Projeto de Graduação

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

21 de dezembro de 2020

Controle com Arduino de um Aparato para Visualização da Dinâmica de Motocicletas e de uma Motocicleta em Escala 1:8

Giulia Mahovlic

PUC RIO

Projeto de Graduação

DEPARTAMENTO
DE ENGENHARIA
ELÉTRICA

Controle com Arduino de um Aparato para Visualização da Dinâmica de Motocicletas e de uma Motocicleta em Escala 1:8

Aluna: Giulia Mahovlic

Orientador: Mauro Speranza Neto

Trabalho apresentado como requerimento parcial para a conclusão do curso de Engenharia de Controle e Automação na Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil.

Agradecimentos

Agradeço primeiramente à Deus por me dar forças para continuar todos os dias.

Agradeço aos meus pais, Katia Cilene Augusto da Costa Mahovlic e Marko Mahovlic, por todo o apoio, incentivo e pela oportunidade de estudar em uma universidade de excelência.

Agradeço à minha irmã Mell Mahovlic, meu namorado Hugo Munhoz e à todos os meus amigos e amigas pela companhia e por estarem lá para mim quando eu precisei e pelos momentos divertidos e de distração que foram muito importantes.

Agradeço à meu orientador, Mauro Speranza Neto, pela confiança, paciência e pelas oportunidades ao longo de todo o curso de engenharia.

Resumo

O projeto consiste em aplicar controle sobre um aparato com 4 graus de liberdade (guinada, rolagem, descolamento lateral e translação) que permite observar a dinâmica de motocicletas. O aparato será modelado em 3D e serão feitas simulações em malha aberta afim de testar todos os seus graus de liberdade. Uma vez completo o aparato modelado será aplicado controle para que o sistema realize diferentes movimentos definidos por sinais de entrada.

Além do aparato a própria motocicleta também será simulada e controlada em alguns graus de liberdade (rotação das rodas, direção do guidão e inclinação do condutor) com os mesmos procedimentos que o aparado e, possivelmente, em conjunto.

Feitas as simulações, os movimentos serão aplicados a motores e servomotores reais através do microcontrolador Arduino integrado ao Simulink, ferramenta do MatLab usada para a realização das simulações.

Palavras-chave: Aparato, Motocicleta, Arduino, Controle, Simulink

Arduino Control of an Apparatus for Motorcycle Dynamics and a Motorcycle of 1:8 Scale Visualization

Abstract

The project consists of applying control over an apparatus with 4 degrees of freedom (yaw, roll, lateral displacement and translation) that allows the observation of the dynamics of motorcycles. The apparatus will be 3D modeled and open loop simulations will be made in order to test all of its degrees of freedom. Once the modeled apparatus is complete, control will be applied so the system can make different movements defined by input signals.

Besides the apparatus the motorcycle itself will also be simulated and controlled on some degrees of freedom (wheels rotation, handlebar steering and conductor inclination) by the same procedures of the apparatus and, possibly, together.

Once all the simulations are done the movements will be applied to real motors and servomotors through Arduino microcontroler integrated to Simulink, a MatLab tool used to make all the simulations.

Keywords: Apparatus, Motorcycle, Arduino, Control, Simulink

Sumário

Lista de Figuras **v**

Lista de Tabelas **vii**

1 Introdução **1**
a Objetivos . . . 1

2 Metodologia **3**
a Simscape . . . 3
1 Rótula (Revolute Joint) . . . 3
2 Junta prismática (Prismatic Joint) . . . 3
b Arduino . . . 4
1 Servomotor . . . 4
2 Motor DC . . . 4
c Validação . . . 5
1 Simulação . . . 5
2 Integração com o Arduino . . . 10
3 Problema encontrado . . . 11
4 Resultados . . . 11

3 Modelagem do aparato **12**
a Partes do modelo . . . 12
1 Base de guinada . . . 12
2 Base de deslocamento lateral . . . 13
3 Base de rolagem . . . 13
4 Suporte . . . 13
5 A motocicleta . . . 14
6 O pêndulo invertido . . . 15
7 O guidão . . . 15
8 As rodas . . . 16
b Juntas do modelo . . . 16
1 Guinada . . . 16
2 Deslocamento Lateral . . . 16
3 Rolagem . . . 17
4 Inclinação do condutor . . . 17
5 Rotação do guidão . . . 18
6 Deslocamento das rodas . . . 19
c Diagrama de blocos . . . 19

4 Controle **20**
a Controle proporcional . . . 21
b Controle proporcional derivativo . . . 23
c Controle proporcional integral . . . 25
d Controle proporcional integral derivativo . . . 27

5 Integração ao Arduino **29**
a Configurações Iniciais . . . 29
b Captura e armazenamento dos dados . . . 30
c Leitura dos dados . . . 31
1 Outras opções de aquisição dos dados . . . 33
d Conversões . . . 33
e Montagem . . . 34
1 Motor DC *MKR Motor Carrier* . . . 35
2 Montagem dos servos sem o *MKR Motor Carrier* e resultados . . . 38

6 Conclusão **40**

7 Referências **41**

Lista de Figuras

1 Controle do Aparato p/ Visualização da Dinâmica de Motocicletas em Escala c/ 4 GL's 1
2 Controle da Motocicleta em Escala 1:8 2
3 Bloco da rótula 3
4 Bloco da junta prismática 4
5 Bloco do servomotor no Simulink 4
6 Servomotor Micro de Arduino, fonte: [1] 4
7 Bloco do servomotor no Simulink 5
8 Mini Motor DC 3V, fonte: [2] 5
9 Guia 5
10 Carro 6
11 Partes do pêndulo 6
12 Todos os elementos do pêndulo duplo invertido montados 6
13 Diagrama de blocos do pêndulo duplo invertido 7
14 Diagrama de blocos do sistema de controle do pêndulo duplo invertido 8
15 Força externa aplicada na parte superior do pêndulo duplo invertido em N, gerando um torque 8
16 Resultados do sistema de controle do pêndulo duplo invertido 9
17 Velocidade do Carro em cm/s 9
18 Bloco que será usado para ler os arquivos que contém os movimentos 10
19 Bloco que será usado para ler os arquivos que contém os movimentos 10
20 Interface Simulink/Arduino com as devidas conversões de dados 11
21 Aparato, moto e condutor 12
22 Base de guinada 12
23 Base de deslocamento lateral 13
24 Base de rolagem 13
25 Suporte 14
26 *Frame* onde ficará o pêndulo 14
27 *Frame* onde ficará a roda traseira 14
28 *Frame* onde ficará o guidão 15
29 Pêndulo que representa o condutor 15
30 Guidão 15
31 Roda 16
32 Movimento de guinada 16
33 Movimento de deslocamento lateral 16
34 Detalhe da junta prismática 17
35 Movimento de rolagem 17
36 Detalhe da rótula de rolagem 17
37 Movimento do condutor 18
38 Detalhe da rótula do condutor 18
39 Movimento do guidão 18
40 Detalhe da rótula do guidão 19
41 Diagrama de blocos do modelo 19
42 Entradas do sistema de controle 20
43 Diagrama de blocos do sistema de controle P 21
44 Resultados do sistema de controle PD 22
45 Diagrama de blocos do sistema de controle PD 23
46 Resultados do sistema de controle PD 24
47 Diagrama de blocos do sistema de controle PD 25
48 Resultados do sistema de controle PD 26
49 Diagrama de blocos do sistema de controle PID 27
50 Resultados do sistema de controle PID 28
51 Lista de Arduinos compatíveis 29
52 Solução do problema de não reconhecimento do arduino 30
53 Onde encontrar o número da porta do Arduino 30
54 Configuração do *scope* 31
55 *Script* qu salva os dados num arquivo 31
56 Seleção do arquivo de dados 31
57 Indicação de onde clicar para editar o sinal 31
58 Os controles da edição de sinal e como os sinais aparecem inicialmente 32
59 Sinais nomeados e selecionados 32

60 Estado final do bloco 33
61 Estado final da interface Simulink/Arduino 34
62 Botão que executa o programa 35
63 Demonstração do uso do adaptador de energia e da bateria 35
64 *MKR Motor Carrier* 36
65 Arduinos compatíveis com o *MKR Motor Carrier* 36
66 Exemplo de um Arduino Nano 33 BLE montado em um *MKR Motor Carrier* 37
67 Qual bloco usar para controlar os servos com o *MKR Motor Carrier* 37
68 Montagem dos motores e servos no *MKR Motor Carrier* 38
69 Ilustração da montagem dos servomotores no Arduino real 39

Lista de Tabelas

1 Memória de cada um dos Arduinos compatíveis com o *MKR Motor Carrier* 40

1 Introdução

Motocicletas tem uma história longa traçada desde 1885, quando a primeira bicicleta motorizada, ou motocicleta, foi construída. Segundo William Harris em seu artigo *How Motorcycles Work* [3], a ideia surgiu a partir da já existente "bicicleta de segurança"que possuía diversas vantagens sobre uma bicicleta "comum", como melhor estabilidade, frenagem e facilidade em montar.

No mesmo ano de 1885, quando a bicicleta de segurança chegou ao mercado, o engenheiro alemão Gottlieb Daimler decidiu que seria uma boa ideia acrescentar um motor ao projeto. Desde então o modelo passou por diversas mudanças e aprimoramentos até se tornar a moto que se conhece nos dias atuais.

É controverso como um meio de transporte que foi originalmente projetado pensando-se em segurança e estabilidade é hoje um dos veículos com o maior número de acidentes fatais no Brasil, segundo o senado brasileiro [4], que gera 20 vezes mais mortes por acidentes do que carros.

Esses dados mostram como é importante o estudo da dinâmica das motocicletas. Um bom entendimento sobre a dinâmica e estabilidade permite o desenvolvimento de soluções que podem amenizar o perigo que as motos representam a seus usuários e é de suma importância até mesmo no projeto de motos autônomas.

Por isso, esse projeto consiste na simulação e controle de um aparato com quatro graus de liberdade que permite visualizar a dinâmica de uma motocicleta em escala através de microcontroladores e softwares de simulação. Espera-se desenvolver, testar a validar algoritmos que ofereçam resultados com informações úteis sobre o comportamento da moto em diferentes situações.

a Objetivos

Este projeto tem como objetivo empregar dois microcontroladores Arduino independentes, para o controle:

1. dos 4 graus de liberdade (guinada, deslocamento lateral, velocidade da esteira/rodas e rolagem) de um aparato para visualização da dinâmica de motocicletas. Nesse caso o Arduino atuará simultaneamente nos 3 motores e 1 servomotor do aparato, a partir dos sinais dos transdutores instalados no equipamento e dos comandos transmitidos via cabo de um computador (Figura 1);
2. de velocidade, atitude do condutor, estabilidade e trajetória de uma motocicleta em escala 1:8. Nesse caso o Arduino atuará simultaneamente em 2 servomotores e um motor CC (com ou sem escovas), a partir das medidas de uma IMU (Unidade de Medida Inercial) instalada na motocicleta e de sinais de comando transmitidos remotamente ou por cabos de um computador (Figura 2).

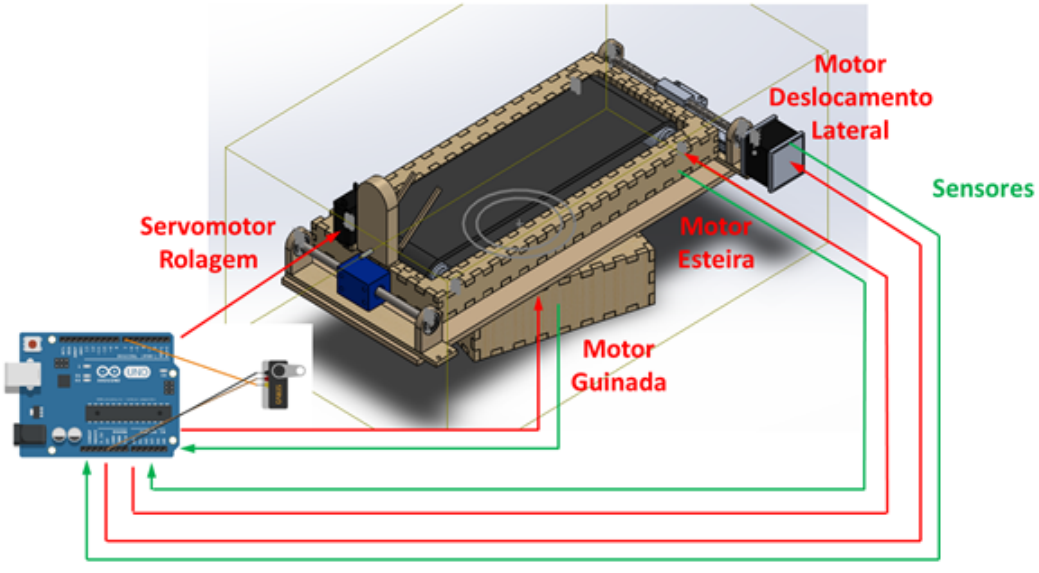

Figura 1: Controle do Aparato p/ Visualização da Dinâmica de Motocicletas em Escala c/ 4 GL's

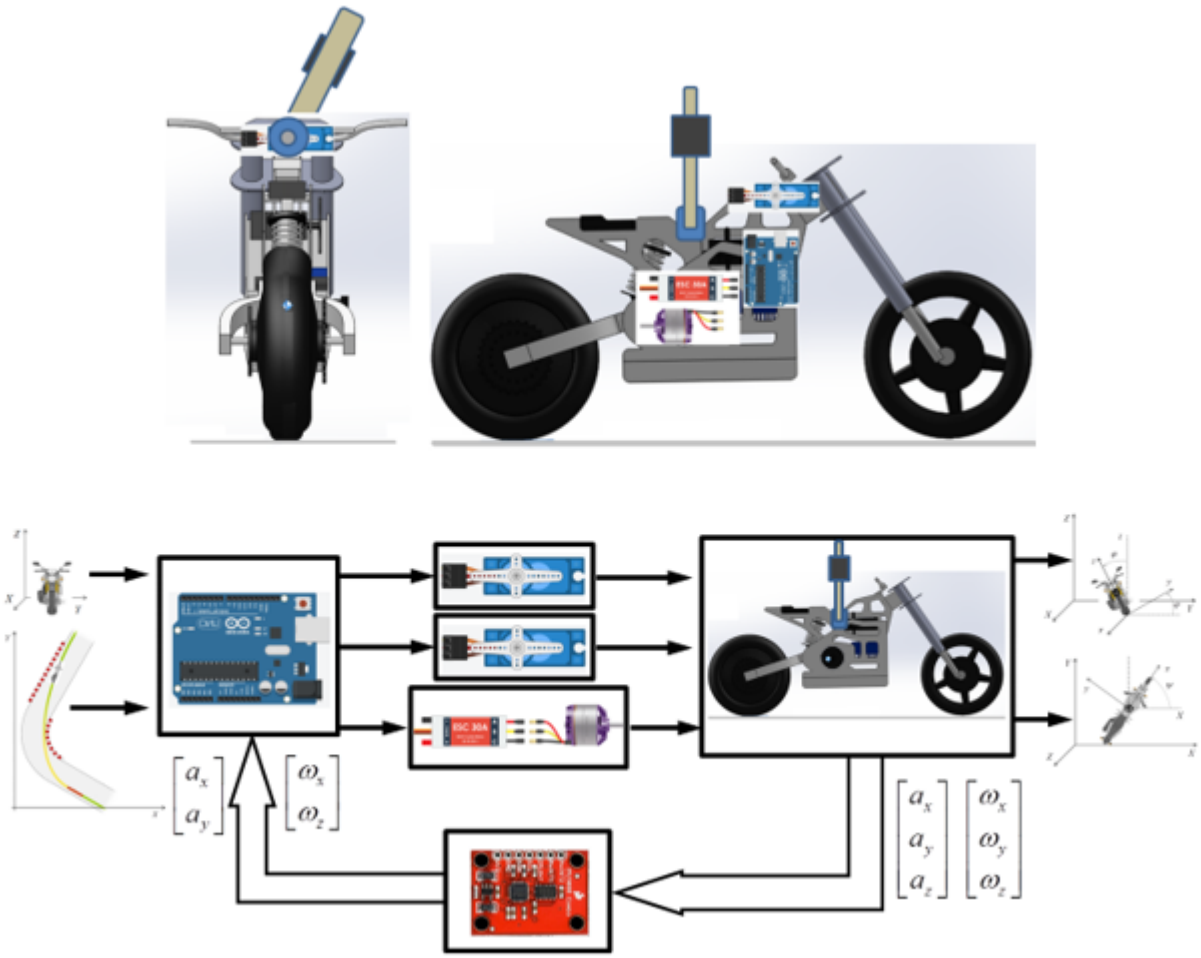

Figura 2: Controle da Motocicleta em Escala 1:8

Os 2 sistemas serão simulados em conjunto no Simscape/Simulink/MATLAB. A sequência de movimentos desejada será armazenada em um arquivo/planilha gerado pelas simulações em Simulink/MATLAB e posteriormente transmitidas aos respectivos Arduinos. Os Arduinos e motores reais serão empregados para verificação dos comandos fornecidos. Não será realizada a integração com o aparato e a motocicleta em tempo real (Hardware-in-the-Loop). Os transdutores reais também não serão empregados.

2 Metodologia

Para a realização das simulações foi utilizado MatLab [5]/Simulink [6], software que proporciona uma boa visualização de sistemas dinâmicos e os simula de forma rápida e prática. A validação das simulações foi realizada no microcontrolador Arduino [7], ideal para protótipos e testes.

a Simscape

Mais especificamente foi utilizado o Simscape [8], uma biblioteca de Simulink que permite a simulação de sistemas dinâmicos através da montagem do próprio sistema físico usando um diagrama de blocos, tornando desnecessária a realização da modelagem matemática do sistema.

Ele possui várias formas pré-existentes, mas também é possível importar modelos 3D projetados em softwares como o SolidWorks da Dassault Systèmes [9] e o Fusion 360 da Autodesk [10]. As relações entre cada objeto do sistema, ou seja, as restrições de movimento, juntas, posicionamento, etc. também são blocos disponíveis no Simscape. Essas relações são dadas entre *frames*, ou seja, pontos específicos nas geometrias que possuem seus próprios eixos de referência.

1 Rótula (Revolute Joint)

A rótula do Simscape cria uma relação de rotação entre um *frame* base e um seguidor. Eles são alinhados de forma que o eixo Z de ambos os *frames* coincida e o movimento angular é realizado em torno desse mesmo eixo.

