

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ENGENHARIA DE COMPUTAÇÃO

ANDRÉ BADENAS DOS SANTOS
DIEGO WESLEY BRAGA
RAFAEL JORGE TRINDADE

**DESENVOLVIMENTO DE UM
ROBÔ MÓVEL DÍCICLO DE AUTO BALANCEAMENTO**

RELATÓRIO FINAL DO PROJETO

CURITIBA
JULHO DE 2015

ANDRÉ BADENAS DOS SANTOS
DIEGO WESLEY BRAGA
RAFAEL JORGE TRINDADE

**DESENVOLVIMENTO DE UM ROBÔ
MÓVEL DICICLO DE AUTO BALANCEAMENTO**

Relatório final do projeto da disciplina Oficina de Integração 2 do curso de graduação em Engenharia de Computação da Universidade Tecnológica Federal do Paraná, ministrada pelos professores Heitor Silvério Lopes e Mário Sérgio Teixeira de Freitas.

CURITIBA
JULHO DE 2015

AGRADECIMENTOS

Primeiramente, agradecemos aos professores da disciplina Heitor Silvério Lopes e Mário Sérgio Teixeira de Freitas pela atenção e aprendizado proporcionado no decorrer do semestre.

Agradecemos ainda à Vinícius Brenner, também estudante de Engenharia de Computação, pela contribuição dos seus conhecimentos.

Por fim, agradecemos à banca examinadora pela disposição e atenção dedicadas a este estudo.

RESUMO

Este relatório apresenta uma abordagem do processo de desenvolvimento de um robô móvel díscio de auto balanceamento. Discute os conceitos teóricos que fundamentam fisicamente a elaboração e desenvolvimento do projeto e apresenta a metodologia de execução utilizada em todas as etapas do processo. Relaciona os materiais, dispositivos e serviços utilizados para a produção do robô, mostrando os métodos responsáveis por integrá-los e os seus respectivos custos e ainda o cronograma de implementação de cada etapa. Por fim, traz os resultados parciais do projeto até o momento de entrega deste documento e uma discussão sobre as dificuldades encontradas.

ABSTRACT

This report presents an approach of the process of development of a self balancing dicyclo movable robot. It discusses the theoretical concepts that physically supports the elaboration and development of the project and presents the implementation methodology used in all stages of the process. Lists the materials, devices and services used to produce the robot, showing the methods responsible for integrating them and their respective costs and also the implementation schedule of each stage. Finally, it brings the partial results of the project to the delivery time of this document and a discussion of the difficulties encountered.

LISTA DE FIGURAS

FIGURA 01 - DIAGRAMA DE BLOCOS DO PROJETO.....	07
FIGURA 02 - DISCO DE UM ENCODER.....	08
FIGURA 03 - MÓDULO DE SENSOR DE CONTROLE MPU-6050.....	09
FIGURA 04 - SIMULAÇÕES DE UM SISTEMA COM CONTROLE PID.....	09
FIGURA 05 - MODELO DE CONTROLE DE UM PÊNDULO INVERTIDO.....	11
FIGURA 06 - MODELOS DE CONTROLE DE UM DICICLO E DE UM SEGWAY.....	12
FIGURA 07 - MÓDULO DE COMUNICAÇÃO RF 433MHz FS-1000a.....	12
FIGURA 08 - VISTAS ESQUEMÁTICAS DA ESTRUTURA DO ROBÔ.....	14
FIGURA 09 - VISTAS ESQUEMÁTICAS DO CONTROLE.....	14
FIGURA 10 - FLUXOGRAMA DE FUNCIONAMENTO DO SOFTWARE.....	16
FIGURA 11 - DIAGRAMA DE GANTT.....	17
FIGURA 12 - VISTA FRONTAL DO ROBÔ MONTADO.....	19
FIGURA 13 - VISTA LATERAL DO ROBÔ MONTADO.....	20
FIGURA 14 - CONTROLADOR MONTADO.....	20

LISTA DE TABELAS

TABELA 1 - ORÇAMENTO DO PROJETO	18
---------------------------------------	----

SUMÁRIO

1 INTRODUÇÃO.....	07
2 DESENVOLVIMENTO.....	08
2.1 FUNDAMENTAÇÃO TEÓRICA.....	08
2.1.1 Encoder.....	08
2.1.2 Acelerômetro e giroscópio.....	08
2.1.3 PID.....	09
2.1.4 Comunicação via radiofrequênci.....	12
2.2 MATERIAIS E MÉTODOS.....	13
2.2.1 Metodologia de execução.....	13
2.2.1.1 Estrutura.....	13
2.2.1.2 Softwares.....	15
2.2.1.3 Testes.....	15
2.2.1.4 Revisão.....	16
2.2.2 Cronograma de implementação.....	17
2.2.3 Orçamento.....	18
3 RESULTADOS E DISCUSSÃO.....	19
4 CONCLUSÃO.....	21
REFERÊNCIAS.....	22

1 INTRODUÇÃO

O projeto tem como objetivo desenvolver um robô móvel díscio de auto balanceamento. O dispositivo será composto de duas grandes partes, o controlador e o robô, como mostrado na Figura 1. Cada uma destas partes será controlada por uma plataforma Arduino. A primeira, composta de um Arduino e um transmissor de radiofrequência, é responsável por receber comandos, do usuário ou pré-programados, e transmitir para a segunda parte. Essa, por sua vez, é responsável por todo movimento e balanceamento do robô. Através das informações recebidas pelos sensores giroscópio, acelerômetro e encoders e dos possíveis comandos recebidos e repassados pelo receptor, o Arduino deverá ser capaz de calcular e enviar à ponte H os comandos necessários para os motores se movimentarem de forma a balancear o dispositivo e paralelamente, quando solicitado, realizar o movimento exigido pelo usuário.

