.Color(0, 255, 0));
        ycargaLabel.setOpaque(true);
        jPanel5.add(ycargaLabel, new
org.netbeans.lib.awtextra.AbsoluteConstraints(60, 80, 20, 20));

        cargaLabel.setFont(new java.awt.Font("Courier New", 1, 18)); // NOI18N
        cargaLabel.setForeground(new java.awt.Color(0, 51, 204));
        cargaLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        cargaLabel.setText("Descarregado!!");
```

```
        jPanel5.add(cargaLabel, new
org.netbeans.lib.awtextra.AbsoluteConstraints(110, 80, -1, -1));

        stattempoLabel.setFont(new java.awt.Font("Courier New", 1, 18)); // NOI18N
        stattempoLabel.setForeground(new java.awt.Color(0, 51, 204));
        stattempoLabel.setText("Tempo");
        jPanel5.add(stattempoLabel, new
org.netbeans.lib.awtextra.AbsoluteConstraints(30, 150, 80, 20));

        valortempoLabel.setFont(new java.awt.Font("Courier New", 1, 18)); // NOI18N
        valortempoLabel.setForeground(new java.awt.Color(0, 51, 204));

valortempoLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        valortempoLabel.setText("20");
        jPanel5.add(valortempoLabel, new
org.netbeans.lib.awtextra.AbsoluteConstraints(110, 150, 50, 20));


jPanel3.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.Borde
rFactory.createEtchedBorder(), "Log", javax.swing.border.TitledBorder.CENTER,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Courier New", 1, 24))); // NOI18N

        loglist.setBackground(new java.awt.Color(0, 0, 0));
        loglist.setForeground(new java.awt.Color(51, 204, 0));

        org.jdesktop.layout.GroupLayout jPanel3Layout = new
org.jdesktop.layout.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(

jPanel3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel3Layout.createSequentialGroup()
                .addContainerGap()
                .add(loglist, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 290,
Short.MAX_VALUE)
                .addContainerGap())
        );
        jPanel3Layout.setVerticalGroup(

jPanel3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel3Layout.createSequentialGroup()
                .add(loglist, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 80,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
```

```
        quantLabel.setFont(new java.awt.Font("Tahoma", 0, 18));
        quantLabel.setForeground(new java.awt.Color(255, 153, 51));
        quantLabel.setText("Quantidade");

        quantmanComboBox.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "1", "3", "5", "10", "-" }));

quantmanComboBox.setBorder(javax.swing.BorderFactory.createCompoundBorder(
));
        quantmanComboBox.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                quantmanComboBoxActionPerformed(evt);
            }
        });

        quantautoComboBox.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "1", "2", "3", "4", "-" }));
        quantautoComboBox.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                quantautoComboBoxActionPerformed(evt);
            }
        });

        org.jdesktop.layout.GroupLayout manualPanelLayout = new
org.jdesktop.layout.GroupLayout(manualPanel);
        manualPanel.setLayout(manualPanelLayout);
        manualPanelLayout.setHorizontalGroup(

manualPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADIN
G)
            .add(manualPanelLayout.createSequentialGroup()

.add(manualPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEA
DING)
                    .add(manualPanelLayout.createSequentialGroup()
                        .add(14, 14, 14)
                        .add(jLabel2)
                        .add(12, 12, 12)

.add(manualPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEA
DING)
                            .add(manualRadioButton)
                            .add(autoRadioButton)))
                    .add(manualPanelLayout.createSequentialGroup()
                        .add(14, 14, 14)
                        .add(bolinhaLabel)
                        .add(19, 19, 19)
```

```
                    .add(freqComboBox,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 50,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(30, 30, 30)

.add(manualPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEA
DING)
                        .add(quantmanComboBox,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 50,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(quantautoComboBox,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 40,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
                .add(manualPanelLayout.createSequentialGroup()
                    .add(14, 14, 14)
                    .add(jLabel3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
100, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(30, 30, 30)
                    .add(quantLabel)))
            .add(32, 32, 32))
        .add(jPanel3, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .add(org.jdesktop.layout.GroupLayout.TRAILING, jPanel5,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 322, Short.MAX_VALUE)
    );
    manualPanelLayout.setVerticalGroup(

manualPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADIN
G)
        .add(manualPanelLayout.createSequentialGroup()
            .add(10, 10, 10)