É possível configurar como o seguidor se moverá em relação à base através de dois parâmetros: torque e movimento. O torque pode ser nenhum, calculado automaticamente ou provido por uma entrada, enquanto o movimento pode ser apenas calculado automaticamente ou provido por uma entrada. Observa-se que nem todas as combinações desses dois parâmetros são possíveis, já que não pode haver contradições. O modo utilizado para as rótulas neste projeto é com o movimento calculado automaticamente e o torque provido pela entrada, que virá do controlador.

O bloco ainda permite a medição da posição, velocidade e aceleração do seguidor em relação à base.

A Figura 3 mostra o bloco que representa a rótula. A entrada B é a base e a F é o seguidor. O bloco pode ter mais entradas dependendo do tipo de atuação a ser aplicado e das medições feitas.

Figura 3: Bloco da rótula

2 Junta prismática (Prismatic Joint)

A junta prismática do Simscape cria uma relação de translação entre um *frame* base e um seguidor. Eles são alinhados de forma que o eixo Z de ambos os *frames* coincida e o movimento linear é realizado ao longo desse mesmo eixo.

A forma como o seguidor vai se mover em relação à base funciona de maneira análoga à rótula, assim como as medições. A diferença é que ao invés de um torque é uma força, e o modo que será utilizado nesse projeto será com a força fornecida por uma entrada e o movimento calculado automaticamente a partir dessa força. Será dessa forma para simular o funcionamento do motor DC (sessão 2.b.2), ele gera um torque, que gera uma força, que gera o deslocamento lateral (sessão 3.b.4) do aparato.

A Figura 4 mostra o bloco que representa a junta prismática. A entrada B é a base e a F é o seguidor. O bloco pode ter mais entradas dependendo do tipo de atuação a ser aplicado e das medições feitas.

Figura 4: Bloco da junta prismática

b Arduino

O Arduino é um placa de prototipagem programada em uma versão adaptada da linguagem de programação C++ ou pode ser controlado diretamente pelo Simulink, através de uma biblioteca [11] que integra o software diretamente ao hardware da placa. Isso torna sua utilização simples e permite mudanças nos projetos sem grandes complicações. A placa possui diversas portas digitais e analógicas que podem ser programadas como entradas ou saídas, permitindo o controle de um sistema completo com sensores e atuadores.

O Arduino usado no projeto é o Arduino UNO [12].

1 Servomotor

O servomotor possui, além dos cabos de alimentação, um cabo por onde ele recebe um sinal que dita o torque que será aplicado. O torque é calculado automaticamente pela biblioteca "Servo.h" [13] do Arduino a partir de apenas o ângulo desejado informado no código. Pela interface Simulink/Arduino basta fornecer um sinal com o ângulo em graus para o bloco.

Pelo Simulink o controle do servo é feito pelo bloco da Figura 5.

Figura 5: Bloco do servomotor no Simulink

A Figura 6 mostra o tipo de servo utilizado no projeto.

Figura 6: Servomotor Micro de Arduino, fonte: [1]

2 Motor DC

O motor DC possui apenas uma entrada de alimentação e uma de aterramento. Ele é controlado de acordo com a voltagem que ele recebe, quando maior a voltagem, maior sua velocidade de rotação. Na interface

Simulink/Arduino, para controlar o motor o sinal de entrada deve ser um número entre -255 e 255, onde o valor absoluto irá controlar a velocidade e o sinal controla o sentido do movimento.

Pelo Simulink o controle do motor é feito pelo bloco da Figura 7.

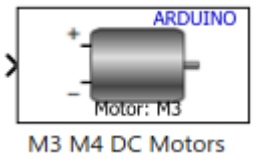

Figura 7: Bloco do servomotor no Simulink

A Figura 8 mostra o motor usado no projeto. Sua velocidade máxima de rotação é de 13100 rpm, segundo o seu *datasheet* [14].

Figura 8: Mini Motor DC 3V, fonte: [2]

c Validação

Para quesito de validação da capacidade de simulação do Simscape e da integração com o Arduino será apresentado um sistema de controle de um pêndulo invertido em um carro que é fornecido [15] no próprio site da MathWorks como exemplo de uso do Simscape.

1 Simulação

O sistema é composto por uma guia (Figura 9), um carro (Figura 10) e um pêndulo duplo (Figura 11). O sistema montado pode ser visto na Figura 12 e seu diagrama de blocos na Figura 13, onde é possível observar as rótulas e junstas prismáticas mencionadas anteriormente.

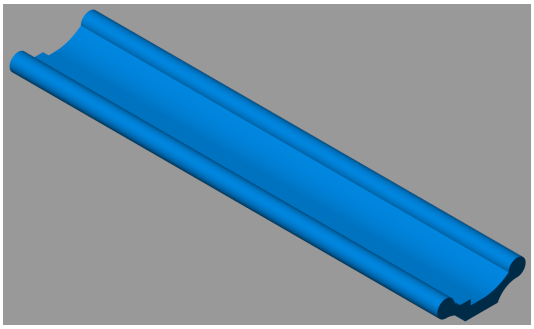

Figura 9: Guia

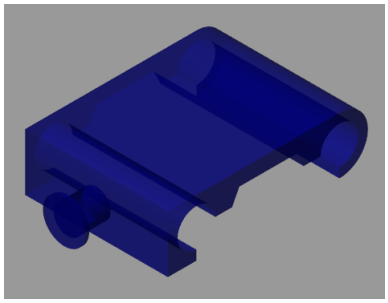

Figura 10: Carro

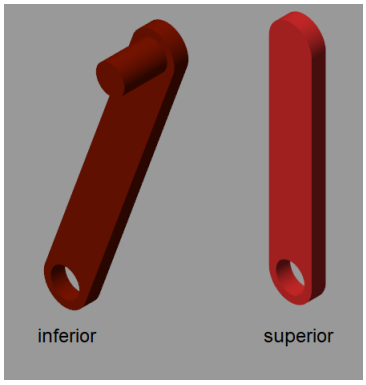

Figura 11: Partes do pêndulo

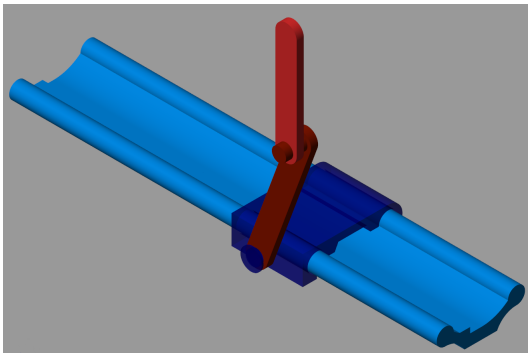

Figura 12: Todos os elementos do pêndulo duplo invertido montados

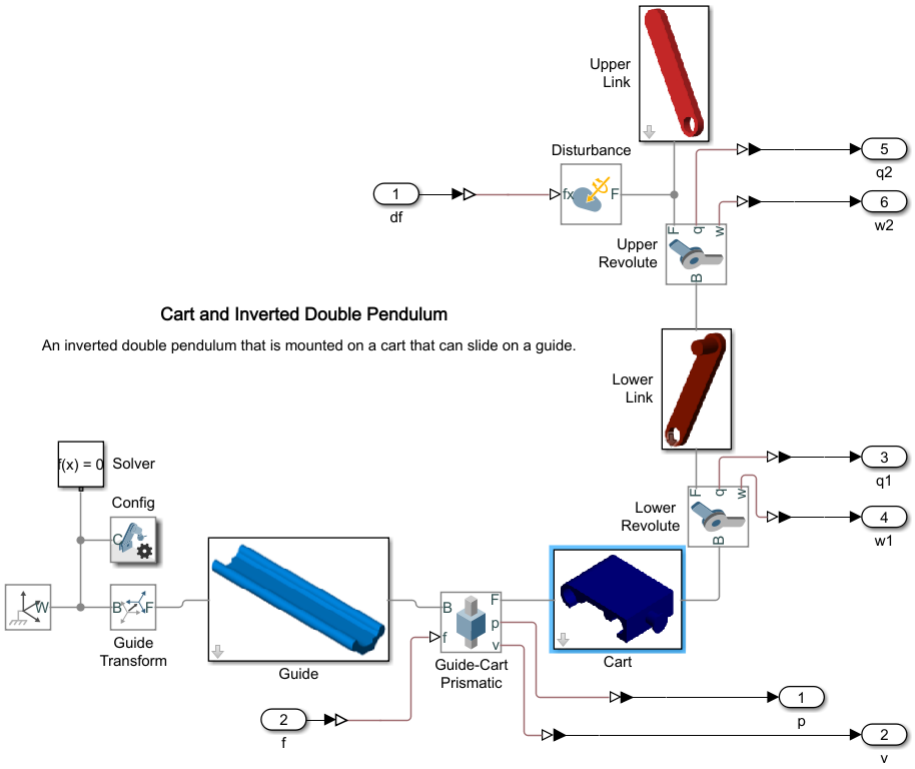

Figura 13: Diagrama de blocos do pêndulo duplo invertido

Na Figura 13 é possível ver que em cada uma das 3 juntas existentes (*Guide-Cart Prismatic*, *Lower Revolute* e *Upper Revolute*) estão sendo medidas a posição e velocidade de seus respectivos seguidores, esses dados serão usando para fechar a malha do sistema.

Na junta *Guide-Cart Prismatic* está sendo aplicada a força f, entrada do sistema, e no seguidor da junta *Upper Revolute* está sendo aplicada uma perturbação externa, com o objetivo de verificar como o sistema reage à ela.

O exemplo aplica um controle proporcional com realimentação negativa conforme o diagrama de blocos da Figura 14, e a perturbação aplicada é a mostrada no gráfico da Figura 15.

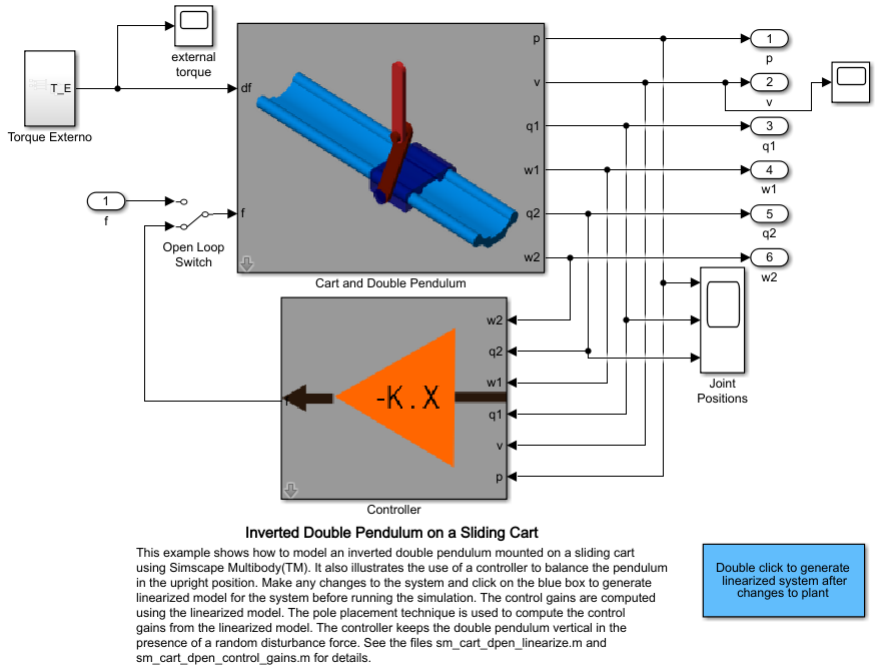

Figura 14: Diagrama de blocos do sistema de controle do pêndulo duplo invertido

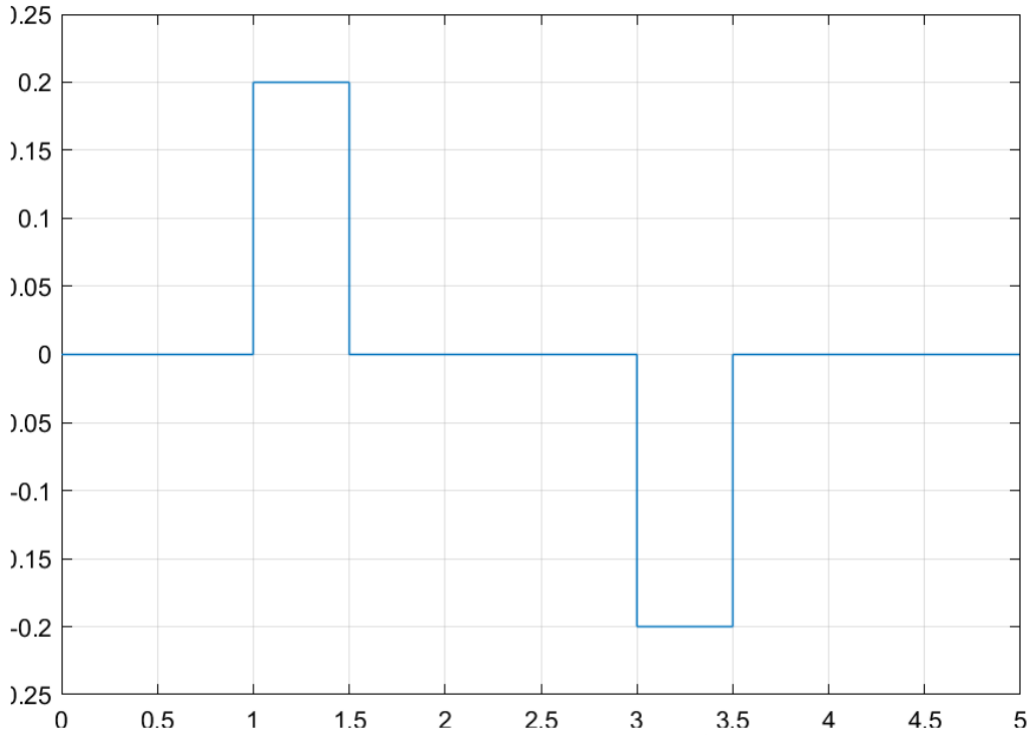

Figura 15: Força externa aplicada na parte superior do pêndulo duplo invertido em N, gerando um torque

O bloco do controlador permite que sejam escolhidos quais serão os polos de malha fechada do sistema, mas a forma é feita internamente a linearização do sistema e o cálculo do ganho do controlador não foi explorada.

O exemplo apresenta os resultados esperados para o conhecido sistema do pêndulo duplo invertido, como

pode ser observado na Figura 16, portanto validando o uso do Simscape para as simulações a serem realizadas no decorrer do projeto.

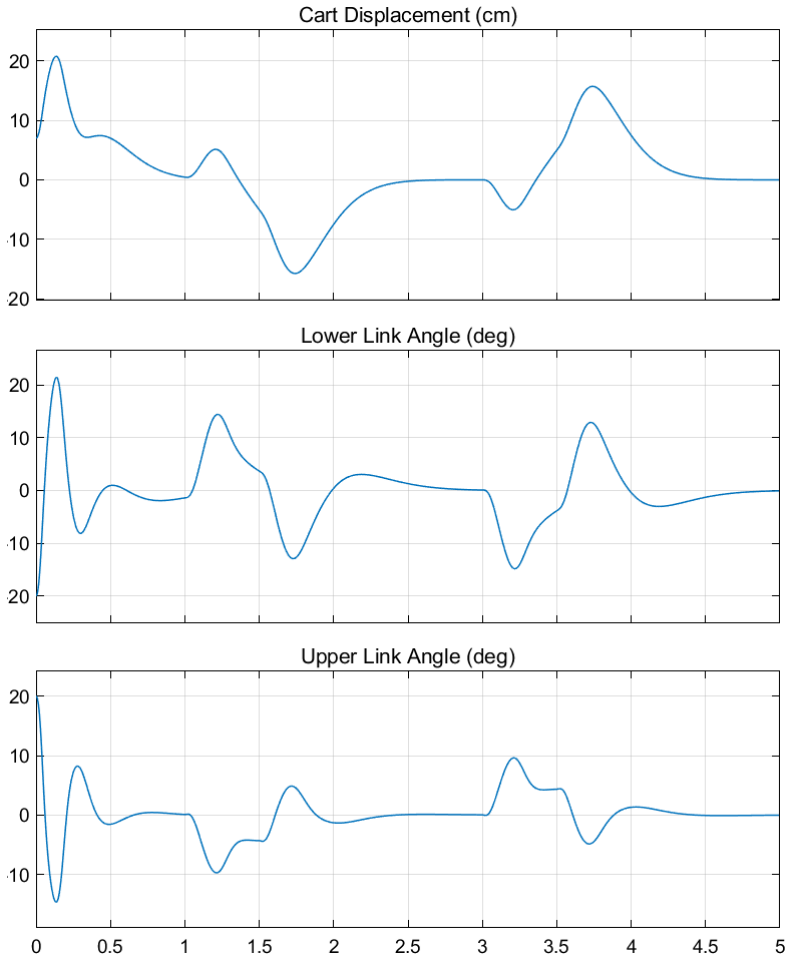

Figura 16: Resultados do sistema de controle do pêndulo duplo invertido

A velocidade do carro também foi medida e pode ser observada na Figura 17.

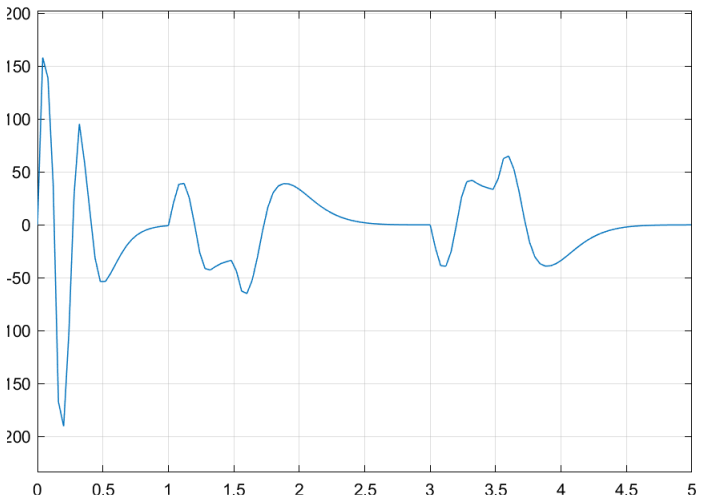

Figura 17: Velocidade do Carro em cm/s

2 Integração com o Arduino

O MatLab possibilita que os dados de um *Scope* (bloco usado para gerar os gráficos da Figura 16) sejam exportados para uma variável no *workspace*, espaço onde as variáveis do MatLab ficam temporariamente armazenadas. Com esses dados no *workspace*, um simples *script* pode salvá-los num arquivo *MAT-File*, que poderá ser posteriormente lido por um modelo do Simulink.