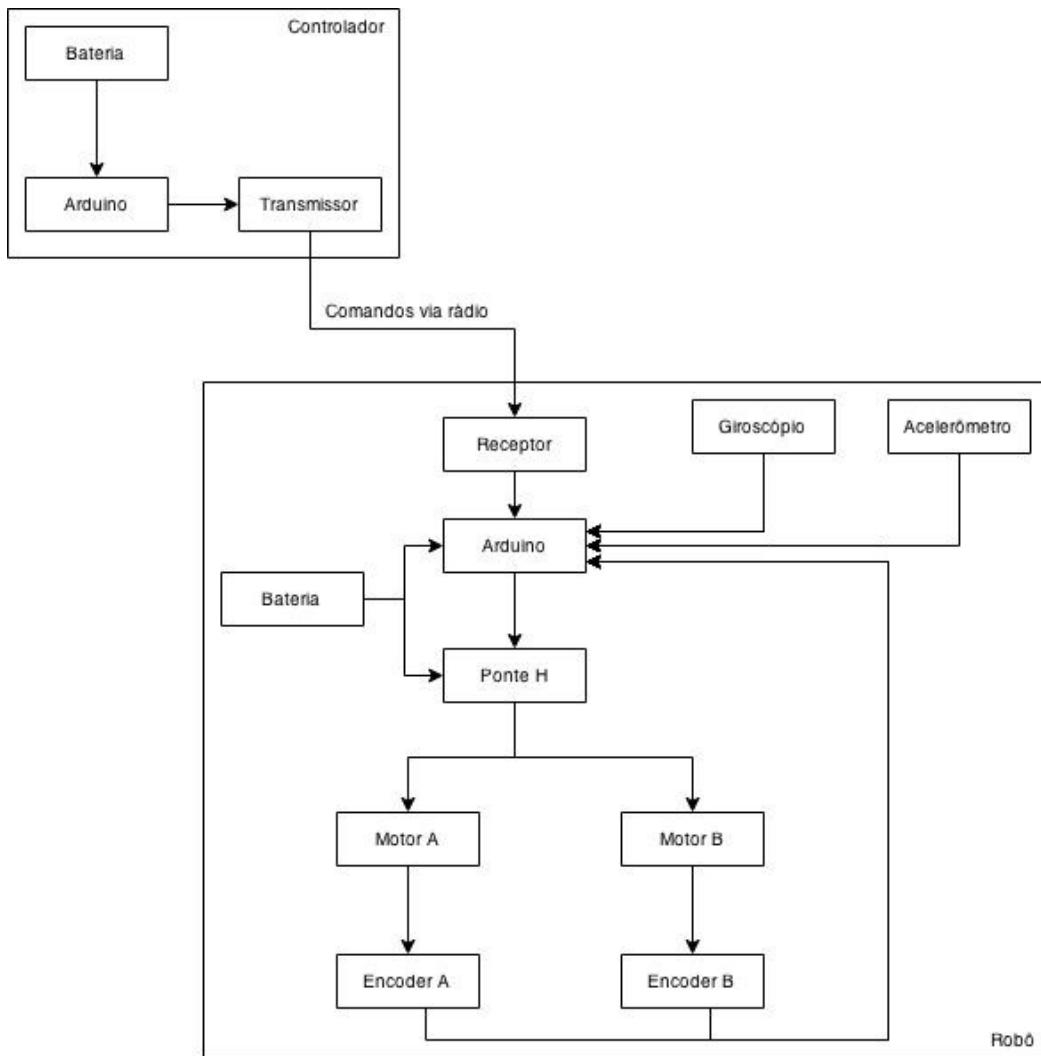


Figura 1: Diagrama de Blocos do projeto.

2 DESENVOLVIMENTO

2.1 FUNDAMENTAÇÃO TEÓRICA

Para a execução do projeto introduzido, é necessária uma diversa gama de conhecimentos da área tecnológica. Antes de tudo, é necessário o conhecimento a respeito dos sensores utilizados, nesse caso, encoders, acelerômetros e giroscópios.

2.1.1 Encoder

Encoder é um dispositivo eletromecânico responsável por gerar pulsos a cada rotação[1]. Seu funcionamento parte da conversão de movimentos rotativos em impulsos elétricos de ondas quadradas. É composto por um disco de encoder e um contador. Encoder serão utilizados no projeto com o objetivo de medir as rotações do motor e assim, encontrar a velocidade de movimentação do robô.

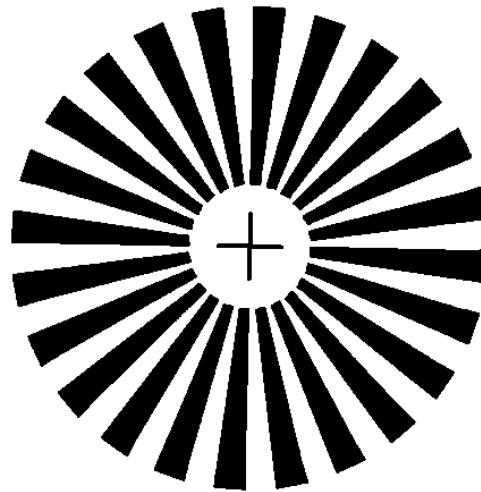


Figura 2: Disco de um encoder.

A figura 2 mostra um disco de encoder com 48 divisões, ou seja cada divisão representa a variação de $7,5^\circ$ no eixo.

2.1.2 Acelerômetro e giroscópio

Acelerômetro é um sensor que mede a aceleração.[2].Já os giroscópios são sensores que medem a taxa de variação do ângulo, ou seja, sua velocidade angular[2].

Ambos os componentes serão utilizados no projeto afim de encontrar o ângulo e a velocidade angular do eixo vertical.



Figura 3: Módulo de Sensor de Controle MPU-6050.

Fonte:

A figura três mostra o MPU-6050, que congrega um acelerômetro de três eixos e um giroscópio de três eixos no mesmo componente.

2.1.3 PID

A partir das informações geradas pelos sensores, é possível analisar o controle do robô. Será utilizado o controle *proportional-integral-derivative*(PID). O controle PID serve para encontrar uma boa forma de o robô partir de um estado *a* e ir para um estado *b* desejado. É representado pela equação $u(t) = K(e(t) + \frac{1}{T_i} \int_0^t e(\tau)d\tau + T_d \cdot \frac{de(t)}{dt})$, onde *u* representa o sinal de controle, *e* o erro entre a situação atual e o resultado esperado e *K*, *T_i* e *T_d* são as constantes reguladas pelo projetista de forma que o sistema seja estável.

As figuras 4a, 4b e 4c mostram o funcionamento de sistemas P, PI, e PID para a função de transferência, isto é, a lei que rege o sistema, dada por $P(s)=1/(1+s)^3$.

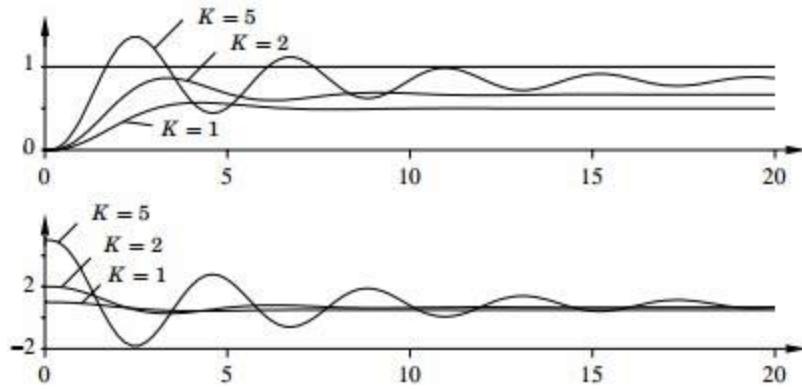


Figura 4a: Simulação de um sistema com controle P para valores diferentes de K .

