Universidade Federal de Minas Gerais

Engenharia de Controle e Automação

Introdução a Robótica

Trabalho Prático 1

Felipe N. Vianna

Luam C. Totti

Belo Horizonte

2008

Objetivo:

Desenvolver habilidades de programação usando a plataforma da handyboard, calibração de movimentos através de odometria. Montagem mecânica capaz de traduzir bem os sinais enviados aos motores através da handyboard. A montagem completa considerando a programação deve ser capaz de proporcionar movimentos em forma de quadrado e círculo.

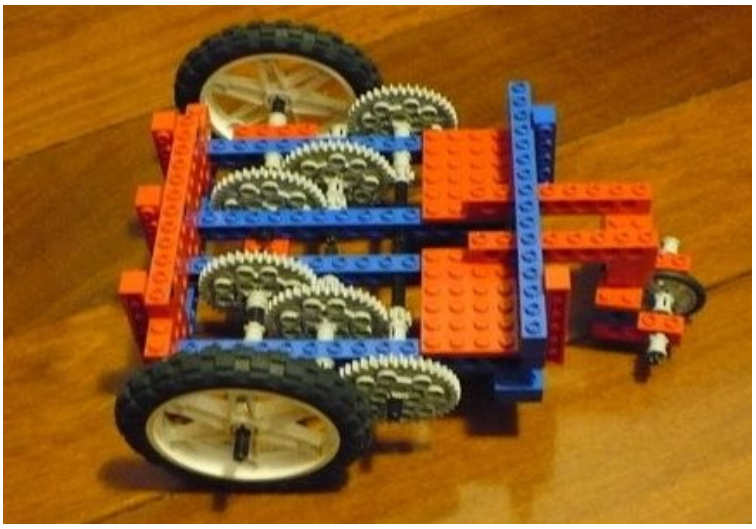
Desenvolvimento:

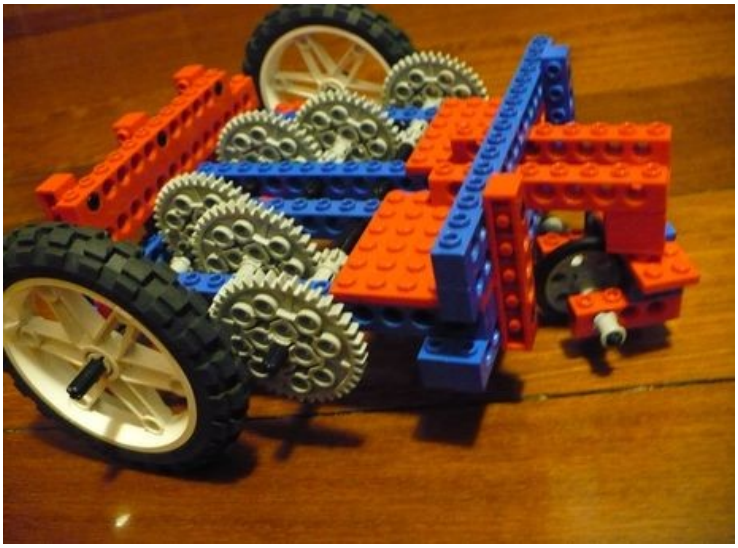
Inicialmente partimos para uma montagem mecânica com o LEGO enquanto estudávamos o manual da handyboard. Para a parte mecânica, foi concebida a idéia de um triciclo que possuísse tração independente nas duas rodas traseiras. A terceira roda seria só um apoio que permitisse curvas em qualquer direção. Para tanto, fez-se uma "roda biruta" assim como as de carrinho de supermercado.

Existia a opção de adquirir um "roll on" pronto com peça LEGO já adaptada para este trabalho, mas consideramos mais desafiador construir uma roda biruta completamente com as peças disponíveis no kit.

