Universidade Federal de Santa Catarina

Departamento de Automação e Sistemas

Projeto Integrador - DAS 5104

Protótipo do Sistema

Acadêmicos:	*Matrícula:*
Daniel Regner	15204839
Igor Althoff Vidal	13104313
Ivan Doria	13204824

Sumário

1 Introdução

Este documento é um registro dos avanços obtidos pelo grupo no desenvolvimento do sistema, de modo a permitir uma avaliação do progresso desenvolvido. Ao longo das últimas semanas, o grupo esteve com foco total no desenvolvimento do algoritmo de movimentação e nas implementações simulada e real da plataforma robótica.

Dado que o projeto original do robô foi realizado para suportar pouca carga já com os servomotores operando nos limiares de sua faixa de operação, foi levantada junto a empresa a possibilidade de desenvolver uma plataforma maior e mais potente, que permitiria a ampliação do projeto no futuro ao disponibilizar o transporte de uma carga maior e ter uma operação que não esteja no limite do suportado pela plataforma.

Como o grupo ainda não possuía a nova plataforma em mãos, já que esta teve que ser fabricada através de uma impressora 3D, o grupo focou no desenvolvimento de um modelo de simulação para facilitar os testes do algoritmo de movimentação desenvolvido, além de também desenvolver a base do código que será utilizado no microcontrolador embarcado na plataforma.

2 Plataforma Quadrupede

Após feito alguns testes com a Plataforma para um robô quadrupede utilizando Servo motores de 1.1 Kgfcm foi diagnosticado a falta de torque dos motores para sustentação da estrutura para efetuar a movimentação. Com este problema diagnosticado a tempo, foi proposto a empresa a impressão de uma nova estrutura para utilização de Servo-motor MG955, os quais possuem torque de aproximadamente 10kgf.cm.

A nova estrutura utilizada foi escolhida através de modelos já prontos e desenvolvidos representado no site https://www.instructables.com/id/A-3D-Printed-Quadruped-Robot/. Como o tempo de produção das peças levaria um prazo não previsto nos cronogramas anteriormente, foi sugerido o inicio da implementação através de uma Simulação computacional utilizando o software V-REP descrito nas seções subsequentes.