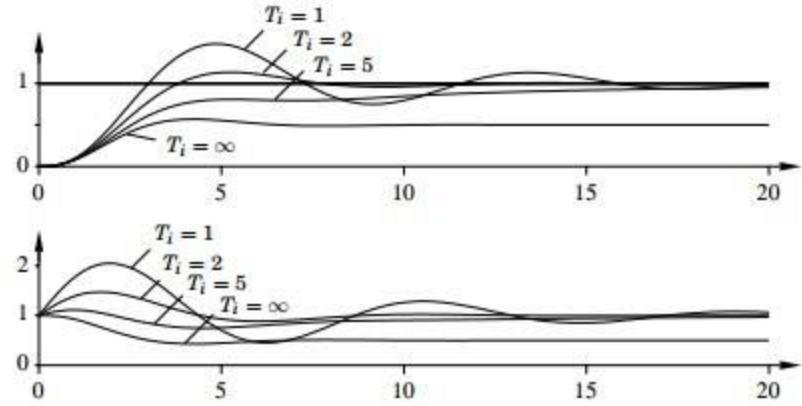


Figura 4b: Simulação de um sistema com controle PI para $K=1$ e valores diferentes de T_i .

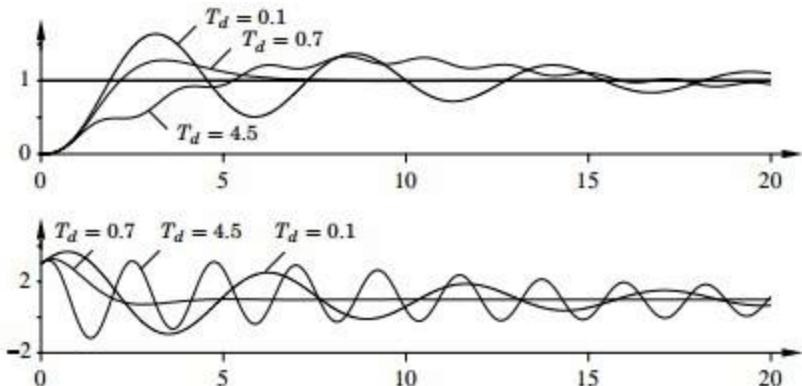


Figura 4c: Simulação de um sistema com controle PID para $K=3$, $T_i = 2s$ e valores diferentes de T_d .

A partir da figura 4a, é possível perceber que um controle P, dependendo da constante K , funciona de forma bastante estável, porém, somente um controle P terá um erro em relação ao estado desejado. Esse erro diminui com o aumento de K , porém quanto maior o K , mais o sistema oscila.

Com a figura 4b é possível perceber que o sistema, quando $T_i = 2s$ não só é estável, como chega no estado desejado. Por isso para muitos casos somente um controle PI já é suficiente. [3][5]

Finalmente, a partir da figura 4c pode-se perceber o funcionamento de um sistema PID, com $K=3$, $T_i = 2s$ e valores variados para T_d . Percebe-se que para tempos muito pequenos, o sistema oscila, e para tempos muito grandes também, sendo necessário encontrar um tempo intermediário no qual o sistema funciona de forma estável. Um dos grandes problemas da derivada do erro é que ela é suscetível a interferências, e por isso normalmente são utilizados filtros passa baixa para esse termo. Para encontrar as constantes ideais para o sistema, geralmente é feita uma bateria de testes até que o sistema se comporte como o esperado.

Porém, para se poder aplicar o PID, é necessária a construção de um modelo de controle. Modelos de controle são representações do sistema abordado utilizando relações conhecidas na física. Em geral são representados por equações diferenciais.

Os modelos de controle levam como base aquilo que se obtém dos sensores para definir o estado do sistema e controlar a variável de estado para se cumprir o objetivo proposto.

Como normalmente os sistemas não variam de forma linear, é necessário utilizar o que é chamado de linearização, onde se estuda o evento localmente utilizando a aproximação de Taylor de primeira ordem para a função de variação, pois a maioria das equações diferenciais não lineares são impossíveis de resolver de forma analítica.[4]

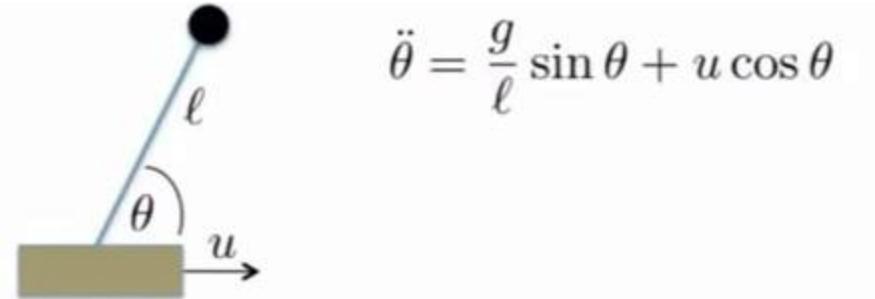


Figura 5: Modelo de controle de um pêndulo invertido.

O modelo apresentado na Figura 5 mostra um pêndulo invertido, onde g é a gravidade, u a aceleração angular da base em relação ao pêndulo, l é o comprimento da haste e θ o ângulo entre a base e a haste. A equação do sistema é obtida através das leis da cinemática. Para $\theta \approx \frac{\pi}{2} rad$, $\sin \theta \approx 1$ e $\cos \theta \approx \frac{\pi}{2} - \theta$, tendo então como linearização possível $\theta'' = \frac{g}{l} + \frac{\pi}{2} - \theta$.

O dispositivo proposto neste trabalho é popularmente conhecido como segway e pode ser descrito como a união de dois dispositivos comuns: díctico e pêndulo

invertido. A figura 6 esquematiza as variáveis presentes no controle em um díctico e um segway. Para o controle total, deverão ser consideradas as variáveis presentes nos dois modelos de controle que o compõem, ou seja, quatro variáveis do estado do sistema: velocidade, rotação do eixo do motor, ângulo e velocidade angular do eixo vertical.