No Simulink há um bloco chamado *Signal Editor* (Figura 18) que é capaz de ler um arquivo *MAT-File* e gerar sinais com ele. Feito esse procedimento com todos os sinais necessários, pode-se aplicar os valores obtidos na interface Simulink/Arduino.

Figura 18: Bloco que será usado para ler os arquivos que contém os movimentos

Para aplicar o movimento dos pêndulos serão usados servomotores. A informação que os blocos dos servos precisam é simplesmente o ângulo em graus, como explicado em 2.b.2, porém o ângulo vindo da simulação tem valores entre -90º e 90º, enquanto o servomotor real trabalha entre 0º e 180º. Por isso, para realizar essa conversão basta somar 90º ao valor vindo da simulação antes de enviá-lo ao bloco do servomotor.

Para aplicar o movimento do carro será usado um motor DC que atua sobre um conjunto de pinhão e cremalheira (como na Figura 19). Pensando numa versão simplificada desse sistema pode-se assumir que a velocidade linear gerada na cremalheira é igual à velocidade angular gerada pela rotação do motor atuando no pinhão vezes o raio do pinhão.

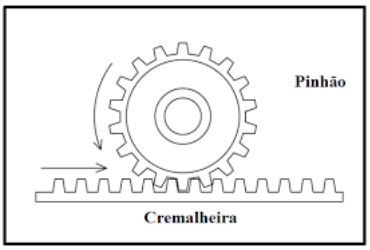

Figura 19: Bloco que será usado para ler os arquivos que contém os movimentos

Como dito anteriormente, a velocidade máxima de rotação do motor utilizado é de 13100 rpm. Considerando um pinhão de 5cm de raio, sua circunferência é de $2\times\pi\times5$cm = 31,42cm, portanto temos:

$$v_{max} = \frac{31,42}{60} \times 13100 = 6859$$

Ou seja, uma entrada de 255 no bloco do motor equivale a uma rotação a 6859cm/s no sentido positivo e uma entrada de -255 tem a mesma velocidade no outro sentido. Então para realizar a conversão da velocidade vinda da simulação para o número que será a entrada do bloco do motor basta multiplicá-la por 255 e dividi-la por 6859.

A interface com o Arduino, enfim, fica como demonstrado na Figura 20.

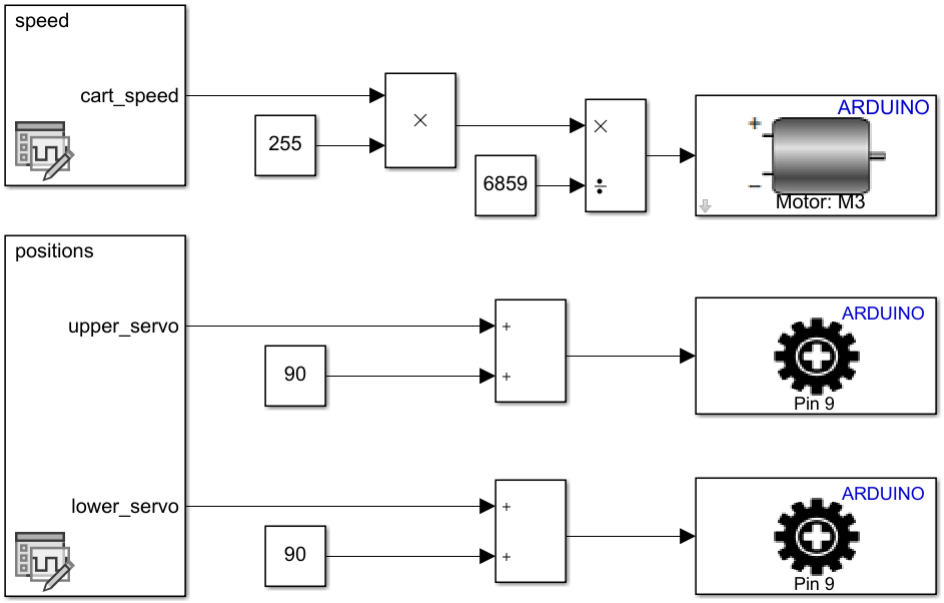

Figura 20: Interface Simulink/Arduino com as devidas conversões de dados

3 Problema encontrado

Na hora de enviar o programa funcionando para o Arduino de verdade um problema surgiu: memória. O Arduino UNO, uma placa compacta e simples, possui pouca memória interna e não é o suficiente para armazenar todos os dados dos movimentos vindos da simulação.

Para contornar esse problema duas medidas foram tomadas:

- A redução do passo de tempo da simulação do Simulink para 0,04. Esse número foi encontrado por tentativa e erro e foi o melhor que gera uma quantidade de dados que o Arduino UNO consegue suportar e não ocorre nenhuma derivada infinita na simulação. É claro que há, infelizmente, perda de dados, mas ainda há dados o suficiente para que o sistema funcione de maneira correta.
- Cada um dos servomotores e motor teve de ser testado separadamente. Para que todos funcionassem simultaneamente o volume de dados de cada um teria de ser ainda menor, o que não é possível pois gera singularidades na simulação.

4 Resultados

Apesar dos obstáculos, os movimentos foram transmitidos perfeitamente aos servomotores, portanto validando o uso do Arduino e da interface Simulink/Arduino para as aplicação dos movimentos gerados pelas simulações a serem realizadas no decorrer do projeto.

A montagem do Arduino físico, os dados transmitidos à ele, o por que de o motor DC não ter sido testado e mais detalhes sobre as configurações em geral e o passo a passo serão explicitadas mais adiante quando aplicados ao projeto real.

3 Modelagem do aparato

A Figura 21 mostra o aparato montado com a motocicleta e o pêndulo invertido que representa o condutor. A seguir serão mostradas todas as partes que compõe o aparado e todas as juntas entre eles. Todos os modelos 3D foram desenhados com o Fusion 360 da Autodesk. Todos os modelos estão sendo mostrados em vista isométrica.

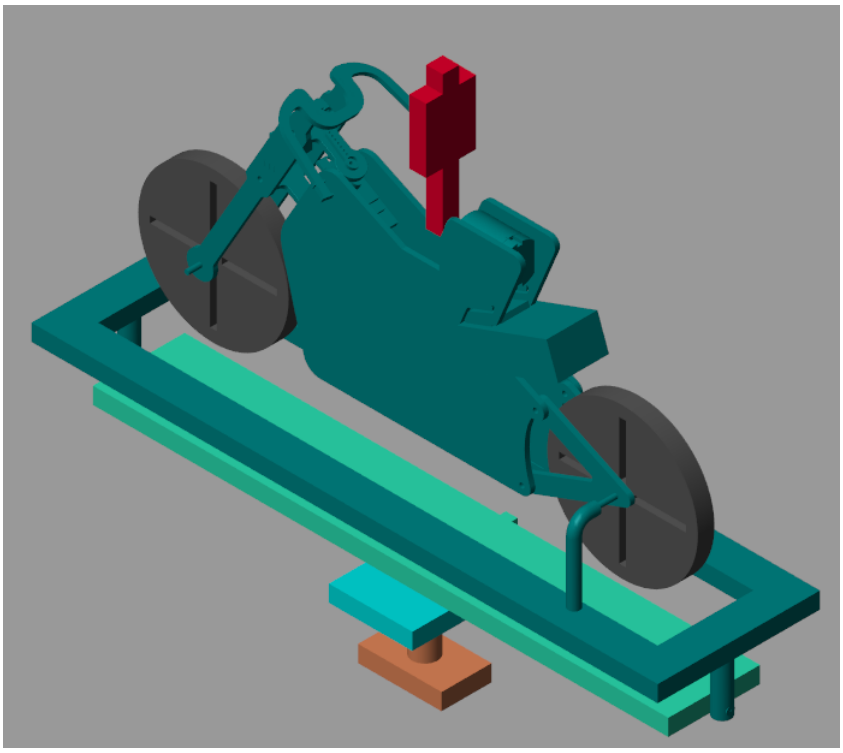

Figura 21: Aparato, moto e condutor

a Partes do modelo

1 Base de guinada

A base de guinada é a mostrada na Figura 22. Ela possui um *frame* de referência (F) e um *frame* que será a a base da rótula responsável pela guinada (F1). É ela quem ficará apoiada no chão.

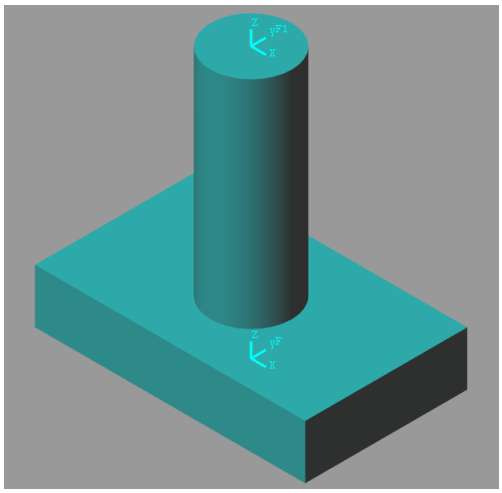

Figura 22: Base de guinada

2 Base de deslocamento lateral

A base de deslocamento lateral é a mostrada na Figura 23. Ela possui um *frame* que será o seguidor da rótula responsável pela guinada (F1) e um *frame* que será a a base da junta prismática responsável pelo deslocamento lateral (F2).

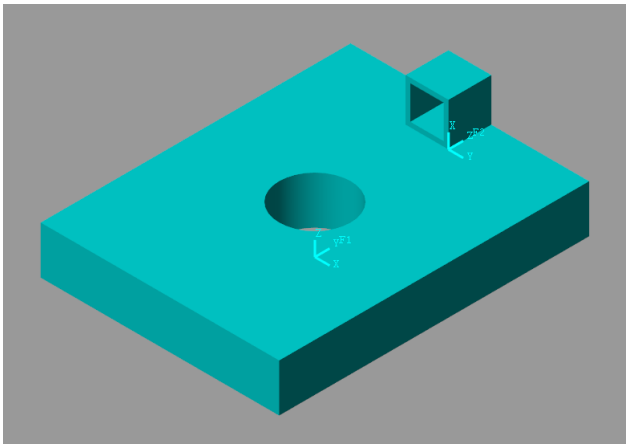

Figura 23: Base de deslocamento lateral

3 Base de rolagem

A base de rolagem é a mostrada na Figura 24. Ela possui um *frame* que será o seguidor da junta prismática responsável pelo deslocamento lateral (F2) e um *frame* que será a a base da rótula responsável pela rolagem (F1).

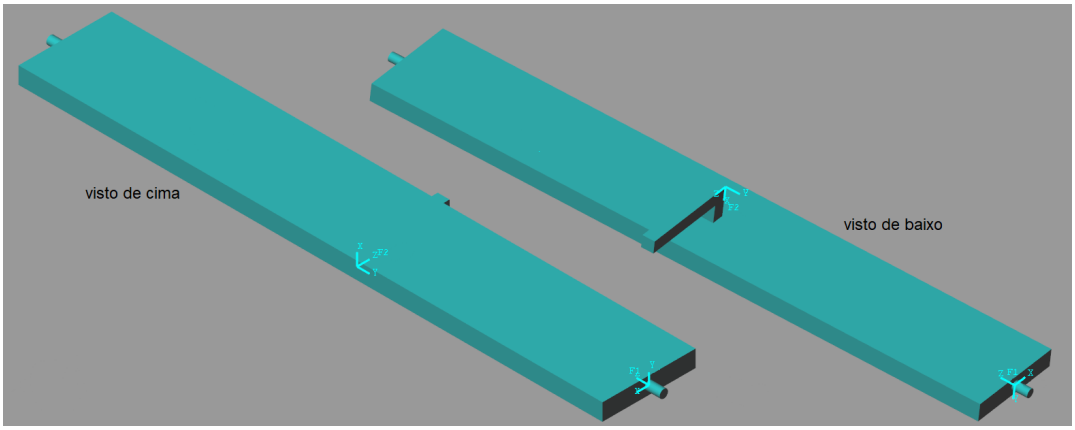

Figura 24: Base de rolagem

4 Suporte

O suporte é a mostrado na Figura 25. Ele possui um *frame* que será o seguidor da rótula responsável pela rolagem (F2) e dois elementos que serão acoplados ao corpo da motocicleta, mantendo-a fixa em relação ao suporte, como mostrado na figura.

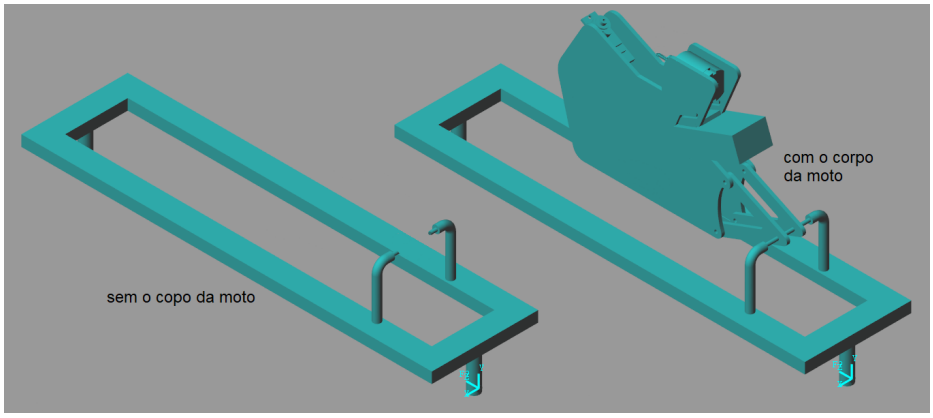

Figura 25: Suporte

5 A motocicleta

A motocicleta possui um *frame* onde será posicionado o condutor, ele será o *frame* base da rótula do pêndulo invertido (Figura 26). Possui também um *frame* que será a base da rótula de rotação da roda traseira (Figura 27). E por fim possui um *frame* que será a base da rótula de rotação do guidão da motocicleta.

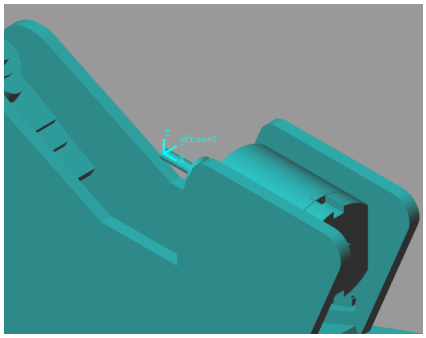

Figura 26: *Frame* onde ficará o pêndulo

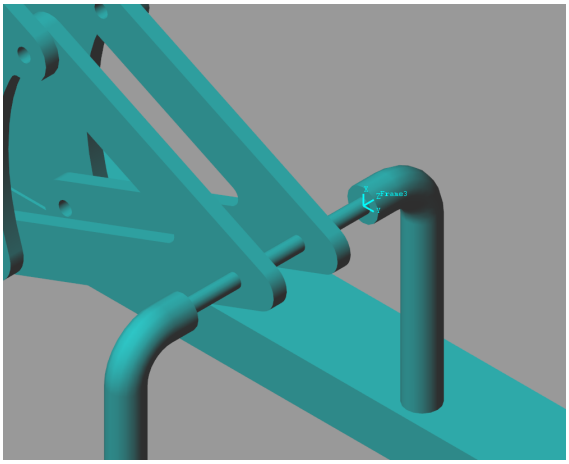

Figura 27: *Frame* onde ficará a roda traseira

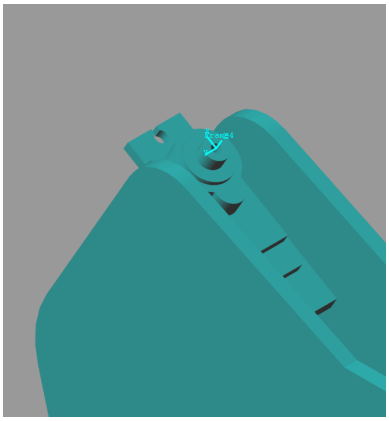

Figura 28: *Frame* onde ficará o guidão

6 O pêndulo invertido

O pêndulo invertido possui um *frame* que será o seguidor da rótula que representa o condutor(Figura 29).

Figura 29: Pêndulo que representa o condutor

7 O guidão

O guidão possui um *frame* que será encaixado no corpo da motocicleta e outro que será a base da rotação da roda dianteira. Ambos podem ser vistos na Figura 30.

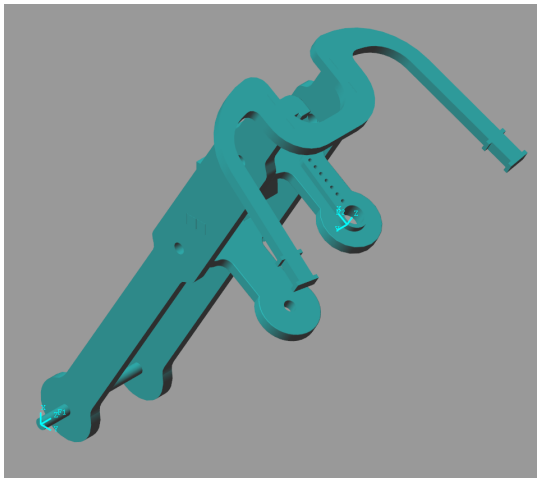

Figura 30: Guidão

8 As rodas

Ambas as rodas são iguais e possuem um *frame* em seu centro. Uma será conectada ao corpo da motocicleta e a outra ao guidão. É possível ver na Figura 31.

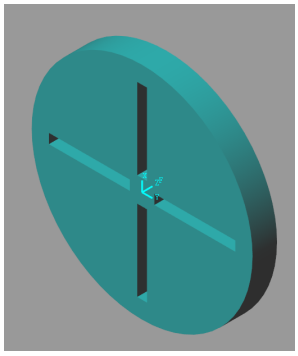

Figura 31: Roda

b Juntas do modelo

1 Guinada

A guinada é realizada através de uma rótula entre o *frame* F1 da base de guinada e o *frame* F1 da base de deslocamento lateral. A Figura 32 mostra como o movimento da rótula da guinada acontece.

