Relatório Parcial PIBIT

1 Identificação

Título do projeto: Robô Humanoide.
Título do sub-projeto: Eletrônica embarcada para Robô Humanoide.
Bolsista: Ana Luiza Buse da Silva.
Orientador: Paulo Fernando Ferreira Rosa, PhD, rpaulo@ime.eb.br.
Data de inicio do bolsista no projeto: 14 de junho de 2019.

2 Introdução

O ser humano levou cerca de sete milhões de anos para conseguir andar. Atualmente, uma criança aprende a fazer o mesmo em um ano. Esse aprendizado é baseado na tentativa e erro até obter o sucesso. No campo da robótica, o desenvolvimento de algorítimos e métodos de caminhada para robôs bípedes começou no final da década de 1970, e ainda se procura um modelo definitivo de locomoção robótica que seja dinâmico, robusto, eficiente e com menor consumo de energia.

Este projeto consiste em desenvolver o robô humanoide da equipe humanoide da RoboIME, que tem como finalidade participar do *Humanoid Robot Racing* 2019. A equipe RoboIME é um pequeno grupo do IME (Instituto Militar de Engenharia) que participa do LARC (Concurso Latino-Americano de Robótica) desde 2010 e conseguiu alcançar muitos bons resultados nas categorias em que ingressou.

Este sub-projeto foca na confecção de um sistema eletrônico a ser embarcado em um robô bípede, que deve ser capaz de correr sem interferência humana. Para isso, serão necessários estudos acerca de motores, microprocessadores, tipos de comunicação, criação de bibliotecas e *firmware* em python, desenvolvimento de placas e sistemas.

3 Objetivos

O objetivo geral desse projeto é o desenvolvimento prático do sistema embarcado para um robô humanoide, visando competir na categoria *Humanoid Robot Racing* (HRR). O principal proposito da competição HRR é motivar as pesquisas para desenvolver e melhorar os movimentos de andar e correr de um robô bípede, a fim de desenvolver um melhor robô humanoide para interações humanas no futuro.

4 Cronograma

Para facilitar a compreensão do projeto, optou-se por dividir o projeto em módulos que seguirão o cronograma abaixo:

- **Tarefa 1:** Estudo do funcionamento do motor e seu acionamento pela Raspberry usando linguagem python;

- **Tarefa 2:** Controle de vários motores com a Raspberry, desenvolvimento da placa e participação na competição;
- **Tarefa 3:** Obtenção de *feedback* dos motores;
- **Tarefa 4:** Estudo dos sensores e obtenção dos retornos destes;
- **Tarefa 5:** Testes com a interpretação dos *feedbacks* dos motores e dos sensores;
- **Tarefa 6:** Testes do robô e elaboração do Relatório Final;

Tarefas	Ago	Set	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago
1	X	X											
2			X	X									
3					X	X							
4							X	X					
5									X	X	X		
6											X	X	

Tabela 1: Cronograma.

Este cronograma foi um pouco modificado do cronograma apresentado na proposta de bolsa, porque notou-se que começar os estudos usando o Arduíno para controle dos motores não seria útil para este projeto, pois seria necessário o uso de um microprocessador, como descrito na seção Atividades realizadas no período.

5 Atividades realizadas no período

Primeiramente, foi feito um estudo para decidir se usar o Arduíno seria útil para o proposito final. Como o robô deveria estar pronto em apenas 3 meses para participar da competição, optou-se por otimizar o tempo dividindo o projeto em três partes, que serão descritas a seguir.

A primeira fase destinou-se a estudar os tipos motores. Depois, pesquisou-se sobre a comunicação entre o AX-12A, motor escolhido para esse projeto e a Raspberry, que é o microprocessador escolhido para esse projeto.

Na segunda fase, foi feito um estudo detalhado do *datasheet* do motor, atentando-se para a forma em que os comandos devem ser enviados para ele. Para isso, foi desenvolvido uma biblioteca em python capaz de enviar esses comandos de forma automática de acordo com o angulo e o tempo de movimento requeridos. Isso foi testado usando uma *protoboard*.

Por fim, foi feita uma placa a ser embarcada no robô já pronto, e a partir disso foram feitos testes e ajustes. O robô foi levado para a *Latin American Robotics Competition*, onde mesmo competindo com outros robôs de plataformas pré-prontas, obteve resultados notórios.