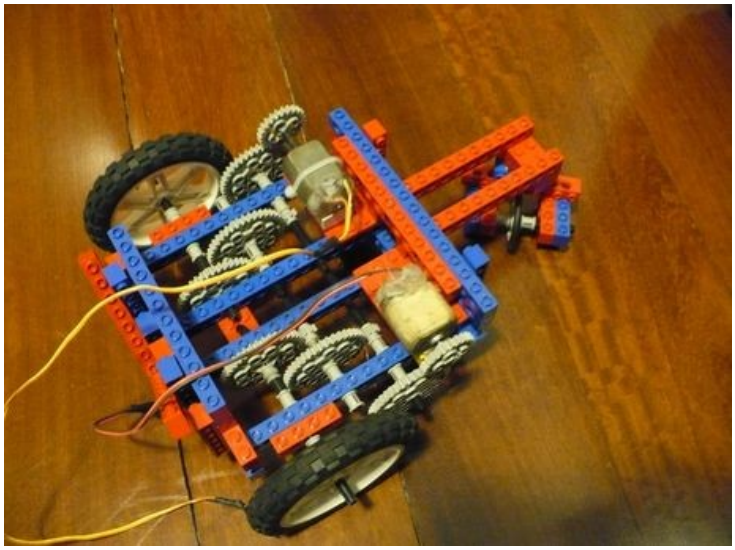
A primeira parte da montagem foi o chassi com reduções e engrenamentos independentes para cada uma das rodas. A redução foi de aproximadamente 40 vezes. Seria de 125 vezes, porém a menor engrenagem acoplada ao eixo do motor não se engrenou bem com a engrenagem da redução e tivemos que usar a engrenagem com quantidade de dentes superior. Visto que o apoio era sobre rodas, não se fez necessário um grande torque.

As ilustrações abaixo mostram em detalhes o chassi muito bem travado e com as reduções para cada uma das rodas:





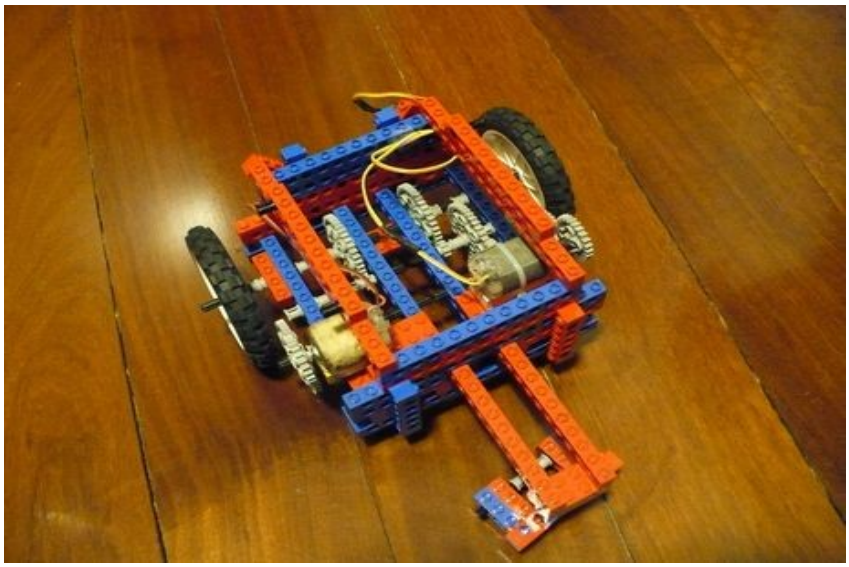
Chassi com reduções e primeira versão da roda biruta