.add(manualPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEA
DING)
                .add(jLabel2)
                .add(manualRadioButton)
                .add(manualPanelLayout.createSequentialGroup()
                    .add(30, 30, 30)
                    .add(autoRadioButton)))
            .add(19, 19, 19)

.add(manualPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEA
DING)
                .add(bolinhaLabel)
                .add(quantmanComboBox,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
```

```
.add(manualPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.TR
AILING, false)
                    .add(org.jdesktop.layout.GroupLayout.LEADING, freqComboBox)
                    .add(org.jdesktop.layout.GroupLayout.LEADING,
quantautoComboBox)))
            .add(13, 13, 13)

.add(manualPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEA
DING)
                .add(jLabel3)
                .add(quantLabel))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(jPanel3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(jPanel5, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 190,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .add(12, 12, 12))
        );


AutoPanel.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.Bo
rderFactory.createEtchedBorder(), "Auto"));
        AutoPanel.setAlignmentX(0.0F);
        AutoPanel.setAlignmentY(0.0F);
        AutoPanel.setPreferredSize(new java.awt.Dimension(310, 460));

        inserirpassoButton.setText("INSERIR PASSO");
        inserirpassoButton.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                inserirpassoButtonMouseClicked(evt);
            }
        });

        repetirButton.setText("REPETIR PASSO");
        repetirButton.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                repetirButtonMouseClicked(evt);
            }
        });

        jLabel11.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/controle/icones/Relógio.JPG"))); //
NOI18N

        jLabel12.setFont(new java.awt.Font("Tahoma", 0, 24));
```

```
        jLabel12.setForeground(new java.awt.Color(255, 153, 0));
        jLabel12.setText("Tempo");


jPanel4.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.Borde
rFactory.createEtchedBorder(), "Programação do Treino",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Tahoma", 0, 24))); // NOI18N

        org.jdesktop.layout.GroupLayout jPanel4Layout = new
org.jdesktop.layout.GroupLayout(jPanel4);
        jPanel4.setLayout(jPanel4Layout);
        jPanel4Layout.setHorizontalGroup(

jPanel4Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel4Layout.createSequentialGroup()
                .addContainerGap()
                .add(proglist, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 242,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        jPanel4Layout.setVerticalGroup(

jPanel4Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel4Layout.createSequentialGroup()
                .add(proglist, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
        );

        jLabel13.setFont(new java.awt.Font("Tahoma", 0, 24));
        jLabel13.setForeground(new java.awt.Color(255, 153, 0));
        jLabel13.setText("Min");

        tempoComboBox.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "-", "1", "2", "3", "4" }));
        tempoComboBox.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                tempoComboBoxActionPerformed(evt);
            }
        });

        resetarButton.setText("RESETAR PASSO");
        resetarButton.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                resetarButtonMouseClicked(evt);
```

```
        }
    });

    org.jdesktop.layout.GroupLayout AutoPanelLayout = new
org.jdesktop.layout.GroupLayout(AutoPanel);
    AutoPanel.setLayout(AutoPanelLayout);
    AutoPanelLayout.setHorizontalGroup(

AutoPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
        .add(AutoPanelLayout.createSequentialGroup()

.add(AutoPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEAD
ING)
            .add(AutoPanelLayout.createSequentialGroup()
                .add(29, 29, 29)

.add(AutoPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEAD
ING)
                    .add(org.jdesktop.layout.GroupLayout.TRAILING,
AutoPanelLayout.createSequentialGroup()
                        .add(jLabel11)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED,
31, Short.MAX_VALUE)