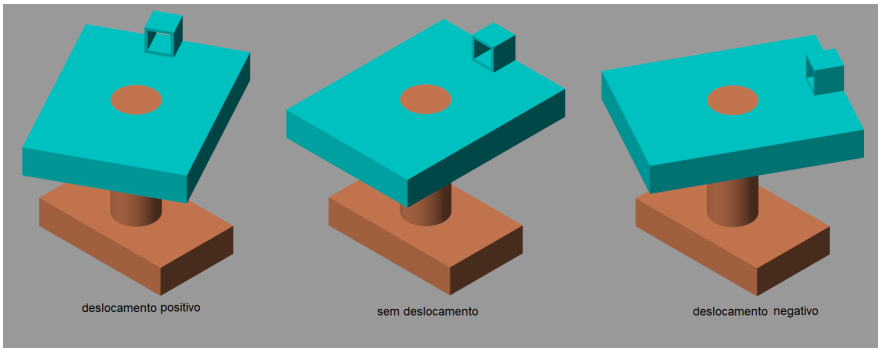

Figura 32: Movimento de guinada

2 Deslocamento Lateral

O deslocamento lateral é realizado através de uma junta prismática entre o *frame* F2 da base de deslocamento lateral e o *frame* F2 da base de rolagem. A Figura 33 mostra como o movimento da junta de deslocamento lateral acontece e a Figura 34 mostra a junta mais de perto.

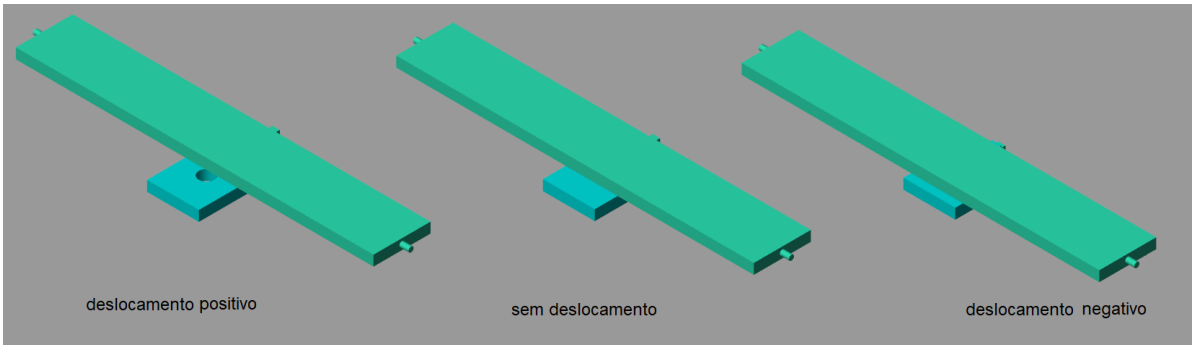

Figura 33: Movimento de deslocamento lateral

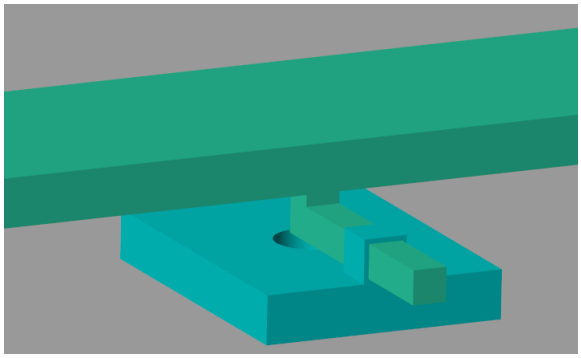

Figura 34: Detalhe da junta prismática

3 Rolagem

A rolagem é realizada através de uma rótula entre o *frame* F1 da base de rolagem lateral e o *frame* F2 do suporte. A Figura 35 mostra como o movimento da rótula de rolagem acontece e a Figura 36 mostra a rótula mais de perto.

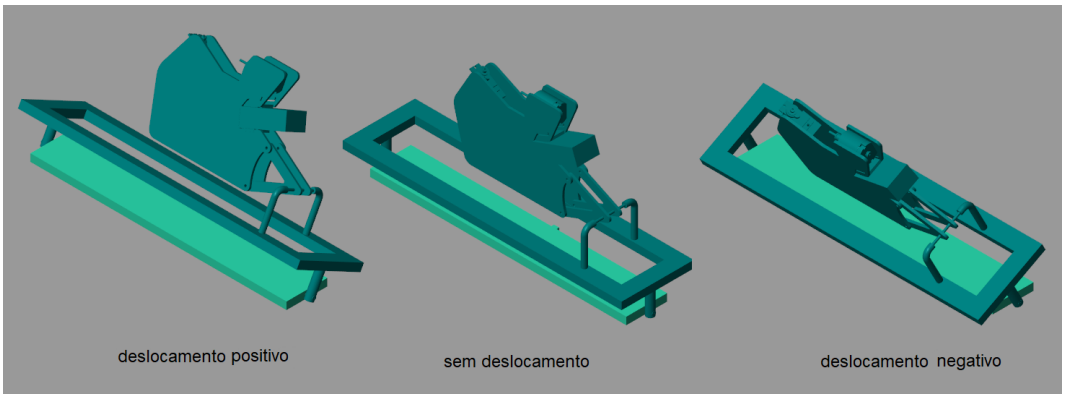

Figura 35: Movimento de rolagem

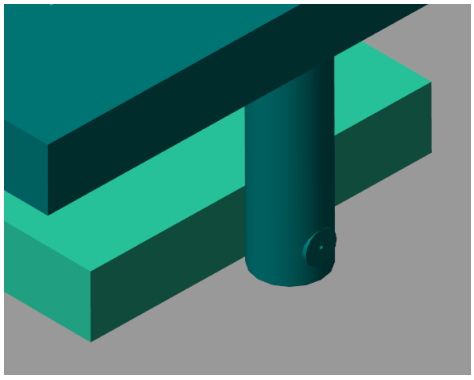

Figura 36: Detalhe da rótula de rolagem

4 Inclinação do condutor

A inclinação do condutor é realizada através de uma rótula entre o *frame* no topo da motocicleta e o único *frame* do pêndulo invertido. A Figura 37 mostra como o movimento da rótula de rolagem acontece e a Figura 38 mostra a rótula mais de perto.

Figura 37: Movimento do condutor

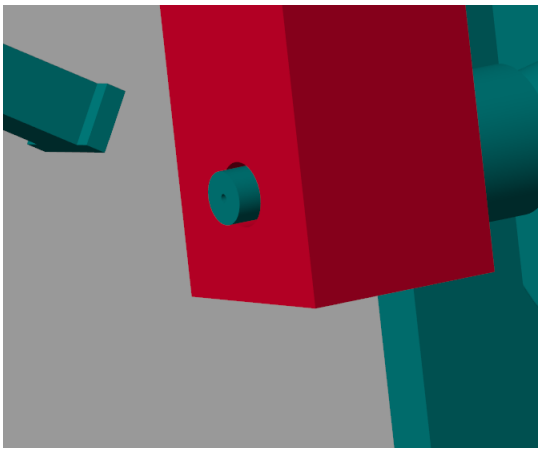

Figura 38: Detalhe da rótula do condutor

5 Rotação do guidão

A rotação do guidão acontece com uma rótula onde a base é o *frame* no topo do corpo da moto e o *frame* de cima do guidão. A Figura 39 mostra como o movimento da rótula do guidão acontece e a Figura 40 mostra a rótula mais de perto.

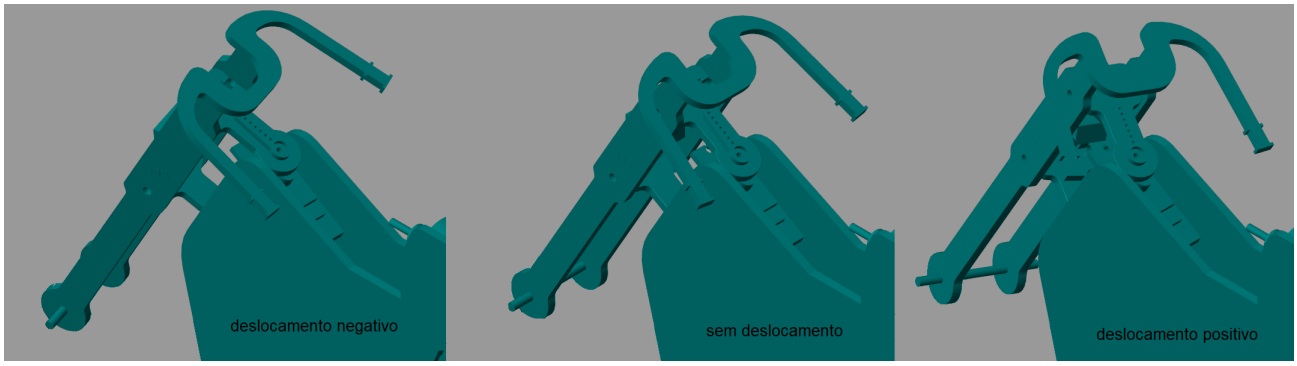

Figura 39: Movimento do guidão

Figura 40: Detalhe da rótula do guidão

6 Deslocamento das rodas

As rodas simplesmente giram em torno do eixo que atravessa seu centro. A roda traseira é a junção por rótula do *frame* da roda com o *frame* base presente no corpo da motocicleta mostrado na Figura 27, e a roda dianteira é a junção por rótula do *frame* da roda com o *frame* base no guisão, mostrado na Figura 30.

c Diagrama de blocos

Para todas as rótulas e a junta prismática haverá uma entrada de torque ou força, respectivamente, e em todas serão medidas a posição, em radianos ou metros, e a velocidade, em radianos ou metros por segundo. O diagrama de blocos, enfim, do sistema montado no Simscape ficou como mostrado na Figura 41.

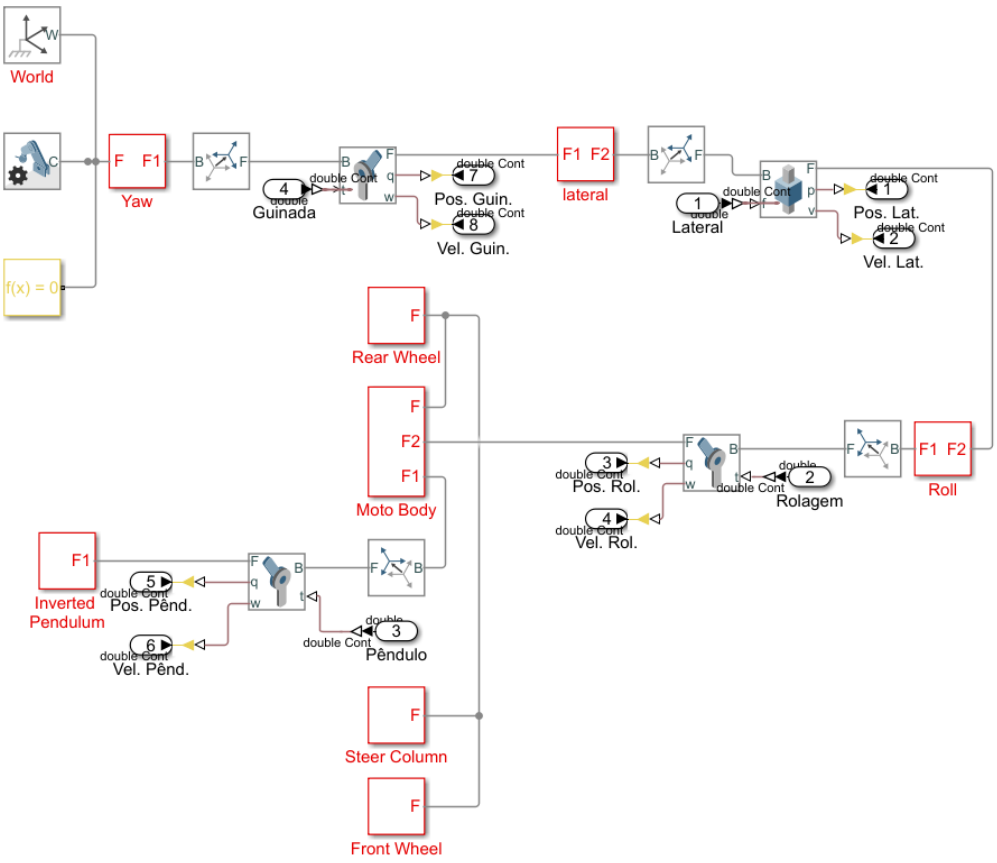

Figura 41: Diagrama de blocos do modelo

4 Controle

A entrada do sistema de controle são as posições desejadas para cada um dos graus de liberdade e a saída são as posições lidas nas juntas do modelo.

Em todos os testes de controladores, as entradas utilizadas como posições desejadas foram as da Figura 42

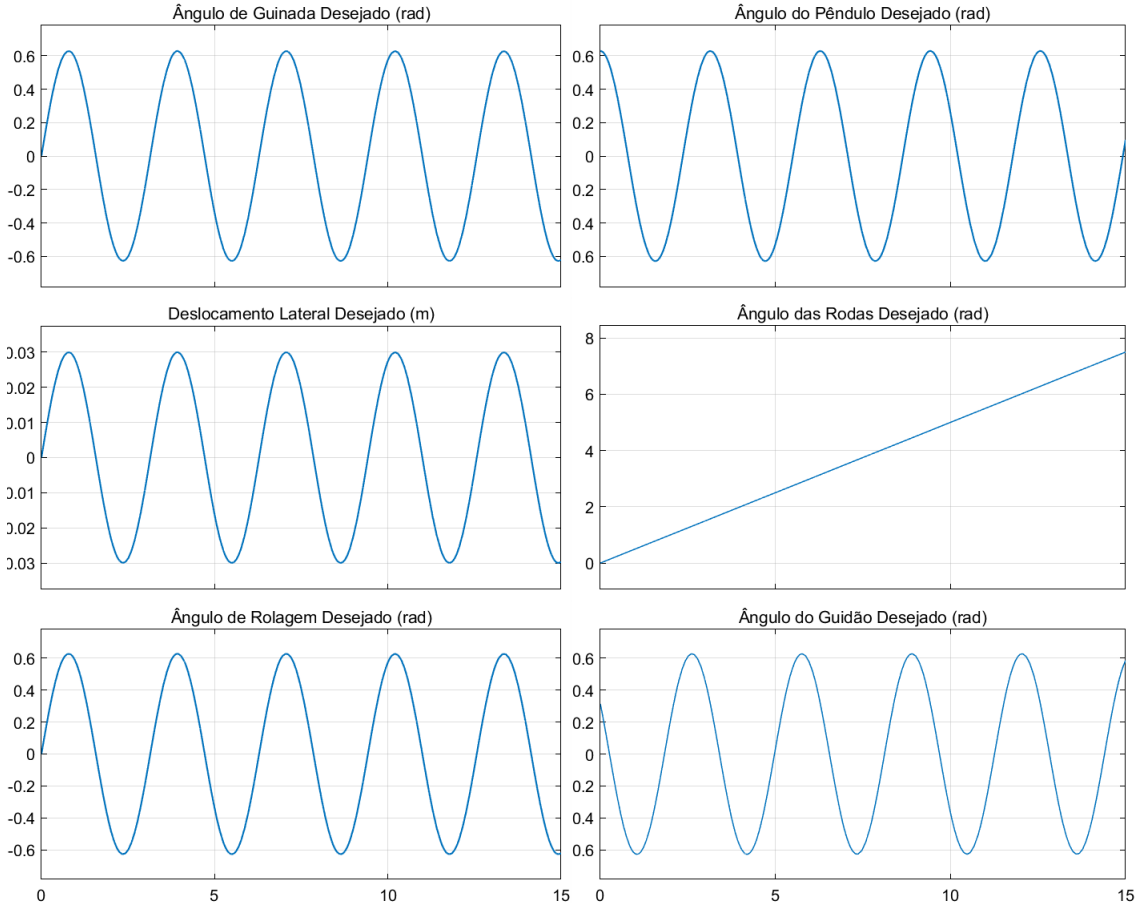

Figura 42: Entradas do sistema de controle

a Controle proporcional

Foi testado inicialmente um controle P com os ganhos todos iguais a 1. O diagrama de blocos do sistema de controle P pode ser visto na Figura 43.

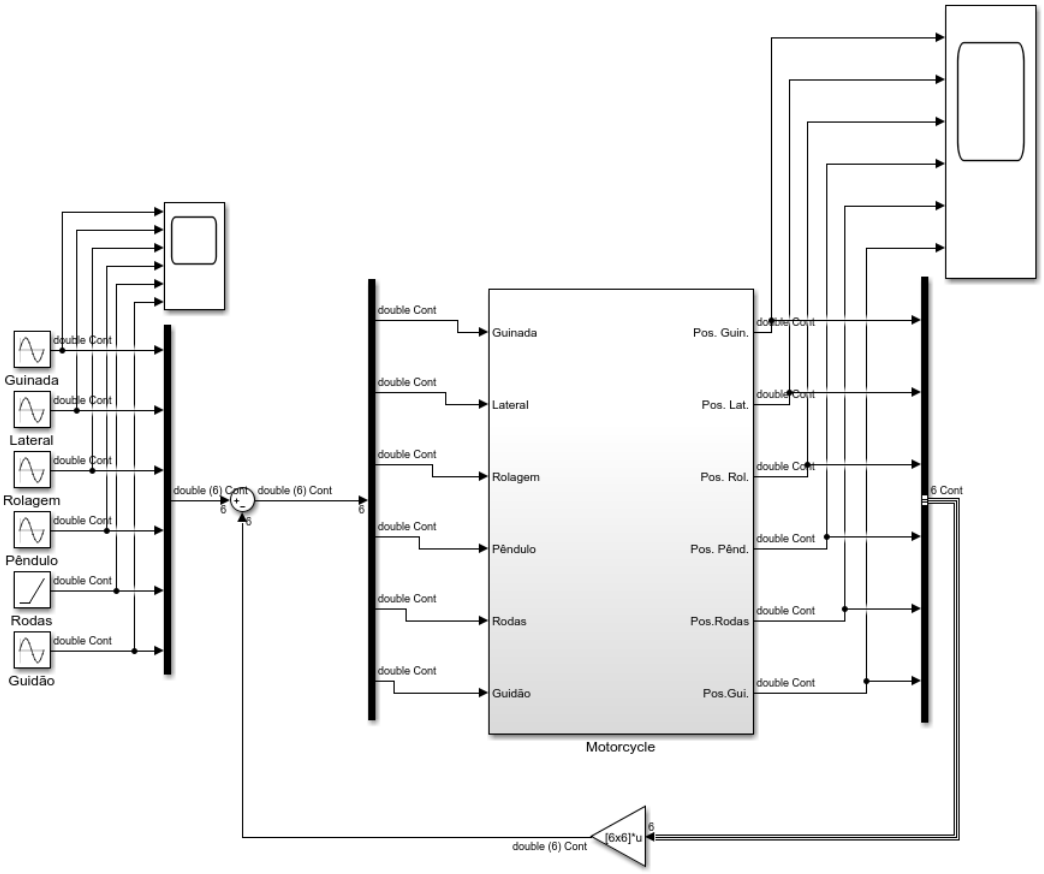

Figura 43: Diagrama de blocos do sistema de controle P

A matriz K é a seguinte:

$$K = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{[6\times6]}$$

Os resultados obtidos foram as da Figura 44.