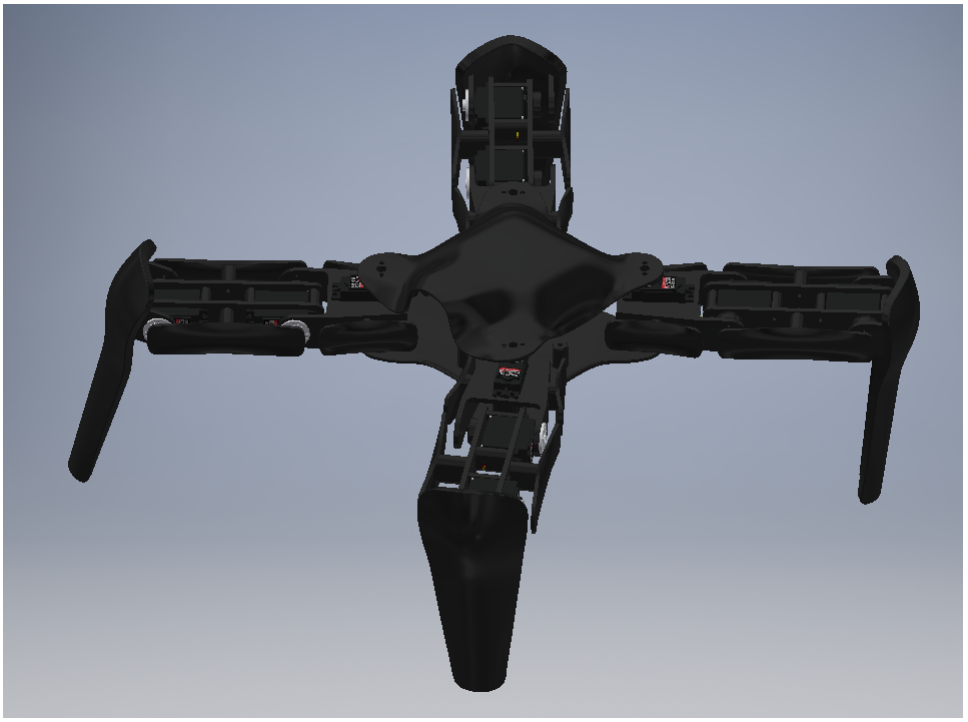

Figura 1: Desenho nova estrutura desenvolvido em Inventor

Para comprovação de que a nova estrutura teria condições de efetuar o movimento e sustentação da estrutura foi desenvolvido um cálculo estrutural sobre os momentos resultantes em cada perna, sendo a força peso de cada componente medida ao longo

da montagem e apresentada na tabela 1. Utilizando da imagem 2 para calcular o momento resultante sobre os motores A e B.

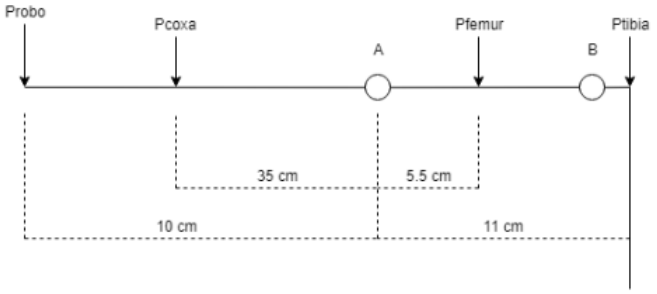

Figura 2: Diagrama Estrutural para cada perna

Efetuado os cálculos para os momentos resultantes sobre os motores representados na imagem, resultando em um momento aplicado sobre os motores de aproximadamente 6 e 8 kgfcm, inferior ao torque nominal de um Servo-motor MG955 com tensão nominal de 5 V.

Componente	Peso
Base	230 g
Coxa	130 g
Fêmur	180 g
Tibia	80 g
Total perna	390 g
Total robo	1790 g

Tabela 1: Tabela de Pesos da Estrutura

3 Movimentação

3.1 Máquina de Estados

Para simplificar a movimentação do robô e dar continuidade ao desenvolvimento do projeto, concluímos que para curto prazo seria mais interessante desenvolver a cinemática do sistema com base em uma máquina de estados.

Dado que, para sustentação do robô, é necessário que não haja deslizamento entre as pernas e a superfície de trabalho, o robô deve levantar cada perna que deva ser movimentada. Entretanto, levantar mais de uma perna ao mesmo tempo pode resultar na perda da estabilidade do sistema, sendo assim recomendado para robôs de mesma configuração, que somente uma perna seja removida da superfície por vez.

De modo a garantir a estabilidade do robô mesmo quando uma de suas pernas não está apoiada na superfície de trabalho, é necessário que em todos os instantes o Centro de Gravidade do sistema esteja dentro do triângulo formado pelos pontos de sustentação do mesmo. Esta situação é ilustrada na Figura 3

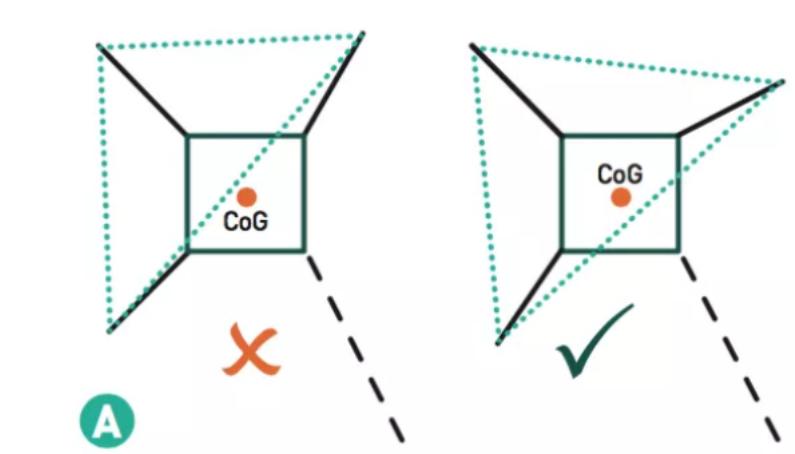

Figura 3: Detalhe do Centro de Gravidade dentro do Robô dentro do triângulo formado pelos pontos seus pontos de sustentação.
Fonte: `https://makezine.com/2016/11/22/robot-quadruped-arduino-program/`