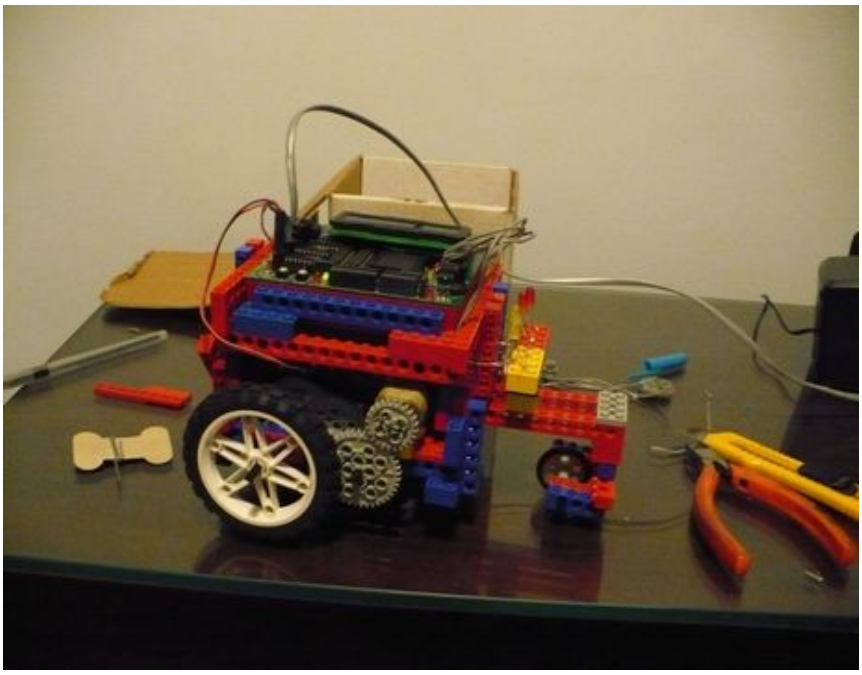
Chassi com motores

A roda biruta sofreu diversas modificações após os testes, pois percebemos que ela não respondia bem às curvas, e também por se mostrar bastante frágil.

Tendo então acoplado os motores com qualidade, conferimos a estrutura e começamos a montar a “cabine” que receberia a handyboard. Por questões de orgulho pessoal não quisemos usar fita adesiva, apenas componentes LEGO para realizar o acoplamento da handyboard à estrutura móvel do robô. Tomando bastante cuidado para manter uma estrutura bem rígida, levantamos paredes laterais mantendo um vão livre para passagem de fios para a handyboard. A estrutura intermediária e a final são mostradas pelas fotos a seguir:



Montagem parcial da cabine



Montagem completa da cabine

Handyboard:

A HandyBoard utliza o Interactive C como linguagem de programação. Essa linguagem se assemelha muito ao ANSI C, incluindo algumas funcionalidades extras como o suporte à multi-threads e o tratamento de erros de execução. Entretanto, a linguagem oferece menos recursos de controle de memória e menos eficiência, pois é interpretada. Além disso, a popularidade da linguagem C fez com que inúmeras bibliotecas se tornassem disponíveis para propósitos diversos, enquanto IC não dispõe da mesma quantidade de recursos.

Apesar das restrições, para programadores habituados com C, a linguagem é extremamente intuitiva. Desenvolver o programa utilizado nesse trabalho não foi uma tarefa muito complicada. As primeiras iniciativas visaram criar um menu interativo e intuitivo. Para isso, utilizou-se a função de leitura do knob e dos botões start e stop.

Um fator complicante da linguagem é que se deve adotar um paradigma de programação orientada a eventos, ou seja, eventos na interface (botões e knob) da handyborad devem acionar tratadores associados dentro do programa. Entretanto, implementar sistemas desse tipo sem tratamentos de interrupções é mais complicado. Dessa forma, deve-se modelar o programa de modo que se espere uma certa entrada em cada momento do programa. Ou seja, como se precisa usar a função start_button() para captar o pressionamento do botão start, devemos permanecer em um loop verificando o botão start para captá-lo eventualmente, enquanto o botão stop não será detectado dentro desse loop. Esse modelo restringe um pouco as possibilidades de interação, porém pode-se fazer aplicações bastante funcionais através de interfaces bem elaboradas.

O menu disponibiliza as seguintes opções, navegáveis através do knob:

1 - Calibrate motors
2 - Set time interval
3 - Set motor power
4 - Start multi-thread
5 - Move Robot

Os nomes descrevem as respectivas funcionalidades associadas. Quando uma opção é escolhida, uma variável armazena essa opção, o loop termina e a funcionalidade é acionada. A função calibrateMotor permite o usuário associar dois valores inteiros de 0 a 255 para cada motor. Esses valores irão descrever a razão de potência entre os dois motores, ou seja, escolher 100 e 50, diz que o motor A deve ser ativado com 2 vezes mais potência para que se obtenha a mesma rotação.

Os itens 2 e 3 permitem ao usuário escolher os valores de tempo e potência utilizados para acionar os motores. Nessa versão do menu não é interessante usar nenhum dos três itens iniciais, pois isso irá desconfigurar os movimentos do item 5, que será em breve explicado.

Item 4 inicia a seqüência de LEDs junto do movimento arbitrário do robô. Para realizar essa tarefa, tivemos que utilizar os recursos multi-thread da linguagem. Basta criar duas funções que realizam as tarefas e chamá-las usando a função start_process(). Isso fará a chamada da função passada como argumento de forma não blocante. Entretanto, para se implementar recursos como o timer de 30 segundos e o controle dos LED através do botão stop, thread adicionais devem ser criadas.