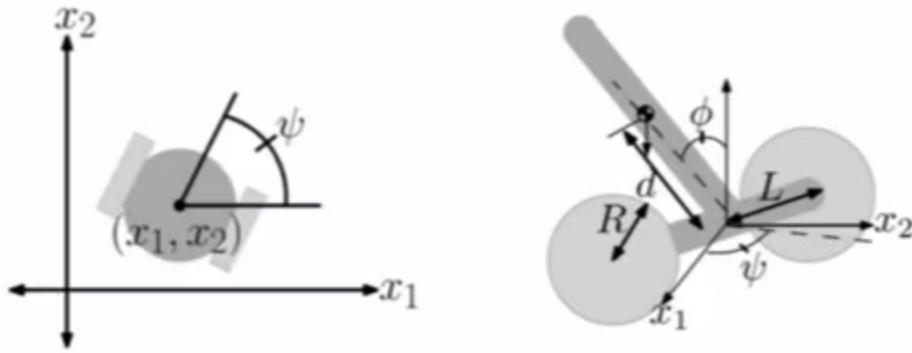


Figura 6: Modelos de controle de um díctico e um segway.

As constantes de ganho proporcional (K), integral (T_i) e diferencial (T_d) deverão ser projetadas de forma a se obter um sistema assintoticamente estável. Porém, a partir dos testes descritos na sessão posterior, poderão ser aperfeiçoadas, de forma a se obter um equilíbrio entre velocidade e oscilação dos ganhos do sistema.

2.1.4 Comunicação via radiofrequência

Para a movimentação do robô através de um controle remoto, será utilizado um módulo de comunicação RF, onde o transmissor recebe um comando e o envia para o receptor, esse realizando a ação designada ao comando em questão[6].

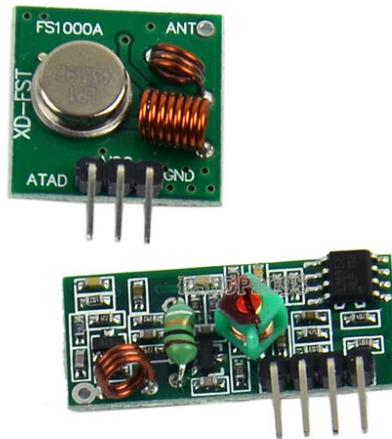


Figura 7: Módulo de comunicação RF 433MHz FS-1000a

2.2 MATERIAIS E MÉTODOS

2.2.1 Metodologia de execução

Nesta seção será apresentada a metodologia de execução do projeto, apresentando os métodos utilizados nas principais etapas de desenvolvimento, teste e revisão.

2.2.1.1 Estrutura

A execução do projeto teve início pela sua estruturação física. A seguir serão apresentadas as etapas de construção e integração dos componentes do robô e do controlador. A figura 8 apresenta as vistas esquemáticas frontal e lateral da estrutura física do robô. A figura 9 apresenta, por sua vez, a vista esquemática da estrutura do controlador.

O robô foi construído sobre três suportes retangulares acoplados verticalmente, com o auxílio de quatro hastes rosqueadas. Os suportes foram feitos a partir de placas fresadas de acrílico, de tamanho 9x14cm. No suporte inferior foram posicionados os dois motores DC com redução, os encoders, a bateria de lítio, sendo os motores fixados à face orientada ao chão, com os encoders já acoplados, e o restante fixados a outra face. Para encaixe das rodas no motor foram utilizados dois adaptadores de eixo fabricados pelos próprios autores.

No suporte central foram anexados à face inferior a ponte H L298N, e à superior o Arduino Mega, juntamente com placa-mãe. Esta foi construída em uma placa perfurada e contém reguladores de tensão para alimentação do robô, o receptor de rádio frequências e módulo de orientação(MPU6050), que agrupa acelerômetros e giroscópios de três eixos), todos devidamente conectados por circuitos feitos na própria placa.

O suporte mais elevado serviu como proteção mecânica dos circuitos imediatamente abaixo.

Para comandar o robô, foi construído um controle composto por uma bateria de 9V, a qual alimenta diretamente o emissor de rádio frequências. A tensão de entrada do Arduino Nano, assim como o circuito que constitui um controle direcional (isto é, cima, baixo, esquerda e direita) foi ajustada com o uso de reguladores. Todos os componentes foram devidamente encaixados em uma placa e conectados por um circuito feito na mesma.

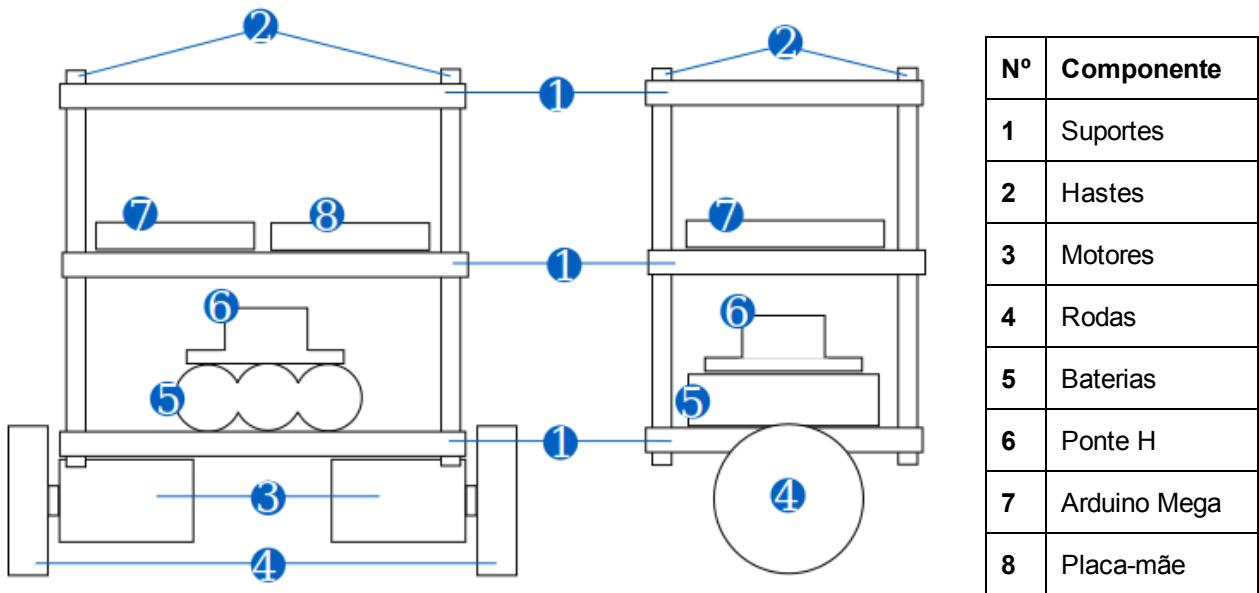


Figura 8: Vistas esquemáticas da estrutura do robô.