Tendo definidas as condições que devem ser respeitas para correta movimentação do robô, definiu-se a sequência de movimentação a ser implementada para o mesmo. Esta implementação foi escolhida dada a sua simplicidade de implementação, além de ter uma maior garantia de funcionamento, apesar de ser menos eficiente do que métodos como a cinemática inversa. Isto acontece pois na primeira os movimentos

são desenvolvidos em sequência, perna a perna, e na segunda todos os servos das pernas são ajustadas ao mesmo tempo.

A imagem 4 representa os passos a serem tomados para iniciar e desenvolver uma caminhada, sendo as pernas representadas por uma letra de A a D.

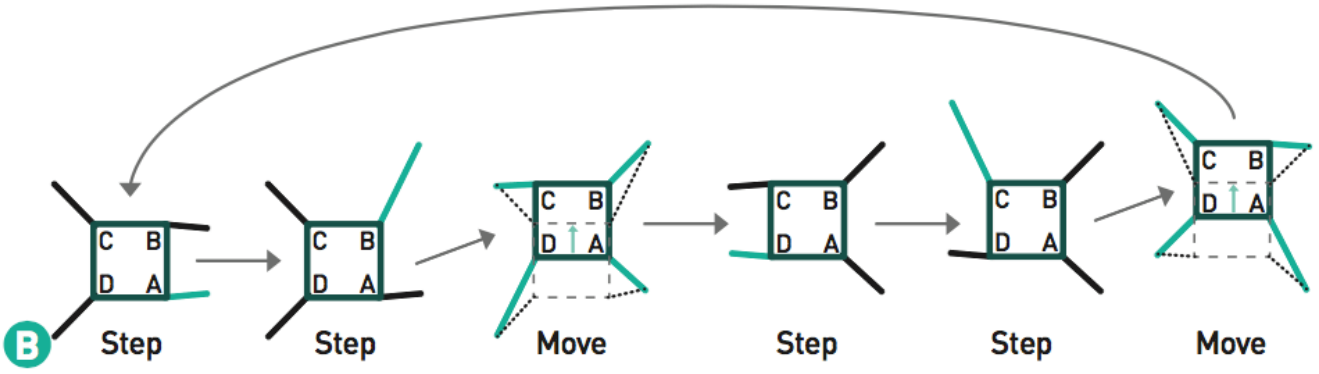

Figura 4: Sequência dos movimentos de perna realizados pelo robô de modo a se deslocar para frente.
Fonte: `https://makezine.com/2016/11/22/robot-quadruped-arduino-program/`

Abaixo, na Figura 5, ilustra-se a máquina de estados obtida.

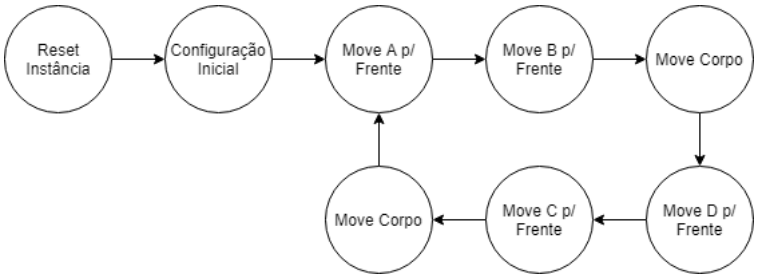

Figura 5: Máquina de Estados para andar

4 Simulação V-REP

A movimentação do robô é uma das questões centrais deste projeto. A disposição das pernas para se alterar o centro de massa e consequentemente o andar do robô não é algo simples. Existe portanto a necessidade de validar a movimentação que já foi descrita teoricamente em relatórios passados. Para tal é necessário simular o robô com o auxílio de ferramentas computacionais. A partir de uma pesquisa encontrou-se dois possíveis simuladores:

- Gazebo;
- V-rep;

O Gazebo é um software de simulação Open-Source utilizado por diversos grupos de pesquisas ao redor do mundo, porém ele somente é acessível em ambiente Linux, além disso a modelagem de novos robôs não é fácil. O V-rep, é gratuito para fins acadêmicos e possui diversas ferramentas que são úteis para o nosso projeto, alguns integrantes do grupo já possuíam familiaridade com essas ferramenta, dessa forma foi escolhido a plataforma do V-rep.