.add(AutoPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEAD
ING)
                            .add(jLabel12,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 95,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                            .add(AutoPanelLayout.createSequentialGroup()
                                .add(10, 10, 10)
                                .add(tempoComboBox,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 44,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                                .add(25, 25, 25)
                                .add(jLabel13)))
                        .add(60, 60, 60))

.add(AutoPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAI
LING, false)
                        .add(org.jdesktop.layout.GroupLayout.LEADING,
inserirpassoButton, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .add(org.jdesktop.layout.GroupLayout.LEADING, repetirButton,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .add(org.jdesktop.layout.GroupLayout.LEADING,
resetarButton))))
```

```
                .add(AutoPanelLayout.createSequentialGroup()
                    .addContainerGap()
                    .add(jPanel4, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
            .addContainerGap())
        );
        AutoPanelLayout.setVerticalGroup(

AutoPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(AutoPanelLayout.createSequentialGroup()
                .add(24, 24, 24)

.add(AutoPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEAD
ING, false)
                .add(AutoPanelLayout.createSequentialGroup()
                    .add(jLabel12)
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.add(AutoPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEAD
ING)
                        .add(tempoComboBox,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(jLabel13)))
                .add(jLabel11, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
73, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
            .add(18, 18, 18)
            .add(inserirpassoButton,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 27,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .add(18, 18, 18)
            .add(repetirButton, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
28, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .add(18, 18, 18)
            .add(resetarButton, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
27, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.UNRELATED)
            .add(jPanel4, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .add(109, 109, 109))
        );

        jLabel4.setFont(new java.awt.Font("Tahoma", 0, 36));
        jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
```

```
        jLabel4.setText("Robô Lançador");

        org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(manualPanel, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(jLabel4)
                    .add(layout.createSequentialGroup()
                        .add(MainPanel,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 280,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.UNRELATED)
                        .add(AutoPanel, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
306, Short.MAX_VALUE)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .add(13, 13, 13)
                .add(jLabel4)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.UNRELATED)

.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING, false)
                    .add(org.jdesktop.layout.GroupLayout.LEADING, AutoPanel,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 510, Short.MAX_VALUE)
                    .add(org.jdesktop.layout.GroupLayout.LEADING, MainPanel,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 510, Short.MAX_VALUE)
                    .add(org.jdesktop.layout.GroupLayout.LEADING, manualPanel,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 510, Short.MAX_VALUE))
                .addContainerGap(org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

private void manualRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {
c.quant = "1";
```

```
manualPanel.setVisible(true);
AutoPanel.setVisible(false);
quantmanComboBox.setVisible(true);
quantautoComboBox.setVisible (false);
resetar();
atualizaLog("Pressione o Botão Start");
if (startstopToggleButton.isSelected()){
   startstopToggleButton.doClick();
}
c.modo = 0;
}

private void autoRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {
   manualPanel.setVisible(true);
   AutoPanel.setVisible(true);
   quantmanComboBox.setVisible (false);
   quantautoComboBox.setVisible (true);
   repetirButton.setVisible(false);
   resetarButton.setVisible(false);
   resetar ();
   atualizaLog("Selecione Inserir Passo.");
   if (startstopToggleButton.isSelected()){
      startstopToggleButton.doClick();
   }
   c.modo = 1;
   c.quant = "1";
}

private void freqComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
   int fcombobox = freqComboBox.getSelectedIndex();
   if (fcombobox == 0.5){
      c.freq = "1";
      c.valorfreq = "0.5";
   }
   else if (fcombobox == 1){
      c.freq = "2";
      c.valorfreq = "1";
   }
   else if (fcombobox == 2){
      c.freq = "3";
      c.valorfreq = "2";
   }
   else if (fcombobox == 4){
      c.freq = "4" ;
      c.valorfreq = "4";
   }
   System.out.println("Frequencia alterada");
}
```

```
private void quantmanComboBoxActionPerformed(java.awt.event.ActionEvent evt)
{
   String qcombobox = (String) quantmanComboBox.getSelectedItem();
   if (qcombobox.equals("1")){
      c.quant = "1";
   }
   else if (qcombobox.equals("3")){
      c.quant = "2";
   }
   else if (qcombobox.equals("5")){
      c.quant = "3";
   }
   else if (qcombobox.equals("10")){
      c.quant = "4" ;
   }
   else if (qcombobox.equals("-")){
      c.quant = "-" ;
   }