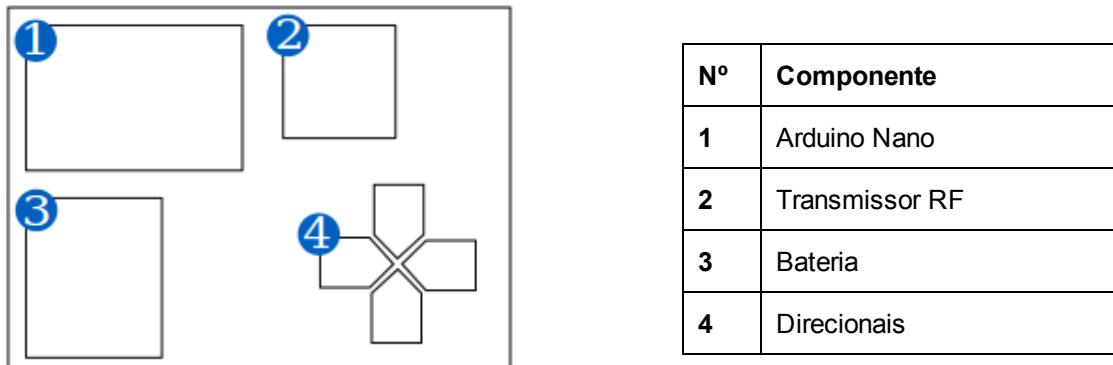


Figura 9: Vista esquemática do controle.

2.2.1.2 Softwares

Foram desenvolvidos para o projeto dois softwares, cada um executado por um Arduino. As diversas funções implementadas são descritas a seguir.

A comunicação e o controle remoto(Arduino Nano) e o robô(Arduino Mega) foi feita utilizando um módulo de comunicação rádio AM e a biblioteca *VirtualWire*, que permite transmissão mais confiável (isto é, sem necessidade de retransmissão e confirmação por parte do receptor). Os dados enviados são os estados dos botões, ou seja, acionados ou não.

Baseado nas informações recebidas do controle, o robô decide que instrução passar para o controlador PID. Visto que o controle possui quatro botões direcionais, parâmetros específicos precisam ser atribuídos a cada uma destas direções e possíveis combinações entre elas. Os botões cima e baixo representam, respectivamente, velocidade de translação relativa positiva e negativa, enquanto os botões direita e esquerda, respectivamente, rotação horária e anti-horária. Combinações que envolvam botões de velocidade e rotação representam movimentos curvilíneos para as direções escolhidas.

Os dados capturados pelos sensores representam o estado do sistema, e, quando repassados ao Arduino, este interpreta-os e trata-os para posterior utilização no controlador PID. Os dois encoders são responsáveis por repassar as rotações dos motores, que, a partir de equações físicas, são convertidas nas velocidades v_1 e v_2 de cada roda. A partir do módulo de orientação é encontrado o ângulo e a velocidade angular do robô.

Posteriormente foi implementada a parte do software responsável pelo controle PID. Esta função é responsável por receber os parâmetros anteriormente recebidos e tratados e aplicados no modelo de controle teórico projetado e então, fornecer os valores das variáveis de velocidade e rotação necessárias para executar o controle do sistema.

A partir das constantes definidas para o controle PID, o Arduino calcula os proporcionais de *Pulse-Width Modulation* (PWM) a serem repassados à ponte H, a qual controla os motores nas velocidades desejadas.

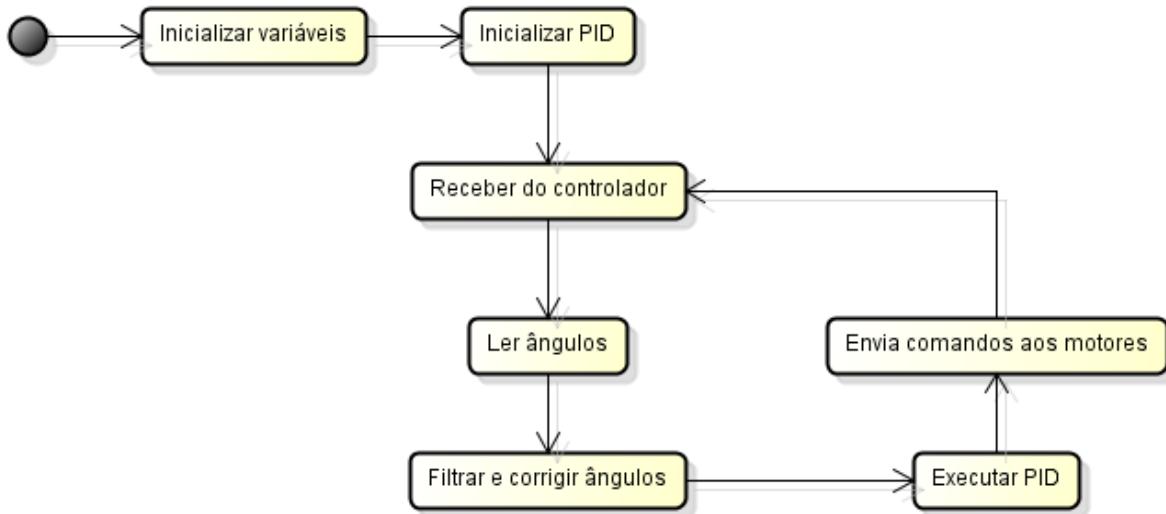


Figura 10: Fluxograma de funcionamento do software.

O fluxograma apresentado na figura 10 mostra a sequência executada pelo software criado. Esse diagrama facilitou a organização das ideias para a criação do programa.

2.2.1.2.1 A criação do software

O código que comanda o robô é dividido nas seguintes funções:

- Comunicação controle remoto -> arduino (virtual wire)
- Filtro de dados (Filtro de Kalman que também já fez a calibragem)
- Controlador PID (Arduino PID Library)
- Velocidade dos motores

Controle remoto

A leitura da tensão sobre os potenciômetros é convertida para valores entre -100 e 100 para os eixos x e y, atribuída de acordo com um plano cartesiano X x Y com centro na origem. Para estabelecer comunicação via rádio entre o controle remoto e o robô foi utilizada a VirtualWire [7], que habilita comunicação digital. Os dados dos potenciômetros são concatenados com identificadores para cada uma das coordenadas e são inseridos em um vetor de caracteres para ser enviados, e. g.:

```
msg = "x-20.0y50.0".
```

O hardware, combinado com esta biblioteca, permite comunicação com velocidade de até 2000 bits/s, mas este não foi um fator limitante no projeto.

Os dados, assim que recebidos pelo robô, são separados usando os identificadores previamente inseridos pelo controle em suas respectivas funções, sendo que os dados do eixo Y são atribuídos à velocidade de translado e os dados do eixo X, rotação.