4.1 Modelagem do robô

O ambiente de simulação V-rep permite importar os arquivos diretamente do SolidWorks, dessa forma, o modelo físico do robô será o mesmo utilizado na simulação.

O V-rep otimiza a simulação permitindo a criação de uma Layer invertida com o intuito de tornar o cálculo do modelo mais simples. Essa Layer invertida é uma versão simplificada do robô, com peças sólidas mais simples, pois, quanto mais peças possui o robô mais complexo é o cálculo realizado pelo simulador. Conforme pode-se observar abaixo:

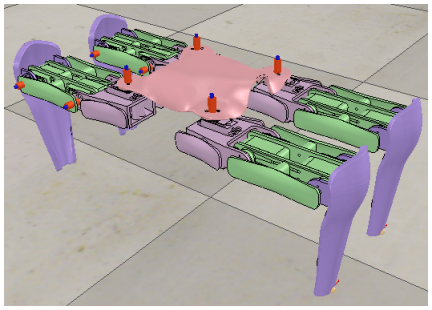

Figura 6: Modelo físico do robô

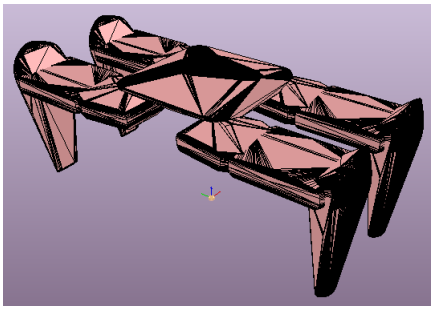

Figura 7: Modelo invertido do robô

4.2 Software da simulação

Com o ambiente de simulação V-rep a programação do robô é feito com linguagem Lua, porém é uma linguagem na qual os integrantes deste trabalho não possuem afinidade, dessa forma o V-rep permite a integração com outras linguagens de programação, tais como:

- Python
- C++
- Matlab
- Octave
- Java

Foi escolhido dessa forma a linguagem Python para o software do robô. Assim sendo utiliza-se no ambiente de simulação do código:

```
1 simRemoteApi.start(19999)
```

Abrindo assim uma porta para comunicação via Socket em 19999. E dentro do código em python escreve-se então:

```
1 vrep.simxStart('127.0.0.1', 19999, True, True, 5000, 5)
```

Possibilitando então a interação entre o ambiente simulado e um código externo. No código disponível no repositório do projeto é então adicionado toda a sequencia de comandos para fazer o robô se mover.

A partir da janela de comando do python é possível inserir a direção e sentido do movimento do robô :

- W - frente;
- S - trás;
- A - gira esquerda;
- D - gira direita;

5 Desenvolvimento do Código para o ESP-32

5.1 Comunicação entre Estação de Operação e a plataforma Robótica

5.1.1 Interface de comunicação Wi-Fi via WebSockets

De modo a permitir a operação remota da plataforma, mostrou-se necessária a implementação de uma interface de comunicação entre o microcontrolador ESP-32 e a estação de operação. Conforme já mencionado anteriormente, o microcontrolador ESP-32 foi escolhido pois este já possui um chip para comunicação *Wireless* com base nos padrões da norma IEEE 802.11.

Devido a prévia experiência do grupo com a utilização de Soquetes de Redes (popularmente conhecido por Sockets) na Raspberry Pi, optou-se por primeiramente tentar utilizar-se desta solução para resolver o problema de comunicação entre o ESP e a Raspberry Pi. Testes foram feitos com implementações dos protocolos TCP e UDP no ESP. porém sem sucesso satisfatório. Esta solução mostrou-se demorada devido a baixa documentação sobre a implementação dos Sockets no ESP-32 e o baixo suporte da comunidade, e por isso novas alternativas foram procuradas.