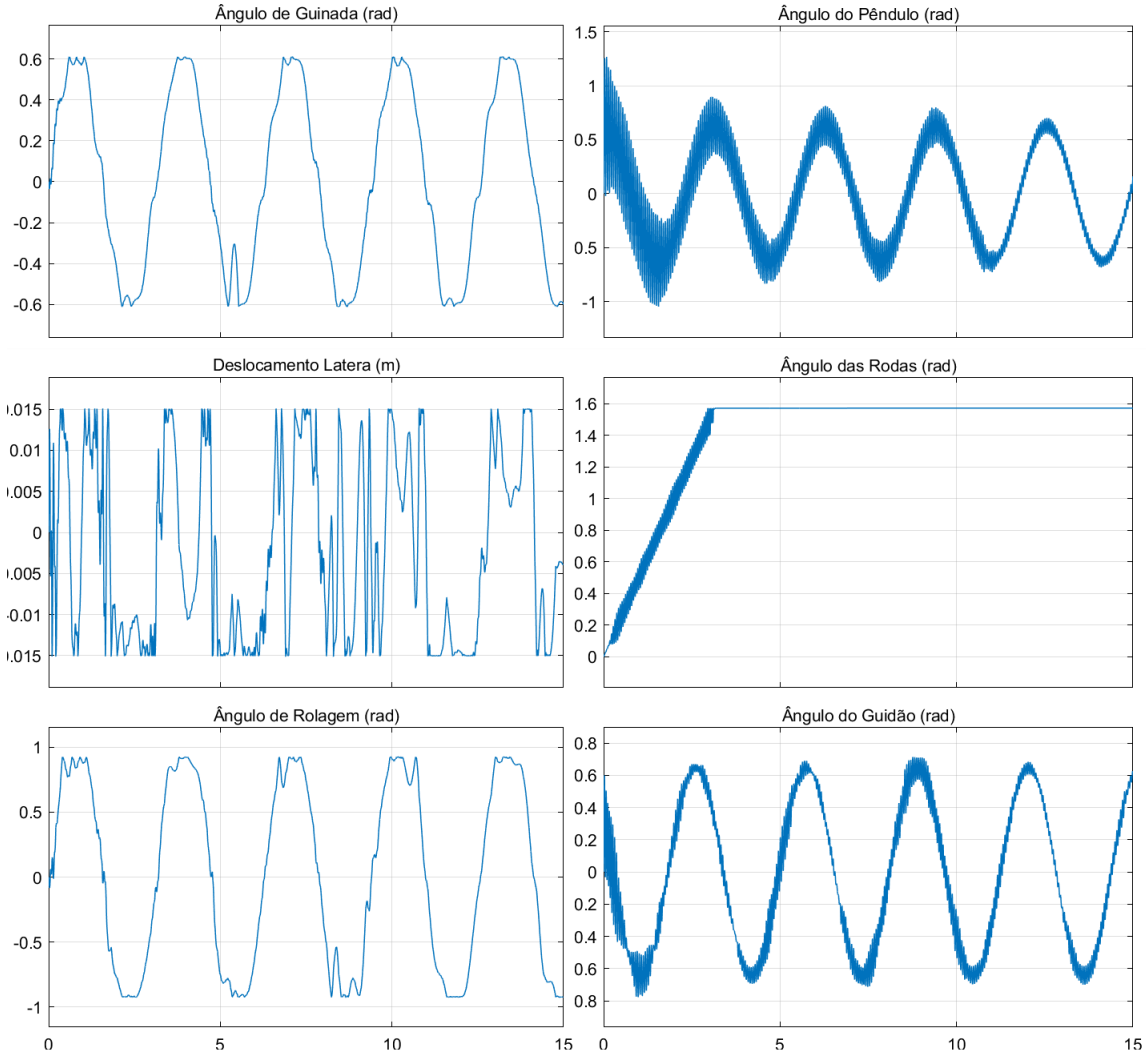

Figura 44: Resultados do sistema de controle PD

Os resultados até tem uma forma parecida com as posições desejadas, mas existe muita oscilação. Para resolver o problema das oscilações deve-se acrescentar um ganho derivativo, como será feito a seguir.

b Controle proporcional derivativo

Foi testado então um controle PD também com os ganhos todos iguais a 1. O diagrama de blocos do sistema de controle PD pode ser visto na Figura 45.

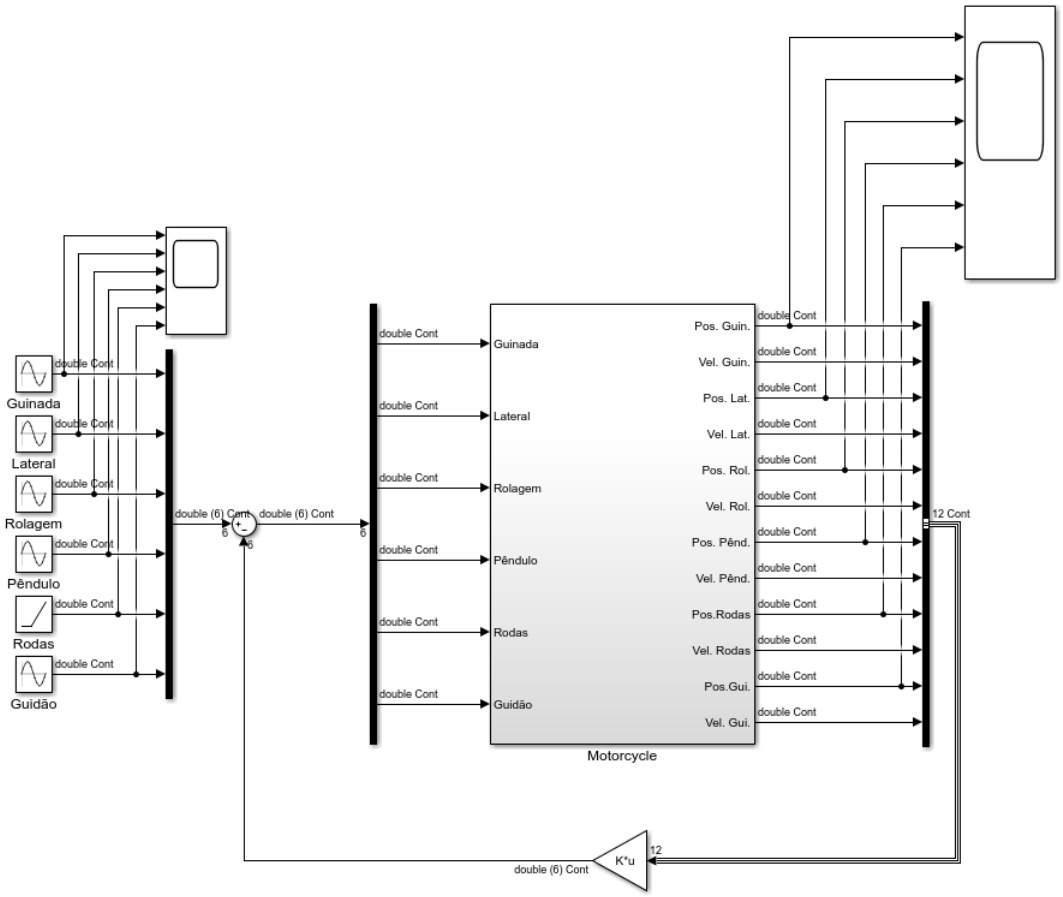

Figura 45: Diagrama de blocos do sistema de controle PD

A matriz K é a seguinte:

$$K = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}_{[6\times 12]}$$

Os resultados obtidos foram as da Figura 46.

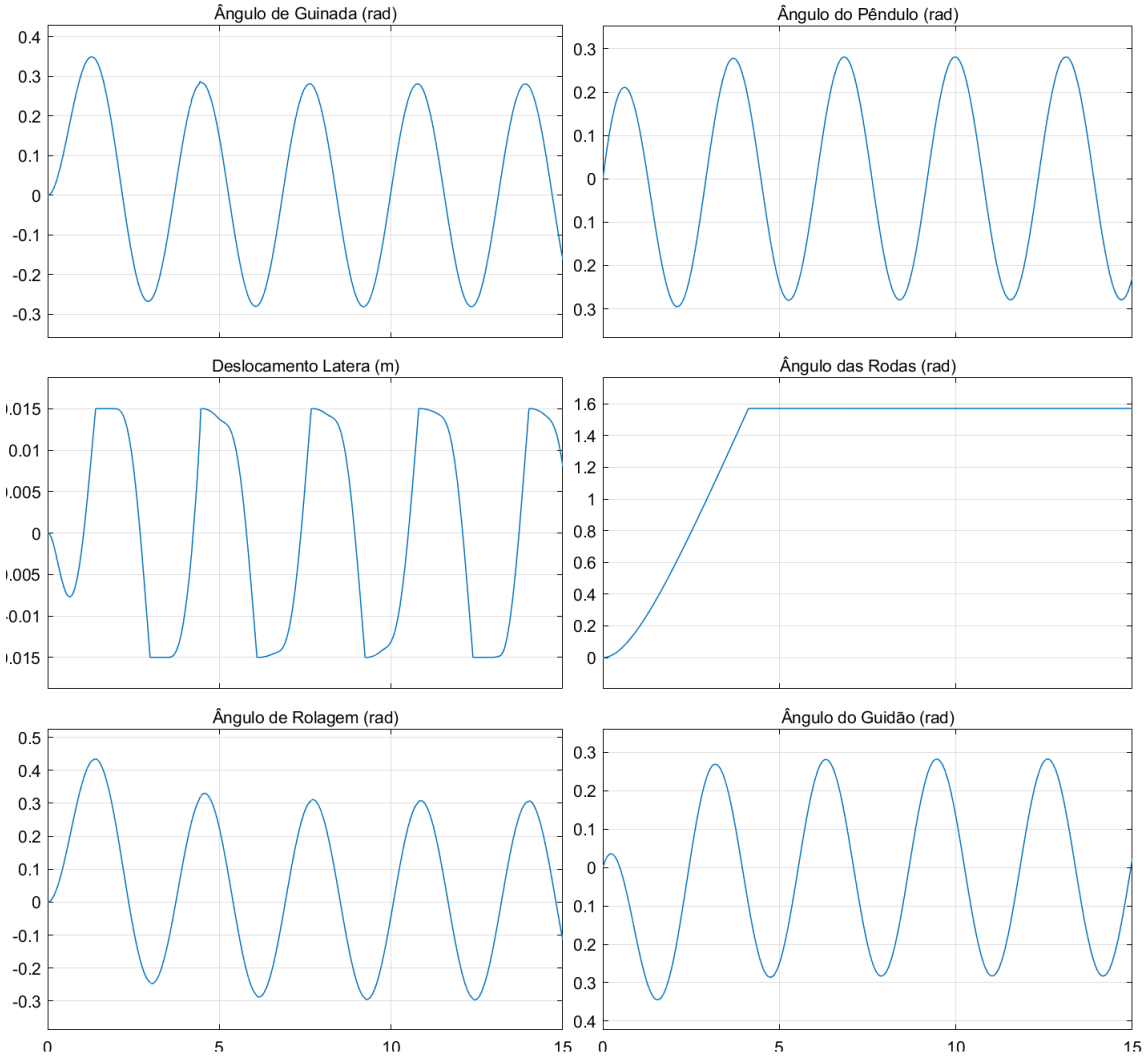

Figura 46: Resultados do sistema de controle PD

A resposta do controlador foi rápida, não oscilatória e satisfatória, mas por via das dúvidas posteriormente será testado também um controlador mais robusto a fim de se comparar resultados.

c Controle proporcional integral

Antes, foi testado um controlador PI também com os ganhos todos iguais a 1. O diagrama de blocos do sistema de controle PI pode ser visto na Figura 47.

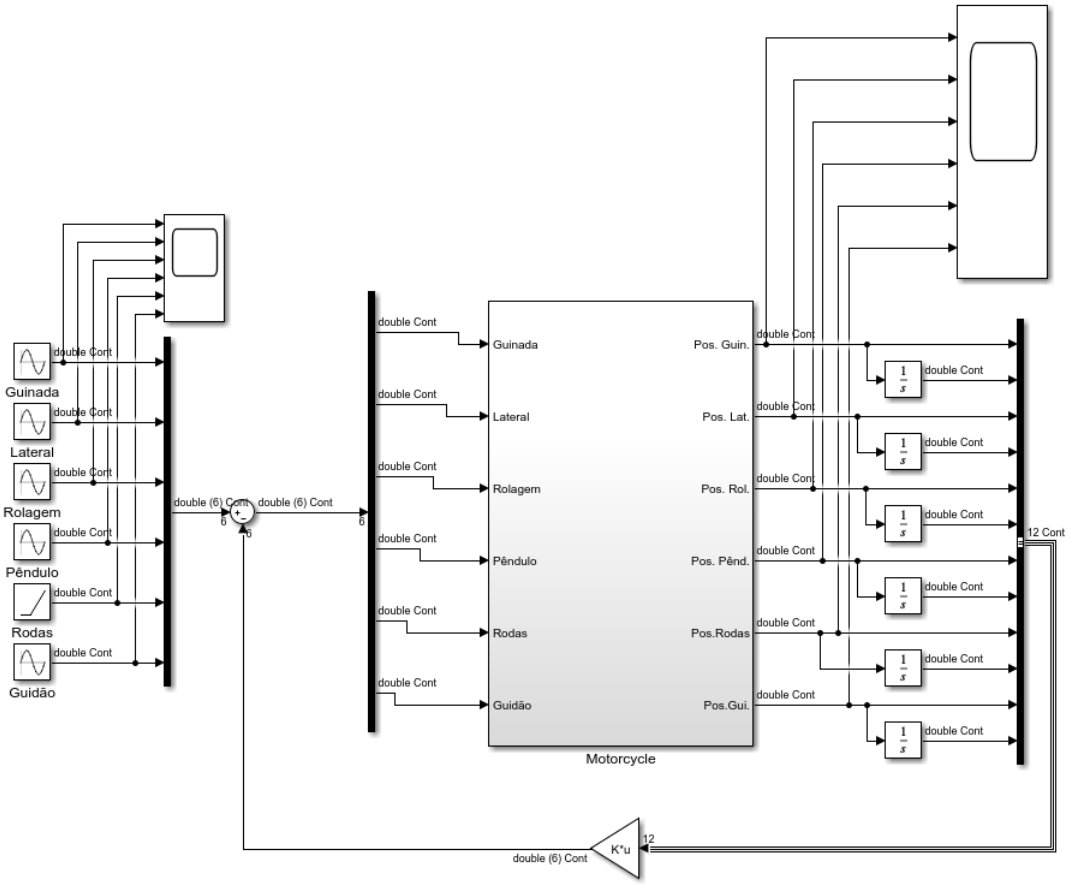

Figura 47: Diagrama de blocos do sistema de controle PD

A matriz K é a seguinte:

$$K = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}_{[6\times 12]}$$

Os resultados obtidos foram as da Figura 48.