5.1 Microprocessadores

São circuitos integrados capazes de executar funções de cálculo e tomada de decisão. Um microprocessador coloca as funções de uma unidade central de computador em um único circuito integrado, sendo um cérebro eletrônico em um chip. Ele pode processa dados de entrada de acordo com o que foi programado, e fornece resultados como saída. (1)

5.1.1 Raspberry

É um computador barato, portátil e versátil, usado principalmente em projetos de programação, robótica e em iniciativas em geral com software e hardware livre. Foi decidido usar a Raspberry Pi 3, pois ela tem algumas características (2) que serão de grande importância, como descrito a seguir.

- WiFi integrado 802.11n: A comunicação deve ser feita sem o uso de cabos,pois o robô deve estar livre para andar sem nenhum tipo de apoio físico do mundo externo. A comunicação foi feita toda com wifi.
- 40 pinos de GPIO: O robô precisa de varias portas de comunicação, pois necessita do uso de vários motores e sensores.
- Interface CSI (câmera): Uma parte muito importante para a correção da caminhada ou o percepção de obstáculos é através da visão computacional.
- Processador 1.2GHz 64-bit quad-core ARMv8 CPU e 1GB RAM : Será necessário que várias coisas sejam processadas ao mesmo tempo, como o algorítimo de caminhada, a visão computacional e a tomada de decisões.

O Arduíno é um microcontrolador, um computador simples que roda um programa por vez. O Raspberry Pi é um computador de uso geral, com sistema operacional e com a possibilidade de rodar vários programas de uma vez (3). Como já mencionado, será necessário processar várias informações ao mesmo tempo, por isso não seria útil para esse projeto usar o Arduíno, pois até a linguagem de programação usada seria diferente da escolhida para a Raspberry. Assim, foi decidido começar o desenvolvimento usando a Raspberry.

5.2 Tipos de motor

No mercado há vários tipos de motores, os mais comuns são motores de corrente contínua, motores de passo e servomotores. Apesar de serem motores elétricos, cada um tem o funcionamento diferente e finalidades diferentes, conforme descrito abaixo.

1. Motores de corrente continua:
   É um motor alimentado por corrente contínua, pode possuir escovas (escovado) ou não (*brushless*). Além disso, e sua velocidade pode ser controlada apenas variando a sua tensão.(4) Podem fornecer bastante velocidade, mas não muita precisão, o que não será interessante para o humanoide, que necessita de muita precisão para manter o equilíbrio.

2. Motores de passo:
   Possuem controle para executar movimentos de rotação muito precisos,já que permitem posicionar o seu eixo muito precisamente em qualquer posição, pois a sua rotação é dividida em vários passos, ou ângulos.O número de passos determina a precisão de ângulo de rotação do motor de passo. Assim, esse motor torna-se muito útil quando precisa-se de movimentos muito precisos, sendo que ainda consegue-se regular a velocidade e o torque do motor. (5)

3. Servomotores:
   São destinados para uso em aplicações de controle de movimento que exigem posicionamento de alta precisão, reversão rápida e desempenho excepcional. Sendo assim, eles são amplamente utilizados na robótica. Sua principal diferença é que possuem incorporado neles um Encoder, que tem a função de fornecer o *feedback* de velocidade e posição, a partir da qual controlam a velocidade e a posição final do motor. (6)

Portanto eles são bem diferentes, com suas próprias características. O motor de corrente contínua serve para aplicações de velocidade e sem precisão. O motor de passo serve para precisão e força. E o servomotor serve para precisão entre 0 a 180 graus e sem força.

5.2.1 AX12-A

Figura 1: Motor Ax12-A. (7)

Para as juntas do robô humanoide, foi decidido usar os servos Dynamixel AX-12A, que são a versão servos básica da empresa coreana Robotis inc. Algumas características diferenciais desses servos para o seu uso no robô:

- Relata posição, velocidade, carga, tensão e temperatura: o *feedback* da posição será de grande importância para a correção durante a caminhada para evitar o tombamento do robô.

- Posição angular de 300 em incrementos de 1024: Resulta uma resolução de 0.29, o que é uma alta precisão.

- Indicador de status / alarme LED embutido: permite visualizar se aconteceu algum erro durante a caminhada.

