Ilana Nigri

Projeto e Controle de um Manipulador Robótico de Dois Graus de Liberdade

Trabalho de Fim de curso

Dissertação apresentada como requisito parcial para obtenção do título de Engenheiro pelo Programa de Graduação em Engenharia de Controle e Automação da PUC-Rio.

Orientador: Marco Antonio Meggiolaro

Rio de Janeiro, 05 de dezembro de 2006

Dedicatória

Dedico este trabalho, a todos os amigos que tornaram esses difíceis 5 anos, em momentos inesquecíveis.

Agradecimento

Este trabalho, que começou como um projeto de iniciação científica, não teria se concluído se não fosse a ajuda de algumas pessoas. Pessoas estas que tornaram minha permanência no laboratório mais agradável, e o meu projeto mais interessante.

- Aos meus pais e irmão que me incentivaram desde o início.
- Ao meu professor orientador Marco Antonio Meggiolaro, por todo conhecimento compartilhado durante esses últimos anos.
- Ao amigo e ex-aluno de mestrado Felipe Augusto Weilemann Belo que sem ele este projeto não teria se concluído.
- Aos amigos Felipe dos Santos Scofano e Júlio Quadrio Guedes por toda paciência, ajuda e dedicação.
- Ao professor do dept. de Engenharia Elétrica Mauro Schwanke por todo incentivo e ajuda.
- A minha querida amiga e ex-aluna de graduação Camilla Bacellar Mello pelo companheirismo e amizade durante esses cinco anos.
- A toda equipe Riobotz que de inúmeras formas, estavam sempre dispostos a ajudar.

Índice

Índice de figuras

Resumo

Este projeto consiste na construção mecânica de um manipulador robótico de 2 graus de liberdade bem como sua eletrônica, tornando hábil o controle pelo computador. Foi realizado também um software para controle de posição dos dois elos, além da movimentação em trajetórias circulares.

Palavras-chave: manipulador, controle PID, torque computado

1. Introdução

Nos últimos anos tem-se observado um vertiginoso crescimento nas aplicações de sistemas robóticos. Sistemas robóticos - como braços mecânicos articulados, veículos terrestres, aéreos ou submarinos - são dispositivos mecânicos versáteis equipados com sensores e atuadores, sob o controle de um sistema computacional. Por serem facilmente re-programáveis, sistemas robóticos possuem grande potencial para a execução de diversas tarefas [1]. O desenvolvimento destes sistemas requer um conhecimento multidisciplinar nas Engenharias Eletrônica, Mecânica e de Computação. A área de robótica industrial traz benefícios como a melhoria da eficácia, da qualidade, redução de mão-de-obra, além de mais eficiência, confiabilidade e redução de custos. Vantagens adicionais incluem a capacidade de realizar tarefas para as quais os humanos teriam grande dificuldade, e a remoção de humanos de tarefas em ambientes potencialmente perigosos.

Mais recentemente, o enfoque tem se voltado na melhoria da precisão e eficiência destes sistemas. Para tanto, é necessário utilizar técnicas de controle que permitam utilizar estes sistemas em velocidades maiores sem sacrificar sua precisão. Dentre as diversas técnicas de controle existentes, as mais comuns são os controles PID (Proporcional-Integral-Derivativo) e de Torque Computado [2, 3, 4]. Estas técnicas vêm sendo utilizadas, por exemplo, em manipuladores de alto desempenho para a indústria de wafers de silício, como visto nas Figuras 1.1 e 1.2.

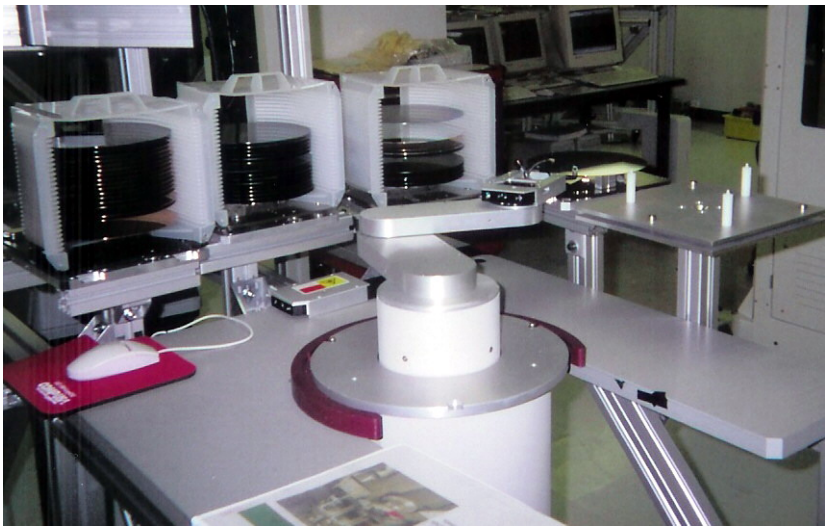


Figura 1.1 – Manipulador de 2 graus de liberdade (Brooks Automation)

Figura 1.2 – Visão interna de um manipulador de 2 graus de liberdade (Brooks Automation)

A principal característica do Controle PID é o fato de ter seu controle isolado para cada junta. Ele é muito utilizado por sua facilidade de programação e por oferecer uma precisão satisfatória para muitos casos. A grande desvantagem desta técnica sobre as demais, é o fato de não levar em consideração os efeitos da dinâmica. Isso pode ser facilmente percebido, no controle de trajetórias onde os pequenos erros se acumulam e a diferença final é visível.

Para uma melhor precisão, é utilizado o Controle de Torque Computado. Este visa determinar um controle de torque baseado em equações bastante complexas da dinâmica e por isso, não são muito utilizados em casos de controles lineares.

2. Objetivos

O presente trabalho teve como objetivo desenvolver um manipulador robótico de dois graus de liberdade controlado por computador, para posicionamento e geração de trajetórias em baixa ou alta velocidade, com compensação de efeitos dinâmicos. Foram utilizadas, as duas técnicas de controle mais comuns (PID e Torque Computado), e observadas as principais diferenças entre elas.

3. Metodologia

O procedimento experimental do projeto foi composto das seguintes etapas:

- Estudo das técnicas de controle
- Desenvolvimento do manipulador
- Estudo da comunicação entre o manipulador e o computador
- Desenvolvimento do programa de controle de posição e trajetórias

3.1. Técnicas de controle

Para ambas as técnicas utilizadas será utilizada a notação q para posição do manipulador, $\dot{q}$ para velocidade (derivada da posição q) e $\ddot{q}$ para a aceleração (segunda derivada da posição).

Controle PID

O controle PID controla cada junta individualmente, e por isso suas equações são as mesmas para todos os elos do manipulador:

$$\tau_i = u_i = Kp_i(q_{des} - q) + Kd_i(\dot{q}_{des} - \dot{q}) + Ki_i\int_0^t (q_{des} - q)dT$$

onde, Kp é a constante de proporcionalidade, que controla a rigidez das juntas do manipulador ao se deslocar para a posição desejada; Kd é a constante derivativa, um termo de amortecimento que influi nas oscilações realizadas pelo manipulador sobre a posição desejada; e Ki é a constante

integral, que é responsável por corrigir o erro residual. Entende-se por erro como sendo a diferença entre a posição para qual o manipulador deve ir e a sua posição real e é caracterizado na equação por $q_{des} - q$.

Controle de Torque Computado

Este controle é descrito pela equação genérica abaixo:

$$\tau_i = \sum_{j=1}^{n} H_{ij}\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}\dot{q}_j\dot{q}_k + G_i$$

onde, H_{ij} são os coeficientes da matriz de inércia do manipulador, h_{ijk} é o coeficiente de três índices de Christoffel, e G_i é o termo gravitacional. Para o projeto foram utilizadas as equações da dinâmica de um manipulador de dois graus de liberdade que possui seu segundo elo (elo mais afastado do eixo) acionado remotamente, ou seja acionado por correias, vide Figura 3.1:

$$\tau_1 = H_{11}\ddot{\theta}_1 + H_{12}\ddot{\theta}_2 - h\dot{\theta}_2^2$$
$$\tau_2 = H_{22}\ddot{\theta}_2 + H_{12}\ddot{\theta}_1 + h\dot{\theta}_1^2$$

$$H_{11} = I_1 + m_1 lc_1^2 + m_2 l_1^2$$
$$H_{22} = I_2 + m_2 lc_2^2$$
$$H_{12} = m_2 l_1 lc_2 \cos(\theta_2 - \theta_1)$$

onde θ_1 e θ_2 são os ângulos das juntas (equivalentes a q_1 e q_2), l é o comprimento do elo, l_c é a distância entre cada junto e o centro de massa do elo seguinte, e I é o momento de inércia de cada elo, onde $I = \frac{ml^2}{12}$, e o coeficiente do termo centrífugo (h) é igual a:

$$\frac{dH_{12}}{d\theta_2} = -m_2 l_1 lc_2 \operatorname{sen}(\theta_2 - \theta_1) = -h$$

$$\frac{dH_{12}}{d\theta_1} = m_2 l_1 lc_2 \operatorname{sen}(\theta_2 - \theta_1) = h$$

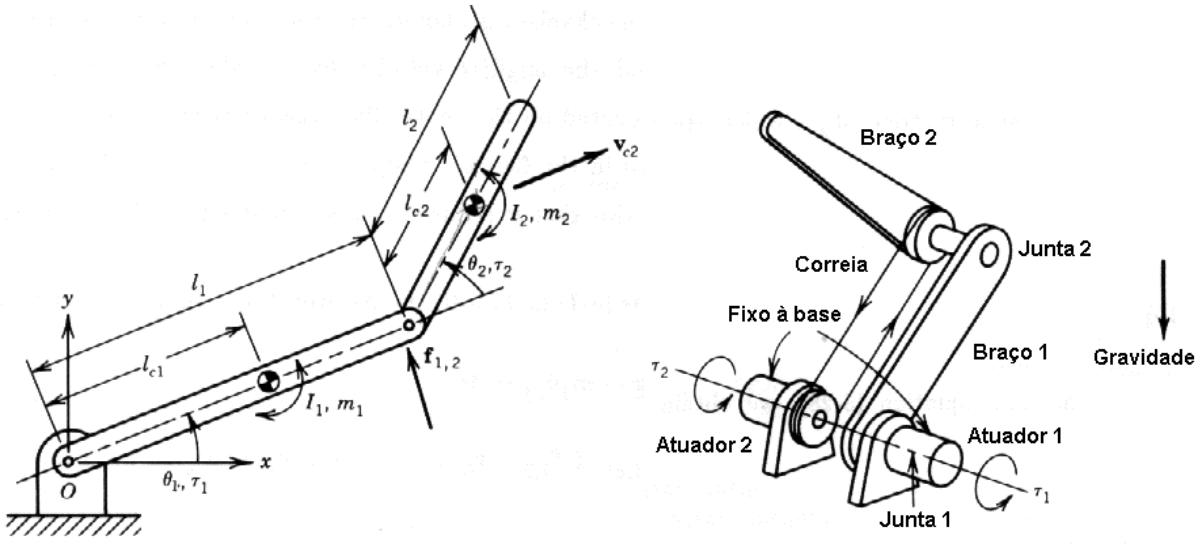


Figura 3.1 – Manipulador de 2 graus de liberdade.

Note que o ângulo θ_2 considerado aqui é medido em relação à horizontal, logo ele equivale ao ângulo $\theta_1 + \theta_2$ da Figura 3.1. Além disso, o manipulador se move em um plano horizontal, logo os efeitos gravitacionais podem ser desprezados ($G_1 = G_2 = 0$).

No controle de torque computado, os termos de aceleração dos ângulos das juntas são substituídos pelas saídas de controladores PID calibrados para massas unitárias.

3.2. Desenvolvimento do manipulador

A estrutura mecânica do manipulador foi construída utilizando dois moto-redutores de corrente contínua (CC) para acionamento do sistema (um para cada grau de liberdade), um par de polias e correia para acionamento remoto do elo 2, e perfis de alumínio para a construção dos elos e da base. Para a leitura da posição foram utilizados encoders, que são sensores óticos de posição, embutidos ao motor, medindo pulsos. Os encoders utilizados continham 2048 pulsos por volta, vide a Figura 3.2.

Figura 3.2 - Disco óptico do encoder de alta resolução utilizado.

Como os motores utilizados para o acionamento do sistema não possuíam encoders, a solução encontrada foi acoplar estes motores a outros motores que possuíssem os sensores. Portanto foram

utilizados mais dois motores CC passivos que além de funcionarem como encoders, funcionando também como sensores de velocidade (tacômetros), ao serem utilizados como geradores de energia elétrica, vide Figura 3.3.

Figura 3.3 – Acoplamento do motoredutor (acima) com o tacômetro/encoder (abaixo, dourado).

O motor no topo da Figura 3.3 é o responsável pelo acionamento de um dos elos do sistema, e o motor mais abaixo é o responsável pela medição de velocidade e posição. O cabo inferior dirige a informação da posição do manipulador para uma placa A/D D/A modelo ServoToGo, dentro de um computador PC. Este procedimento foi feito da mesma maneira para o outro elo.

Por fim foi adicionada uma ponteira laser na ponta do elo 2 para melhor visualizar as trajetórias. As Figuras 3.4, 3.5 e 3.6 apresentam a parte mecânica do manipulador desenvolvido.

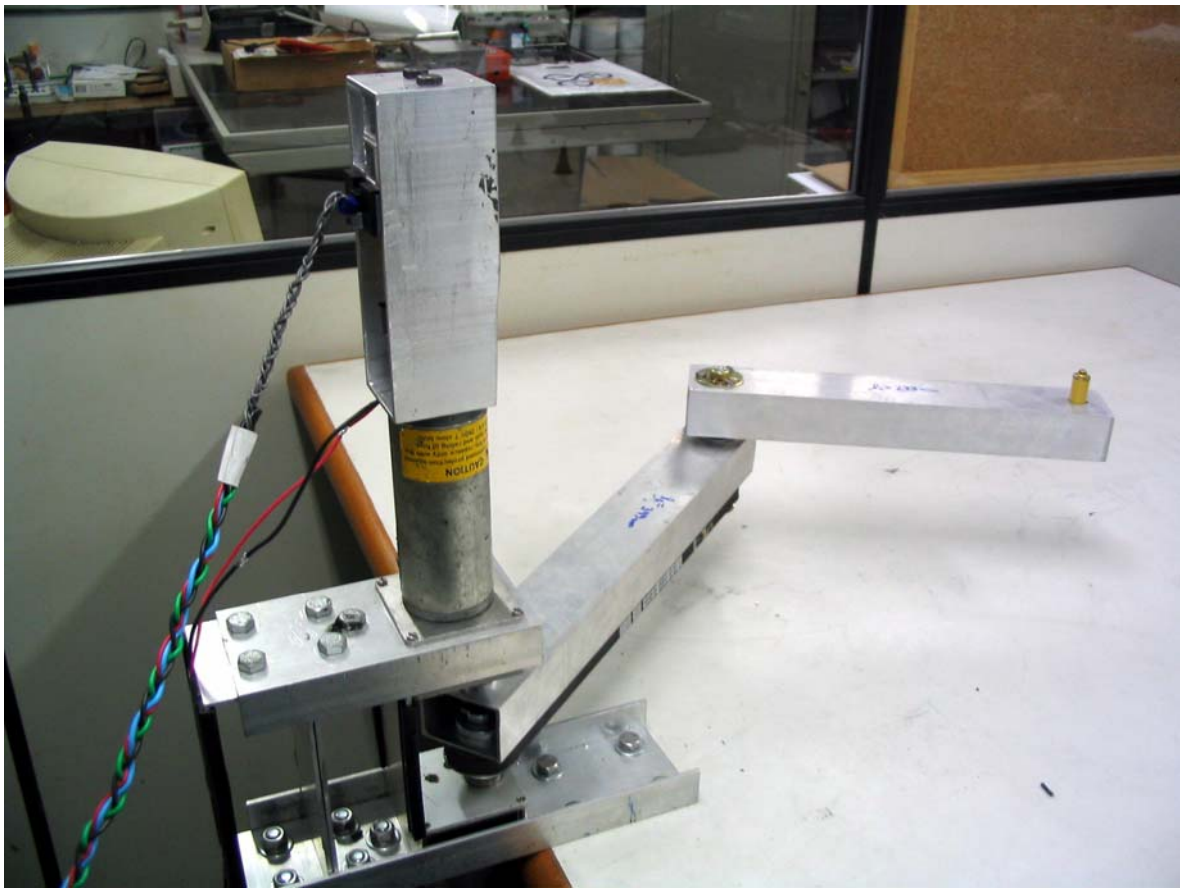


Figura 3.4 – Vista do manipulador desenvolvido.

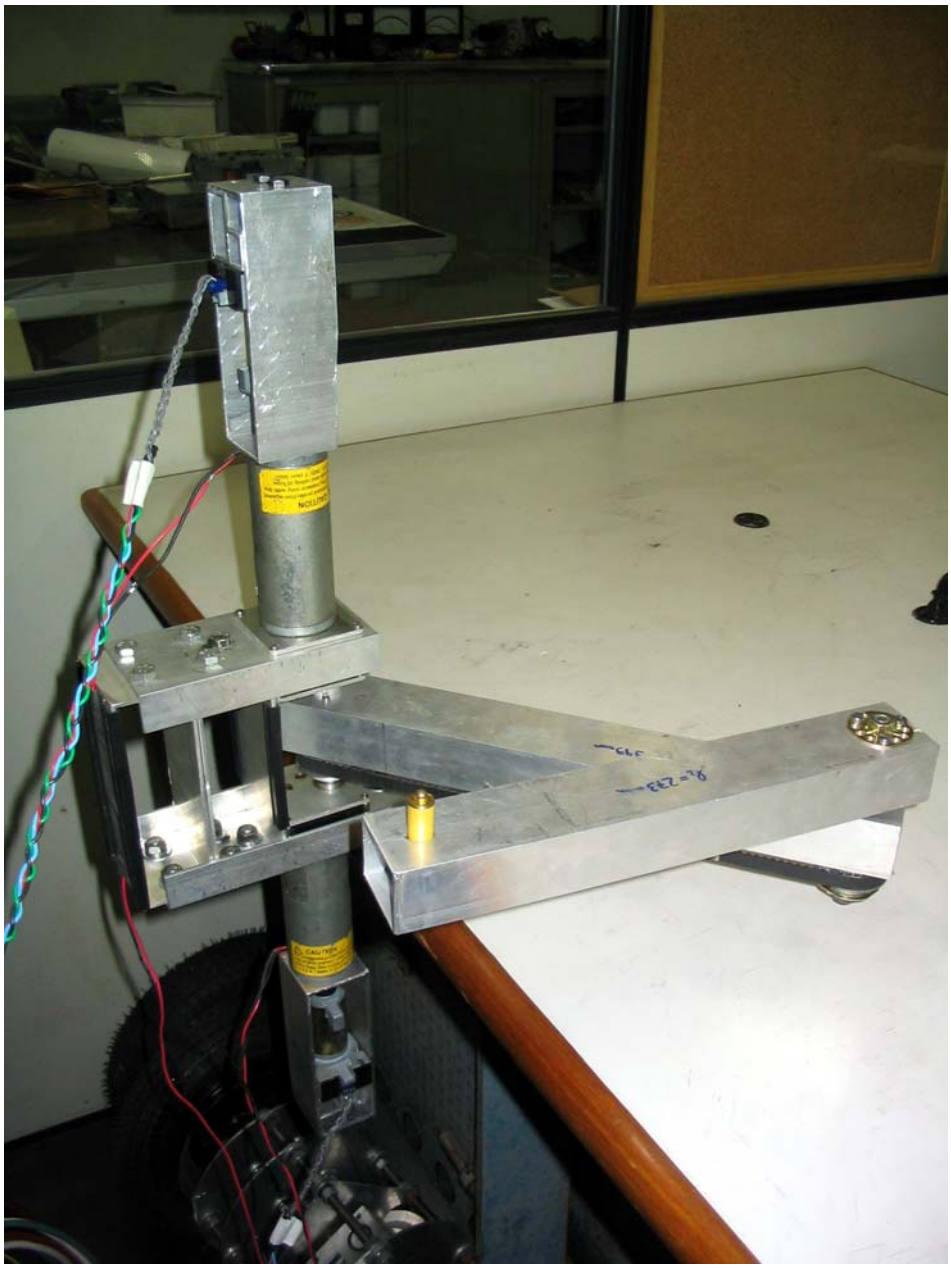


Figura 3.5 – Manipulador em outra configuração.