   System.out.println("Quantidade alterada");
}

private void quantautoComboBoxActionPerformed(java.awt.event.ActionEvent evt)
{
  String qcombobox = (String) quantautoComboBox.getSelectedItem();

      if (qcombobox.equals("-")) {
       tempoComboBox.setVisible(true);
       quantautoComboBox.setVisible(false);
   }
   else {
      tempoComboBox.setVisible(false);
      c.tempo = "-";
   }

  if (qcombobox.equals("1")){
      c.quant = "1";
   }
   else if (qcombobox.equals("2")){
      c.quant = "2";
   }
   else if (qcombobox.equals("3")){
      c.quant = "3";
   }
   else if (qcombobox.equals("4")){
      c.quant = "4" ;
   }
```

```
    else if (qcombobox.equals("-")){
        c.quant = "-" ;
    }
}

private void startstopToggleButtonMouseClicked(java.awt.event.MouseEvent evt) {
    if (c.control == 0) c.inicio = 1;
        P1.setVisible(true);
        P2.setVisible(true);
        P3.setVisible(true);
        P4.setVisible(true);
        freqComboBox.setVisible(true);
    if (startstopToggleButton.isSelected()){
        c.start = 1;
        if (c.modo == 1) {
            if (c.quant.equals("-") && c.tempo.equals("-")){
                if (c.quant.equals("-")) quantautoComboBox.setVisible(true);
                else tempoComboBox.setVisible(true);
                c.start = 0;
                startstopToggleButton.doClick();
                atualizaLog("Selecione Tempo ou Quantidade.");
            }
            else {
                if (c.ind_seq <= 1){
                    c.start = 0;
                    startstopToggleButton.doClick();
                    atualizaLog("Insira Posições e Frequências.");
                }
                else {
                    if (c.quant.equals("-")){
                        valortempoLabel.setVisible(true);
                        valortempoLabel.setText(c.tempo);
                        stattempoLabel.setVisible(true);
                        statquantLabel.setVisible(false);
                        valorquantLabel.setVisible(false);
                        c.ntempo = tempoComboBox.getSelectedIndex();

                    }
                    else if (c.tempo.equals ("-")){
                        valortempoLabel.setVisible(false);
                        stattempoLabel.setVisible(false);
                        statquantLabel.setVisible(true);
                        valorquantLabel.setVisible(true);
                        valorquantLabel.setText(c.quant);
                        c.nquant = quantmanComboBox.getSelectedIndex();
                    }
                    c.enviarComando("8");
                    atualizaLog("Selecione Repetir Passo ou Inserir Passo.");
```

```
                repetirButton.setVisible(true);
            }
        }
    } else if (c.modo == 0){
        atualizaLog("Selecione Freqüência, Quantidade e Posição.");
    }
} else {
    if (c.modo == 1) {
        c.start = 0;
        c.stop = 1;
        atualizaLog("Selecione Repetir Passo ou Inserir Passo.");
    }
    if (c.modo == 0) {
        c.start = 0;
        c.stop = 1;
        atualizaLog("Selecione Start para Reiniciar.");
    }

}
}

private void P3MouseClicked(java.awt.event.MouseEvent evt) {
    c.enviarComando("3");
    if (c.ind_seq >=6){
        P1.setVisible(false);
        P2.setVisible(false);
        P3.setVisible(false);
        P4.setVisible(false);
        freqComboBox.setVisible(false);
        quantautoComboBox.setVisible(false);
        tempoComboBox.setVisible(false);
    }
}

private void P4MouseClicked(java.awt.event.MouseEvent evt) {
    c.enviarComando("4");
    if (c.ind_seq >=6){
        P1.setVisible(false);
        P2.setVisible(false);
        P3.setVisible(false);
        P4.setVisible(false);
        freqComboBox.setVisible(false);
        quantautoComboBox.setVisible(false);
        tempoComboBox.setVisible(false);
    }
}