- Desligamento automático para proteção quando em tensão carga máxima ou mínima ou temperatura limite: sistema de proteção que impede danos ao motor durante a caminhada.
- Todo o gerenciamento do sensor e controle de posição são controlados pelo microcontrolador integrado do servo (ATMega 8)

Eles se conectam a um barramento serial e cada um tem um número de identificação, sendo controlados por um link serial de 1Mbaud e podem ser conectados entre si para criar cadeias de servomotores (Daisy-Chain). Até 254 servomotores podem ser conectados em série ao Dual-POB com um único cabo, como na figura 2 (8).

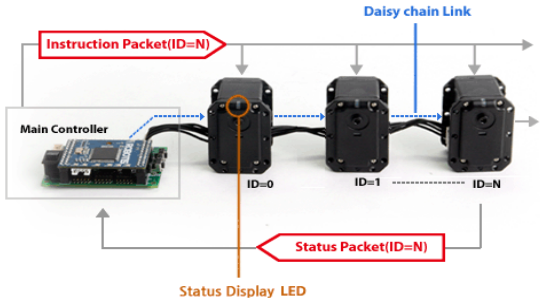

Figura 2: Cadeia de Servos AX12-A (7)

5.3 Comunicação

Ao contrário de outros servos, o Dynamixel não responde aos sinais PWM, mas um protocolo de instruções mais complicado para ler e gravar em sua memória. Essa comunicação ocorre através de uma porta UART half-duplex, usando apenas um fio para envio e recebimento. Então,é preciso construir um circuito, como na figura 3, que converte full-duplex em half-duplex, para usar um microcontrolador com uma interface serial full-duplex, como a Raspberry pi, para controlar esses motores.

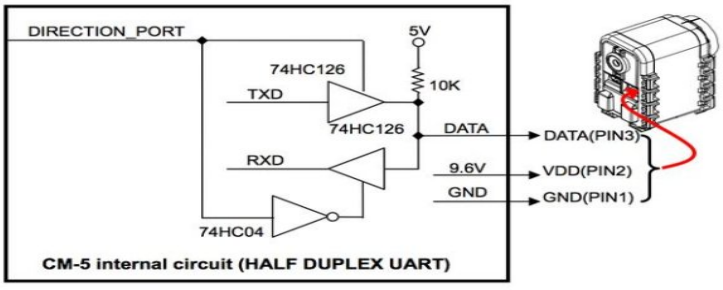

Figura 3: Circuito para conversão de sinal. (9)

O circuito é um buffer de três estados para garantir que, quando o controlador estiver transmitindo, o barramento não esteja conectado ao pino Rx e que, quando estiver esperando receber, não seja acionado pelo pino Tx.

Foi usado o buffer de três estados 74LS24, porque ele já possui a capacidade interna de habilitar metade de seus buffers com sinais altos e a outra metade com sinal baixo.Para isso, conforme a referência (10) ele foi conectado a Raspberry da seguinte forma:

- Pino 2 ao Pino 3 (dados enviados para o AX-12)
- Pino 1 ao pino 19 (conectado à porta RPi 12 (GPIO 18))
- Pino 18 (conectado ao RPi pino 10 RX)
- Pino 17 (conectado ao pino RPi 8 TX)
- Pino 10 ao terra (pino RPi 6 GND)
- Pino 20 a Vcc (5 Volts, pino RPi 4 5V)

A figura 4 mostra os pinos do 74LS24.

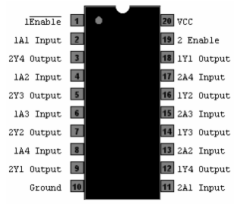

Figura 4: Pinos do 74LS24. (10)

5.4 Bateria

A bateria comprada para uso no robô é o TURNIGY 5000mAh 3S 20C Lipo Pack com XT-60. É equipado com condutores de descarga reforçados para minimizar a resistência e sustentar cargas de alta corrente, conectores XT-60 e conectores de balanceamento estilo JST-XH.

Figura 5: Bateria.

As especificações da bateria estão disponíveis na tabela 2:

ITEM	Especificações
Idade recomendada	12 anos ou mais
Peso	432 g
Descarga constante	20C
Pico de descarga	30C
Capacidade	5000mAh
Configuração	3S1P \ 11.1 V \ 3 Cell

Tabela 2: Especificações da bateria.