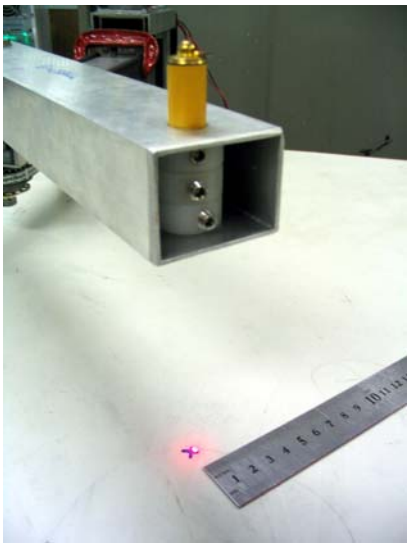


Figura 3.6 – Ponteira laser colocada na extremidade do elo 2

3.3. Comunicação entre o manipulador e o computador

Esta é à parte do projeto, considerada mais crítica. Foram gastos cerca de 7 meses no estudo da comunicação manipulador – computador, de forma que o software pudesse trabalhar em tempo real com os dados do manipulador.

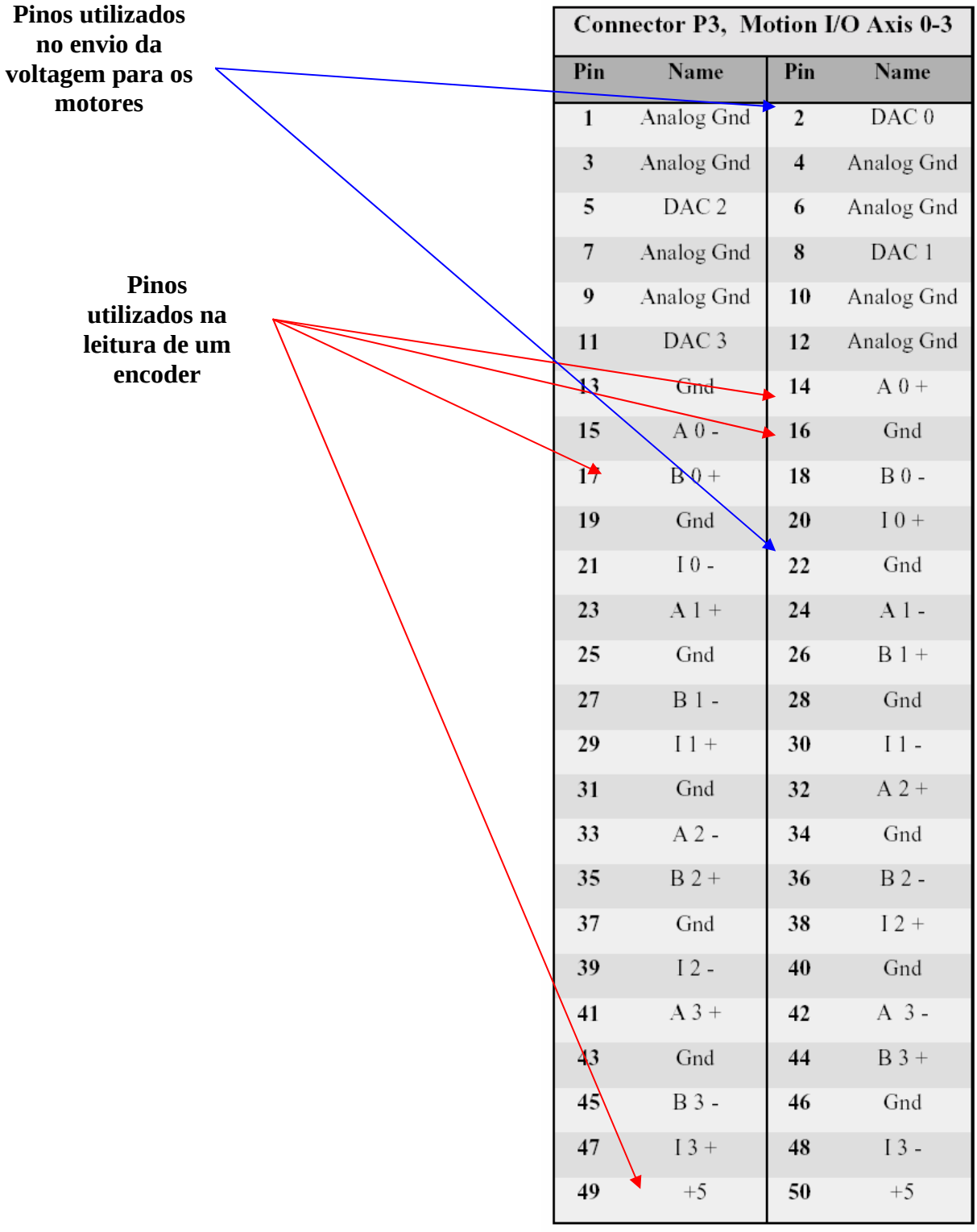
A primeira decisão tomada foi à utilização e posteriormente a aquisição de uma placa conversora analógico-digital, digital-analógica da marca ServoToGo. Esta possui cerca de 8 canais de entradas analógicas, 8 canais de saídas analógicas, 8 canais de entradas para encoders, e 32 bits de entrada e saída digital.

Figura 3.7 – Placa ServoToGo

A utilização da placa neste projeto se restringiu a duas entradas para encoders, para determinar a posição real de cada elo, e dois canais de saídas digitais, para o envio da voltagem final, decorrente dos controles, aos motores respectivos.

Para cada leitura de encoder, foram utilizados 4 pinos: 2 de sinal, um VCC (alimentador) e 1 pino terra. Já para a saída, apenas 1 pino de sinal e 1 pino terra.

Para um melhor entendimento das pinagens utilizadas segue abaixo o esquemático dos dois conectores da placa, utilizados:



Connector P3, Motion I/O Axis 0-3			
Pin	**Name**	**Pin**	**Name**
1	Analog Gnd	**2**	DAC 0
3	Analog Gnd	**4**	Analog Gnd
5	DAC 2	**6**	Analog Gnd
7	Analog Gnd	**8**	DAC 1
9	Analog Gnd	**10**	Analog Gnd
11	DAC 3	**12**	Analog Gnd
13	Gnd	**14**	A 0 +
15	A 0 -	**16**	Gnd
17	B 0 +	**18**	B 0 -
19	Gnd	**20**	I 0 +
21	I 0 -	**22**	Gnd
23	A 1 +	**24**	A 1 -
25	Gnd	**26**	B 1 +
27	B 1 -	**28**	Gnd
29	I 1 +	**30**	I 1 -
31	Gnd	**32**	A 2 +
33	A 2 -	**34**	Gnd
35	B 2 +	**36**	B 2 -
37	Gnd	**38**	I 2 +
39	I 2 -	**40**	Gnd
41	A 3 +	**42**	A 3 -
43	Gnd	**44**	B 3 +
45	B 3 -	**46**	Gnd
47	I 3 +	**48**	I 3 -
49	+5	**50**	+5

Figura 3.8 – Conexões utilizadas pela placa AD/DA

Partiu-se então para a fase real de comunicação, que consiste basicamente, em tentar ler os sinais do encoder no computador. Posteriormente tentar mandar pelo computador um sinal capaz de acionar um motor.

A primeira providência foi instalar o driver da placa a partir dos arquivos recebidos. Depois disso foi possível testar os diversos canais da placa, a partir de exemplos executáveis fornecidos pelos produtores.

Para a comunicação estudaram-se os arquivos fontes, que continham toda a programação de acesso a placa. Porém para utilizá-los teríamos que re-programar as rotinas mais utilizadas, e conseqüentemente re-compilar o driver.

Foi adquirido então um software “Driver Development Kit” para Windows, porém mesmo com o auxilio do software não foi possível compilar o driver.

Depois de outros estudos mais profundos, chegou-se a conclusão que a forma mais prática de se comunicar com a placa, seria através de funções já pré-estabelecidas. O mesmo modo utilizado pelos exemplos.

A comunicação é composta de uma função, existente tanto para Visual Basic, C e Labview, detentora dos seguintes campos:

- nCommand (inteiro)
- nAxis (inteiro)
- lErrorCode (long)
- acErrorString (cadeia de no máximo 100 caracteres)
- lTimeStamp (long)

- lDataCount (long)
- lParams (vetor de long de 1000 posições)
- lBuffer (vetor de long de 1000 posições)

Para o Labview (software escolhido), temos a seguinte SubVi STG.vi :

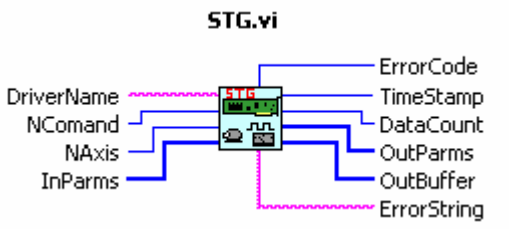


Figura 3.9 – Função STG.vi

A placa disponibiliza uma série de funções referenciadas por um número (NComand), e para a utilização de cada função suas entradas e saídas variam. No caso da leitura do encoder, temos a função GET_ENCODER_ONE. Nas especificações das funções temos o seguinte:

- Valor numérico: 213
- Parâmetros de Entrada: nAxis
- Parâmetros de Saída: lParams[0] = posição., lDataCount = 1

Para acessar o exemplo mostrado acima, teríamos a seguinte função:

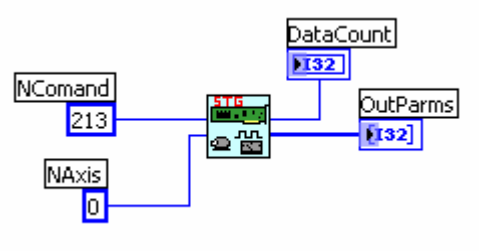


Figura 3.10 – Exemplo do uso da função GET_ENCODER_ONDE

Já para o caso de saídas de voltagens, temos a função SET_DAC_ONE, que possui as seguintes especificações:

- Valor numérico: 114
- Parâmetros de Entrada: nAxis, lParams[0] = valor
- Parâmetros de Saída: nenhum

Sendo o range de 13 bits temos que fazer uma conversão entre os valores fornecidos à função para os reais em volts:

Value	Volts
0x0FFF	+10
0	0
0xF000	-10

Ou seja, se no exemplo acima quiséssemos mandar 10 volts para o canal 0, teríamos algo similar a:

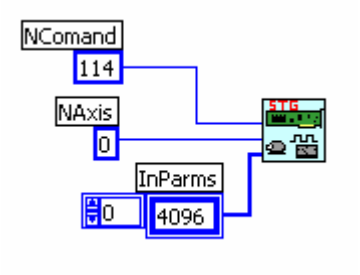


Figura 3.11 – Exemplo da função SET_DAC_ONE

Amplificação de sinal para os motores:

O sinal de saída da placa varia apenas de -10 a 10 volts (conforme visto acima), sendo assim sentiu-se a necessidade de amplificar este sinal, já que iríamos trabalhar com motores de -24 a 24 volts.

Foi utilizado então um controlador/amplificador da marca Roboteq que transformava proporcionalmente cada sinal de 0 à 5 volts em voltagens entre -24 e 24 volts. **Ou seja, para se enviar 0 (zero) volts aos motores, devemos ter como entrada 2,5 volts.**

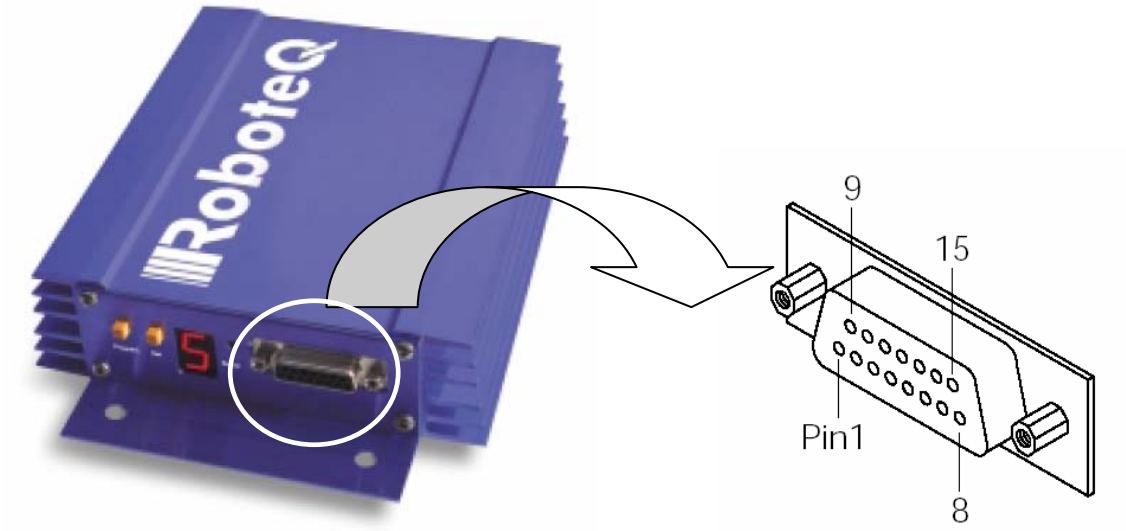


Figura 3.12 - Controlador / Amplificador da marca RoboteQ

Pino	Sinal	Entrada ou Saída	Descrição
10	Vel/ Pos/ T 1	Entrada Analógica	Velocidade, Posição e Tempo do canal 1
11	Vel/ Pos/ T 2	Entrada Analógica	Velocidade, Posição e Tempo do canal 2
5/13	Terra	Força	Terra de Controle (-)

Figura 3.13 - Tabela de pinagens do RoboteQ

Para sua devida utilização foram realizadas as seguintes conexões: o sinal (DAC) vindo da placa é ligado diretamente ao pino 10 do RoboteQ. Analogamente o sinal de Gnd oriundo da placa ServoToGo se conecta ao pino 5 do amplificador.

As saídas associadas às estas conexões são os fios branco e verde referentes ao motor 1 (conforme figura 3.14) e já amplificados. Da mesma forma que para a entrada no pino 11 do RoboteQ, temos a saída no motor 2.

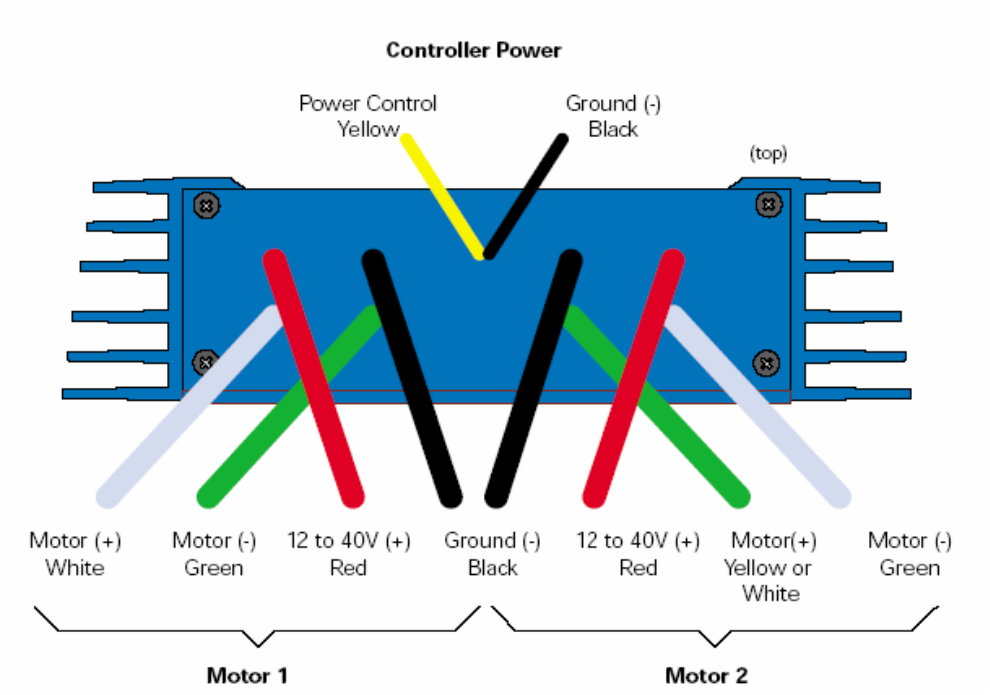


Figura 3.14 – Saídas do Controlador

Para a alimentação dos motores, são conectadas baterias nos canais vermelho e preto. Pode-se obter uma amplificação colocando mais de uma bateria em série, porém neste caso foi utilizada apenas uma bateria para os dois motores. **É uma exigência do controlador que sejam utilizadas baterias e não fontes, por conta de possíveis ruídos no sinal.**

É importante, que ao ligar o RoboteQ, esteja sendo enviados 0 volts para os motores (ou seja, 2,5 volts nas entradas analógicas). Isso é uma proteção do próprio aparelho, para que não sejam enviadas às saídas possíveis voltagens altas.