private void P2MouseClicked(java.awt.event.MouseEvent evt) {
```

```
    c.enviarComando("2");
    if (c.ind_seq >=6){
        P1.setVisible(false);
        P2.setVisible(false);
        P3.setVisible(false);
        P4.setVisible(false);
        freqComboBox.setVisible(false);
        quantautoComboBox.setVisible(false);
        tempoComboBox.setVisible(false);
    }
}

private void P1MouseClicked(java.awt.event.MouseEvent evt) {
    c.enviarComando("1");
    if (c.ind_seq >=6){
        P1.setVisible(false);
        P2.setVisible(false);
        P3.setVisible(false);
        P4.setVisible(false);
        freqComboBox.setVisible(false);
        quantautoComboBox.setVisible(false);
        tempoComboBox.setVisible(false);
    }
}

private void resetarButtonMouseClicked(java.awt.event.MouseEvent evt) {
    resetar();
    atualizaLog("Selecione Inserir Passo.");
    if (startstopToggleButton.isSelected()){
        startstopToggleButton.doClick();
    }
}

private void tempoComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
    String tcombobox = (String) tempoComboBox.getSelectedItem();

    if (tcombobox.equals("-")) {
        quantautoComboBox.setVisible(true);
        tempoComboBox.setVisible(false);
    } else {
        quantautoComboBox.setVisible(false);
        c.quant = "-";
    }

    if (tcombobox.equals("1")){
        c.tempo = "1";
    } else if (tcombobox.equals("2")){
        c.tempo = "2";
```

```
    } else if (tcombobox.equals("3")){
        c.tempo = "3";
    } else if (tcombobox.equals("4")){
        c.tempo = "4" ;
    } else if (tcombobox.equals("-")){
        c.tempo = "-" ;
    }
}

private void repetirButtonMouseClicked(java.awt.event.MouseEvent evt) {
    c.enviarComando("9");
}

private void inserirpassoButtonMouseClicked(java.awt.event.MouseEvent evt) {
    resetar();
    c.addPasso = 1;
    c.nPasso++;
    atualizaLog("Selecione Quantidade ou Tempo, Freqüências e Posições.");
    atualizaLog("Então, Pressione Start.");
    repetirButton.setVisible(false);
    resetarButton.setVisible(true);
    if (startstopToggleButton.isSelected()){
        startstopToggleButton.doClick();
    }
}

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new HMI().setVisible(true);
            }
        });
    }

    private Controle c;
    private Comunicacao comm;
    // Variables declaration - do not modify
    private javax.swing.JPanel AutoPanel;
    private javax.swing.JPanel MainPanel;
    private javax.swing.JLabel Mesa;
    private javax.swing.JButton P1;
    private javax.swing.JButton P2;
    private javax.swing.JButton P3;
    private javax.swing.JButton P4;
    private javax.swing.JLabel StartStop;
    private javax.swing.JRadioButton autoRadioButton;
    private javax.swing.JLabel bolinhaLabel;
    private javax.swing.ButtonGroup buttonGroup1;
```

```
    public javax.swing.JLabel cargaLabel;
    private javax.swing.JFileChooser fileChooser;
    private javax.swing.JComboBox freqComboBox;
    public javax.swing.JLabel inicialLabel;
    private javax.swing.JButton inserirpassoButton;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    public java.awt.List loglist;
    private javax.swing.JPanel manualPanel;
    private javax.swing.JRadioButton manualRadioButton;
    private javax.swing.JLabel ncargaLabel;
    private javax.swing.JLabel ninicializaLabel;
    private java.awt.List proglist;
    private javax.swing.JLabel quantLabel;
    private javax.swing.JComboBox quantautoComboBox;
    public javax.swing.JComboBox quantmanComboBox;
    private javax.swing.JButton repetirButton;
    private javax.swing.JButton resetarButton;
    private javax.swing.JToggleButton startstopToggleButton;
    private javax.swing.JLabel statquantLabel;
    private javax.swing.JLabel stattempoLabel;
    public javax.swing.JComboBox tempoComboBox;
    private javax.swing.JLabel valorquantLabel;
    private javax.swing.JLabel valortempoLabel;
    private javax.swing.JLabel ycargaLabel;
    private javax.swing.JLabel yinicialLabel;
    // End of variables declaration