Após discussão do problema com a empresa Seashell, o grupo foi recomendado a implementar o protocolo WebSockets, dado a sua simplicidade e a sua grande utilização. Por se tratar de um protocolo mais documentado para o ESP, sua implementação foi feita de maneira rápida e com uma performance aceitável.

Ainda aproveitando-se da vasta documentação do uso dos WebSockets, foi criado um *script* em Python que constantemente envia mensagens para o ESP-32 através dos WebSockets e recebe a resposta do mesmo. Este *script* foi até o momento foi testado em um notebook com Linux e por isso não deve apresentar problemas ao ser utilizado em uma Raspberry Pi que também dispõe de Linux e a mesma versão do Python.

Através de uma conversa com a empresa, também ficou decidido que a implementação da interface de usuário (UI) se daria pela parte deles, visto que eles possuem maior domínio sobre as ferramentas necessárias e que eles podem desenvolver a aplicação de uma maneira que facilite a integração com o sistema desenvolvido por eles.

5.1.2 Dados Transmitidos

Para garantir que o usuário tenha uma boa experiência ao operar a plataforma robótica, foi realizado um brainstorm, com os membros do grupo e da empresa, de modo a definir quais seriam os dados que deveriam ser disponibilizados para o usuário e quais informações este iria transmitir para o robô.

A informação mais importante que o robô deve mandar para o usuário é a leitura dos sensores ultrassônicos, para que o operador possa ter uma ideia do meio onde o robô se encontra e validar as decisões a tomadas quanto aos movimentos a serem desenvolvidos. Sendo assim, esta informação é útil na parte de desenvolvimento e aperfeiçoamento do produto como também no produto final.

Além disso, mostrou-se interessante disponibilizar ao usuário os dados relativos a posição de cada uma das juntas do robô. Assim, é possível recriar um modelo da plataforma na UI e permitir um acompanhamento do status do robô remotamente.

Por último, seria interessante também adicionar um dispositivo de medição por inércia (IMU), para permitir a visualização da orientação da plataforma nos 3 eixos, melhorando a experiência do usuário.

De modo a permitir que o robô desenvolva as tarefas desejadas pelo operador, chegou-se a conclusão da necessidade de enviar a plataforma o modo de operação desejado (Automático ou Manual), para definir se o robô deve tomar suas próprias decisões quanto ao movimento a ser executado ou se este deve aguardar a passagem de uma referência para se movimentar.

No caso da operação manual, foi definido que a passagem de referência será a direção para a qual o robô deve movimentar-se. Como o robô não possui um *header*, ou seja, qualquer lado pode ser sua frente, este pode desempenhar um padrão de movimentação *headerless*, mudando sempre sua definição de frente quando uma referência solicitando movimentação em outra direção é recebida.

Esta característica ainda facilita o desenvolvimento do código de movimentação, pois é necessário somente desenvolver a sequência de movimentos para cada perna de modo com que o robô ande para frente, restando somente alterar a ordem de movimentação de cada perna quando este tiver que se movimentar em outra direção, sem a necessidade de rotacionar o corpo do robô. Esta situação é melhor ilustrada na Figura 8.