Segue abaixo uma foto que permite visualizar as conexões do motor com o controlador RoboteQ, e a conexão deste com a placa ServoToGo, dentro do computador:

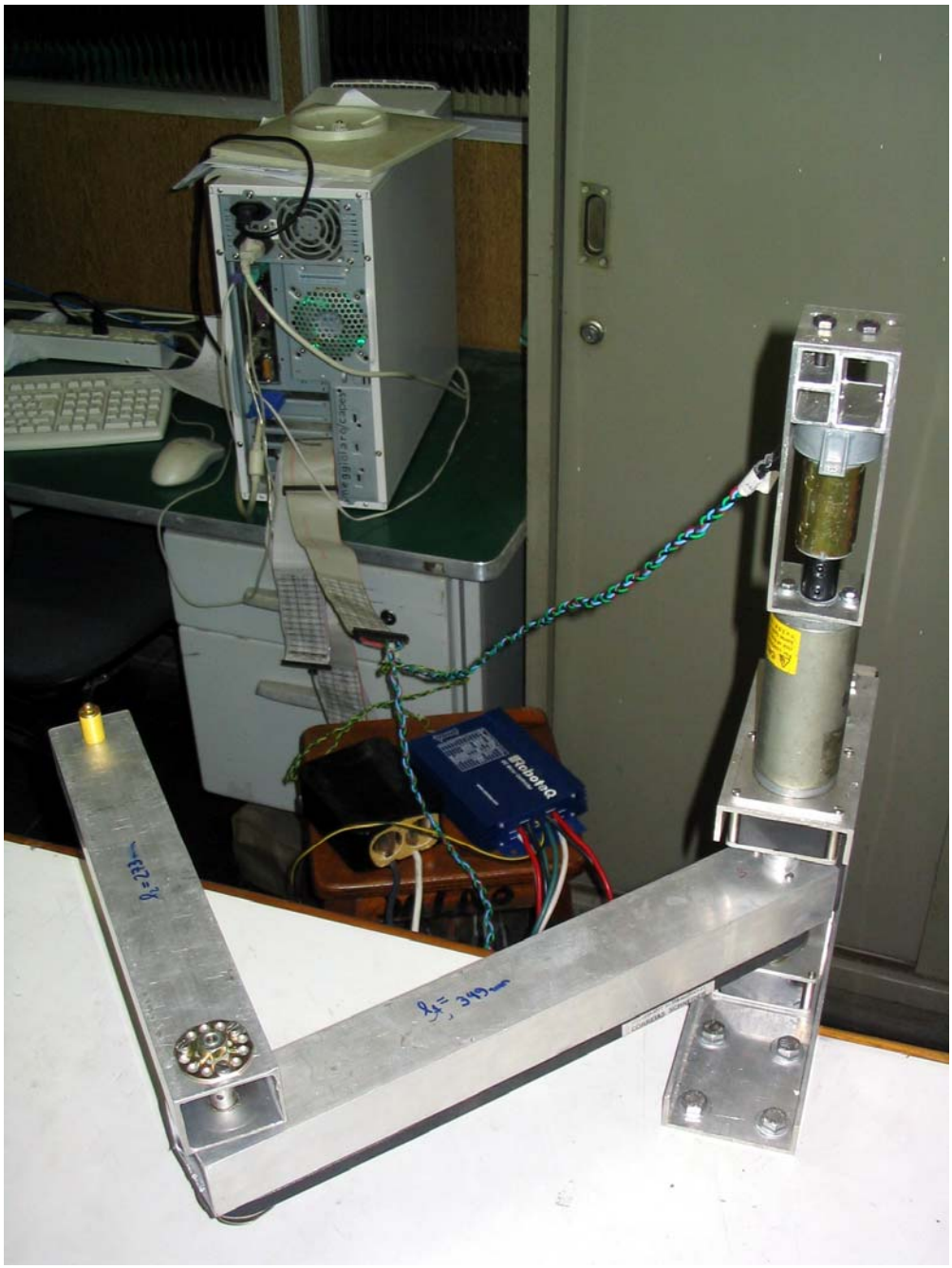


Figura 3.15 – Manipulador, Controlador/Amplificador e Computador

3.4. Software de controle de posição

O *software* desenvolvido possibilita a interface entre um computador PC e a eletrônica de potência. O *software* foi desenvolvido utilizando o programa LabVIEW da *National Instruments*, escolhido pela rapidez de implementação e interface gráfica intuitiva. A Figura 3.16 apresenta a interface principal do programa.

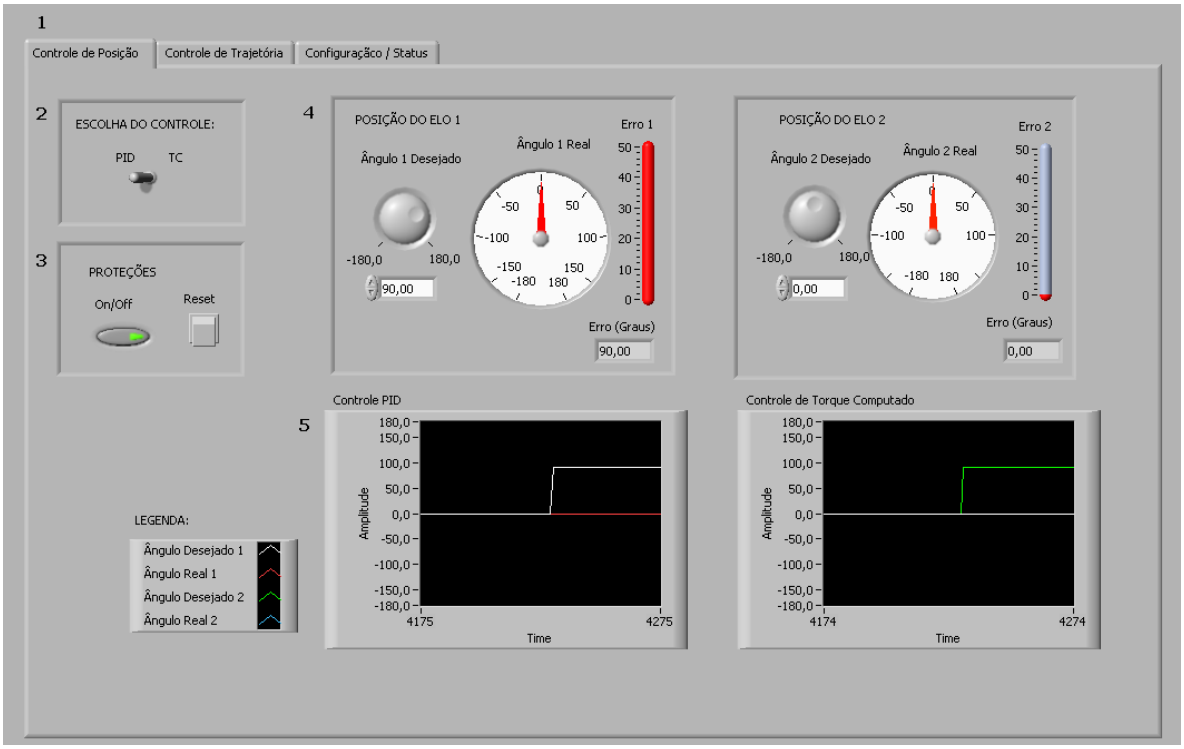


Figura 3.16 - Tela de Controle de Posição

Os números 1 a 5 da Figura 3.16 representam:

1- Navegação: Através dessa tabela, o usuário é capaz de navegar entre as opções: controle de posição, controle de trajetória e configuração/status.

2- Controle: Primeiro passo do programa. Neste botão é possível escolher qual dos dois controles utilizar: Controle PID e Controle de Torque Computado (TC).

3- Proteções:

- Botão On/Off: Caso este botão seja ativado, o computador envia 0 (zero) volts para os motores e estes param o movimento. Este botão foi criado para segurança do manipulador.
- Botão Reset: Antes de iniciar o programa, deve ser gravado a posição de referência do manipulador. Para isso, o manipulador deve ser colocado na posição considerada como referência, e pressionar este botão. Este procedimento fará com que os encoders (sensores de posição) zerem a contagem.

4- Posições dos elos 1 e 2:

- Ângulo Desejado: Através destes botões são escolhidos os ângulos desejados para cada elo do manipulador independentemente.
- Ângulo Real: Este indicador mostra o ângulo real dos elos.
- Erro: Diferença entre a posição para qual o manipulador deve ir e a sua posição real.

5- Gráficos gerados

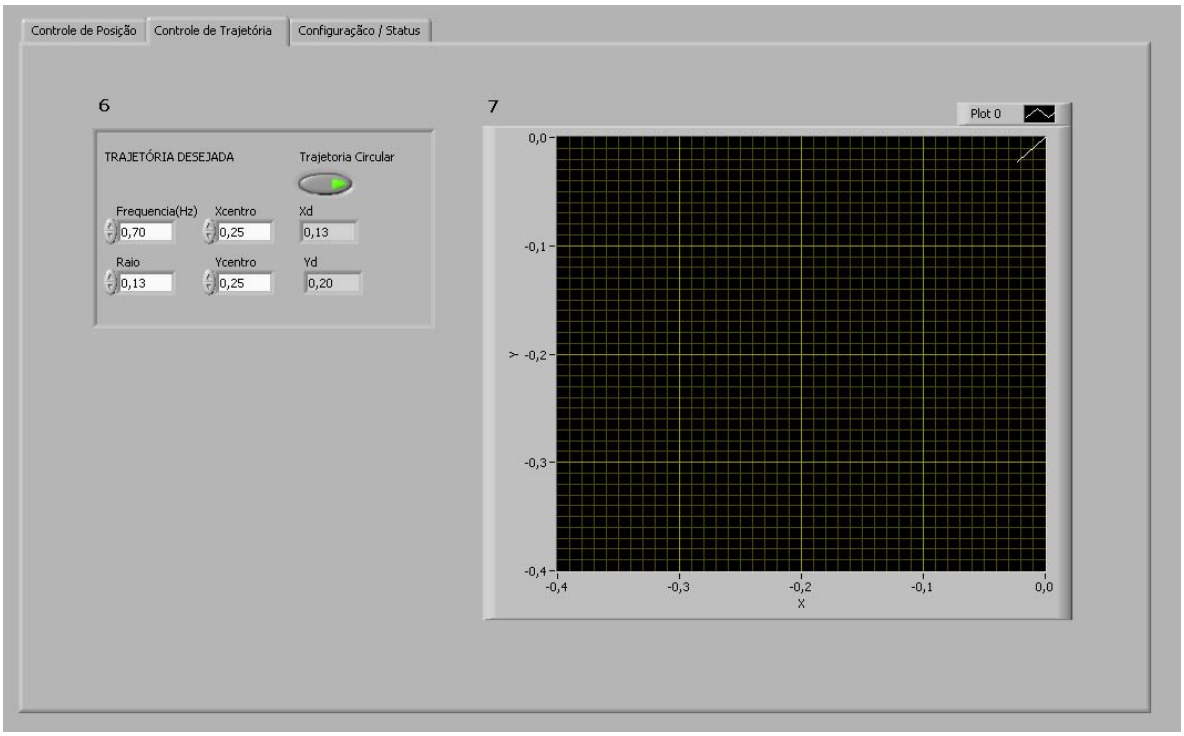


Figura 3.17 - Tela de Controle de Trajetória

Os números 6 e 7 da Figura 3.17 representam:

6- Trajetória Desejada:

- Trajetória Circular: Com este botão acionado, o manipulador inicia uma trajetória circular, de acordo com os parâmetros fixados.
- Freqüência: Escolha da freqüência desejada para a trajetória.
- Raio: Escolha do raio desejado para o círculo.
- Xcentro, Ycentro: Escolhas das coordenadas centrais do círculo.

7-Gráfico X versus Y: Neste gráfico estão representadas as trajetórias circulares real e desejada.

Figura 3.18 - Tela de Configuração / Status

Os números 8 a 11 da Figura 3.18 representam:

8- Constantes: Nestes quadros, pode-se calibrar as constantes do sistema (Kp, Kd, Ki), explicadas no item 1.1. Estas constantes podem variar com o sistema, com o elo e com o controle.

9- Ganhos: Nestes quadros, são indicados os ganhos proporcionais, derivativos e integrais de ambas as juntas.

10-Saídas: Nestes quadros, indicam-se as voltagens que estão sendo enviadas aos motores 1 e 2. Estes indicadores são mostrados independentes do controle utilizado, porém a voltagem real transmitida para os motores é a do controle escolhido na tela inicial (Controle de Posição) .

11-Proteção: Apesar do motor suportar voltagens de –24 a 24 volts, este botão permite que a voltagem final se limite a um valor menor. O botão indica o módulo da voltagem.

4. Experimentos e Resultados

Para iniciar os testes, deixou-se o elo 1 imobilizado, possibilitando apenas que o elo 2 se movimentasse. Com isso, foi possível configurar todas as constantes de modo satisfatório para o elo 2.

Nesta primeira etapa, aplicou-se um ângulo desejado de 45 graus e os resultados se encontram a seguir.

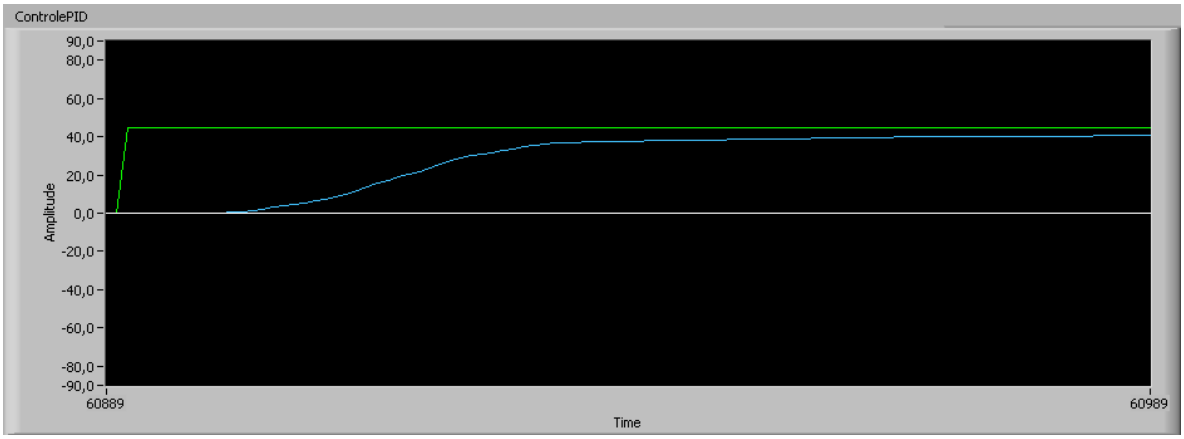


Figura 4.1 - Controle PID – Elo 1 imobilizado

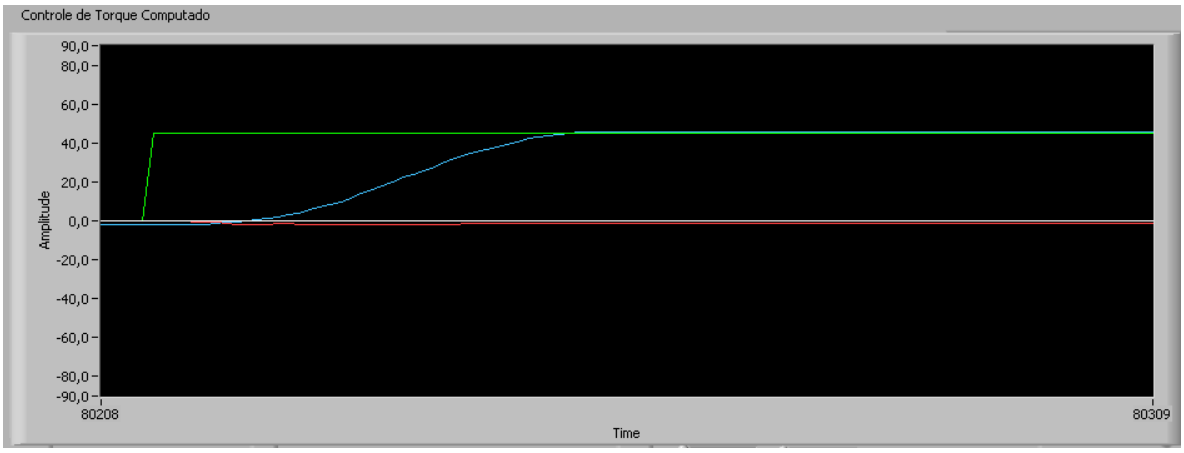


Figura 4.2 - Controle de Torque Computado – Elo 1 imobilizado

Feito isso, pode-se então soltar o elo 1 e calibrar suas constantes, observando porém que os dois elos estariam em movimento. Para testar esta etapa, foi aplicado um ângulo desejado ao elo 1 de 45 graus. Nota-se a superioridade do Controle de Torque Computado, que conseguiu manter baixos os erros da junta 2 (linha azul) enquanto a junta 1 (linha vermelha) se movimentava, compensando os efeitos do acoplamento dinâmico.

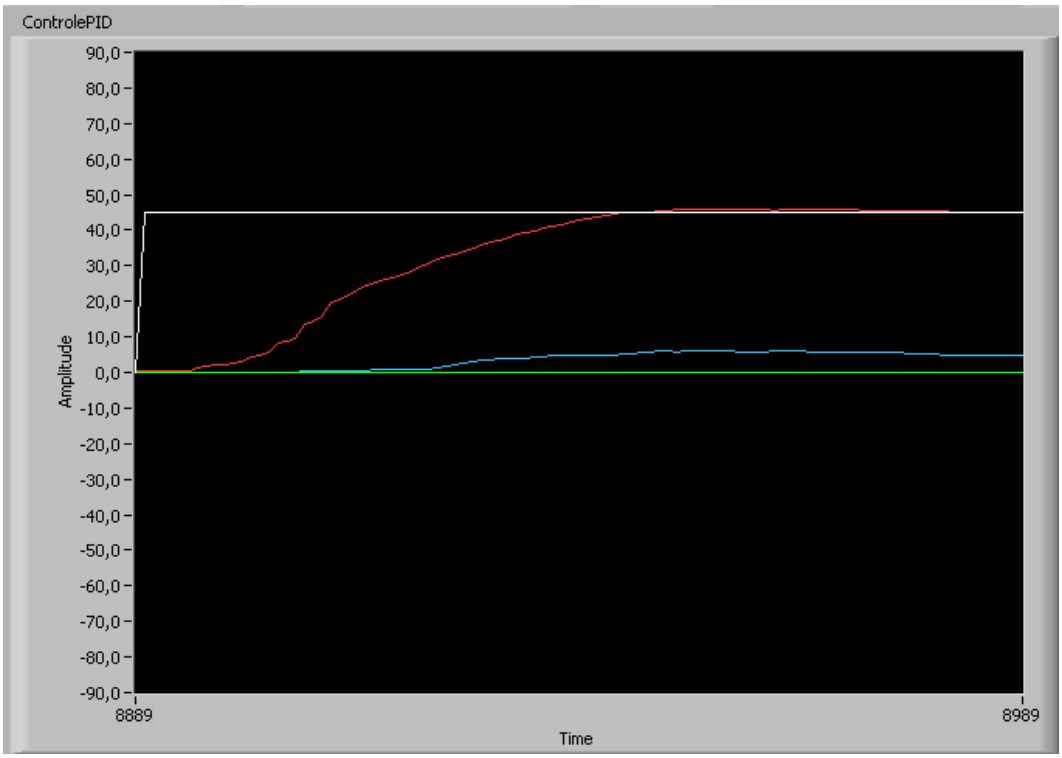


Figura 4.3 - Controle PID – Elos 1 e 2 livres

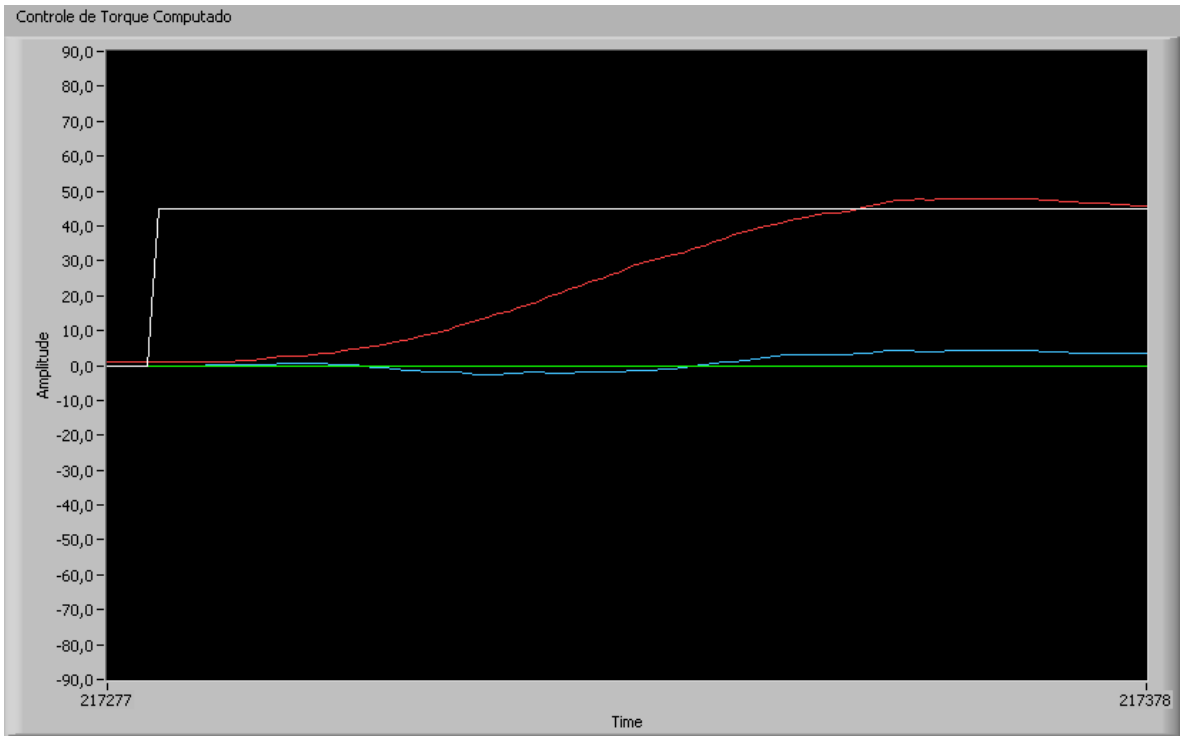


Figura 4.4 - Controle de Torque Computado – Elos 1 e 2 livres

Tendo já calibrado as constantes de ambos os elos, implementou-se uma trajetória circular com diferentes freqüências. Para comparar o desempenho dos dois controles foi realizada uma trajetória circular de raio igual a 0.18 metros e freqüência igual a 0.8 Hz.