}



/*************************************************************/
/*  Classe Controle                                          */
/*                                                           */
/*  Ultima alteracao: 15/11/09                               */
/*************************************************************/

package controle;

import java.io.IOException;
```

```
public class Controle {

   HMI HMIObj;
   Comunicacao comObj;
   String Comando  = "0";
   String cod = "0";
   String[] codauto = new String [20];
   String ComandoFinal = "";
   String seqpas = "";
   String seqfreq = "";
   int envio = 0;
   int modo = 0;
   int caso = 0;
   String freq = "1";
   String quant = "1";
   String tempo = "-";
   int addPasso = 0;       // Passo adicionado
   int nPasso = 0;         // no. de Passos
   int ind_seq = 1;        // Índice da Seqüência
   int inicio = 0;         // Início da Execução do Programa
   int control = 0;        // Flag de Inicialização
   int start = 0;          // Start
   int stop = 0;           // Stop
   int repetirpasso = 0;
   String valorfreq = "0.5";
   int ntempo = 0;         // tempo lido na serial
   int qtempo = 0;         // quociente do tempo
   int rtempo = 0;         // resto do tempo
   int nquant = 0;         // quantidade lida na serial


/*************************************************************************
**/
   /*  Construtor. Precisa receber o objeto Comunicacao gerado no main  */
   /*  para assim poder informar a HMI eventos.                         */
   /*                                                                   */
   /*  Ultima alteracao: 20/11/09                                       */

/*************************************************************************
**/
   Controle( HMI HMIObjConstrutor){
      HMIObj = HMIObjConstrutor;
      NCObj  = new ControleNC();
   }

   public void setComm(Comunicacao comm){
   comObj=comm;
   }
```

```
/*************************************************************/
/*  Funcao para passar comando para o PIC              */
/*                                                     */
/*  Ultima alteracao: 20/11/09                         */
/*************************************************************/
void enviarComando(String s){
   codauto [0] = "";

   if (modo == 0 && start == 1) {
   switch (returnIndex(s, "1", "2", "3", "4")) {
         case 0 : cod = "Am0"+"q"+quant+"p1f"+freq+"B"; caso = 1; break;
         case 1 : cod = "Am0"+"q"+quant+"p2f"+freq+"B"; caso = 2; break;
         case 2 : cod = "Am0"+"q"+quant+"p3f"+freq+"B"; caso = 3; break;
         case 3 : cod = "Am0"+"q"+quant+"p4f"+freq+"B"; caso = 4; break;
    }

   if (caso != 0) {
       HMIObj.atualizaLog("Comando enviado: "+s);
       HMIObj.atualizaLog("Enviado: "+cod);
       caso = 0;
       envio = 1;
       Comando = cod;
   }
   else {
      envio = 0;
   }
   }
   else if (modo == 1 && addPasso == 1){
      switch (returnIndex(s, "1", "2", "3", "4", "8")) {
         case 0 : codauto [ind_seq] = "p1f"+freq;
         System.out.println (codauto[ind_seq]);
                seqpas =seqpas + "1  ";
                seqfreq = seqfreq + valorfreq +"  ";
                caso = 1;
                break;
         case 1 : codauto [ind_seq] = "p2f"+freq;
         System.out.println (codauto[ind_seq]);
                seqpas =seqpas + "2  ";
                seqfreq = seqfreq + valorfreq +"  ";
                caso = 2;
                break;
         case 2 : codauto [ind_seq] = "p3f"+freq;
         System.out.println (codauto[ind_seq]);
                seqpas =seqpas + "3  ";
                seqfreq = seqfreq + valorfreq +"  ";
                caso = 3;
```