Uma das formas de se controlar o tempo de execução de uma tarefa é iniciá-la junto de outra thread que contém apenas uma função sleep e uma atribuição de variável após o sleep. Com isso, a tarefa a ser controlada deve verificar periodicamente o valor da variável de controle. Isso se assemelha a verificar as funções start_button e stop_button, porém oferece uma flexibilidade maior, ao custo de precisão no controle. Esse mesmo mecanismo pode ser usado para observar o botão stop enquanto a seqüência de LED é tocada. Basta que a cada nota tocada seja verificada a variável de controle, definida por uma thread que é executada todo tempo e apenas observa o botão stop.

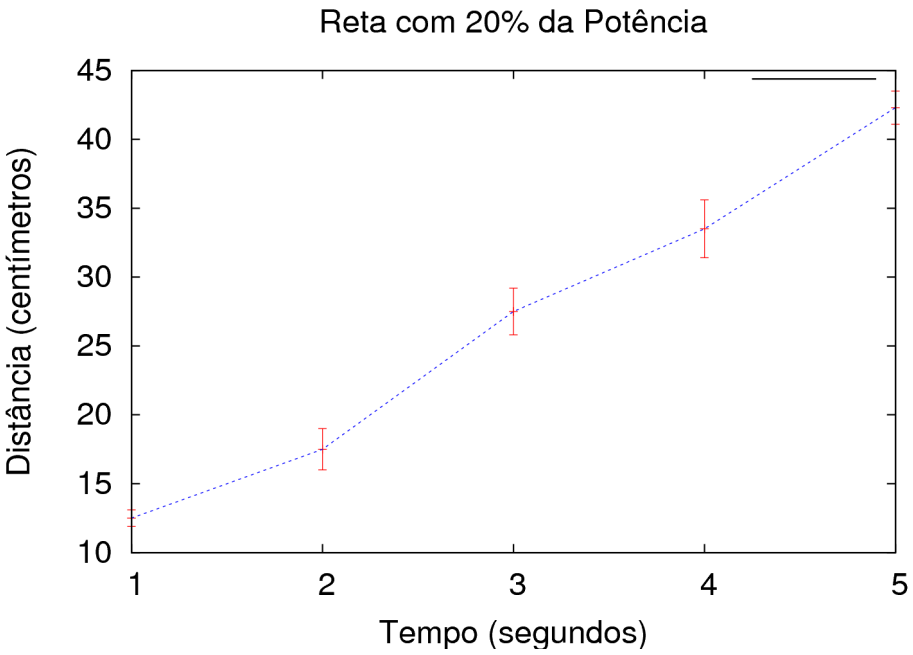
O último item é responsável pelo movimento em quadrado do robô, seguido do movimento circular. A fim de simplificar essa tarefa, uma função foi criada. Essa função recebe dois inteiros, a potência e o tempo de acionamento dos motores. Para se realizar os movimentos pedidos na proposta, basta configurar uma seqüência de chamadas a essa função corretamente. Para se realizar curvas, basta associar os motores com fatores previamente obtidos para realizar um certo raio, assim como o tempo, para percorrer um certo comprimento de curvatura.

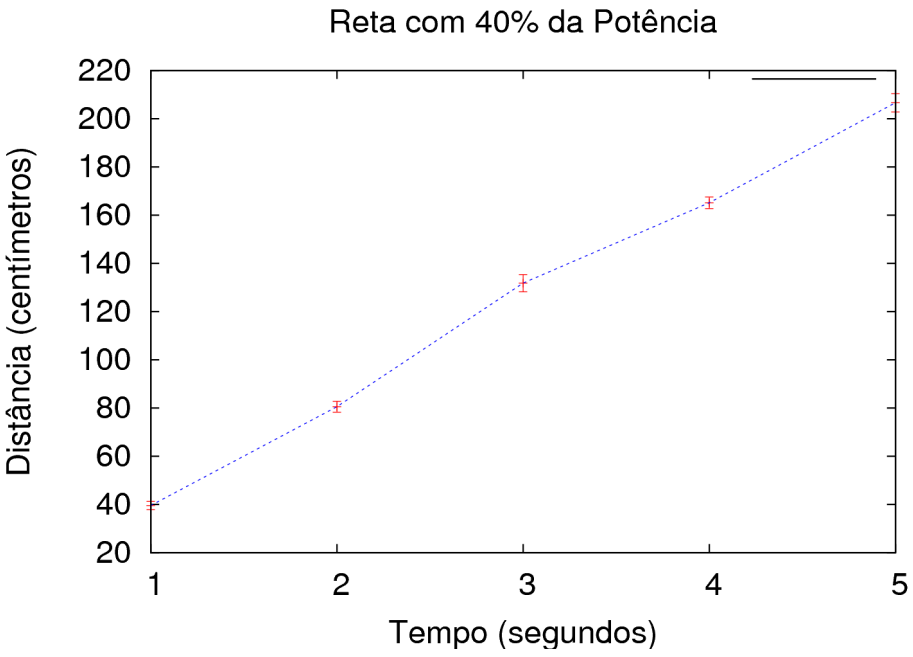
Calibragem:

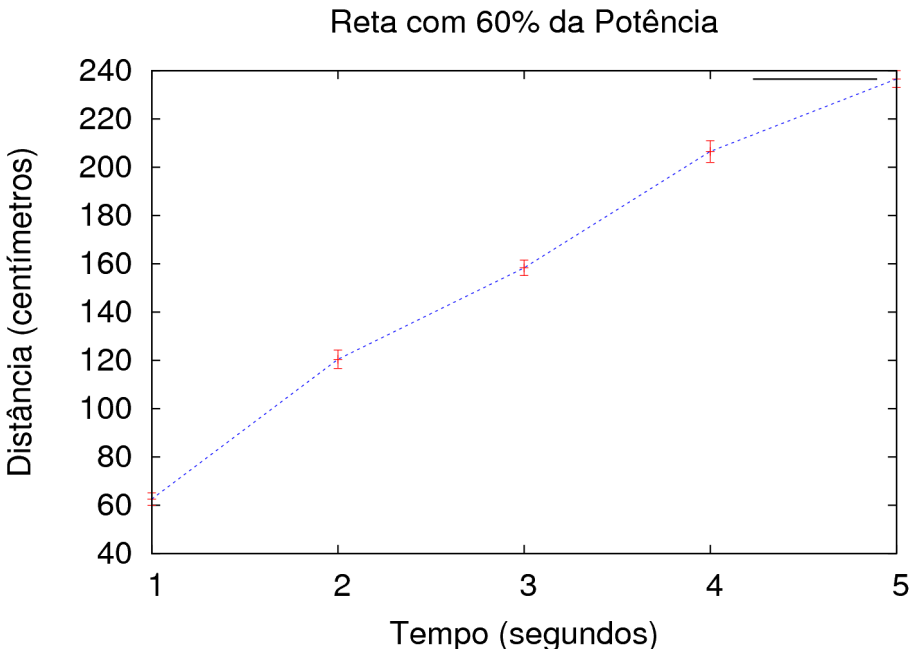
Terminada a montagem mecânica e a programação, partimos para a calibração dos movimentos. Como cada roda se movimenta independente da outra, devem ser calibradas para fazer a trajetória desejada. Os motores, engrenagens e eixos apesar de teoricamente serem iguais, não são idênticos. Isso faz com que seja necessário fornecer potências diferentes a cada motor, para que o robô ande em linha reta, por exemplo.

O menu feito através da programação permite variar a potência fornecida a cada motor e o tempo que ficam ligados. Com esses parâmetros é possível, teoricamente, calibrar qualquer movimento retilíneo ou circular. Mais adiante serão mostrados os gráficos da calibração.

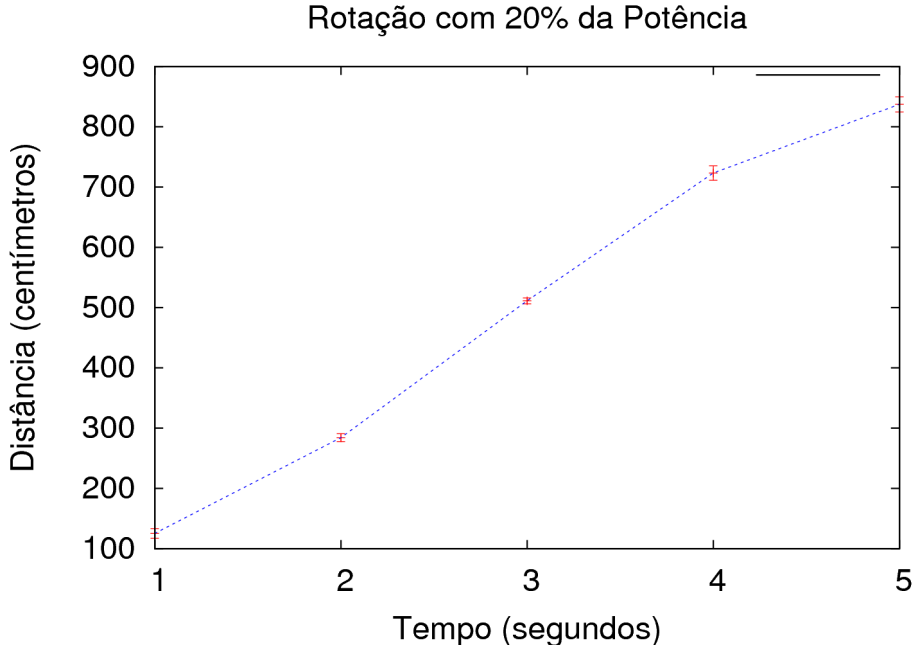
Para os dados coletados no movimento linear, medimos as distâncias percorridas para cada potência fixa variando-se o tempo de motor ligado. As distâncias foram medidas utilizando um barbante para obtenção de uma precisão razoável. Os gráficos abaixo mostram os valores médios e desvio padrão.