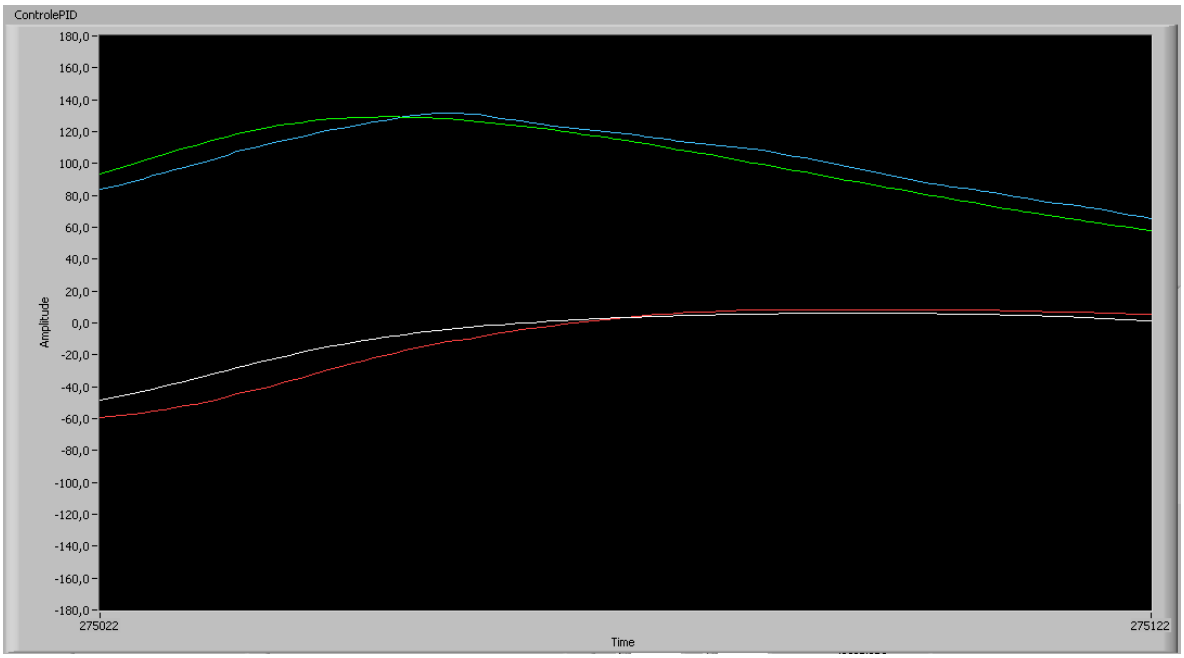


Figura 4.5 - Controle PID - Trajetória circular, raio 0.18 m e freqüência igual a 0.8 Hz

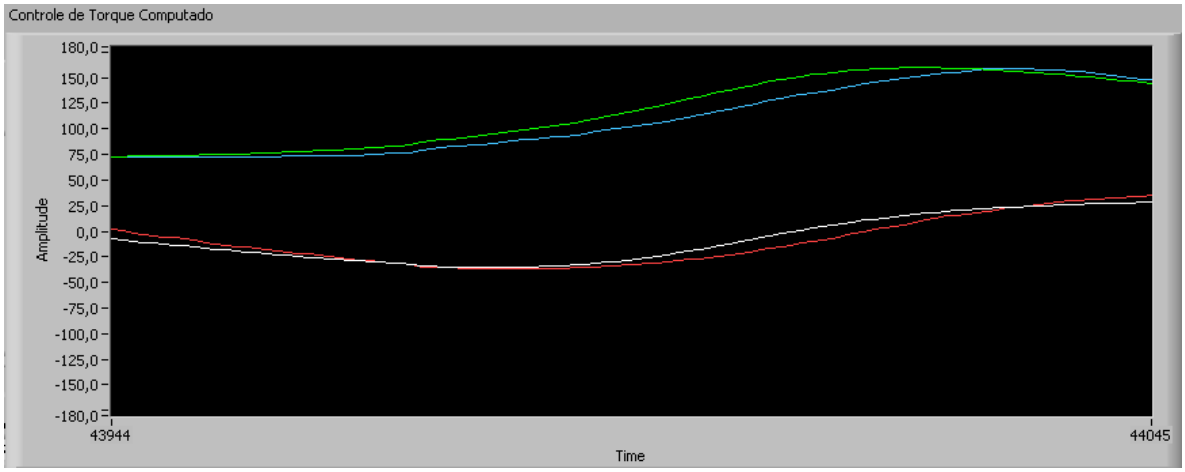


Figura 4.6 - Controle de TC – Trajetória circular, raio 0.18 metros e freqüência igual a 0.8 Hz

Desta vez, ambos os controles apresentaram resultados bem similares. Os resultados, no entanto, divergem quando é variada a freqüência.

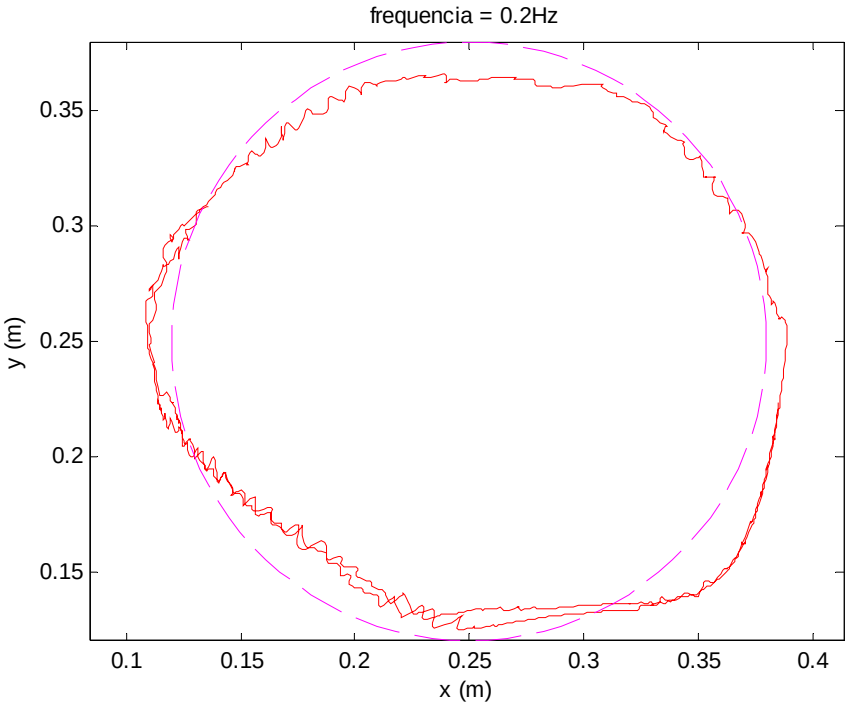


Figura 4.7 – Trajetória circular referente a freqüência de 0.2Hz

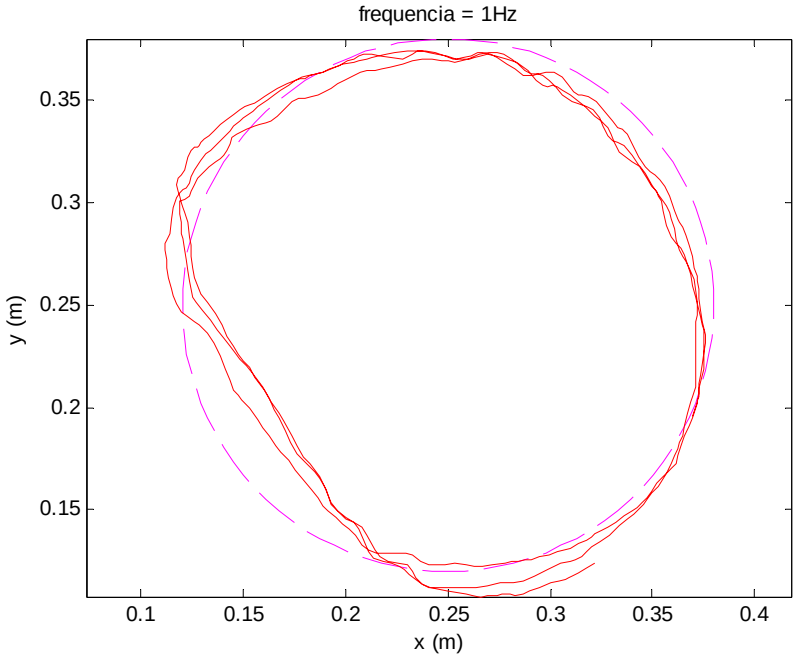


Figura 4.8 –Trajetória circular referente a freqüência de 1.0Hz

Estes resultados mostram que para baixas freqüências os erros residuais são justificados pelo atrito nas juntas do robô. Para altas freqüências esse atrito torna-se desprezível. Entretanto, encontram-se problemas relativos à inércia, o que tende a tornar o círculo em uma elipse. Outros dois fatores importantes para explicar os erros são a falta de estimativas precisas dos parâmetros de inércia e a falta de um controlador de corrente para os motores (diferente do utilizado que era um controlador de voltagem).

5. Discussões e Conclusões

Para a primeira parte, onde foram testadas apenas as mobilidades dos elos, o controle de torque computado apresentou resultados superiores ao PID, tendo a repetibilidade na extremidade do robô de 1,7mm. Já para os testes de trajetória, ambos os controles obtiveram resultados semelhantes sofrendo bastante influência tanto do atrito (baixas freqüências), como da dinâmica e da falta de precisão dos parâmetros de inércia.

Devido a grande quantidade de atrito nas juntas do manipulador, tanto o controle de torque computado quanto o controle PID obtiveram resultados pouco precisos. Para o PID esse resultado já era esperado visto que este não considera os efeitos dinâmicos. Porém para controle de torque computado (que leva em conta a dinâmica) o atrito superou esses efeitos, deixando os seus resultados pouco satisfatórios. Ainda assim no controle de torque computado obteve-se resultados melhores que no PID.

6. Conclusões Finais

Neste trabalho foram desenvolvidos a parte mecânica, eletrônica e *software* de um manipulador de 2 graus de liberdade. Foram implementados controles PID e de Torque Computado. O controle PID não considera os efeitos dinâmicos, portanto só apresentou bons resultados para baixas velocidades angulares (tipicamente menor que 90 graus por segundo). Já o Controle de Torque Computado, que compensa os efeitos dinâmicos, apresentou melhores resultados que o PID, em especial ao percorrer trajetórias em alta velocidade. Ambos os controles ainda assim apresentaram erros de posicionamento, que podem ser atribuídos a um grande atrito nas juntas, e a falta de estimativas precisas quanto aos parâmetros da inércia. Vale ressaltar que para altas velocidades os controles sofreram grandes alterações devido à dinâmica. Estes controles não conseguem compensar estes efeitos, a menos do uso de um ganho integral, o qual prejudica a velocidade de resposta do sistema e pode gerar instabilidade.

7. Referências

1 - ASADA, H., SLOTINE, J.J. **Robot Analysis and Control**. Wiley, 1986.

2 - SPONG, W., VIDYASAGAR, M. **Robot Dynamics and Control**. Wiley, 1989.

3 - CRAIG, J.J. **Introduction to Robotics: Mechanics and Control**. Addison-Wesley, 1986.

4 - OGATA, K. **Engenharia de Controle Moderno**. Prentice-Hall do Brasil, 1990.

8. Anexo

8.1. Código de Programação - LabView

O software de controle do manipulador foi desenvolvido em labview utilizando a função STG.vi para a comunicação direta com o manipulador.

A estrutura principal do programa é composta de dois frames, divididos em inicialização das variáveis e do RoboteQ. É necessário inicializar o amplificador com 2,5 volts (correspondente a zero volts) para que não haja nenhum distúrbio indesejado na saída (motores do manipulador).

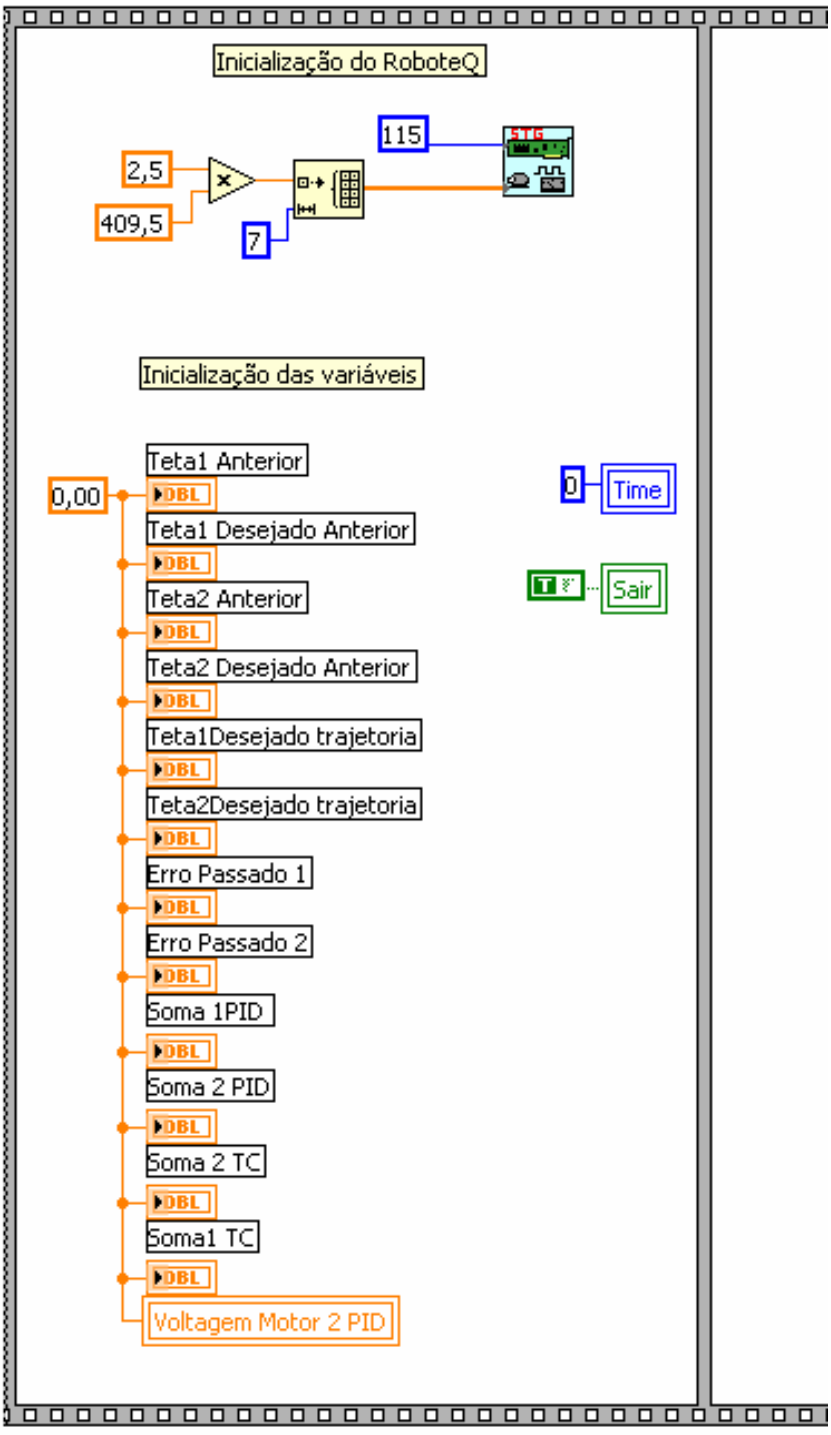


Figura 8.1 – Estrutura principal do programa

O segundo frame é constituído de um loop aonde são realizados os controles PID, Torque computado e controle de trajetórias.

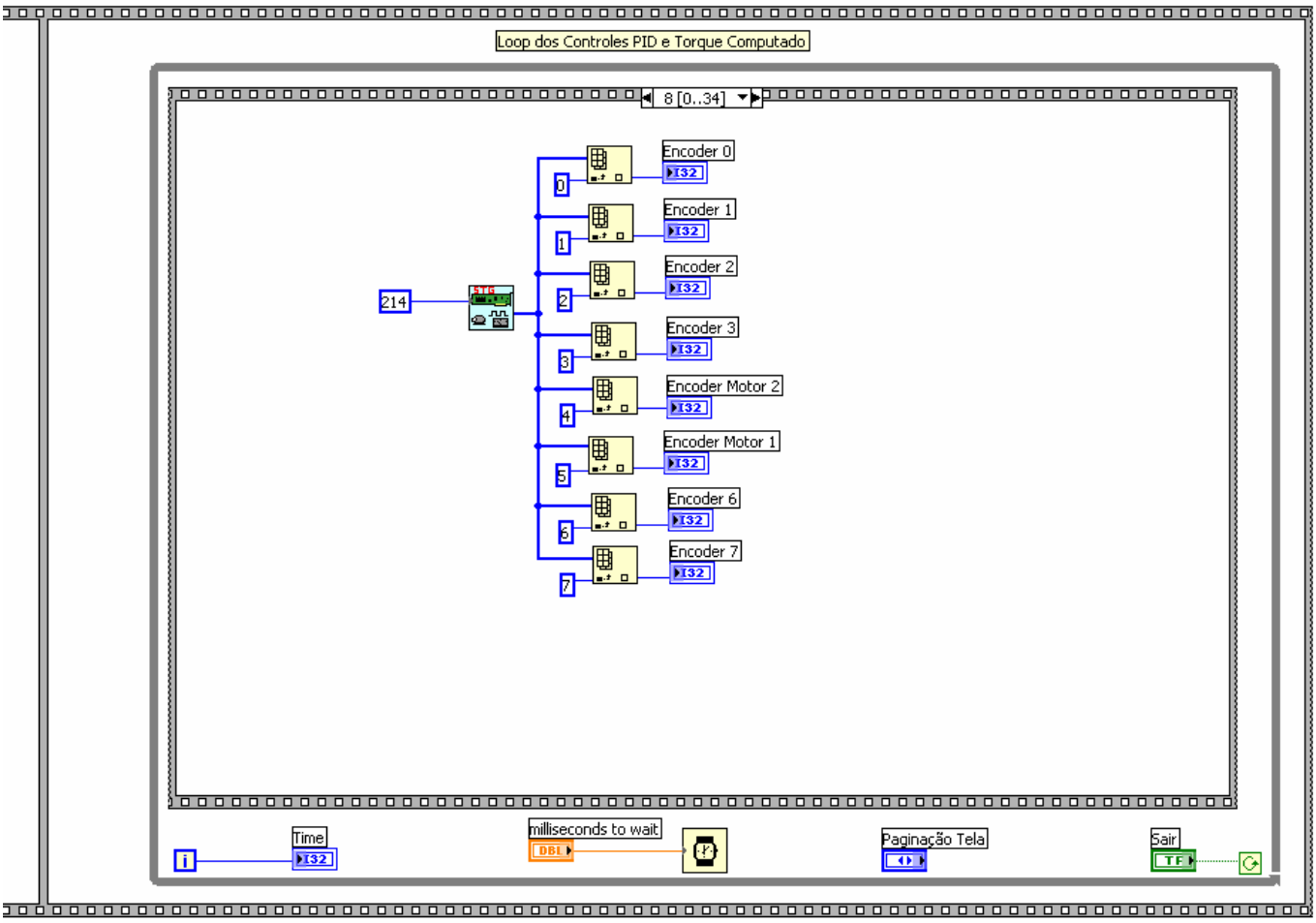


Figura 8.2 – Leitura dos encoderes

Na rotina acima é realizada a leitura dos encoders.