```
                break;
            case 3 : codauto [ind_seq] = "p4f"+freq;
            System.out.println (codauto[ind_seq]);
                seqpas =seqpas + "4  ";
                seqfreq = seqfreq + valorfreq +"  ";
                caso = 4;
                break;
            case 4 : caso = 8;
             System.out.println ("caso9");
                break;
        }
        if (caso != 0) {
            if (caso == 8){
                if (tempo.equals("-")){
                    ComandoFinal = "Am1q" + quant + ComandoFinal + "B";
                }
                else {
                    ComandoFinal = "Am1t"+tempo+ComandoFinal + "B";
                }
                HMIObj.atualizaLog("Comando enviado: "+ComandoFinal);
                envio = 1;
                ind_seq = 1;
            }
            else {
                caso = 0;
                HMIObj.atualizaProg(1);
                ComandoFinal = ComandoFinal + codauto[ind_seq];
                System.out.println (ComandoFinal);
                ind_seq++;
            }
        }
    }
    else if (s.equals("9")){
        HMIObj.atualizaLog("Comando enviado: "+ComandoFinal);
        repetirpasso = 1;
        envio = 1;
    }
}

 private static int returnIndex(String toIndex, String... args) {
    for (int i=0; i<args.length; i++) {
        if (toIndex.equals(args[i] ) )
            return i;
    }
    return -1;
 }

void propagaStatus(String status){
```

```
        HMIObj.atualizaLog(status);
        if (status.equals("AychB")) {
           HMIObj.cargaLabel.setText("Carregado!!");
        }
        else if (status.equals("AnchB")){
           HMIObj.cargaLabel.setText("Descarregado!!");
        }
        else if (status.equals("AsokB")){
           HMIObj.inicialLabel.setText("Inicializado!!");
        }
        else if (status.equals("Aquant--B")){
           nquant--;
           HMIObj.inicialLabel.setText(""+nquant);
        }
        else if (status.equals("Atempo--B")){
           ntempo--;
           qtempo = ntempo / 60;
           rtempo = ntempo % 60;
           HMIObj.inicialLabel.setText(qtempo+"h"+rtempo);
        }
     }
}


/*************************************************************/
/*  Classe Comunicação                                       */
/*                                                           */
/*  Ultima Alteração: 20/11/09                               */
/*                                                           */
/*************************************************************/


package controle;

import javax.comm.*;
import java.io.*;
import java.util.*;


public class Comunicacao extends Thread implements SerialPortEventListener{

   CommPortIdentifier cp;
   SerialPort porta;
   OutputStream saida;
   InputStream entrada;
   int nodeBytes;
   HMI HMIObj;
```

```
Controle cont;
String portName;
String msg2;
int inicializa = 0;

//contrutor para primeira instanciação

public Comunicacao (String s,Controle c){
   super();
   cont=c;
   portName=s;
}


//Obtém o ID da PORTA
public void ObterIdDaPorta(){

   try {
      cp = CommPortIdentifier.getPortIdentifier(portName);

      if ( cp == null ) {
         cont.propagaStatus("A porta escolhida nao existe!" );
      }
   } catch (Exception e) {
      cont.propagaStatus("Erro durante o procedimento: " + e );
   }
}

//abre uma porta serial para comunicaçao
public void AbrirPorta(){
   try {
      porta = (SerialPort)cp.open(portName,1000);

      //configurar parâmetros
      porta.setSerialPortParams(9600,porta.DATABITS_8,
                                porta.STOPBITS_1,
                                porta.PARITY_NONE);
   } catch (Exception e) {
      cont.propagaStatus("Ocorreu um erro ao abrir a porta: " + e);
   }
}

//função que fecha a conexão
public void FecharCom(){
   try {
      porta.close();
   } catch (Exception e) {
      cont.propagaStatus("Ocorreu um erro ao fechar a porta: " + e);
```