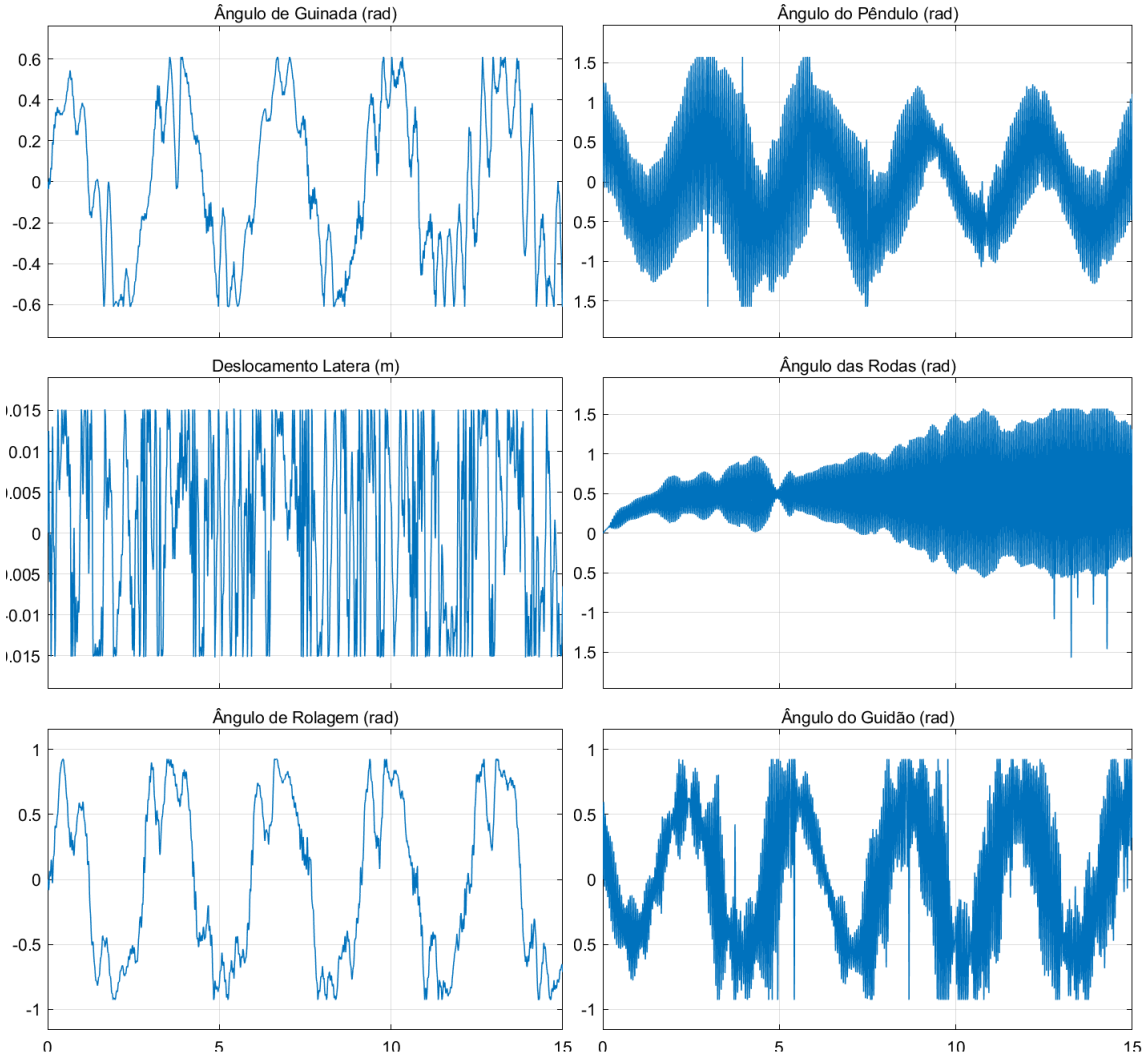

Figura 48: Resultados do sistema de controle PD

Desta vez o problema da oscilação está ainda pior que no controlador puramente proporcional, demonstrando a importância do ganho derivativo nesse sistema. A seguir foi testada uma combinação dos três.

d Controle proporcional integral derivativo

Por fim, foi testado um controlador PID também com os ganhos todos iguais a 1. O diagrama de blocos do sistema de controle PID pode ser visto na Figura 49. O controlador recebe as posições e velocidades, ou seja, as derivadas, e retorna os valores dos atuadores

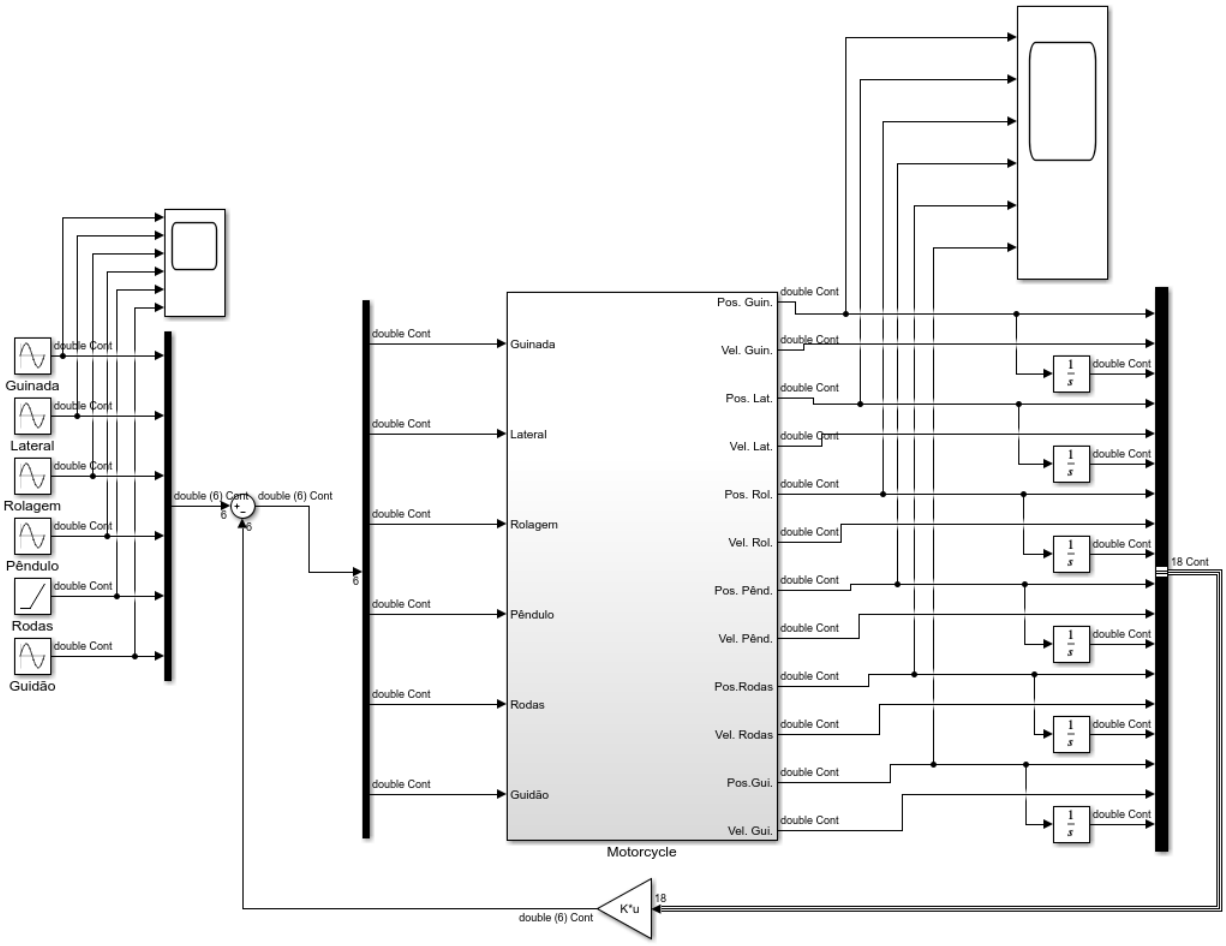

Figura 49: Diagrama de blocos do sistema de controle PID

A matriz K é a seguinte:

$$K = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}_{[6\times 18]}$$

Os resultados obtidos foram as da Figura 50.

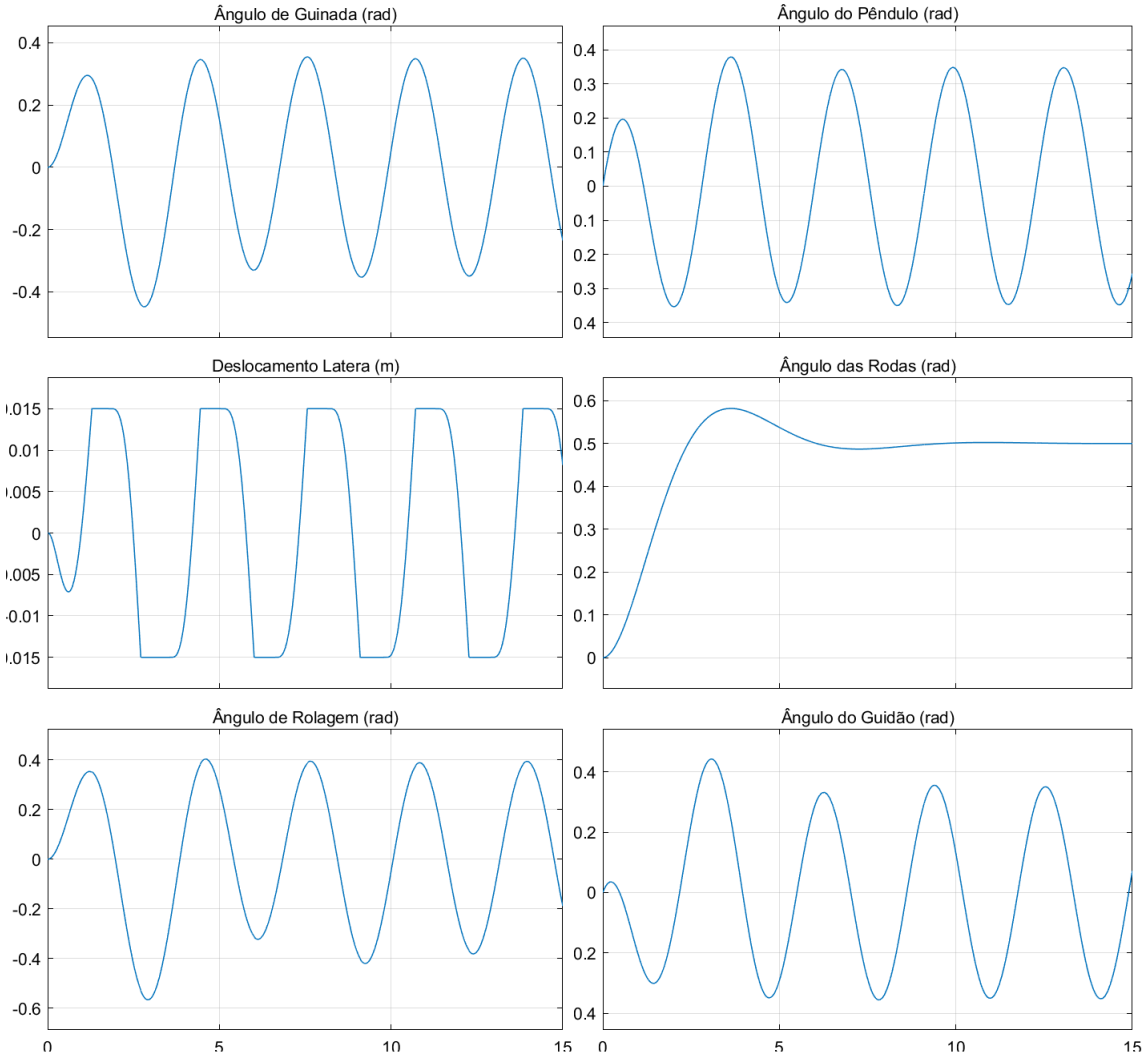

Figura 50: Resultados do sistema de controle PID

Os resultados foram praticamente idênticos ao controlador PD, portanto não é necessário um controlador robusto como o PID se o PD tem resultados igualmente satisfatórios.

5 Integração ao Arduino

Para reproduzir em motores e servomotores reais os movimentos obtidos em 4 através do Arduino, será preciso reproduzir os passos explicados em 2.c.2:

1. Configurar o *scope* que captura os movimentos para exportar seus dados para o *workspace*;
2. Escrever e executar um *script* que salva esses dados em um arquivo;
3. Implementar no Simulink um bloco que lê esse arquivo, fazer as devidas conversões, e aplicar os valores corretos nas entradas do blocos que controlam o Arduino.

a Configurações Iniciais

Primeiramente deve-se configurar o Simulink para trabalhar com o Arduino. Para isso é preciso instalar o *add-on* de suporte ao Arduino, basta clicar em *Add-Ons* na aba *Home* do MatLab, e na tela que abrir pesquisar por *Simulink Support Package for Arduino Hardware* e instalar esse pacote. Durante a instalação do pacote é necessário ter um Arduino em mãos e seguir as instruções de configuração na tela.

Quando tudo que for pedido estiver feito já se pode criar um novo modelo Simulink em branco. Deve-e então abrir as configurações do modelo e acessar a parte *Hardware Implementation*. No topo dessa seção há o campo *Hardware board*, se tudo deu certo com a instalação do pacote, deve aparecer toda a lista de placas de Arduino compatíveis com o Simulink, como na Figura 51

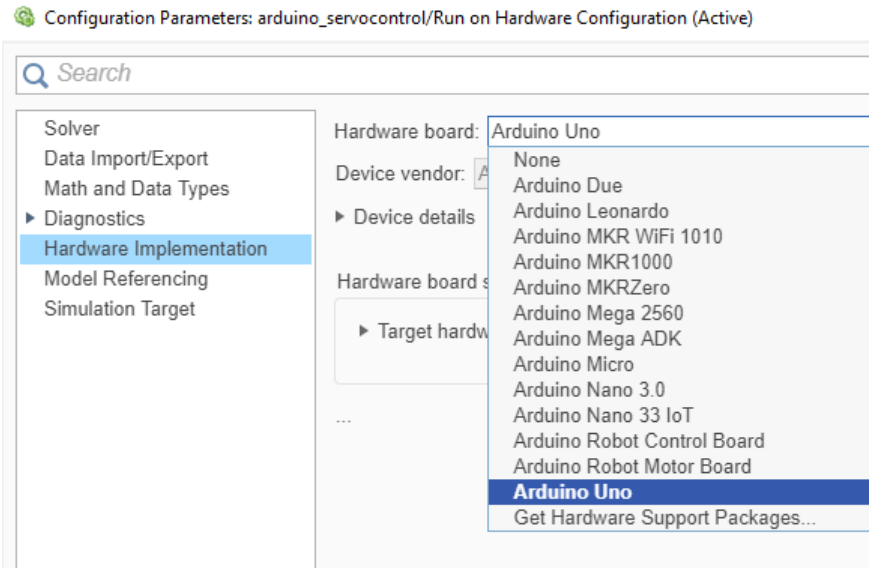

Figura 51: Lista de Arduinos compatíveis

Feito isso a interface já deve funcionar corretamente. Entretanto, há a possibilidade de ocorrer um erro frequente em que o Simulink não consegue encontrar o Arduino. Quando isso ocorre a solução é, na mesma área de configurações de hardware, ir até a seção *Target Hardware resources*, depois selecionar *Host-board connection*, aterar a opção do *Set host COM port* de *Automatic* para *Manually* e no campo abaixo digitar o número da porta em que o Arduino está conectado, como na Figura 52.

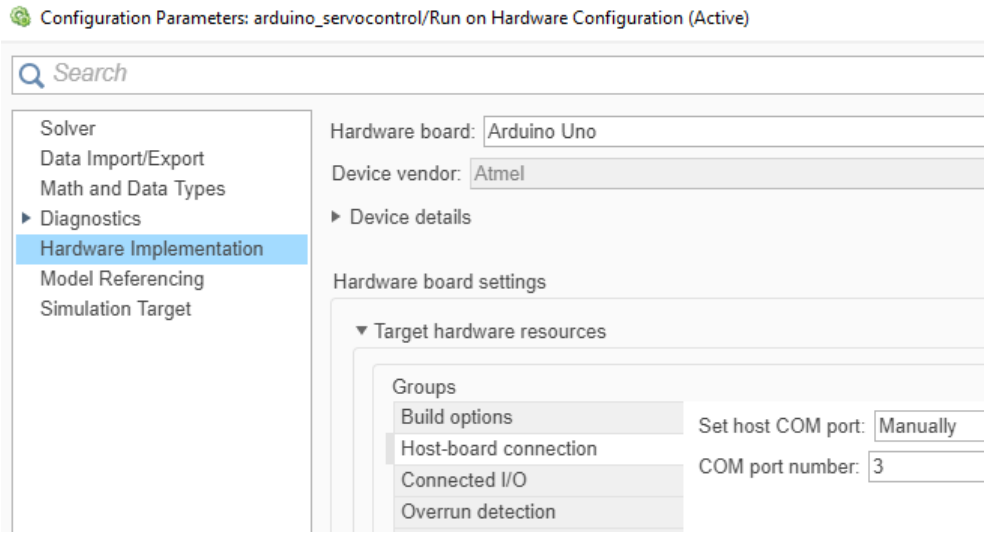

Figura 52: Solução do problema de não reconhecimento do arduino

Para saber o número correto da porta basta abrir a IDE do Arduino, com o Arduino conectado ao computador, abrir a aba *tools* e então olhar o tópico *Port*, como mostrado na Figura 53.

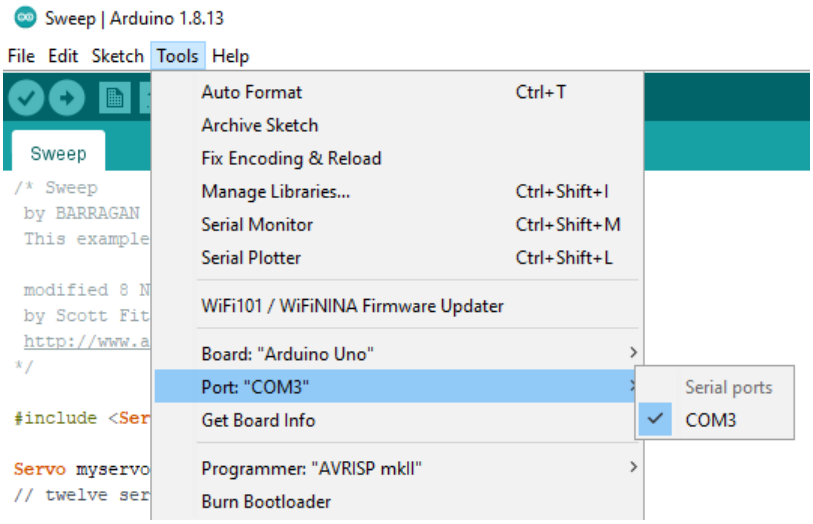

Figura 53: Onde encontrar o número da porta do Arduino

b Captura e armazenamento dos dados

A configuração que deve ser feita no *scope* é mostrada na Figura 54, deve-se selecionar a opção *Log data to workspace* e escolher o formato *Dataset.*

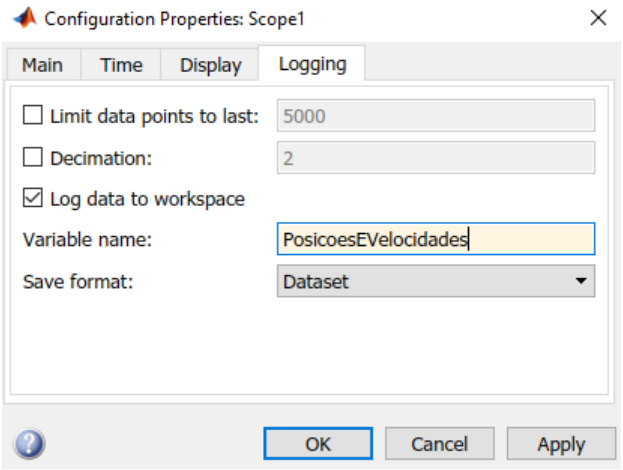

Figura 54: Configuração do *scope*

A seguir, basta executar o pequeno *script* de duas linhas mostrado na Figura 55, e os dados estarão salvos no arquivo *"posAndVel.mat"*.

```
posAndVel = evalin('base', 'PosicoesEVelocidades');
save('posAndVel.mat', 'posAndVel','-v7.3');
```

Figura 55: *Script* qu salva os dados num arquivo

c Leitura dos dados

O próximo passo é criar um novo modelo Simulink e inserir o bloco *Signal Editor*, citado em 2.c.2. Deve-se dar um clique duplo no bloco para abrir sua tela de parâmetros e na seção *Scenario* escolher o arquivo que foi gerado no passo anterior, como na Figura 56 e então clicar no ícone da direita da seção *Signal properties*, mostrado na Figura 57.