6 Resultados alcançados no período

Os ganhos finais obtidos foram a placa, o firmware e os resultados das corridas.

6.1 Montagem da placa

Foi montado um circuito em uma *protoboard*, de forma a conectar a bateria com a Raspberry e com os motores, conforme o esquemático da figura 6.

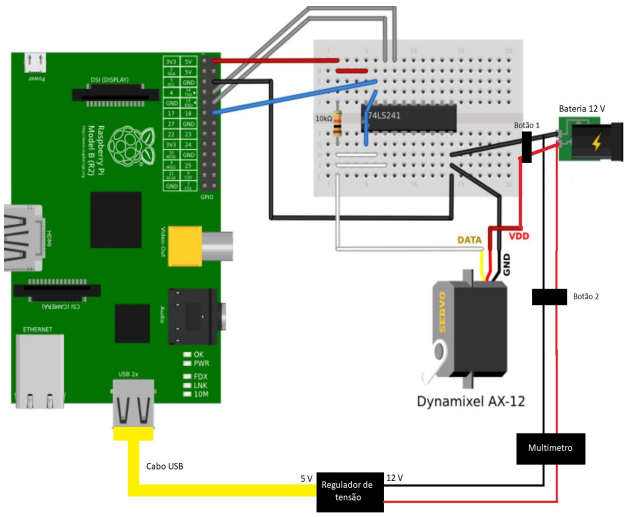

Figura 6: Esquemático da placa.

Tem-se algumas observações a serem feitas:

1. A bateria usada é de 12V, enquanto a Raspberry é de 5V, por isso precisa-se de um regulador de tensão que tem como entrada uma porta USB.
2. Foi colocado um multímetro display digital para o controle do nível de bateria, que não pode ficar abaixo de 9,6V.

3. É usado 2 botões para ligar/desligar: um para os motores e outro para a Raspberry.

4. Foram feitas 4 entradas de motor ligados diretamente na bateria, uma para cada membro.

A figura 7 é uma foto da placa final, que apresenta os componentes soldados em uma placa de fenolite.

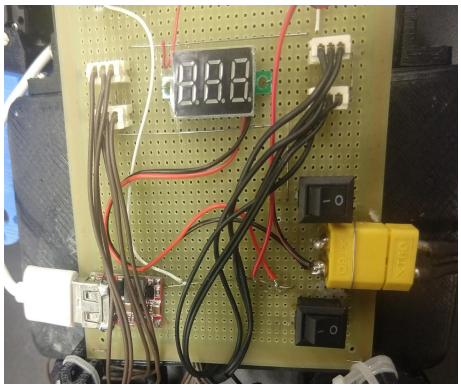

Figura 7: Foto da placa desenvolvida.

6.2 Desenvolvimento da biblioteca

Como os motores se comunicam com comandos de forma bem especificas, foi necessário desenvolver uma biblioteca e python para automatizar algumas ações, como mandar um motor se movimentar e depois mandar vários motores se movimentarem ao mesmo tempo, explorando o fato de que eles podem ser conectados em serie.
Um passo do robô será definido pela inteligência do robô, usando algorítimos de inteligência artificial, será colocado em um arquivo csv e deverá ser lido pela eletrônica, interpretando e mandando os movimentos para cada motor.Cada linha desse arquivo será composta pelo numero do id, seguido do angulo e da velocidades. O firmware deverá ser baseado em ler esse arquivo, salvar em uma matriz, que deverá ser lido linha por linha, e mandado para os motores, como demonstrado no código abaixo:

```
    f=45 #f     a frequencia do arquivo
    entrada = csv.reader(open("arquivo.csv", "r"))
    for linha in entrada:
        matriz.append(linha)
    x= len(matriz)
    m = int(matriz[0][0])
    for i in range (0,x):  #transforma os valores string em float
        for j in range (0,3):
            matriz[i][j]=float(matriz[i][j])
```

```
    while 1:
        print('Come ou! o passo')
        delta=0
        t1=time.time()
        for j in range(0,x-1):
            if j==0:
                tinicio=time.time()
            t.move(matriz[j])
            if j%9==0:
                tantes=tinicio
                tagora=time.time()
                delta=tagora-tantes
                if delta<(1/f):
                    time.sleep((1/f)-delta)
                tinicio=time.time()
    ser.close()
    GPIO.cleanup()
```

O pacote de instruções do motor deve ter a seguinte forma:

Header1	Header2	ID	Length	Instruction	Param 1	...	Param N	Checksum
0xFF	0xFF	ID	Length	Instruction	Param 1	...	Param N	CHKSUM

Figura 8: Formato do pacote de instruções. (7)

O comando é dado em hexadecimal e sempre começa com a sequencia FF FF, seguido do ID do motor, o tamanho do pacote de instrução, os parâmetros e o Cheksum.