Para iniciar o controle do manipulador, se faz necessário estabelecer um ponto de referência. Isso ocorre porque o encoder conta pulsos, porém não é capaz de guardar posições iniciais. Desta forma toda vez que o programa for iniciado deve-se mover o manipulador à posição considerada

como “zero”, e zerar os encoderes (tornado, assim, a posição como referencial). O botão que reapresenta esta operação no software encontra-se abaixo:

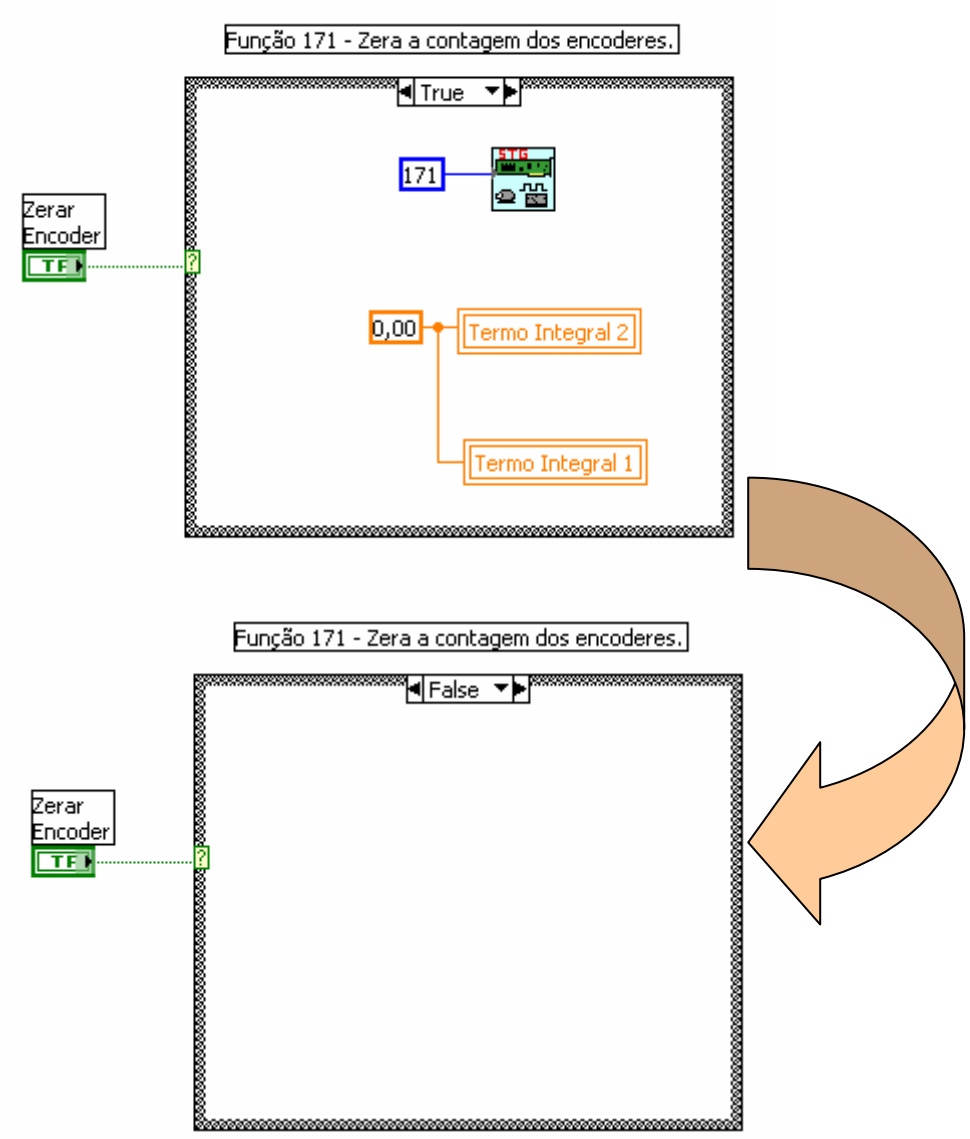


Figura 8.3 – Rotina que zera a contagem dos encoderes

O software em questão foi desenvolvido de forma a atuar tanto com controle PID, como também com Controle de Torque Computado. Para isso o programa internamente realiza as operações das duas formas, porém só emite como resposta a selecionada pelo usuário. Na tela de

operações o comando pode ser emitido através do botão de escolha. Este botão é representado no código de programação da seguinte forma:

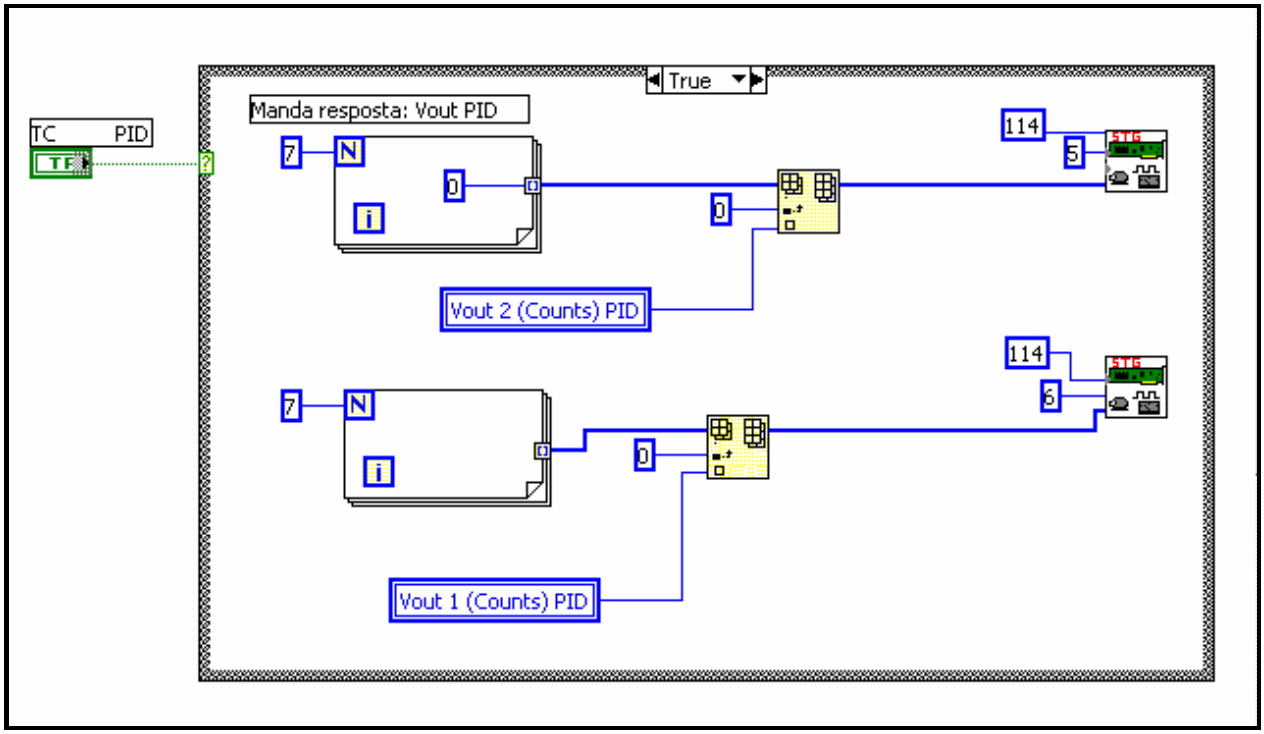


Figura 8.4 – Rotina que envia resposta do Controle PID

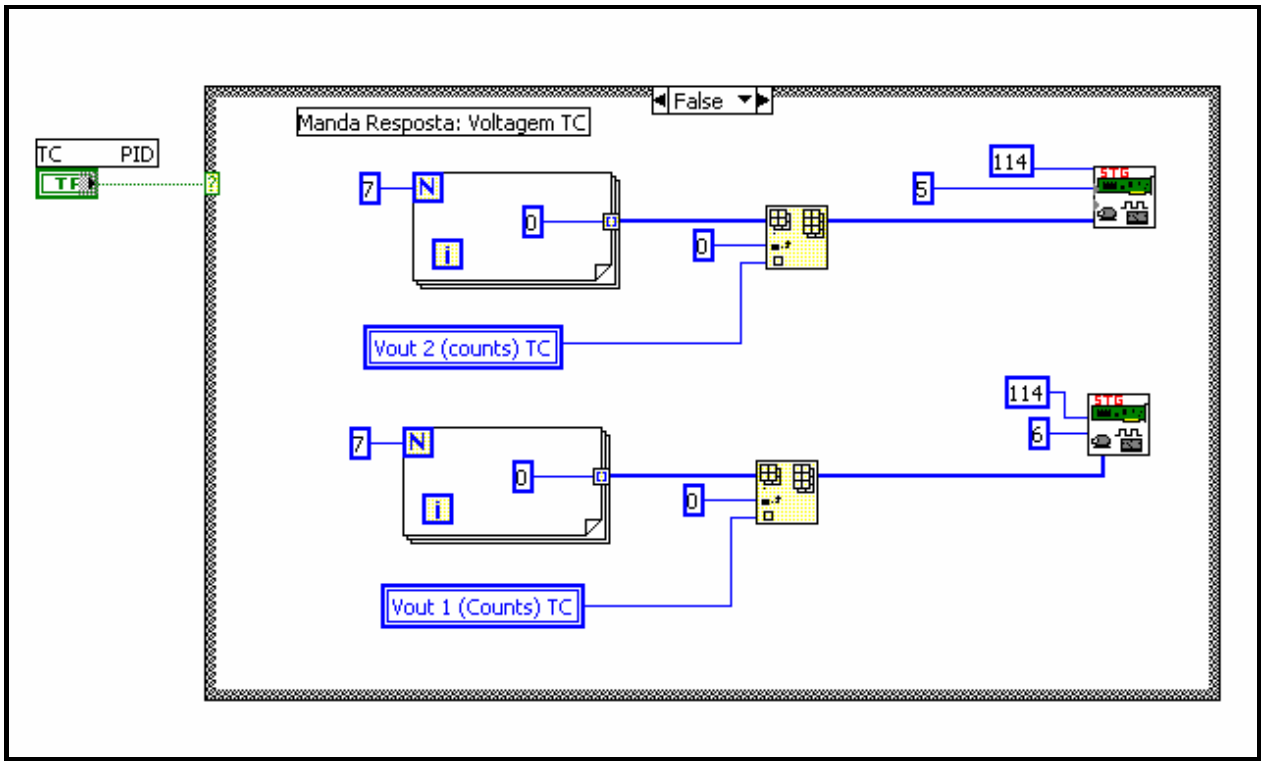


Figura 8.5 – Rotina que envia resposta do Controle TC

Sabendo que a fórmula do controle PID é:

$\tau_i = u_i = Kp_i(q_{des} - q) + Kd_i(\dot{q}_{des} - \dot{q}) + Ki_i \int_0^t (q_{des} - q)dT$, o primeiro passo foi o cálculo do erro:

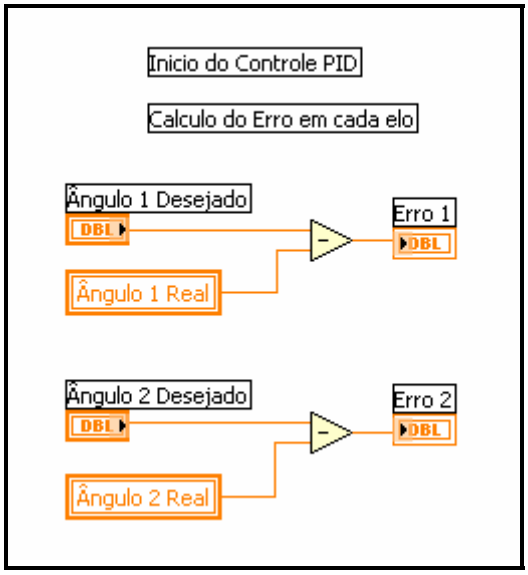


Figura 8.6 – Cálculo do erro de cada elo (Controle PID)

Por se tratar de um programa computacional fez-se necessário discretizar as funções para que o programa fosse capaz de realizar tarefas a cada iteração de tempo. Para isso transformou-se a derivada do erro, na diferença entre o erro atual e o erro da iteração anterior. Analogamente, a integral do erro foi transformada no erro atual mais o somatório do erro das iterações anteriores. Abaixo segue o código de programação referente o controle PID:

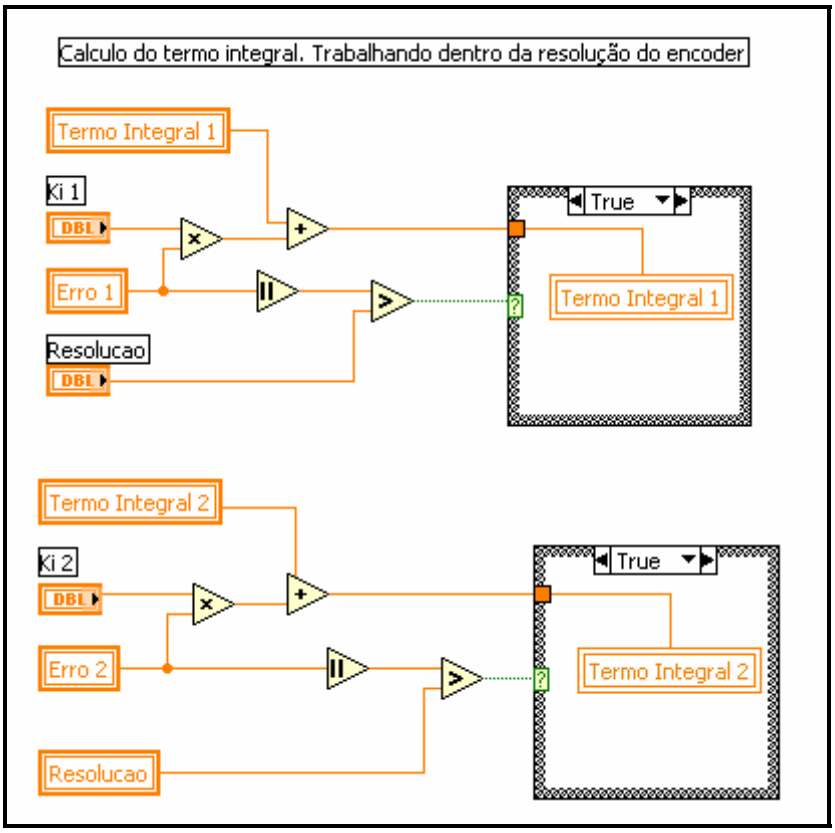


Figura 8.7 – Cálculo do termo integral (Controle PID)

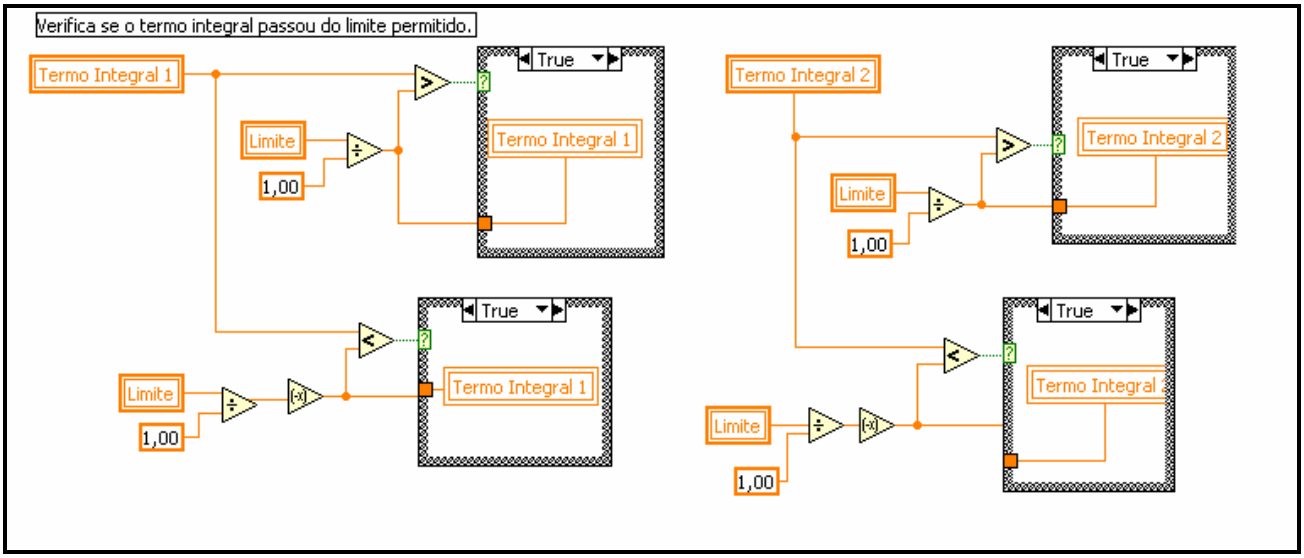


Figura 8.8 – Verifica limite do termo integral

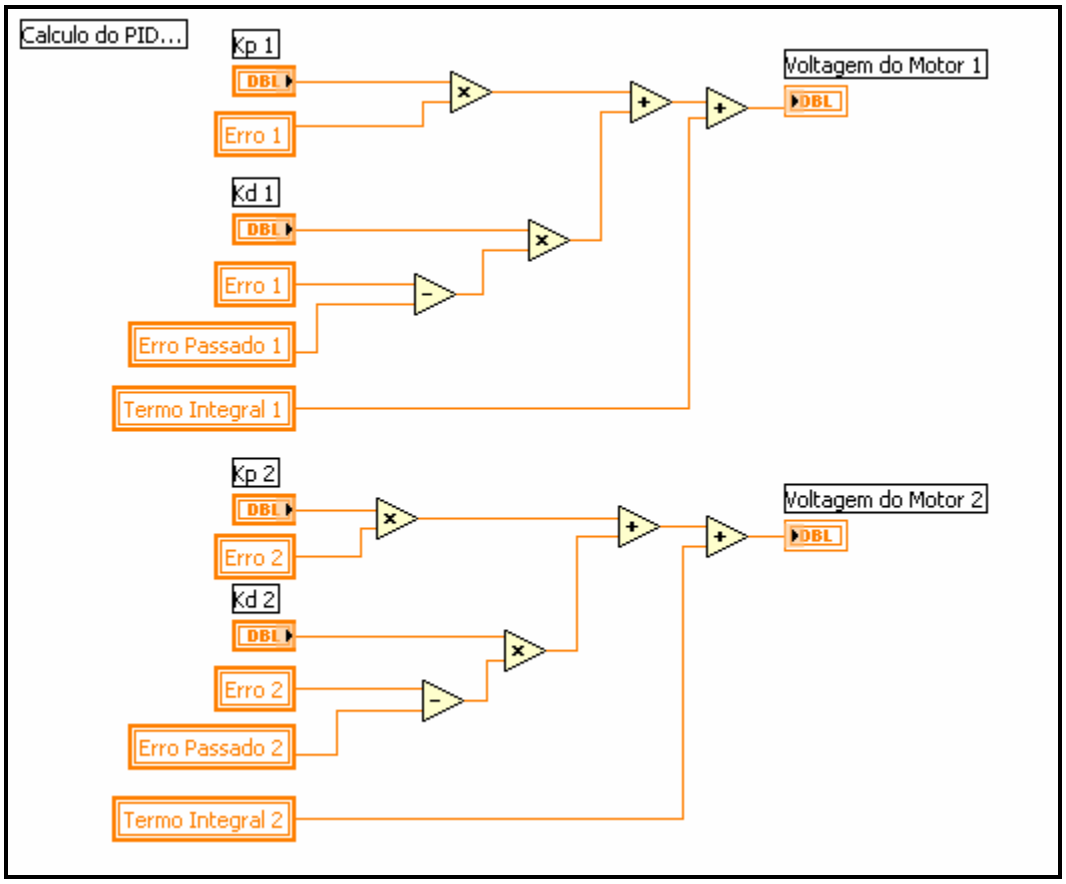


Figura 8.9 – Cálculo da voltagem final (Controle PID)

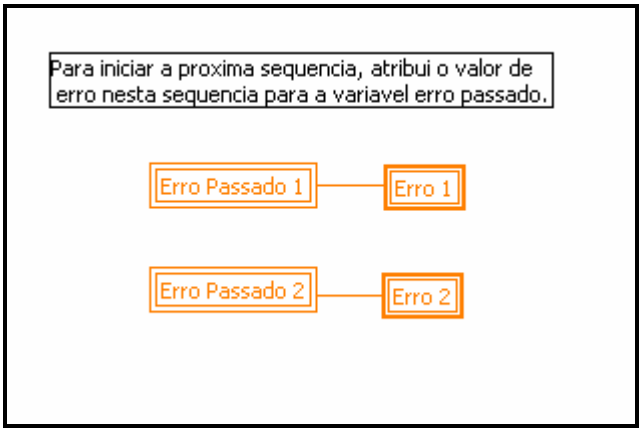


Figura 8.10 – Atualização das variáveis para a próxima iteração (Controle PID)

Após ter calculado a saída final, seria necessário uma calibração da saída, de forma que representasse a transformação do torque (resultante da formula do controle) em voltagem. Esse passo no entanto não foi realizado já que todos os termos do controle (proporcional, derivativo e integral) eram diretamente proporcionais a saída. Sendo assim, ao calibrar as constantes a saída seria diretamente ajustada.