Figura 56: Seleção do arquivo de dados

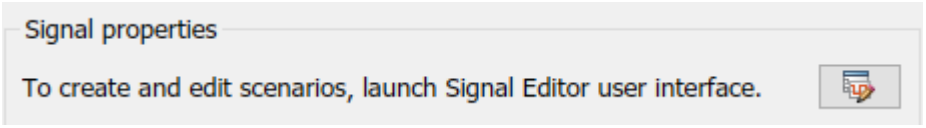

Figura 57: Indicação de onde clicar para editar o sinal

Ao clicar no ícone será aberta uma tela onde é possível editar os valores presentes no arquivo. Essa tela possui diversas informações e é possível modificar os sinais de várias maneiras, mas a única coisa que será útil para este cenário é poder nomear os sinais e escolher quais serão saídas do bloco.

Quando essa tela é aberta pela primeira vez aparecem, como é possível ver na Figura 58, todos os sinais que vieram da simulação, porém sem nome. Deve-se então atentar à ordem em que os sinais estão sendo passados para o *scope* onde eles foram originalmente capturados, pois é nessa ordem que eles aparecem nessa lista.

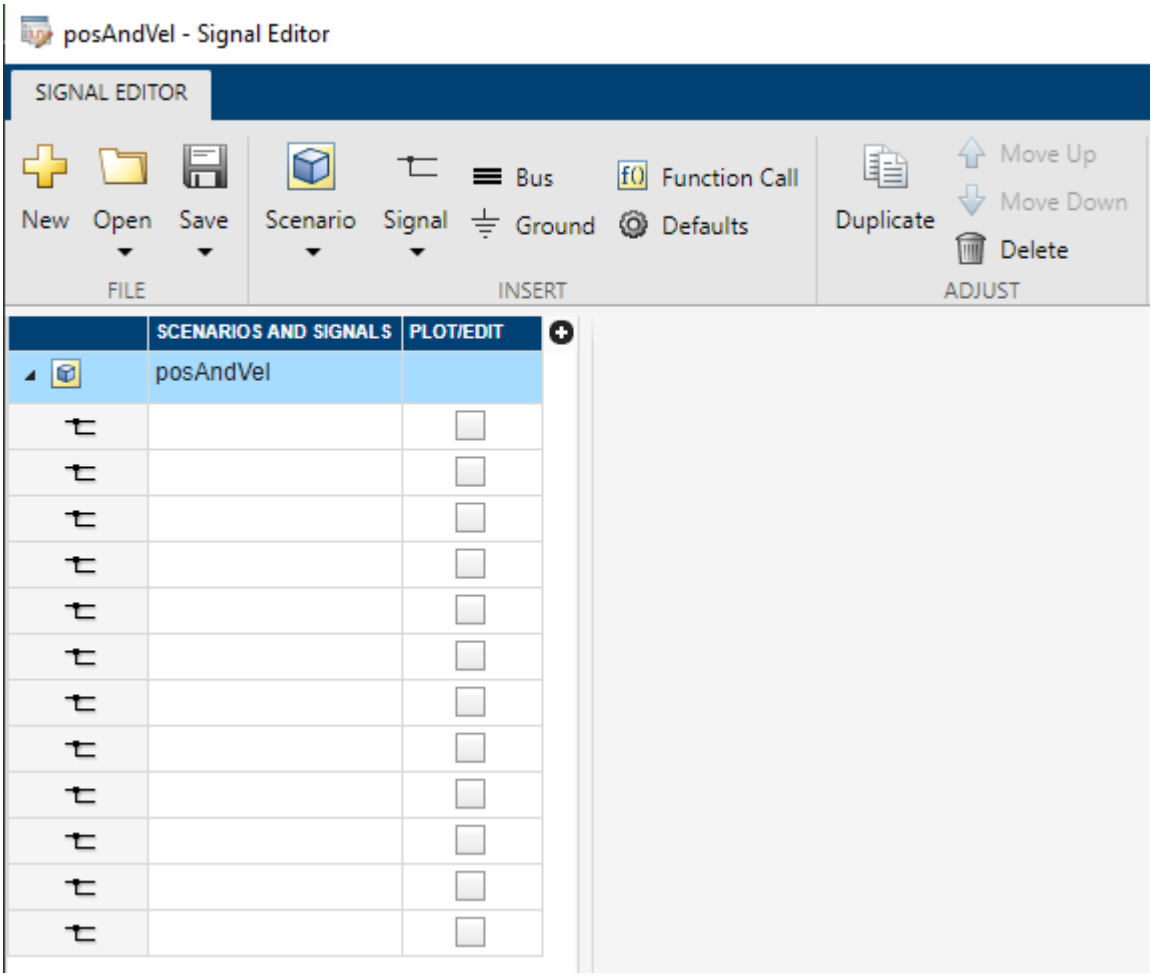

Figura 58: Os controles da edição de sinal e como os sinais aparecem inicialmente

Sabendo da ordem pode-se nomear os sinais e então selecionar apenas aqueles que serão de fato utilizados. Para isso, os sinais que não vão ser usados devem ser excluídos, clicando neles então em *Delete*, e os que serão utilizados devem estar com sua *checkbox* marcada na aba *PLOT/EDIT*, como na Figura 59.

	SCENARIOS AND SIGNALS	PLOT/EDIT
◢	posAndVel	
	Ângulo de Guinada	☑
	Velocidade Lateral	☑
	Ângulo de Rolagem	☑
	Ângulo do Pêndulo	☑
	Velocidade Angular das Rodas	☑
	Ângulo do Guidão	☑

Figura 59: Sinais nomeados e selecionados

Quando tudo estiver correto, deve-se clicar em *Save*, fechar essa tela e clicar em *OK* na tela dos parâmetros do *Signal Editor*. Feito tudo isso, tem-se um bloco como o da Figura 60, com os sinais gerados e pronto para serem convertidos e enviados aos blocos do Arduino.

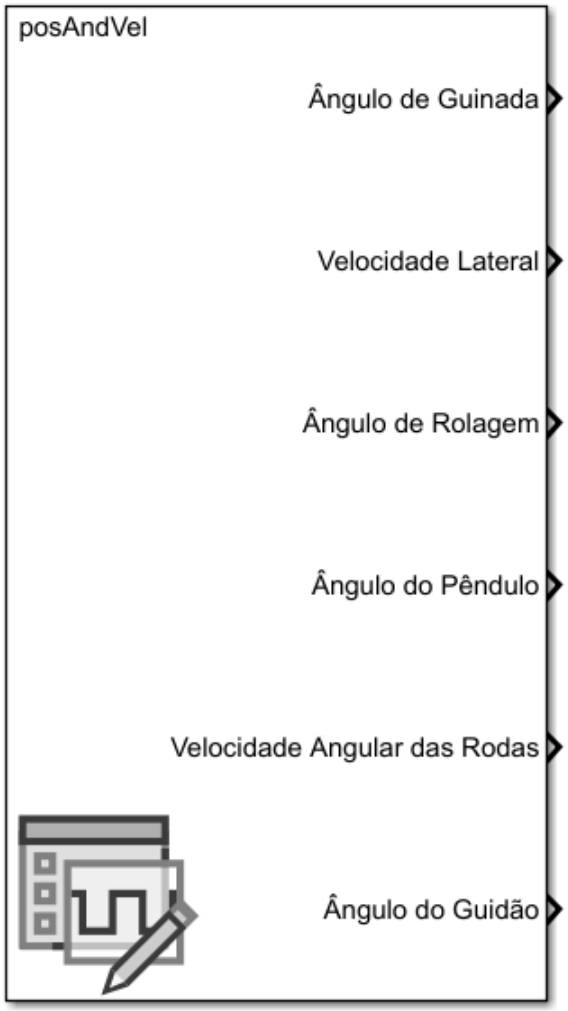

Figura 60: Estado final do bloco

1 Outras opções de aquisição dos dados

O MatLab e Simulink possuem diversas outras opções para se salvar dados e depois acessá-los, muitas até mesmo mais simples do que o método utilizado. Entretanto, a interface com o Arduino apresenta um número limitado de blocos que funcionam com a chamada geração de código.

O que essa interface faz na verdade é traduzir o modelo do Simulink para o código que o Arduino entende, então cada um dos blocos do Simulink é "traduzido"independentemente, e não exite tradução para 100% dos blocos existentes. O método do bloco *Signal Editor* foi usado pois ele é um dos blocos compatíveis com o Arduino. Felizmente, esse também é o método mais organizado e versátil, apesar de não ser o mais simples.

d Conversões

Quanto aos ângulos de guinada, rolagem, guidão e pêndulo, será feita a mesma conversão simples feita anteriormente. Já que o servo do Arduino trabalha entre 0º e 180º e os sinais destes ângulos estão entre -90º e 90º, basta somar 90 a cada um deles.

Já com os motores será diferente. Lembrando que a velocidade máxima de rotação do motor utilizado é de 13100 rpm e que o sinal que deve ser enviado ao bloco que controla o motor é 0 ou um número

positivo ou negativo entre 1 e 255, onde 255 se refere à velocidade máxima e o sinal diz o sentido de rotação.

Sabendo disso, para as rodas deve-se converter a velocidade vinda da simulação de rad/s para rpm e então fazer uma regra de três simples:

$$V_{rpm} = V_{rad/s} \times \frac{60}{2\pi} \qquad S_{rodas} = 225 \times \frac{V_{rpm}}{13100} = 225 \times \frac{V_{rad/s} \times \frac{60}{2\pi}}{13100}$$

$$S_{rodas} = \frac{135}{262\pi} \times V_{rad/s}$$

Já para o deslocamento lateral a conversão será semelhante à feita anteriormente para o carro do pêndulo duplo invertido: com um sistema de pinhão e cremalheira. Considerando um pinhão de 0.015 m de raio, sua circunferência é de 2×π×0,015 m = 0,094 m, portanto temos:

$$v_{max} = \frac{0,094}{60} \times 13100 = 20,58$$

Ou seja, uma entrada de 255 no bloco do motor equivale a uma rotação a 20,58 m/s no sentido positivo e uma entrada de -255 tem a mesma velocidade no outro sentido. Então para realizar a conversão da velocidade vinda da simulação para o número que será a entrada do bloco do motor basta multiplicá-la por 255 e dividi-la por 20,58.

Por fim deve-se dar um clique duplo em cada bloco do Arduino e designar onde que cada componente está conectado. Feito isso, interface com o Arduino fica como na Figura 61.

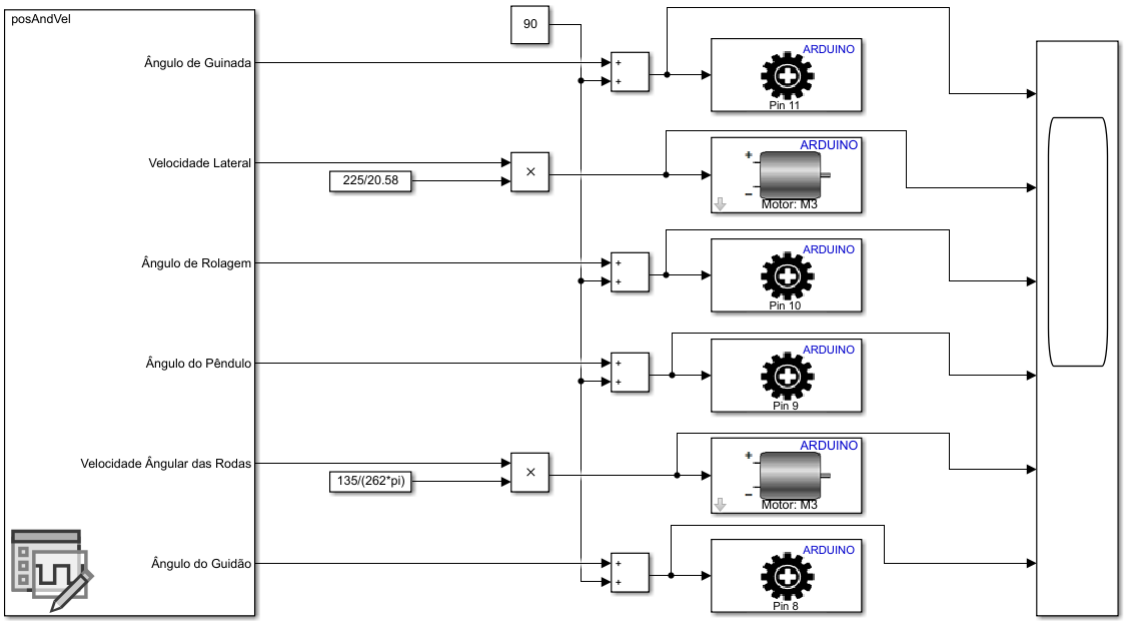

Figura 61: Estado final da interface Simulink/Arduino

e Montagem

Para enfim fazer tudo funcionar a montagem dos componentes deve estar feita, o Arduino deve estar conectado ao computador via cabo USB e então basta clicar no botão mostrado na Figura 62, que irá compilar, gerar o código do Arduino e enviá-lo à ele.

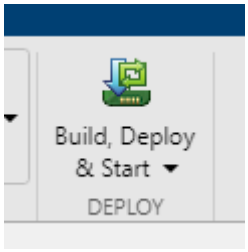

Figura 62: Botão que executa o programa

Depois que o código é enviado ele fica salvo na memoria do Arduino e sua execução fica se repetindo continuamente. Nesse momento é possível desconectar o Arduino no computador e conectar uma bateria de 9V através de um adaptador na entrada de energia da placa, como é possível ver na foto da Figura 63

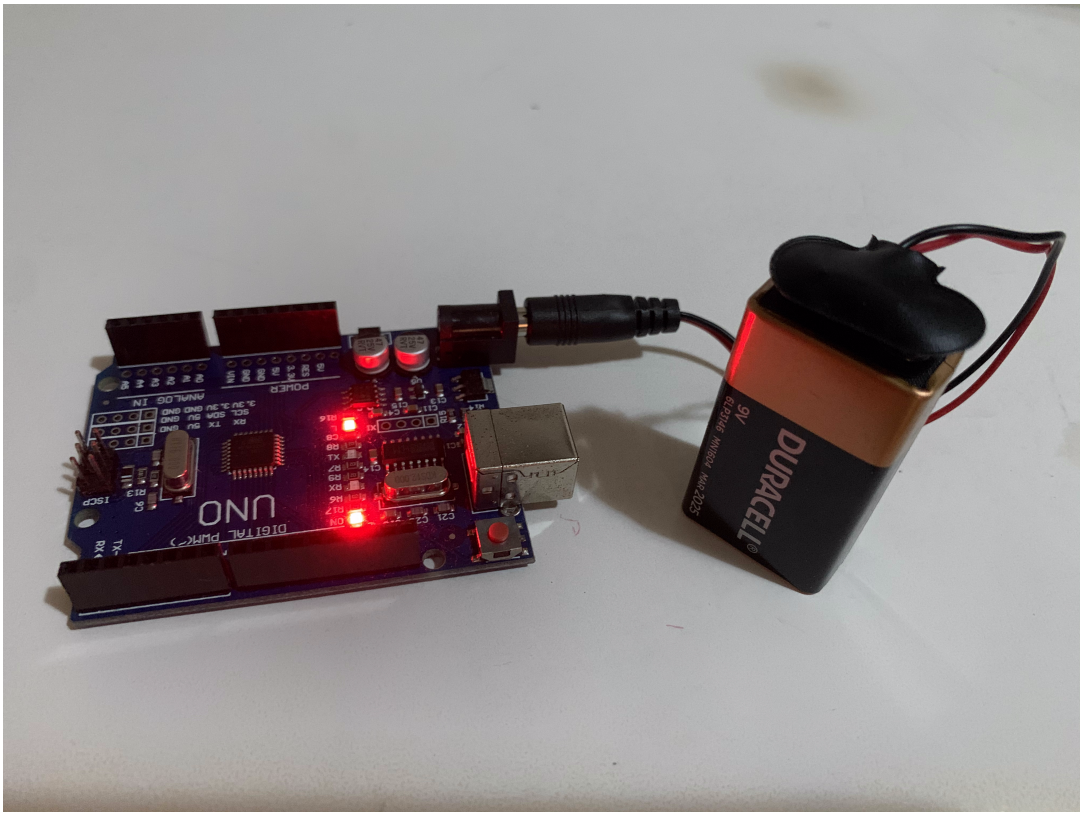

Figura 63: Demonstração do uso do adaptador de energia e da bateria

1 Motor DC *MKR Motor Carrier*

Os blocos de controle de motor DC do Simulink são parte de um pacote do Arduino que controla uma placa chamada *MKR Motor Carrier* [16]. Essa é a única forma de controlar a velocidade de um motor DC usando o Simulink. Exitem outras maneiras de se controlar se o motor gira ou não e o sentido da rotação, mas para controlar a velocidade só com o *MKR Motor Carrier*.

O *MKR Motor Carrier* é uma placa de apoio ao Arduino capaz de controlar diversos motores DC, motores de passo e servomotores simultaneamente, além de também ler as entradas de alguns sensores, ele é mostrado na Figura 64.

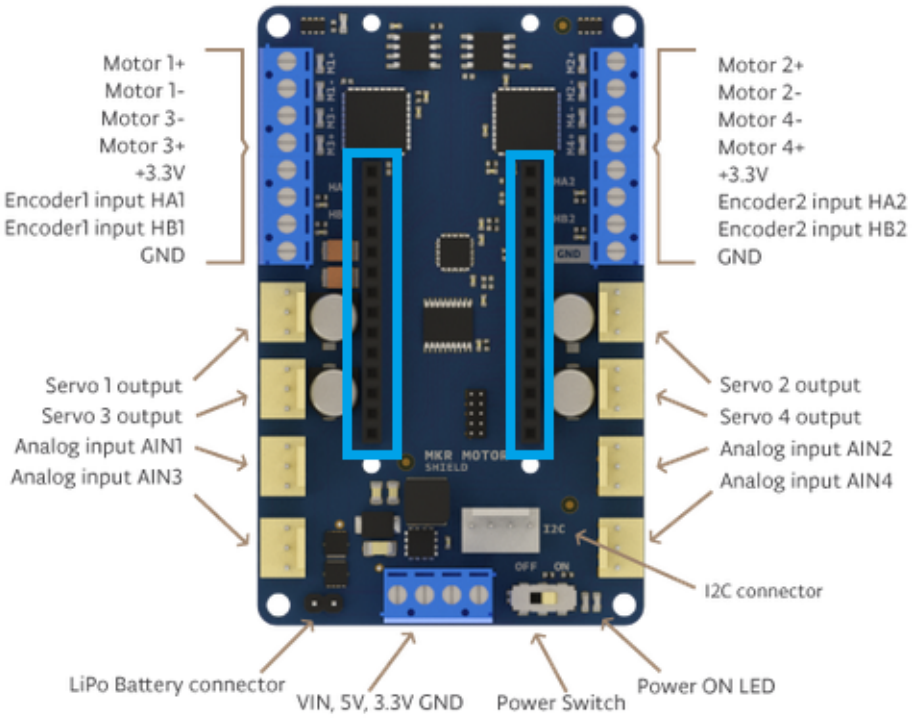

Figura 64: *MKR Motor Carrier*

Infelizmente, pela ausência dessa placa, não será possível testar a implementação dos motores DC. Mesmo que a placa estivesse presente, ela é compatível com uma quantidade limitada de Arduinos, mais especificamente os listados na Figura 65, e o Arduino UNO não está na lista.