Deve-se então desenvolver funções capazes de transformar cada linha do arquivo csv para uma string nesse formato.

6.2.1 Funções base

Primeiramente, deve-se criar função para transformar um numero decimal para hexadecimal (transformahex) e outra para fazer o contrario (transformadec). Os códigos são os seguintes:

```
def transformahex(numero):
    hex = ["0","1","2","3","4","5","6","7","8","9","A","B","C","D","E",
        "F"]
    numerohex=''
    n = int(numero)
    r = []
    if n==0:
        numerohex='0'
    else:
        while n > 0:
            r.append(hex[(n % 16)])
            n = n // 16
```

```
        for i in range(len(r)-1,-1,-1):
            numerohex= numerohex+r[i]
    return numerohex

def trasformadec(hex):
    dec=int(hex, 16)
    return dec
```

Outro desafio é pelo fato de os parâmetros ângulo e velocidade após terem sido colocados em hexadecimal, devem ser transformados para uma forma certa, onde cada valor deve ser separado em dois conjuntos. Por exemplo, o valor de angulo de 300, será colocado na função transformahex, que retornará o valor de 12C, que será colocado na forma 2C 01, a ser recebido pelo motor. O código que faz essa mudança esta abaixo.

```
def poenaformacerta(numero):
    formacerta=''
    z=0
    numerohex=transformahex(numero)
    z=len(numerohex)
    forma='{0}{1} {2}{3}'
    if z==1:
        formacerta=forma.format(0, numerohex, 0, 0)
    if z==2:
        formacerta=forma.format(numerohex[0], numerohex[1], 0, 0)
    if z==3:
        formacerta=forma.format(numerohex[1], numerohex[2], 0,
            numerohex[0])
    if z==4:
        formacerta=forma.format(numerohex[2], numerohex[3], numerohex
            [0], numerohex[1])
    return formacerta
```

6.2.2 Movendo um motor

A função de mover um motor, aqui chamada como Move, deverá ter como entrada uma linha do arquivo, e deverá executar o comando. Por exemplo, ao mandar o motor de ID 1, para o angulo 512 com velocidade de 30, o comando fica FF FF 01 07 03 1E 00 02 2C 01 A7 ,onde após o conjunto FF FF vem o id (01), seguido do pacote de instrução que significa mover para um determinada posição (07 03 1E), seguido do angulo (00 02) e da velocidade (2C 01). O ultimo pacote é o Checksum (A7), que um valor de checagem dado pelo somatório dos pacotes anteriores. A seguir o código da função que monta esse comando e da função Move.

```
def calculacomando(id, angulo, speed):
    angulo=fator_ang*float(angulo)
    speed=fator_vel*float(speed)
    id=dectohex(id)
```

```
    angulohex=poenaformacerta(angulo)
    speedhex=poenaformacerta(speed)
    com='FF FF {0} 07 03 1E {1} {2}'
    comando=com.format(id, angulohex, speedhex)
    comsep=comando.split(" ")
    x=0
    for i in [2,3,4,5,6,7,8,9]:
        x=x +trasformadec(comsep[i])
    check=''
    if x<255:
        check=transformahex(255-x)
    else:
        x=transformahex(x)
        x=x[len(x)-2:len(x)]
        x=trasformadec(x)
        check=transformahex(255-x)
    if trasformadec(check)<16:
        comando=comando+' 0'+check
    else:
        comando=comando+' '+check
    return comando

def move(excel):
    id=int(excel[0])
    angulo=float(excel[1])
    speed=float(excel[2])
    angulo = angulo + correcao[id][2]
    comando=calculacomando(id, angulo, speed)
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(channel, GPIO.OUT)
    ser = serial.Serial("/dev/ttyAMA0", baudrate=1000000, timeout=10.0)
    GPIO.output(18, GPIO.HIGH)
    t1=time.time()
    ser.write(bytearray.fromhex(comando))
    t2=time.time()
    time.sleep(0.0009) #0.0009
    GPIO.output(18, GPIO.LOW)
```

6.3 Resultados das corridas

O objetivo da categoria *Humanoid RobotRacing* (HRR), é fazer o robô correr em uma pista de 4 metros no menor tempo possível. Não foi possível andar a pista toda, por isso foi contabilizado a distância máxima percorrida em 3 minutos. Na tabela abaixo estão listadas os resultados obtidos na final da LARC 2019, no dia 26 de outubro.