Para a emissão das voltagens ao amplificador e posteriormente aos motores, foram necessárias várias conversões. A primeira delas foi pelo fato do amplificador aceitar apenas sinais de 0 a 5 volts. Determinou-se então uma faixa para a saída do torque (-24 a 24 volts) e fez-se então a conversão.

A segunda conversão necessária foi a conversão para utilização das funções da placa de comunicação (como descrito no item 3.3 - Comunicação entre o manipulador e o computador). Realizou-se então a conversão entre os valores fornecidos à função para os reais em volts segundo o range de 13 bits:

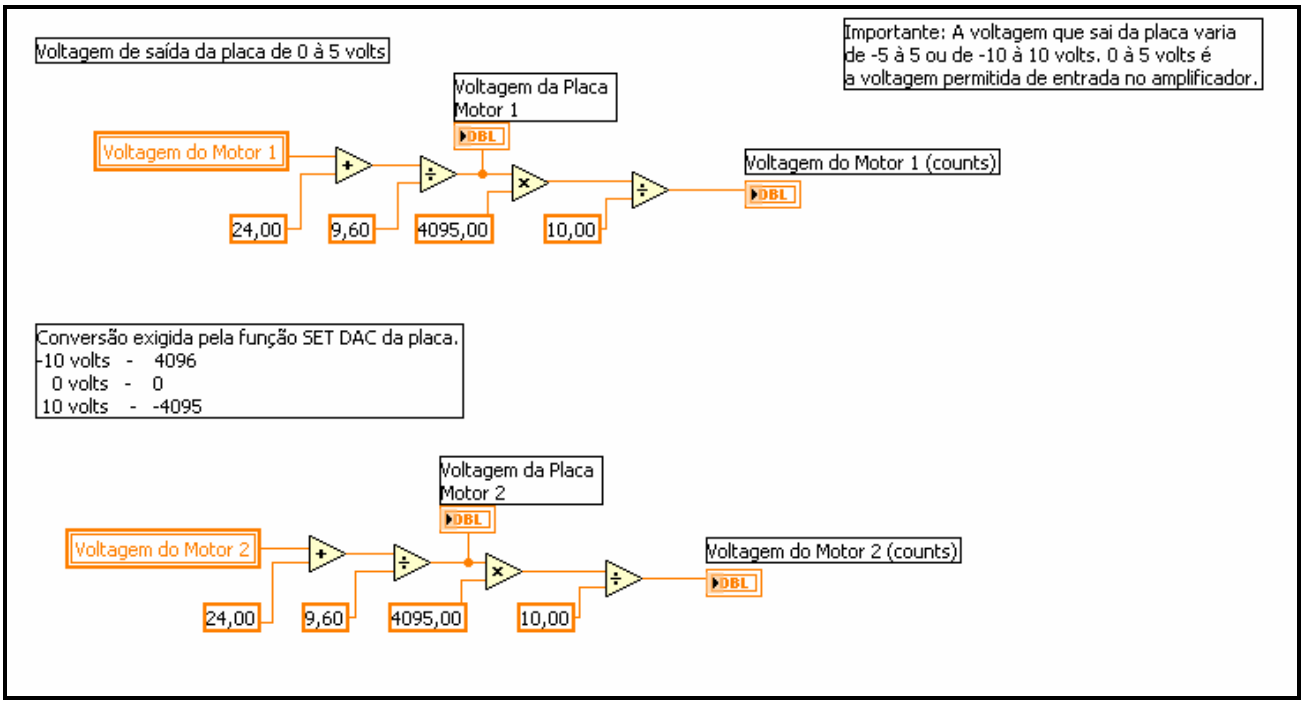


Figura 8.11 – Conversões finais (Controle PID)

Já para o controle de torque computado, foi necessário calcular alguns valores referentes ao peso, a inércia, e a gravidade. Tendo em vista a fórmula do Controle:

$\tau_i = \sum_{j=1}^{n} H_{ij}\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}\dot{q}_j\dot{q}_k + G_i$, onde, H_{ij} são os coeficientes da matriz de inércia do manipulador, h_{ijk} é o coeficiente de três índices de Christoffel, e G_i é o termo gravitacional, foram realizadas as seguintes rotinas de programação:

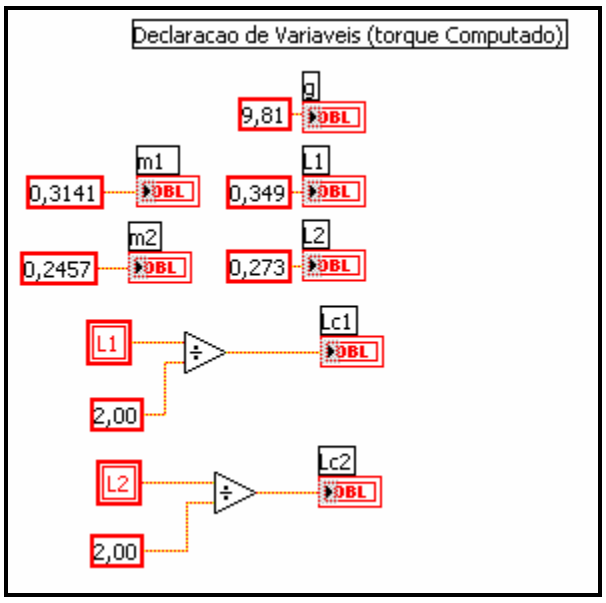


Figura 8.12 – Declaração das variáveis (Controle TC)

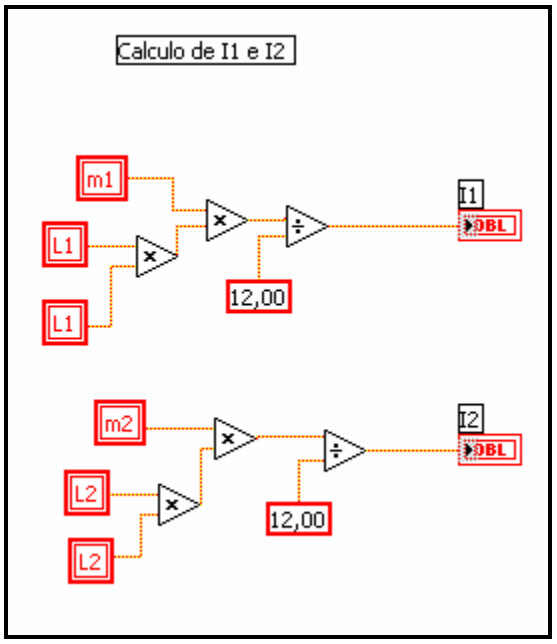


Figura 8.13 – Cálculo de I1 e I2 (controle TC)

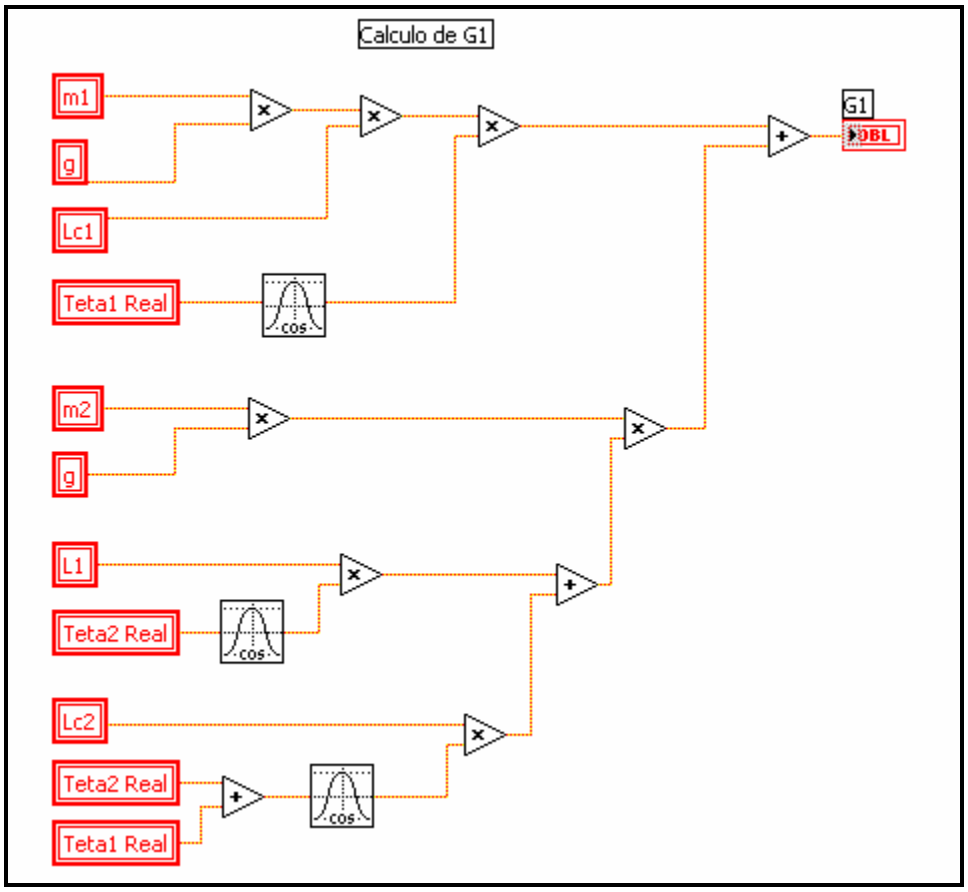


Figura 8.14 – Cálculo de G1 (Controle TC)

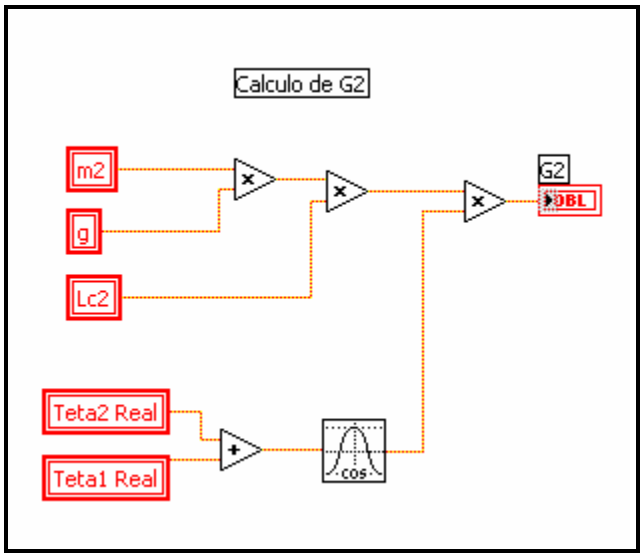


Figura 8.15 – Cálculo de G2 (Controle TC)

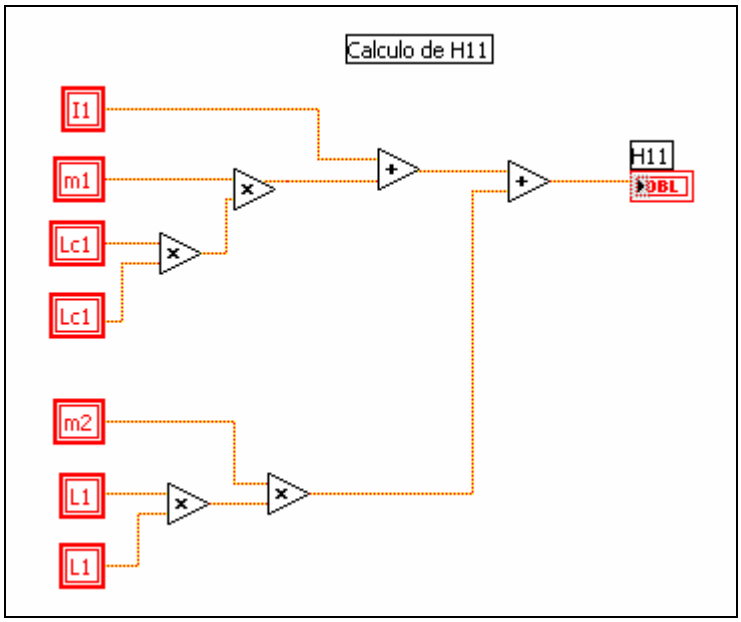


Figura 8.16 – Cálculo de H11 (Controle TC)

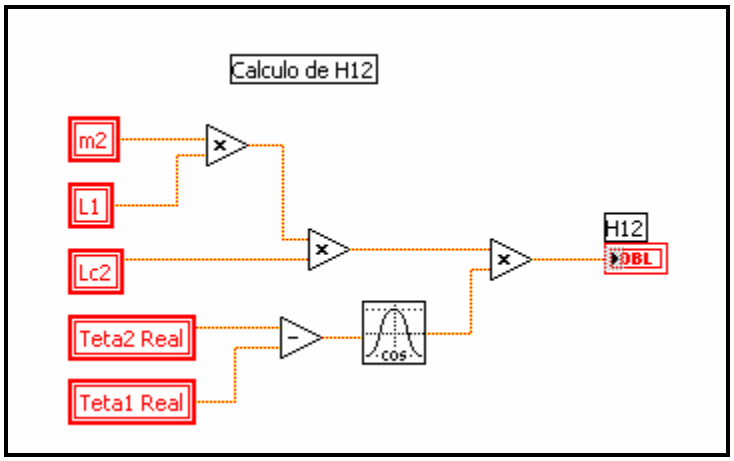


Figura 8.17 – Cálculo de H12 (Controle TC)

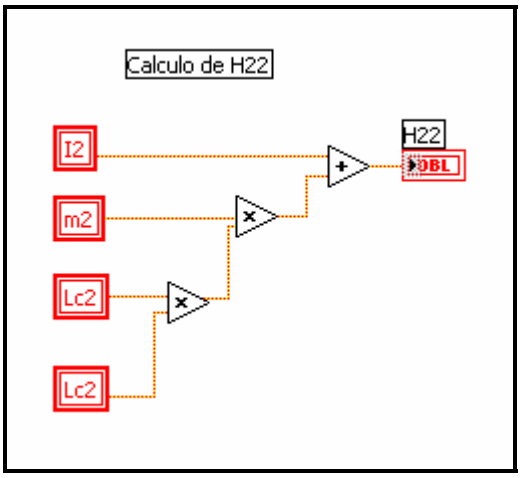


Figura 8.18 – Cálculo de H22 (Controle TC)

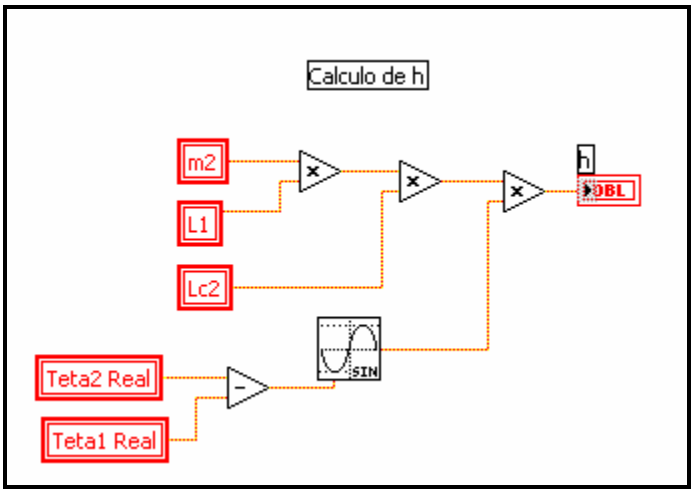


Figura 8.19 – Cálculo de h (Controle TC)

Após a inicialização das variáveis, partiu-se então para o cálculo da do torque final:

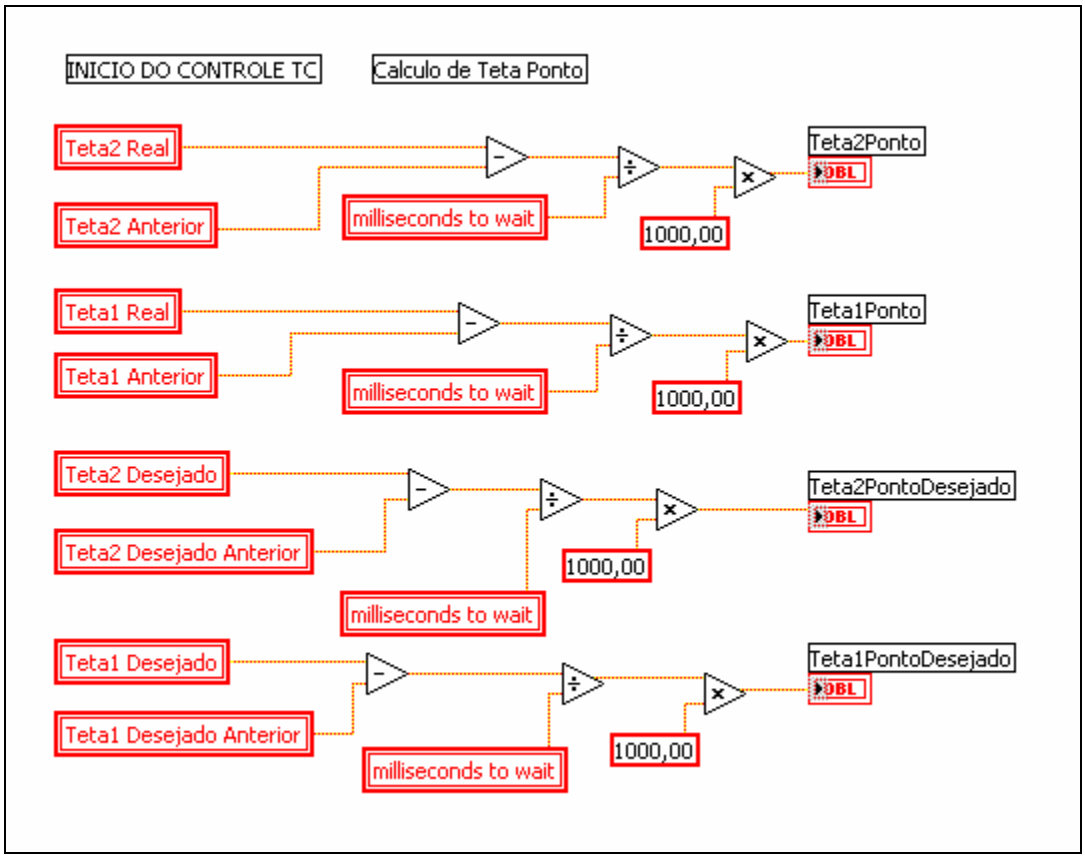


Figura 8.20 – Cálculo dos Tetas Pontos (Controle TC)