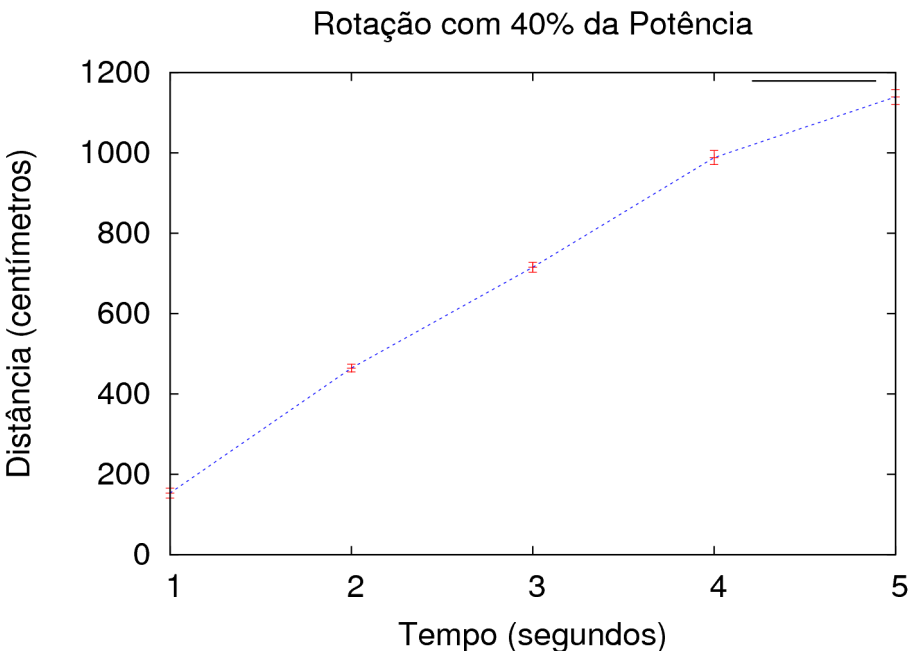

Reta com 20% da Potência
Distância (centímetros)
45
40
35
30
25
20
15
10
1
2
3
4
5
Tempo (segundos)