```
    }
}

//funçao que envia uma string
public void EnviarUmaString(String msg){
    try {
        saida = porta.getOutputStream();
    } catch (Exception e) {
        cont.propagaStatus("ENVIO: Ocorreu um erro no envio de dados: " + e);
    }

    try {
        saida.write(msg.getBytes());
        Thread.sleep(20);
        saida.flush();
    } catch (Exception e) {
        cont.propagaStatus("ENVIO: Houve um erro durante o envio: " + e);
      }
 }

//leitura de dados na serial
public void LerDados(){
    try {
        entrada = porta.getInputStream();
    } catch (Exception e) {
        cont.propagaStatus("Ocorreu um erro no recebimento de dados: " + e);
    }

    try {
        porta.addEventListener(this);
    } catch (Exception e) {
        cont.propagaStatus("Ocorreu um erro ao criar um listener: " + e);
    }

    porta.notifyOnDataAvailable(true);

    try {
        Thread.sleep(15);
    } catch (Exception e) {
        cont.propagaStatus("Ocorreu um erro ao iniciar leitura: " + e);
    }
}


public void serialEvent(SerialPortEvent ev){
    switch (ev.getEventType()) {
    case SerialPortEvent.BI:
    case SerialPortEvent.OE:
```

```
        case SerialPortEvent.FE:
        case SerialPortEvent.PE:
        case SerialPortEvent.CD:
        case SerialPortEvent.CTS:
        case SerialPortEvent.DSR:
        case SerialPortEvent.RI:
        case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
        break;
        case SerialPortEvent.DATA_AVAILABLE:
        byte[] bufferLeitura = new byte[20];

        boolean inicio = false;
        int i,j;
        int indice_inicio = 0;
        int indice_fim = 0;


        try {

           while ( entrada.available() > 0 ) {
              nodeBytes = entrada.read(bufferLeitura);
           }


         if (bufferLeitura.length == 0) {
           cont.propagaStatus("Nada lido!");
        } else {
              String Dadoslidos = new String(bufferLeitura);
              System.out.println ("Leitura: "+Dadoslidos);
              for (i = 0; i < 20; i++){
                 if (Dadoslidos.charAt(i) == 'A') {
                    inicio = true;
                    indice_inicio = i;
                 }
                 if (Dadoslidos.charAt(i) == 'B' && inicio == true) {
                    indice_fim = i;
                    char[] dadosuteis = new char[(indice_fim -1) - (indice_inicio + 1) +
1];
                    for (j = indice_inicio+1; j <= indice_fim-1; j++) {
                       dadosuteis[j-indice_inicio-1] = Dadoslidos.charAt(j);
                    }
                    String Dados = new String (dadosuteis);
                    System.out.println("Dados = "+Dados);
                    cont.propagaStatus("Lido : "+Dados);
                    inicio = false;
                 }
              }
        }
```

```
        } catch (Exception e) {
            cont.propagaStatus("Ocorreu um erro durante a leitura: " + e);
        }
        break;
        }
    }

    public void run() {
        while(true){

            ObterIdDaPorta();
            AbrirPorta();

            if (cont.inicio == 1) {
                cont.control = 1;
                cont.inicio = 0;
                EnviarUmaString("AiiiB");
            }
            if (cont.modo == 0) { // Modo Manual
                if (cont.envio == 1 && cont.start == 1){
                    EnviarUmaString(cont.Comando);
                    System.out.println(cont.Comando);
                    cont.envio = 0;
                }
            }
            else if (cont.modo == 1) { // Modo Automático
                if (cont.start == 1 && cont.envio == 1){
                    cont.envio = 0;
                    cont.addPasso = 0;
                    EnviarUmaString(cont.ComandoFinal);
                    System.out.println(cont.ComandoFinal);
                }
              if (cont.start == 1 && cont.repetirpasso == 1){
                    EnviarUmaString(cont.ComandoFinal);
                    cont.repetirpasso = 0;
                    cont.envio = 0;
              }
            }
            if (cont.stop == 1) {
                cont.stop = 0;
                EnviarUmaString("AsssB");
            }
            LerDados();
            FecharCom();

            }
      }
}
```