Tentativa	Tempo(min)	Quedas	Distância(cm)
1	3	1	65
2	3	1	85
3	3	2	80

Tabela 3: Resultados das corridas

7 Próximos passos

Conforme explicitado no cronograma, a partir de agora deve-se preocupar com a implementação dos sensores: sensor de pressão nos pés, IMU e sensor de distância . Também é preciso estudar sobre a obtenção do *feedback* dos motores. É necessário a construção de uma placa melhor a ser embarcada, e para isso serão necessários conhecimentos do software Altium Designer.
Deseja-se, com isso, fazer o robô competir novamente em 2020, completando a pista no menor tempo possível, objetivando um lugar no pódio.

8 Dificuldades encontradas

As dificuldades encontradas foram aqui divididas nos seguintes tópicos:

8.1 Movendo vários motores

No *datasheet* do motor é mostrado um comando para mover todos os motores ao mesmo tempo. Então, foi desenvolvido uma função abaixo, que lê uma linha de um arquivo csv e transforma em um comando útil para o motor.

```
def sync_n_motors(excel): #n    a quantidade de motores
    n=int(excel[0])
    tamanho=4+5*n
    tamanhohex=dectohex(tamanho)
    com='FF FF FE {0} 83 1E 04'
    comando=com.format(tamanhohex)
    for i in range (0,n):
        id=int(excel[3*i+1])
        angulo=float(excel[3*i+2])
        angulo=int(angulo*fator_ang+0.5)
        speed=float(excel[3*i+3])
        speed=int(speed*fator_vel+0.5)
        angulohex=forma_certa(angulo)
        speedhex=forma_certa(speed)
        id=dectohex(id)
        com='{0} {1} {2}'
        com_motor=com.format(id, angulohex, speedhex)
        comando= comando+' '+com_motor
    tamanho=tamanho+3
    check= calcula_check(tamanho,comando)
    comando=comando+check
    GPIO.output(18, GPIO.HIGH)
    ser.write(bytearray.fromhex(comando))
    time.sleep(0.1)
    GPIO.output(18, GPIO.LOW)
```

Porém, durante os testes teve-se muitos problemas causados pela demora para execução dessa função, fato que atrasava o envio do comando, sendo este muito grande, por se

tratar de um comando para 10 motores. Assim, muitas vezes, acontecia de uma nova ordem estar sendo enviada para o motor enquanto este ainda não tinha terminado de receber a ordem anterior, o que gerava perda de pacotes e tombamento do robô. Para resolver esse problema começou-se então a não usar esse artificio de mover vários ao mesmo tempo, e sim enviando um comando de cada vez para cada motor em uma frequência muito alta.
Colocando contadores de tempo no código, foi possível ver que o tempo de ler a matriz que guardava os valores do arquivo, gerar o comando útil e enviá-lo para o motor era de 0.0029 segundos em média. Um conjunto de comando para os 10 motores não poderia acontecer em um tempo menor que 0,023 segundos,tempo determinado pela frequência de 45 Hz.Isso é forçado a acontecer por um *time.sleep* colocado no primeiro código da seção Desenvolvimento da biblioteca. Um passo é constituído de 490 linhas, ou seja, existem 49 conjuntos de 10 comandos e demora cerca de 1,30 segundos para terminar.

8.2 Motores com pouco torque

O modelo inicial do robô contava com 12 motores nas pernas. Porém durante os testes foi necessário remover 2 desses motores, que estão indicados na figura 9, por causa que, apesar deles darem mais graus de liberdade, eles não eram completamente necessários para a caminhada e estavam gerando muitos problemas, pois aumentavam o tamanho das pernas em torna de 6 centímetros, o que acabava por aumentar o torque do peso do corpo sobre os demais motores. Assim, o que acontecia bastante durante os testes era de os motores do joelho ou das coxas terem seu limite de torque ultrapassado, limite esse de 15.3 kg·cm, e acabavam desligando, pois, conforme já apresentado, o AX12-A possui desligamento automático para proteção. Isso era percebido através do LED do motor, que começava a piscar, indicando o erro.