Filtragem dos dados sensoriais

O giroscópio e o acelerômetro, respectivamente, possuem as seguintes falhas:

- Não mantém a orientação em relação ao início da execução do programa;
- Mesmo que ajustado para manter relativa baixa sensibilidade (+2g), é muito sensível a variações e a ruídos inerentes a eletrônica do conjunto. Mesmo “completamente” estático, retornava leves (porém muito frequentes) medições de aceleração em todos os eixos.

Para reduzir quase que completamente esta fonte de erros no software foi utilizado um Filtro de Kalman, convenientemente agrupado em uma biblioteca para Arduino [8]. De forma resumida, através da combinação dos dados sensoriais com os modelos físicos que os relacionam, o filtro reduz os ruídos, aproximando os valores obtidos aos valores reais produzidos pelo equacionamento físico.

No contexto de nosso projeto, os valores processados por este filtro foram convertidos para graus ($^{\circ}$), os quais representam a orientação do robô em relação aos eixos padrão do circuito MPU6050. Graças a isso, nenhuma calibragem adicional precisou ser implementada para que o robô identificasse sua orientação em relação ao chão toda vez que fosse ligado.

Controlador PID

Este controlador foi aplicado utilizando a biblioteca para controladores PID para Arduino [9]. Este software permitiu simples implementação deste controlador à nossa proposta, através da criação de um objeto do tipo PID, que tem como parâmetros de inicialização:

```
PID controlador(&input, &output, &setpoint, Kp, Ki, Kd, DIRECTION);
```

- **&input:** endereço da variável que registra o valor fornecido pelo filtro de dados, variando entre -90° e $+90^{\circ}$ neste contexto;
- **&output:** endereço da variável que armazenará a resposta do controlador ao sistema;
- **&setpoint:** endereço da variável que determina o valor que deseja ser mantido, atribuído como 0° para o robô. Os dados enviados pelo controle servem para alterar este valor para permitir que o robô se movimentasse;
- **Kp, Ki e Kd:** constantes do parâmetro proporcional, derivativo e integral, respectivamente, conforme o modelo do controlador;
- **DIRECTION:** direção para qual a saída (output) deverá apontar em função da entrada (input) fornecida;

Velocidade dos motores

A velocidade dos motores foi feita através da modularização de largura de pulso (mais conhecido por PWM, sigla para Pulse Width Modularization), sendo que para a plataforma Arduino este valor pode ser configurado entre 0 e 255. A implementação utilizada em nosso projeto considerou este intervalo entre 40 e 255, pois a resposta dos motores para valores muito baixos desse parâmetro era praticamente nula, devido a inércia inerente ao robô. Outras consequências disso foram o superaquecimento do *driver* dos motores, o que também significa que energia das baterias era desperdiçada.

A velocidade das rodas era diretamente proporcional a saída do controlador PID, variando adequadamente à orientação do robô em relação ao *setpoint* estabelecido.

2.2.1.3 Testes

Posteriormente às etapas de construção estrutural e desenvolvimento de softwares, iniciá-se a fase de testes. Esta etapa consiste em testar o dispositivo e analisar a velocidade de oscilação do sistema, provenientes das constantes escolhidas para o PID .

2.2.1.4 Revisão

Após a coleta das análises realizadas na etapa de testes as constantes do PID poderão ser substituídas no software, almejando um sistema que atenda uma maior estabilidade, equilibrando a velocidade e oscilação do controle. Posteriormente, iniciou-se um novo ciclo de testes e revisões, até que se encontre uma boa condição de funcionamento.

2.2.2 Cronograma de implementação

Com base na metodologia de execução apresentada na seção anterior foi elaborado um cronograma de implementação. As figuras Z.a e Z.b apresentam um Diagrama de Gantt referente ao cronograma proposto, contendo as etapas de execução e seus respectivos tempos de implementações previstos.

Etapa	Data de início	Data de término
Fresagem das chapas	20/05/15	26/05/15
Acoplagem dos suportes	27/05/15	02/06/15
Montagem da placa-mãe	27/05/15	02/06/15
Montagem e integração dos componentes do Robô	03/06/15	09/06/15
Montagem e integração dos componentes do Controlador	03/06/15	09/06/15
Implementação de Software - Coleta de dados do direcional	10/06/15	14/06/15
Implementação de Software - Comunicação entre Arduinos	12/06/15	16/06/15
Implementação de Software - Transformação do estado dos direcionais	17/06/15	20/06/15
Implementação de Software - Tratamento dos dados dos Sensores	18/06/15	23/06/15
Implementação de Software - Comunicação entre Arduinos	24/06/15	27/06/15
Implementação de Software - Cálculos e controle	24/06/15	30/06/15
Implementação de Software - Transmissão dos comandos para os atuadores	28/06/15	30/06/15
Testes	01/07/15	06/07/15
Revisão	02/07/15	07/07/15
Defesa	08/07/15	08/07/15

Figura 11.a: Diagrama de Gantt parte 1.

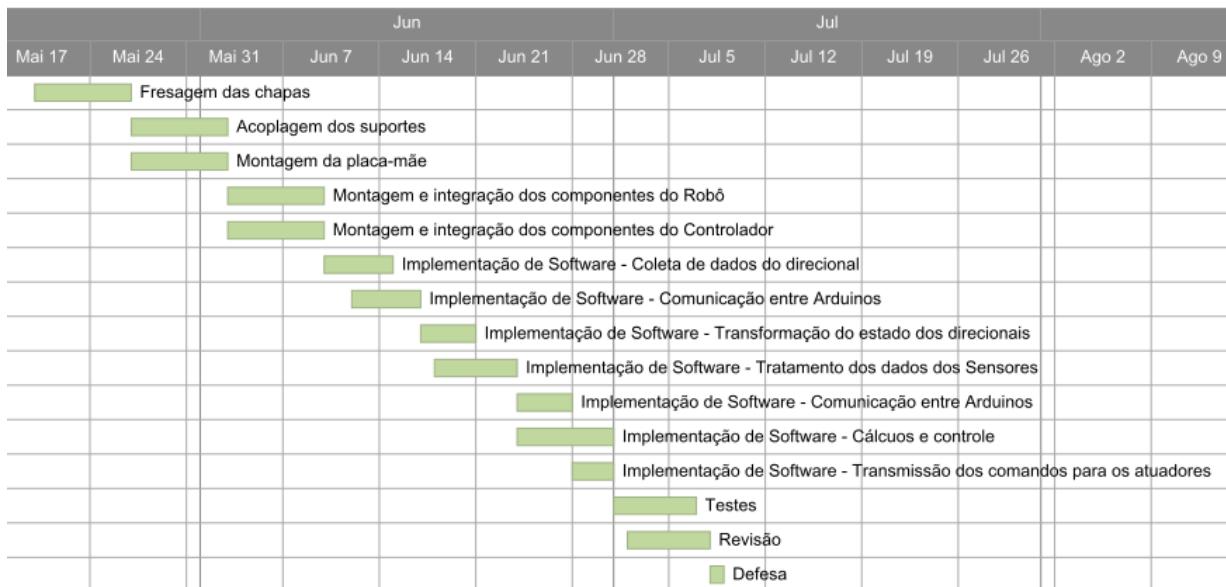


Figura 11.b: Diagrama de Gantt parte 2.