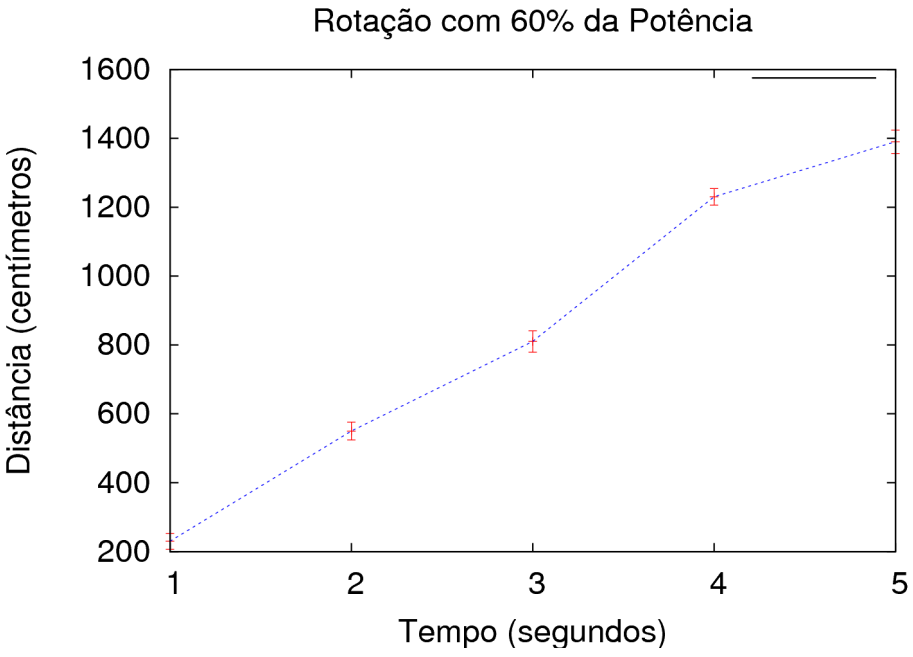

Reta com 40% da Potência
Distância (centímetros)
220
200
180
160
140
120
100
80
60
40
20
1
2
3
4
5
Tempo (segundos)



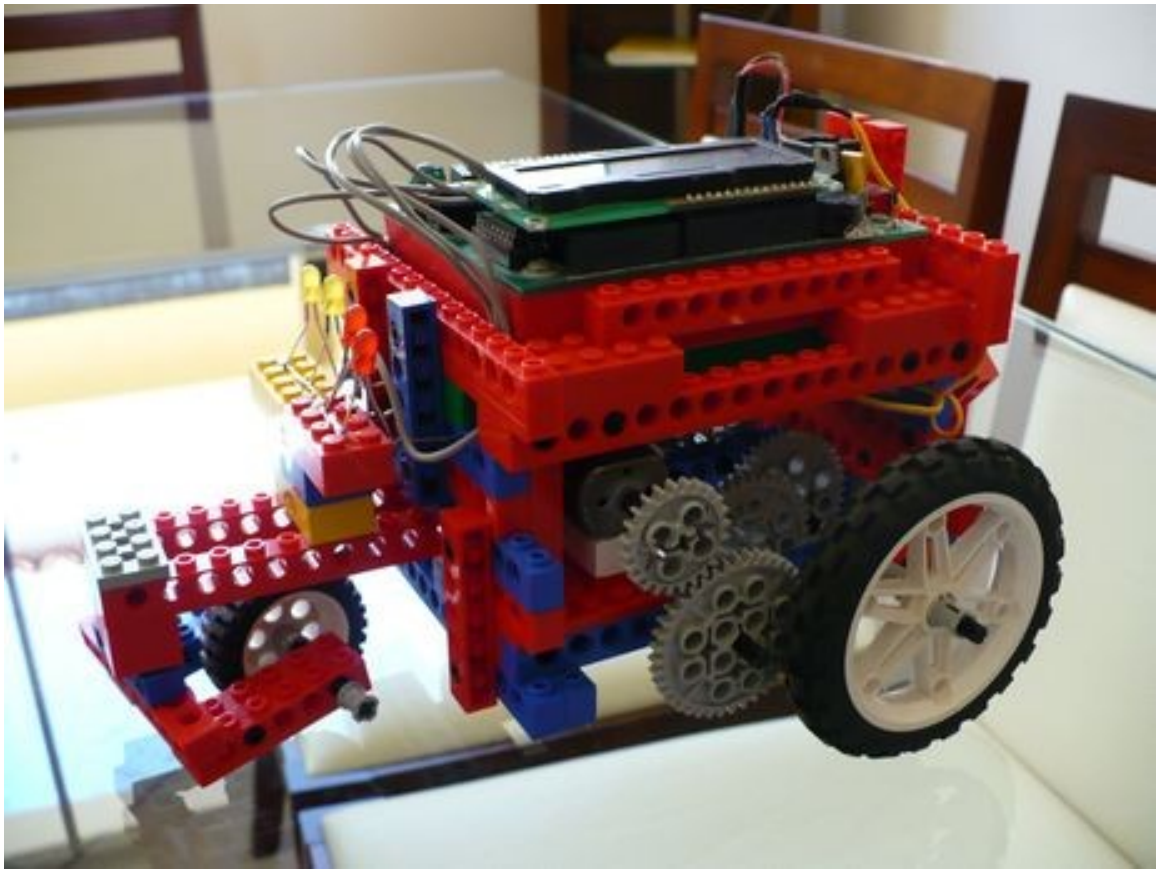
No caso da rotação, medimos com um transferidor o ângulo de giro efetuado pelo robô para cada potência fixa, variando-se o tempo. Vimos que as relações entre potencia e ângulo são bem menos lineares do que no caso do movimento em linha reta. Os gráficos abaixo mostram os resultados obtidos.