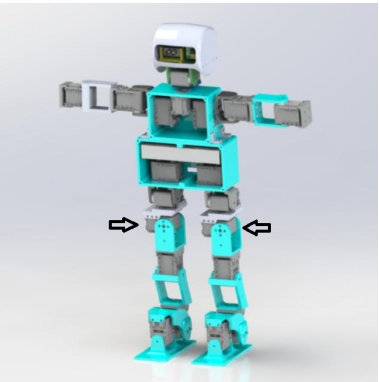

Figura 9: Modelo inicial com 11 graus de liberdade nas pernas.

8.3 Conexão em serie dos motores

O projeto inicial da placa era para que os 5 motores de cada perna ficassem em série entre si. Porém, durante os testes percebeu-se que quando muitos motores são ligados em série, era frequente acontecer o desligamento dos motores por proteção. Isso porque a corrente tinha que ser dividida para muitos motores, o que resultava que alguns motores, principalmente os do joelhos e coxas, que são os motores que fazem mais força, recebiam pouca corrente para o torque necessário. Assim, fez-se conforme o esquematizado na figura 10, separando uma perna em dois conjuntos em paralelo, um com 2 motores em série, representado pela cor verde, e outro com 3 motores em série, representado pela cor vermelha.

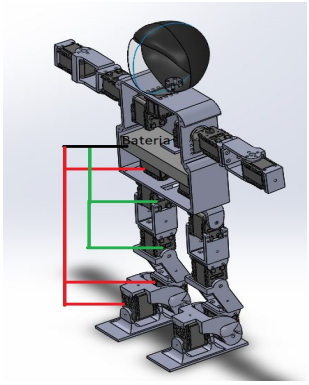

Figura 10: Representação da conexão série/paralelo dos motores.

Referências

[1] Hardware: O que é microprocessador? Disponível em: https://guiadatecnologia.com/2012/02/hardware-o-que-e-microprocessador.html Acessado em:Julho de 2019.

[2] Arduino vs Raspberry Pi: entenda as diferenças Disponível em: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ Acessado em:Julho de 2019.

[3] Arduino vs Raspberry Pi: entenda as diferenças Disponível em: https://www.embarcados.com.br/arduino-vs-raspberry-pi/ Acessado em:Julho de 2019.

[4] Motor CC: Saiba como Funciona e de que forma Especificar. Disponível em: https://www.citisystems.com.br/motor-cc/ Acessado em: Junho de 2019.

[5] Motor de Passo – O que é e como funciona. Disponível em: https://athoselectronics.com/motor-de-passo-como-funciona/ Acessado em: Junho de 2019.

[6] Servo Motor: Veja como Funciona e Quais os Tipos. Disponível em: https://www.citisystems.com.br/servo-motor/ Acessado em: Julho de 2019.

[7] Robots e-manual. Disponível em: http://emanual.robotis.com/docs/en/dxl/protocol1/status-packet Acessado em: Novembro de 2019.

[8] AX-12A Dynamixel Servo. Disponível em: https://www.robot-r-us.com/vmchk/motor-robot-servo/ax-12a-dynamixel-servo.html Acessado em:Julho de 2019.

[9] How to drive Dynamixel AX-12A servos (with a RaspberryPi). Disponível em: https://projects-raspberry.com/how-to-drive-dynamixel-ax-12a-servos-with-a-raspberrypi/ Acessado em: Outubro de 2019.

[10] Dynamixel AX-12A and Arduino: how to use the Serial Port. Disponível em: https://robottini.altervista.org/dynamixel-ax-12a-and-arduino-how-to-use-the-serial-port Acessado em: Agosto de 2019.

[11] Conheça os tipos de atuadores e motores elétricos (Servo, motores de passo e corrente continua) e suas aplicações. Disponível em: http://labdegaragem.com/profiles/blogs/tutorial-sobre-tipos-de-motores-servo-motores-de-passo-e-corrente Acessado em: Junho de 2019.