2.2.3 Orçamento

O quadro 1 apresenta o orçamento dos materiais adquiridos na execução do projeto. Nele estão listados todos componentes do dispositivo, as quantidades utilizadas e os custos unitários e totais. Conforme mostrado na tabela, o projeto apresentou um custo de R\$ 420,50.

Componente	Quantidade	Preço (R\$)	Total (R\$)
Bateria de Li-ion 2600 mAh	3	19,25	77,00
Motor de redução (1.19 N.m, 191 rpm em 5V, 3V ~ 12V)	2	30,00	60,00
Arduino Nano v3	1	35,00	35,00
Arduino Mega v3	1	73,90	73,90
MPU6050 (acelerômetro e giroscópio)	1	18,50	18,50
Ponte H L298N	1	18,90	18,90
Módulo de comunicação RF 433mHz fs1000a	1	12,00	12,00
Encoder	2	5,00	10,00
Chapa de acrílico	1	20,00	20,00
Roda	2	21,00	42,00
Haste rosqueada M4x200mm	4	1,50	6,00
Porca M4	24	0,30	7,20
Total (R\$)			420,50

Tabela 1: Orçamento do projeto.

3 RESULTADOS E DISCUSSÃO

Até o momento em que este relatório foi redigido as estruturas do robô e controlador estavam devidamente montadas, conforme mostrado nas figuras 11, 12 e 13, que apresentam as vistas frontal e lateral do robô e vista total do controlador, respectivamente. Em relação aos softwares, o responsável pelo controlador já se encontrava finalizada e em funcionamento, captando dos potenciômetros os comandos e transmitindo-os para o robô. O software responsável pelo robô se encontra incompleto, com os sensores sendo lidos com sucesso mas seus valores ainda não interpretados corretamente.

Neste momento vale ressaltar as dificuldades encontradas no decorrer do desenvolvimento. A construção da estrutura física se complicou por diversos empecilhos, como indisponibilidade dos laboratórios de mecânica para fresagem das chapas e fabricação dos eixos e dificuldade de encontrar peças compatíveis com o projeto. Contudo, ao final não é possível garantir a estabilidade da parte mecânica, justamente pois não se sabe qual o problema que faz com que o robo não ficasse estável. Em relação à teoria e sua aplicação aos software destaca-se a dificuldade de utilização dos conceitos de controle PID na conversão dos valores recebidos.

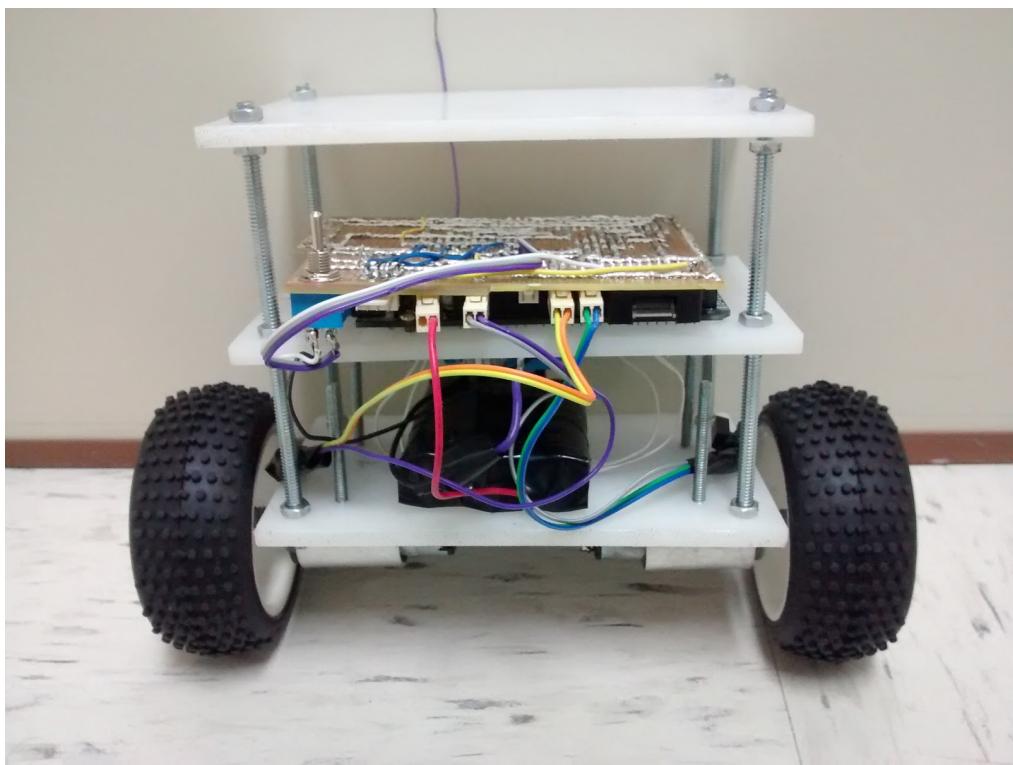


Figura 12: Vista frontal do robô montado.