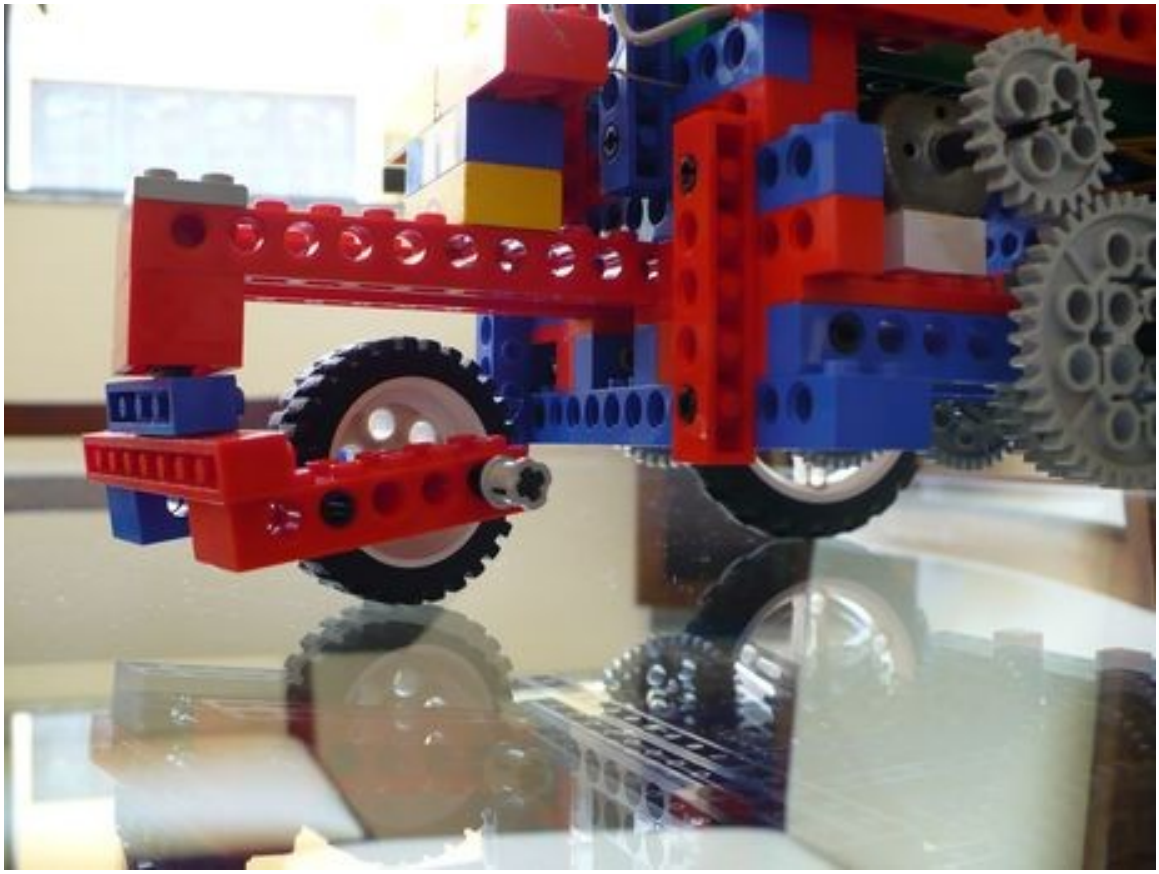

Rotação com 40% da Potência
Distância (centímetros)
0
200
400
600
800
1000
1200
1
2
3
4
5
Tempo (segundos)


Rotação com 60% da Potência
Distância (centímetros)
200
400
600
800
1000
1200
1400
1600
1
2
3
4
5
Tempo (segundos)

Versão Final



Versão final



Detalhe da roda biruta da versão final

A figura acima retrata as mudanças feitas para maior robustez da roda biruta. Utilizamos um esquema para transladar a peça e encaixá-la com uma peça de travamento. Com isso, a roda conseguiu resistir ao torque de carga devido à massa de toda a estrutura apoiada sobre ela. Também utilizamos uma roda maior, que passou a responder melhor no caso da rotação.