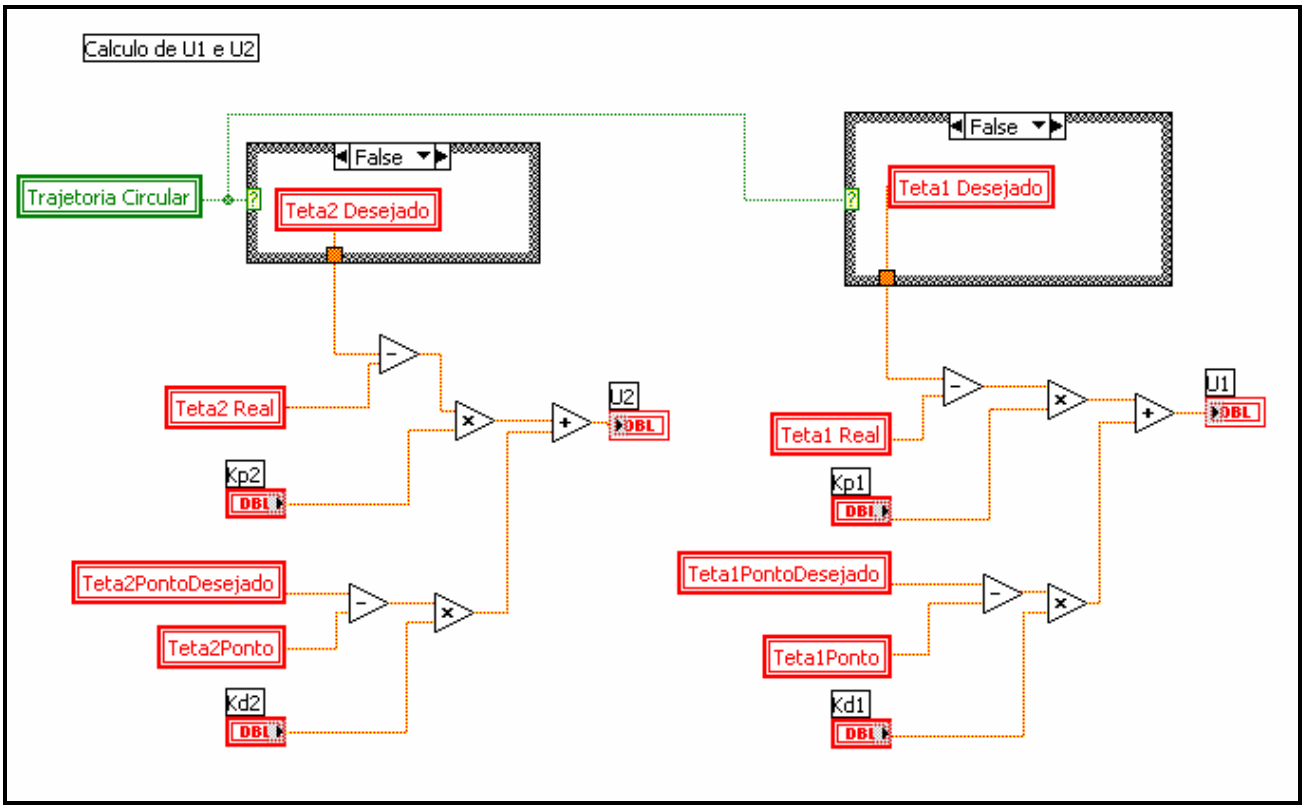


Figura 8.21 – Cálculo de U1 e U2 (Controle TC)

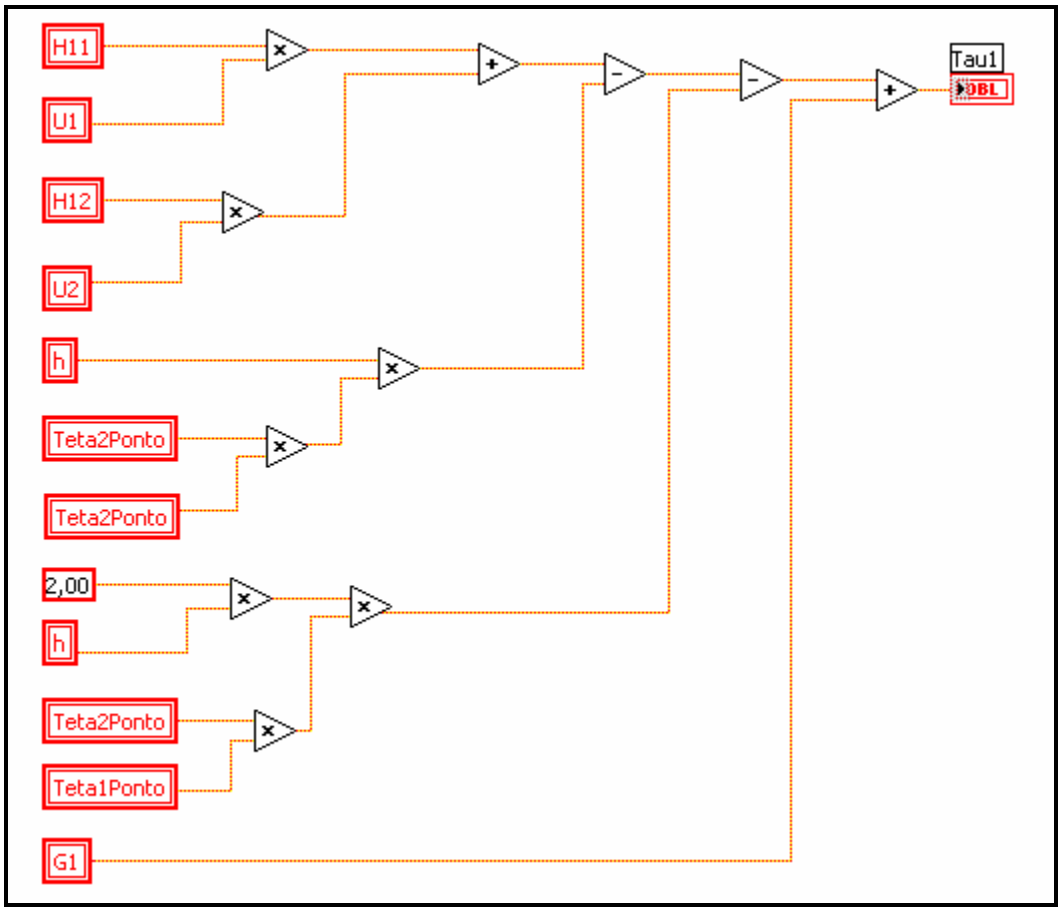


Figura 8.22 – Cálculo do torque Tau1 (Controle TC)

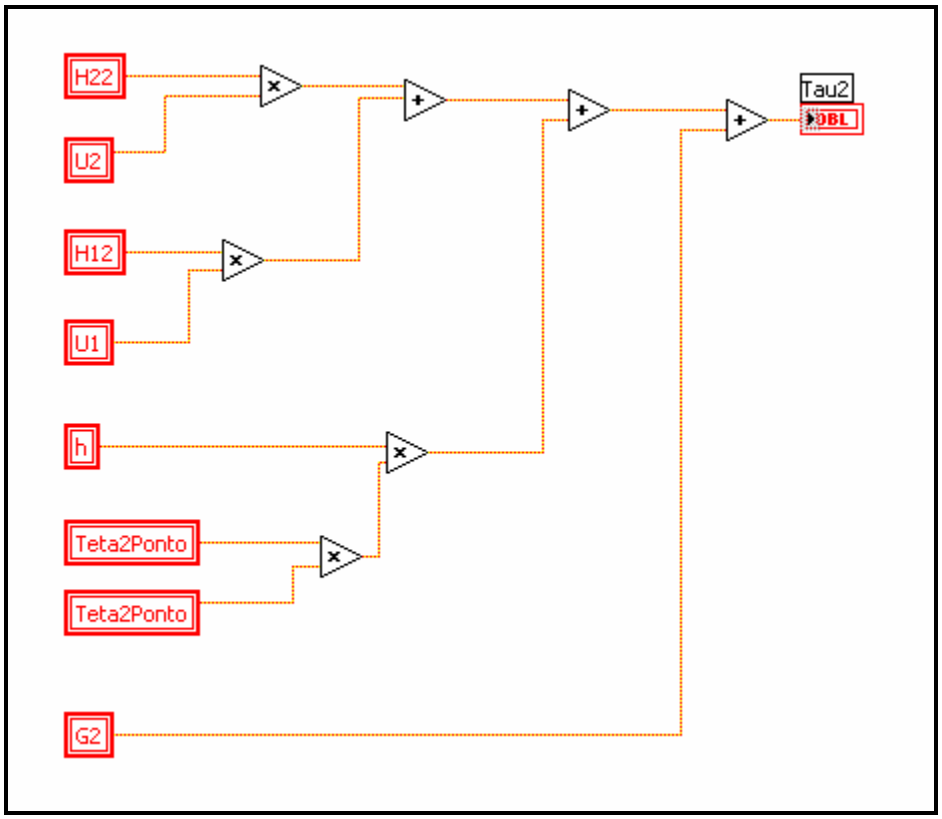


Figura 8.23 – Cálculo do Torque Tau2 (Controle TC)

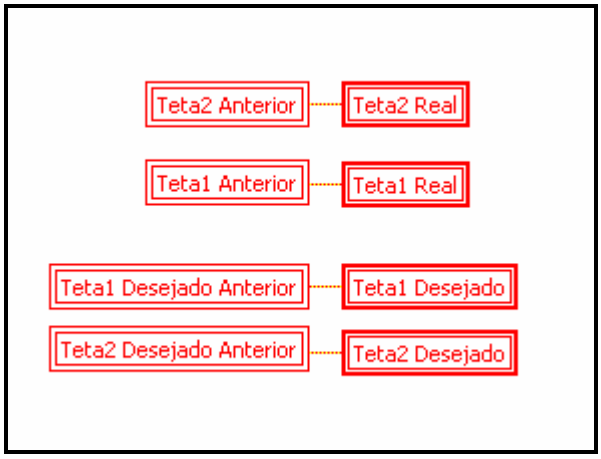


Figura 8.24 – Atualização das variáveis para a próxima iteração (Controle TC)

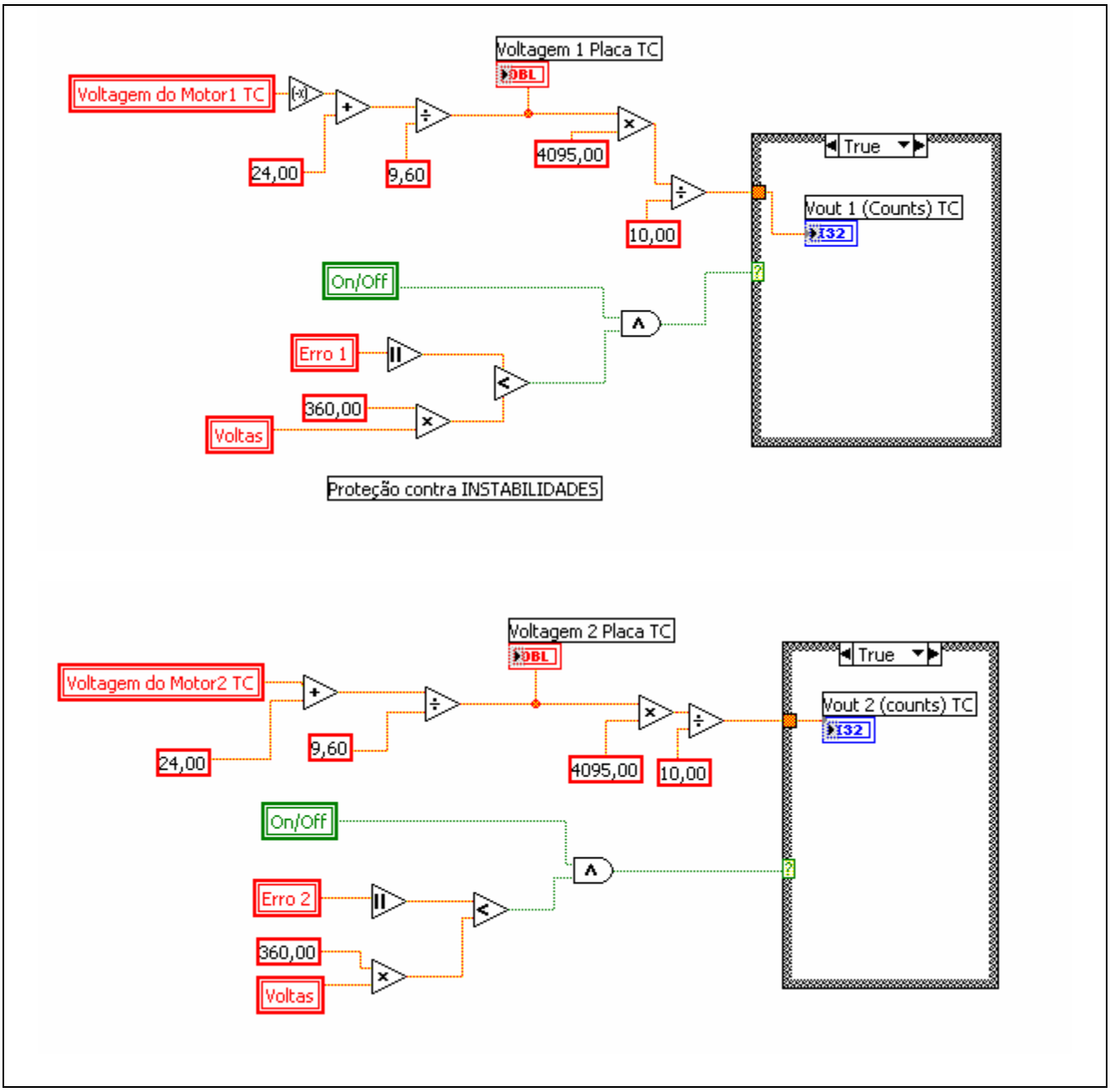


Figura 8.25 – Conversões finais (Controle de TC)

Já para realização da trajetória circular, algumas rotinas tiveram que ser modificadas na programação. Para o cálculo da trajetória, foram utilizadas as equações da cinemática inversa. Para a

utilização do controle de trajetórias era necessário que o usuário escolhesse as coordenadas centrais do círculo, bem como o raio e a freqüência da trajetória. Após esses termos definidos parti-se então para o cálculo das coordenadas da extremidade desejadas:

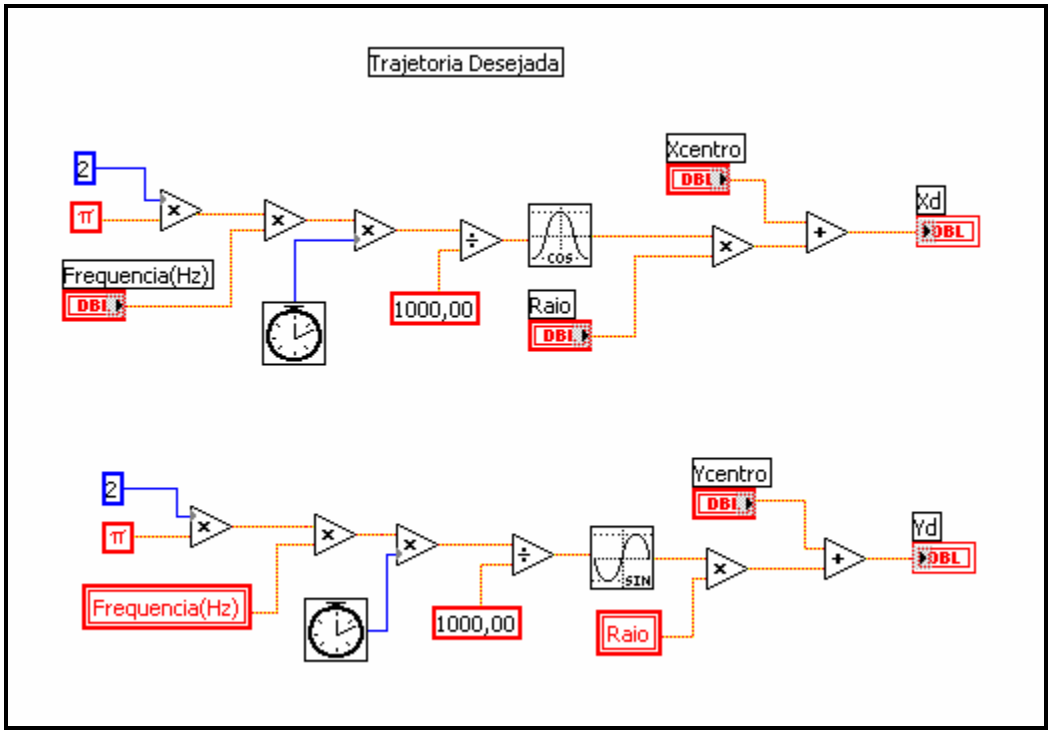


Figura 8.26 – Cálculo das coordenadas desejadas (Controle de trajetória)

Tendo então as coordenadas da extremidade desejadas, utilizam-se as equações da cinemática inversa para obter as coordenadas em forma de ângulos:

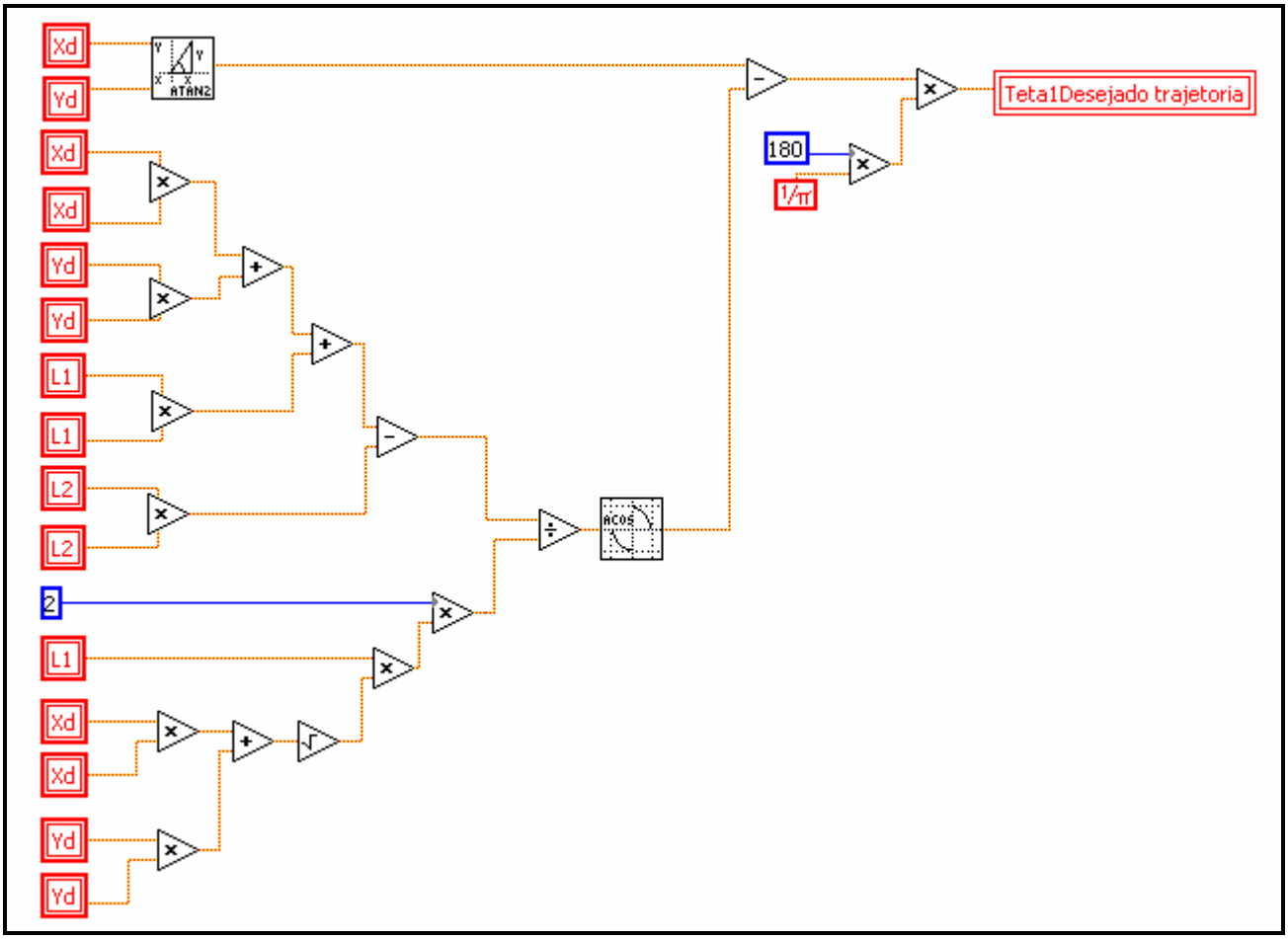


Figura 8.27 – Cálculo de Teta1 Desejado (Controle de Trajetória)

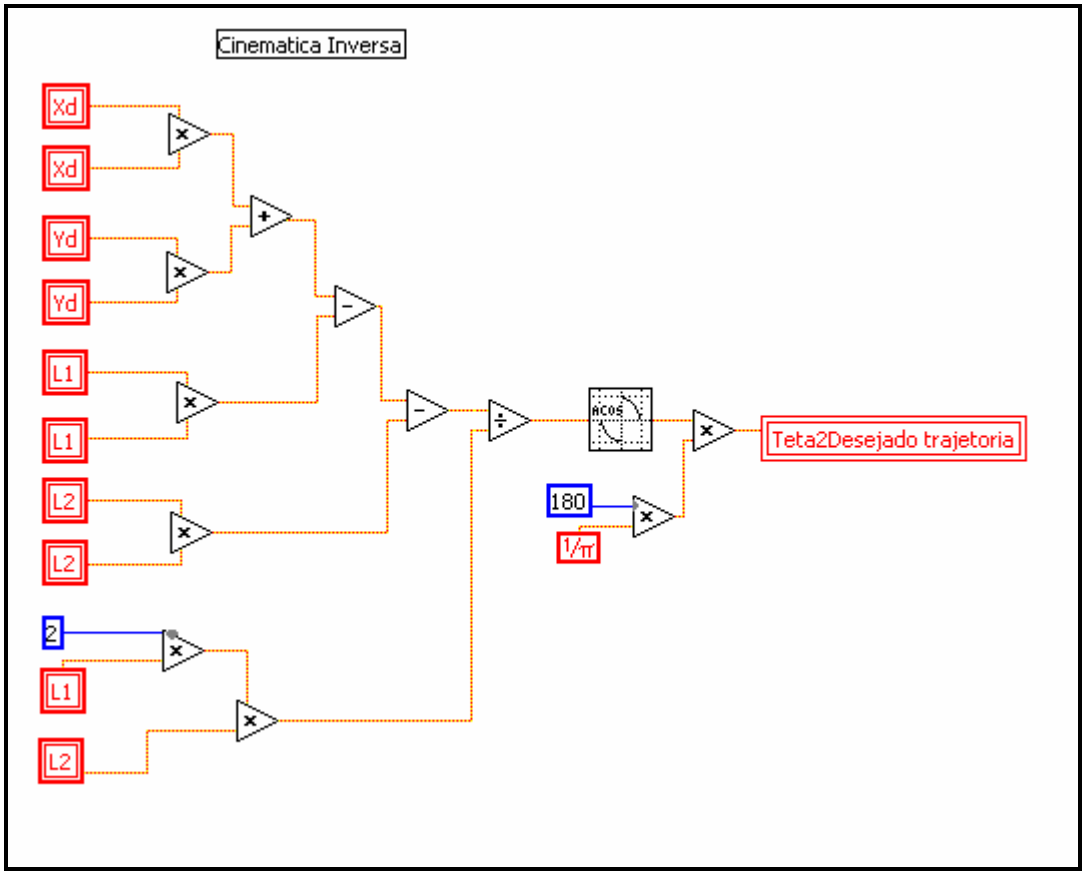


Figura 8.28 – Cálculo de Teta2 Desejado (Controle de Trajetória)

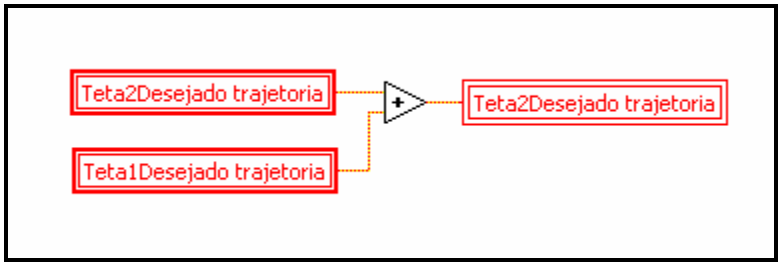


Figura 8.29 – Cálculo de Teta2 segundo as configurações do manipulador (Controle de Trajetória)

Tendo essas informações, atribuímos a um botão a escolha entre o controle da trajetória e o controle de posição simples. Este botão fica encarregado de selecionar os ângulos desejados apropriados, e encaminhá-los ao controle escolhido:

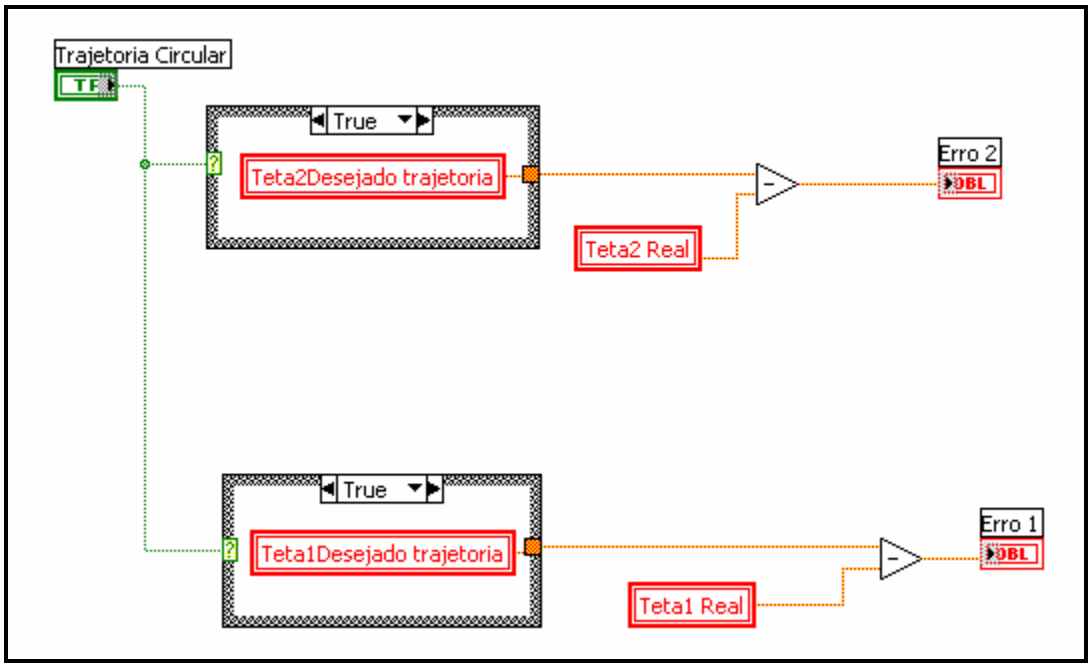


Figura 8.30 – Cálculo do erro para o Controle de Trajetória

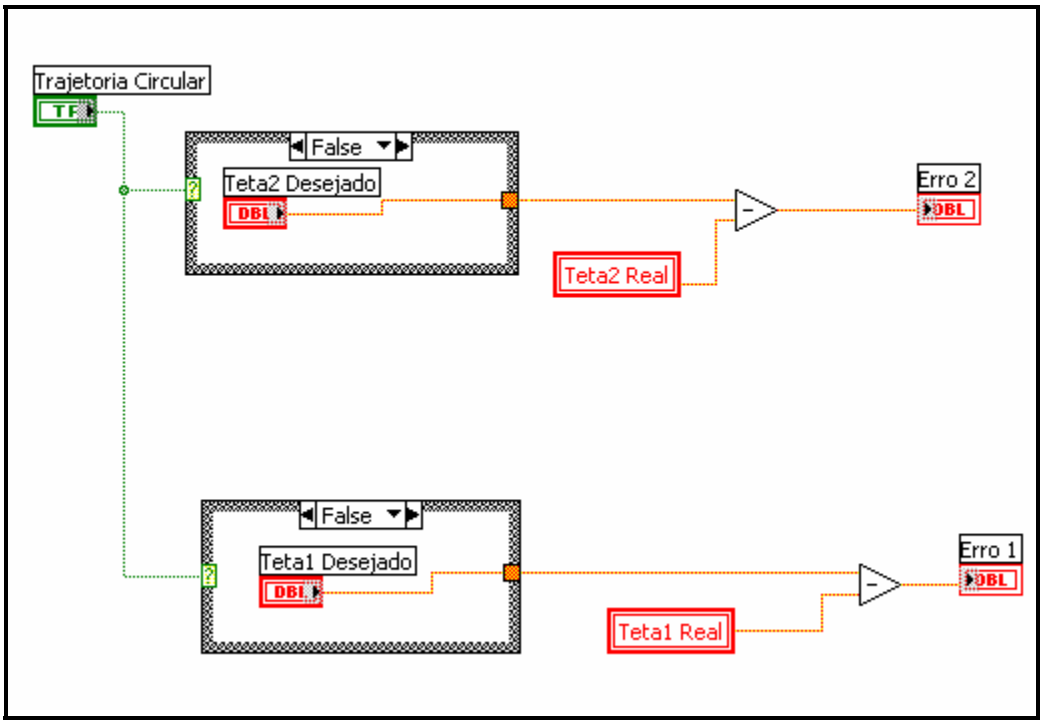


Figura 8.31 – Cálculo do erro para o Controle de posição

Para uma maior segurança foram elaboradas algumas chaves que permitem parar o sistema em caso de emergência, ou que simplesmente limitam a voltagem passada ao manipulador. No primeiro caso é um simples botão que quando acionado leva as saídas para zero volts. No segundo caso é escolhida uma voltagem que será considerada a máxima, sendo assim qualquer valor acima deste não será emitido aos motores.

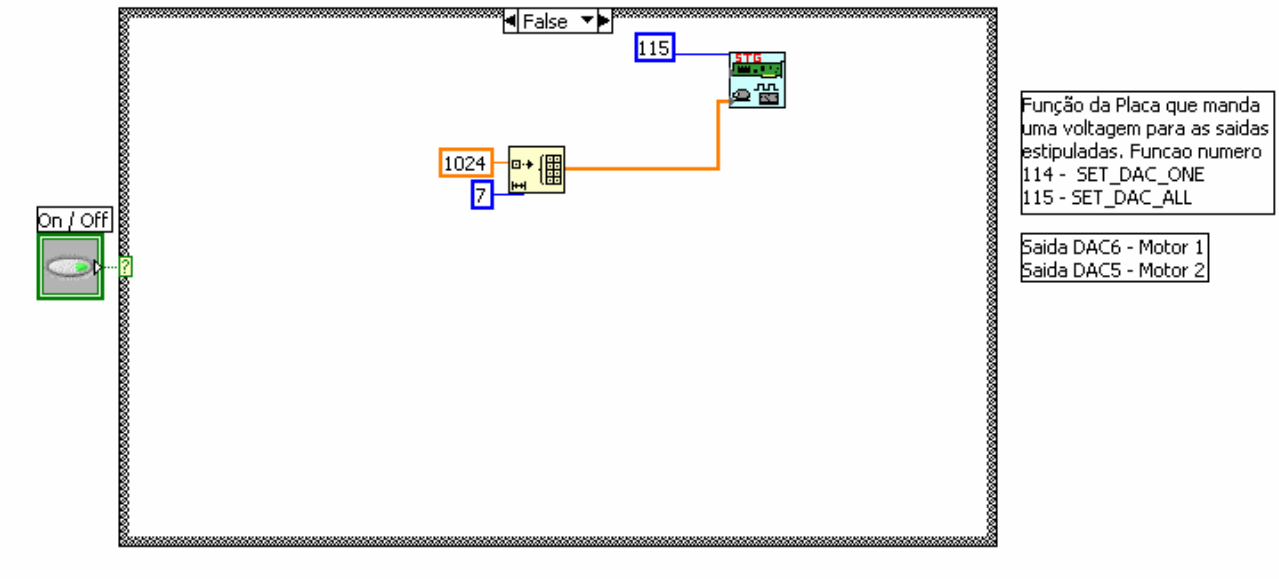


Figura 8.32 – Envio de Zero volts para todas as saídas da placa (Proteção)

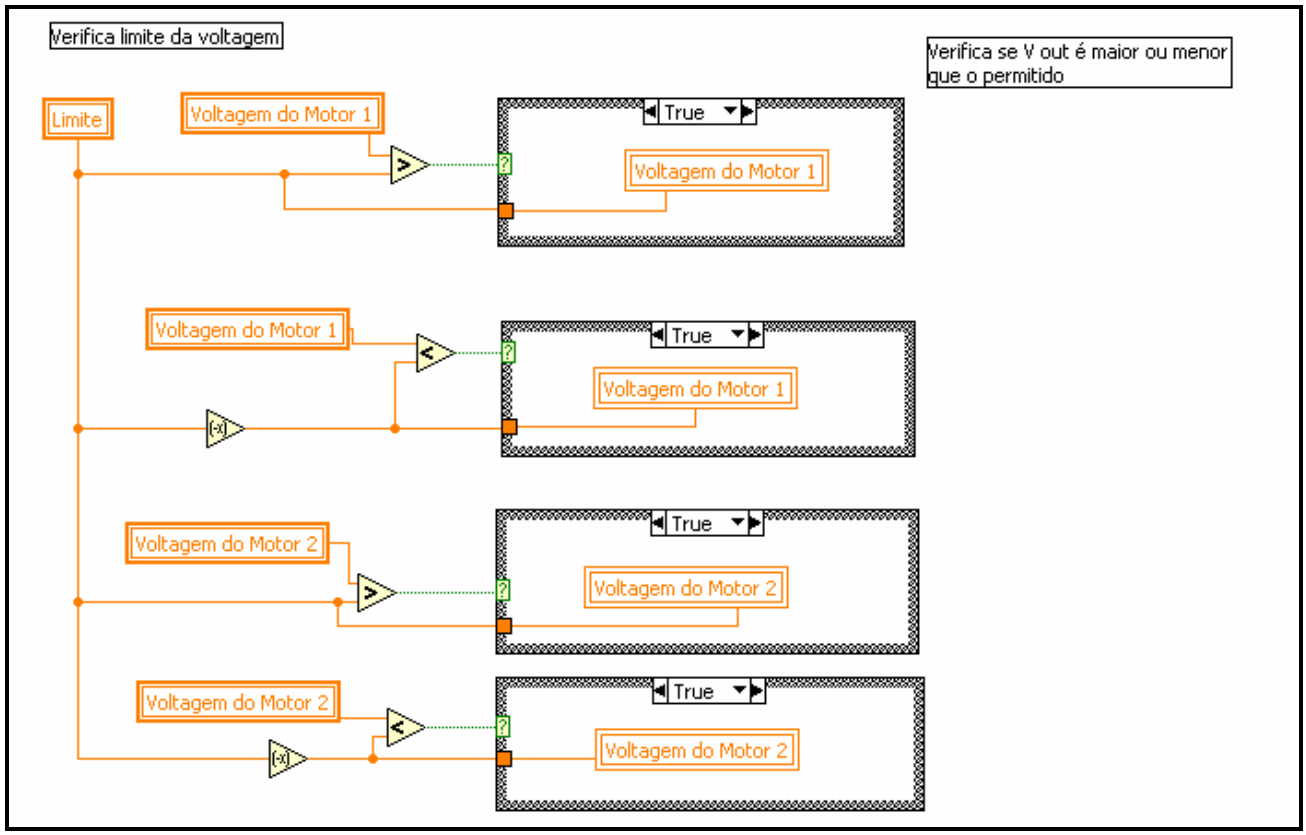


Figura 8.33 – Verifica limite da voltagem (Proteção)

8.2. Especificações do manipulador

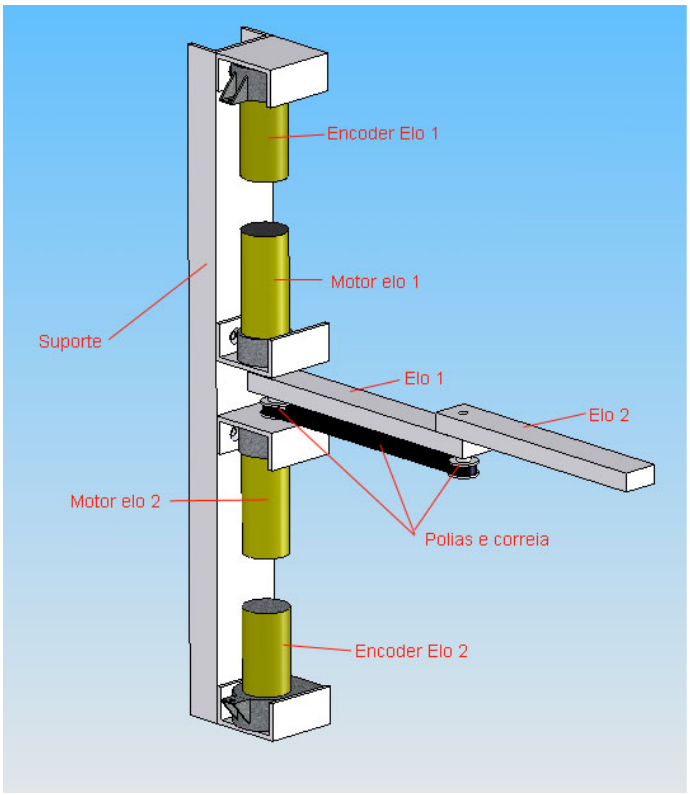


Figura 8.34 – Projeto do Manipulador

<u>Elo 1</u>:

Comprimento: 349mm
Peso: 0,3141g

<u>Elo 2</u>:

Comprimento: 273 mm
Peso: 0,2457g

<u>Motores Atuadores</u>:

Marca: VDO Anteriebstechnik GmbH
24 Volts

<u>Motores Encoderes</u>:

Marca: Buehler
24 Volts