- Arduino MKR FOX 1200
- Arduino MKR GSM 1400
- Arduino MKR NB 1500
- Arduino MKR VIDOR 4000
- Arduino MKR WAN 1300
- Arduino MKR WAN 1310
- Arduino MKR WiFi 1010
- Arduino MKR ZERO
- Arduino MKR1000 WIFI
- Arduino Nano 33 BLE
- Arduino Nano 33 IoT
- Arduino Zero
- Portenta H7

Figura 65: Arduinos compatíveis com o *MKR Motor Carrier*

Qualquer uma das placas listadas acima se conecta em cima do *MKR Motor Carrier*, nos pinos marcados em azul na Figura 64, como é possível ver na foto de exemplo da Figura 66, onde um Arduino Nano 33 BLE, presente na lista, está sendo aplicado no controle de um pequeno carro [17].

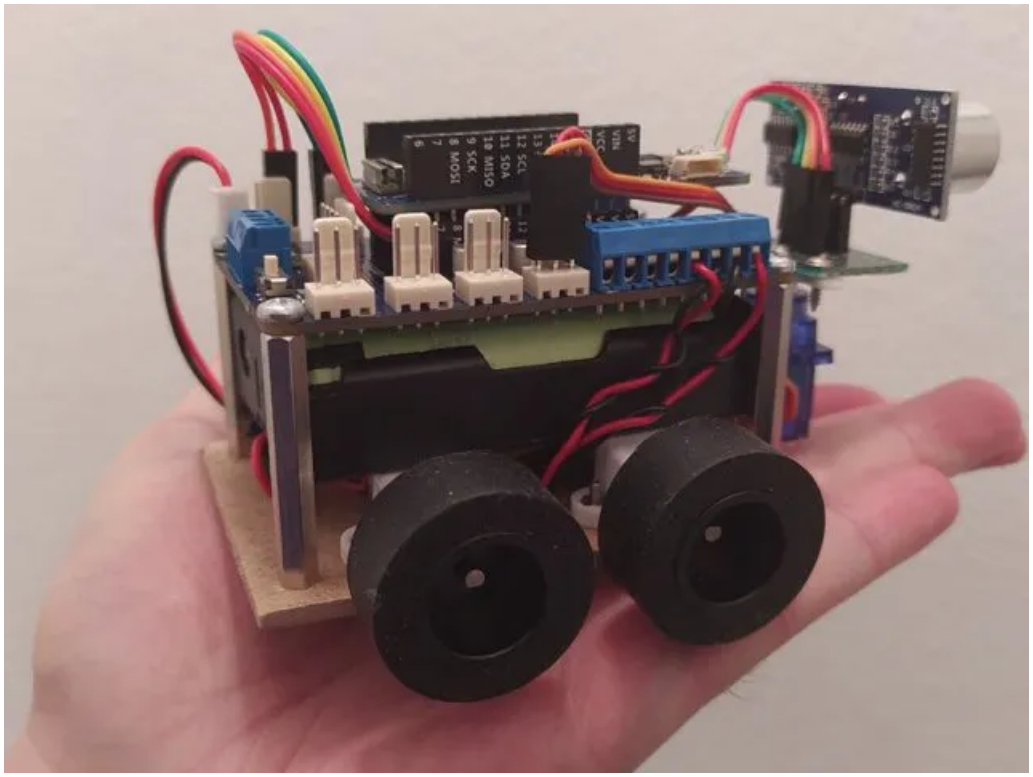

Figura 66: Exemplo de um Arduino Nano 33 BLE montado em um *MKR Motor Carrier*

É possível também controlar todos os servos, além dos dois motores usando essa configuração, e isso seria o ideal. Para controlar também os servos, a única diferença no modelo do Simulink seria trocar o bloco do servo *standard* pelo bloco do*MKR Motor Carrier*, como na Figura 67, já que eles esperam receber o exato mesmo tipo de sinal. Deve-se atentar também ao número das portas.

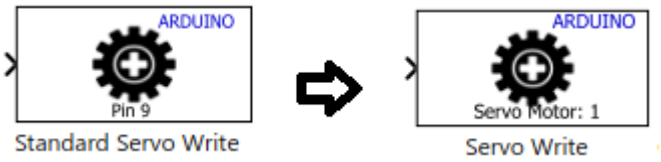

Figura 67: Qual bloco usar para controlar os servos com o *MKR Motor Carrier*

A montagem dos motores e dos servos, ou seja, de todo o modelo usando o *MKR Motor Carrier* é a mostrada na Figura 68, porém com um dos Arduinos da lista encaixados em cima dele e devidamente alimentado, seja por bateria ou via cabo USB. A partir daí, a integração com o Simulink deverá funcionar integralmente.

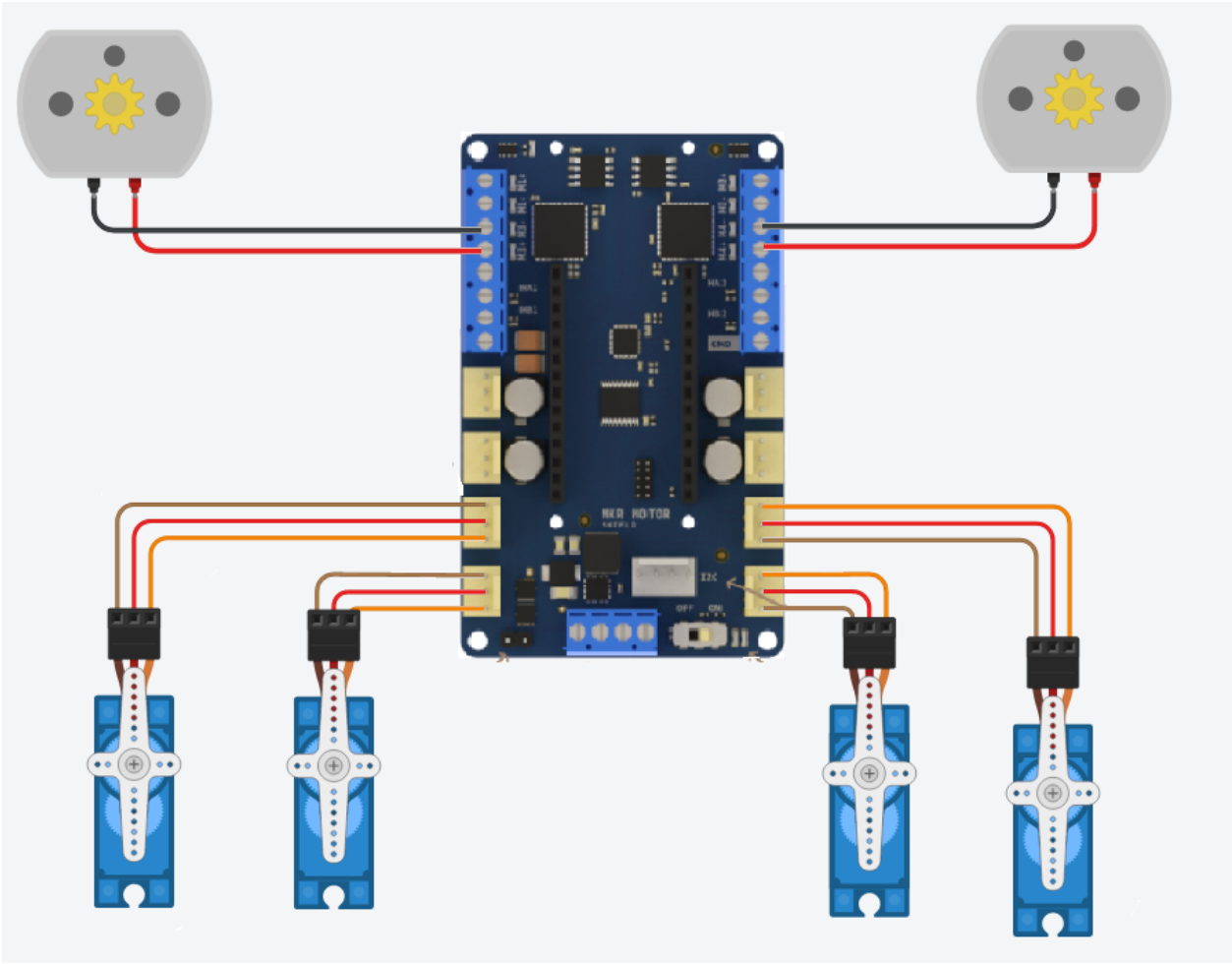

Figura 68: Montagem dos motores e servos no *MKR Motor Carrier*

2 Montagem dos servos sem o *MKR Motor Carrier* e resultados

Apesar de não ser possível testar os motores DC, os servomotores foram testados. A Figura 69 mostra uma ilustração feita no Tinkercad [18], software web da Autodesk que permite simular circuitos simples, que demonstra de maneira legível como os servomotores foram montados no Arduino real. É possível acessar esse modelo no link: *https://cutt.ly/thCA7aL* (Acessado em Dezembro de 2020).

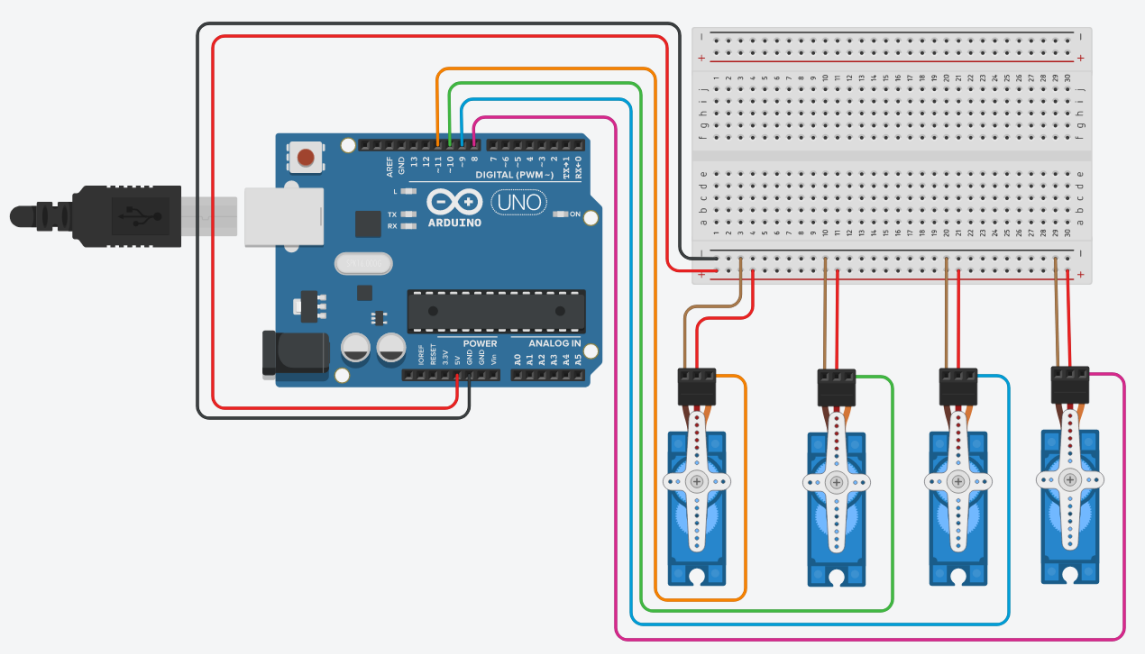

Figura 69: Ilustração da montagem dos servomotores no Arduino real

Devido ao problema de memória do Arduino UNO mencionado anteriormente, os servos foram testados separadamente e com dados vindos de uma simulação igual à mostrada antes, porém com 3 segundos de duração ao invés de 60 segundos e também com a tolerância do passo da simulação bem maior, aumentada de 0,0001 para 0,1.

Mesmo com esse obstáculo, cada um dos movimentos foi testado e funcionou como esperado.

6 Conclusão

Pode-se dizer que, apesar da falta de alguns testes físicos, o objetivo central do projeto foi alcançado: desenvolver um método capaz de transferir movimentos simulados no MatLab/Simulink/Simscape para componentes reais, através de uma interface com o microcontrolador Arduino.

Futuramente seria interessante que os componentes fossem montados num modelo real da motocileta e aparato e, já com todos os componentes funcionando, fosse adicionado um IMU (*inertial measurement unit*), para ter um *feedback* do movimento ocorrido para verificar se ele é condizente com os movimentos desejados. Isso exigiria uma modelagem mais rigorosa da motocicleta e do aparato.

Uma conclusão concreta é que o Arduino UNO não é cabível a essa aplicação, tanto pelo fato de ter pouca memória, quanto por não ser compatível com *MKR Motor Carrier*, necessário para o controle de velocidade dos motores. Cada um dos modelos compatíveis e suas respectivas memórias está listado na Tabela 1. Para fins de comparação, o Arduino UNO possui uma memória total de 34 Kb.

modelo	memória total
Arduino MKR FOX 1200	300 Kb
Arduino MKR GSM 1400	300 Kb
Arduino MKR NB 1500	300 Kb
Arduino MKR VIDOR 4000	300 Kb
Arduino MKR WAN 1300	300 Kb
Arduino MKR WAN 1310	300 Kb
Arduino MKR WiFi 1010	300 Kb
Arduino MKR ZERO	300 Kb
Arduino Nano 33 BLE	1.2 Mb
Arduino Nano 33 IoT	300 Kb
Arduino Zero	300 Kb
Portenta H7	1.3 Mb

Tabela 1: Memória de cada um dos Arduinos compatíveis com o *MKR Motor Carrier*

Os proximos passos seriam obter um *MKR Motor Carrier*, e uma das placas compatíveis com ele e então testar as implementações. Recomenda-se a obtenção de um Arduino Nano 33 BLE ou um Portenta H7, já que possuem memória mais de 3 vezes maior que a das outras opções e quase 40 vezes maior que a do Arduino UNO.

Entre as duas melhores opções citadas a mais recomendada é o Arduino Nano 33 BLE (mostrado na Figura 66) por se adequar melhor à aplicação, pois ele vem com um IMU embutido, Bluetooth integrado e baixo consumo de energia tornando ele ideal e possibilitando implemenações de controle remoto. Já o Portenta H7 é mais indicado para aplicações de inteligencia artificial e é consideravelmente mais caro (5 vezes mais). Sua única vantagem é a memória, mas a diferença entre ele e o Nano 33 BLE é bem pequena e não compensa a diferença de preço, além de não ter o IMU embutido, que seria extremamente importante na evolução do projeto.

7 Referências

[1] E. Oliveira, "Como usar com arduino – micro servo motor sg90 9g," Acessado em Novembro de 2020. [Online]. Available: https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-micro-servo-motor-sg90-9g/

[2] F. Flop, "Motor dc 3v," Acessado em Novembro de 2020. [Online]. Available: https://uploads.filipeflop.com/2017/09/7MS73-1.jpg

[3] W. Harris, "How motorcycles work," *HowStuffWorks.com*, 2005, Acessado em Outubro de 2020. [Online]. Available: https://auto.howstuffworks.com/motorcycle.htm

[4] Senado, "Estudos e dados estatísticos apontam aumento do número de vítimas fatais de acidentes com motos no trânsito, mas risco de morte sobre duas rodas é menor para motociclistas profissionais," Acessado em Outubro de 2020. [Online]. Available: https://cutt.ly/ZhCAMjy

[5] MathWorks, "Matlab," Acessado em Outubro de 2020. [Online]. Available: https://www.mathworks.com/products/matlab.html

[6] ——, "Simulink," Acessado em Outubro de 2020. [Online]. Available: https://www.mathworks.com/products/simulink.html

[7] Atmel, "Arduino," Acessado em Outubro de 2020. [Online]. Available: https://www.arduino.cc/

[8] MathWorks, "Simscape," Acessado em Outubro de 2020. [Online]. Available: https://www.mathworks.com/products/simscape.html

[9] D. Systèmes, "Solid works," Acessado em Outubro de 2020. [Online]. Available: https://www.solidworks.com/

[10] Autodesk, "Fusion 360," Acessado em Outubro de 2020. [Online]. Available: https://www.autodesk.com/products/fusion-360/overview

[11] MathWorks, "Arduino support from simulink," Acessado em Novembro de 2020. [Online]. Available: https://www.mathworks.com/hardware-support/arduino-simulink.html

[12] Atmel, "Arduino uno rev 3," Acessado em Novembro de 2020. [Online]. Available: https://store.arduino.cc/usa/arduino-uno-rev3

[13] Arduino, "Servo," Acessado em Novembro de 2020. [Online]. Available: https://www.arduino.cc/reference/en/libraries/servo/

[14] Rapid, "3v dc motor datasheet," Acessado em Novembro de 2020. [Online]. Available: https://img.filipeflop.com/files/download/Datasheet-MotorDC-3V-37-0140.pdf

[15] MathWorks, "Inverted double pendulum on a sliding cart," Acessado em Novembro de 2020. [Online]. Available: https://www.mathworks.com/help/physmod/sm/ug/inverted-double-pendulum-on-a-sliding-cart.html

[16] Arduino, "Mkr motor carrier," Acessado em Dezembro de 2020. [Online]. Available: https://store.arduino.cc/usa/mkr-motor-carrier

[17] Avilmaru, "Mini 4wd arduino robot controlled by bluetooth," *Hackster.io*, 2020, Acessado em Dezembro de 2020. [Online]. Available: https://www.hackster.io/Avilmaru/mini-4wd-arduino-robot-controlled-by-bluetooth-d2512f

[18] Autodesk, "Tinkercad," Acessado em Dezembro de 2020. [Online]. Available: https://www.tinkercad.com/