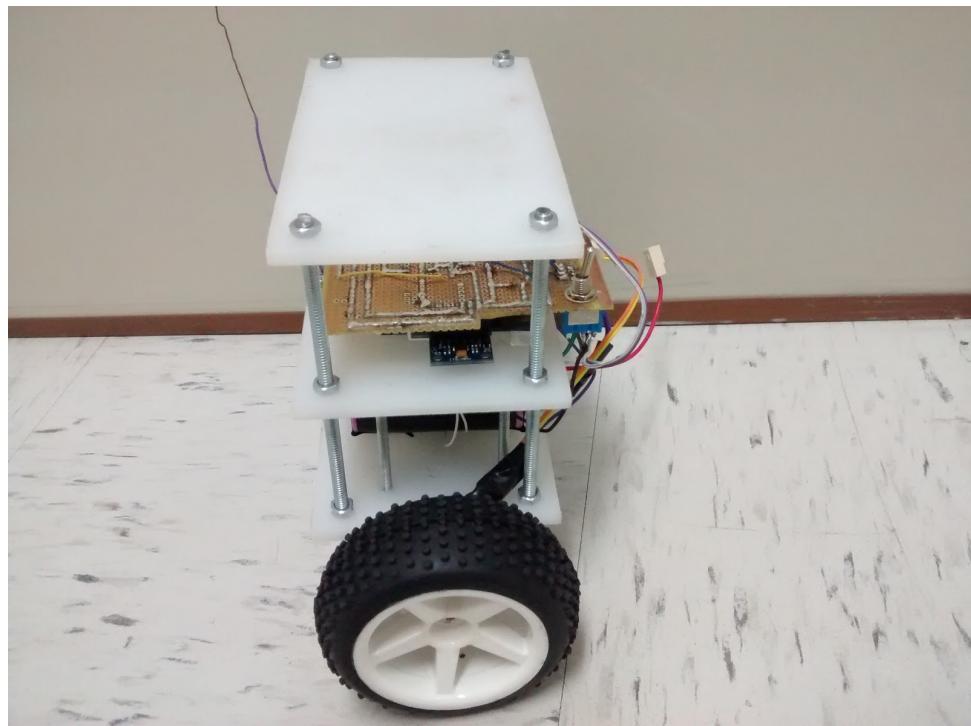


Figura 13: Vista lateral do robô montado.

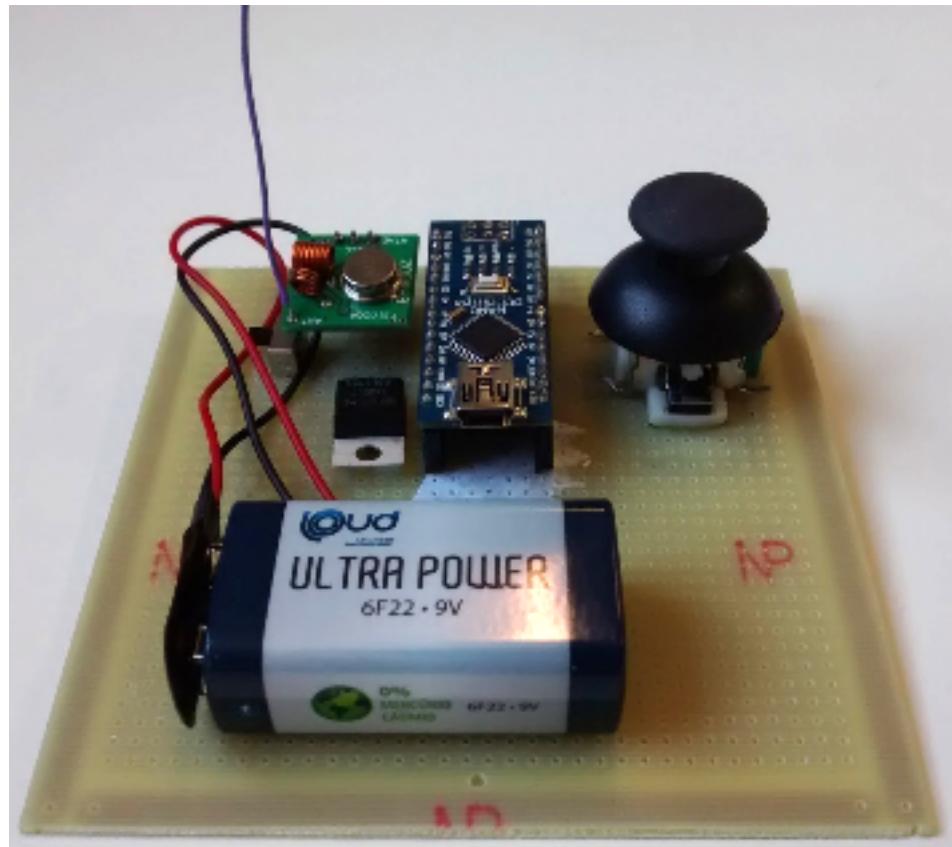


Figura 14: Controlador montado.

4 CONCLUSÃO

Ao final, pode-se concluir acerca do planejamento que todo e qualquer aspecto do robô, seja da construção física ou do software que o controla é absolutamente essencial, pois uma falha em qualquer das partes gera um problema no robô como um todo. Várias etapas que pareciam ser simples, como a fixação das rodas, acabaram se transformando em fontes de atraso, interferindo na qualidade do projeto. Peças customizadas que dependiam de serviço externo foram cruciais neste aspecto. Conclui-se ainda, que mesmo com um planejamento detalhado do cronograma de implementação, as etapas podem fugir ao controle e perpetuar atrasos no projeto todo. Ao final, conclui-se a importância do trabalho em equipe e da divisão de tarefas para alcançar participação de todos de forma eficiente, otimizando os resultados.

REFERÊNCIAS

[1] Society of Robots,SENSORS - ROBOT ENCODER (SLOT, ROTARY, LINEAR), Disponivel em: <http://www.societyofrobots.com/sensors_encoder.shtml>. Acesso em 12/05/2015

[2] Hyung Gi Min, Eun Tae Jeung, "Complementary Filter Design for Angle Estimationusing MEMS Accelerometer and Gyroscope", Robotics Lab., NTREX Ltd. Korea.

Disponível em:

<http://www.academia.edu/6261055/Complementary_Filter_Design_for_Angle_Estimation_using_MEMS_Accelerometer_and_Gyroscope>. Acesso em 24/04/2015.

[3] Magnus Egerstedt. PID Control. Disponivel em:
<<https://class.coursera.org/conrob-002/lecture/8>>. Acesso em 05/05/2015.

[4] Magnus Egerstedt.Linearizations. Disponivel em:
<<https://class.coursera.org/conrob-002/lecture/21>>. Acesso em 07/05/2015.

[5] Karl Johan Åström. PID Control. California Institute of Technology, p216-251.
Disponível em:
<<http://www.cds.caltech.edu/~murray/courses/cds101/fa02/caltech/astrom-ch6.pdf>>.
Acesso em 24/04/2015.

[6]Arduino Basics.433MHz Tutorial 1. Disponível em:
<<https://www.youtube.com/watch?v=l5ytC2zKj7M>>. Acesso em 12/05/2015.

[7]Mike McCauley - VirtualWire Library
<https://www.pjrc.com/teensy/td_libs_VirtualWire.html>. Acesso em 22/07/2015.

[8]Kristian Lauszus - Kalman filter library
<http://www.tkjelectronics.dk/uploads/Kalman_SR.zip>. Acesso em 22/07/2015.

[9]Arduino PID Library
<<http://playground.arduino.cc/Code/PIDLibrary>>. Acesso em 22/07/2015.