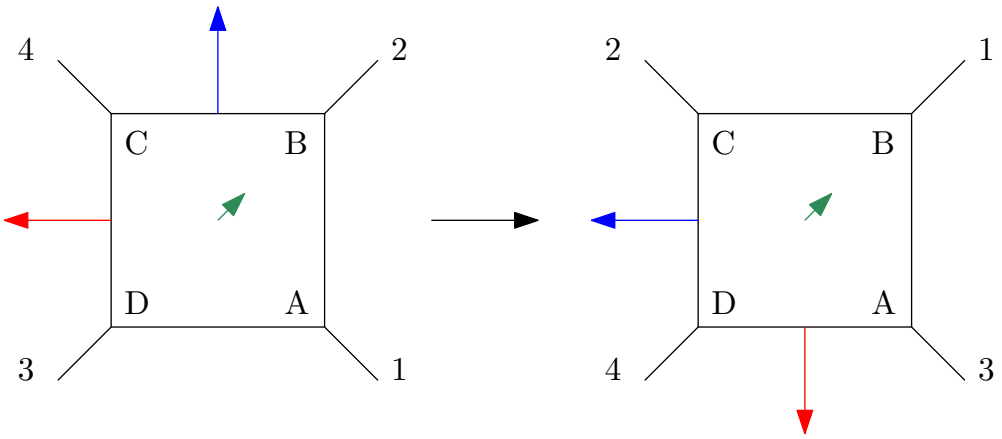

Figura 8: Exemplo da mudança de orientação do robô. Com o robô se movimentando para a frente, temos a ordem de movimentação A,B,D,C. Quando solicita-se movimento para a esquerda, teremos a ordem B,C,D,A.
https://www.overleaf.com/2652991939zqsjhmjvjskx

Foi requisitado pela empresa que os dados trocados entre a estação de operação e a plataforma robótica fossem formatados no padrão *JSON*, facilitando sua legibilidade por parte dos usuários, além de possuir um bom suporte da comunidade e possuir vários métodos disponíveis para o desenvolvimento.

5.2 Implementação da Máquina de Estados

Para a implementação da sequência de movimentação do robô, foi desenvolvida no ESP-32 a lógica da movimentação para a frente. Como ilustrado na Figura 5, o robô possui 8 estados em que ele irá desenvolver algum tipo de movimentação. Uma função *HandleFSM()* foi implementada para executar as mudanças de estados, com a utilização de um *Switch:Case* para determinar as ações tomadas em um certo estado.

Como o robô ainda não encontrava-se montado para a realização dos testes, um pequeno circuito foi montado com 1 LED representando cada perna do robô. Assim, toda vez que uma perna deveria ser movimentada, o LED correspondente acenderia. O código foi implementado e executado com sucesso, usando como transição entre os estados um certo período de tempo transcorrido.

Com a finalização da montagem do robô, o próximo passo é colocar a máquina de estados para operar as pernas e verificar a necessidade de outros estados não levantados até agora, além da determinação da condição de transição entre os estados,

garantindo que o robô tenha tempo suficiente para executar o movimento de uma perna antes de iniciar o movimento de outra.

5.3 Execução paralela via Threads

Para garantir que o robô possa desenvolver suas funções continuamente sem ter que parar para receber comandos do usuário, tirando proveito dos dois núcleos de processamento do ESP, foi definido que 2 *threads* seriam executadas paralelamente: Uma para a movimentação e outra somente para comunicação com a interface de usuário.

Para isto, optou-se pela utilização da biblioteca *ArduinoThread*, dado a experiência prévia com a mesma. Para sua utilização, são definidas as funções que serão executadas paralelamente ao código principal, depois a thread é inicializada recebendo como parâmetros a respectiva função e o requisito temporal para que a função seja executada novamente. Por último, é criado o controlador, que tem as threads criadas atribuídas a ele.

A primeira thread criada é referente a comunicação, tendo em vista que já existia um prótotipo do código da comunicação funcionando. Assim, a função que previamente era chamada no *loop* do Arduino para escutar novas mensagens e respondê-las, agora foi associada a thread *handleCommunication()*.

A segunda thread é a relativa à máquina de estados. O código mencionado na Seção 5.2 foi implementado como uma thread. O novo estado do sistema é sempre setado com a finalização do estado anterior, e somente quando a thread é executada novamente as ações do novo estado são desenvolvidas. Para garantir a execução contínua das chamadas para posicionamento dos servos, a thread será responsável somente por realizar o cálculo das novas posições, enquanto a atualização das posições irá acontecer repetidamente no *loop* principal.

A implementação destas duas threads em paralelo ocorreu com sucesso, demonstrando ser capaz de atender os requisitos temporais do sistema. Entretando deve-se tomar cuidado com chamadas do tipo *delay* dentro do código das threads, já que o Arduino bloqueia todas as execuções durante o período informado.

Um problema da implementação atual é que não existe nenhum tipo de proteção quanto as variáveis que são manipuladas pelas Threads, o que pode causar inconsistências. Para resolução deste problema, será implementado um mutex para cada uma das variáveis que serão modificadas, impedindo assim a mudança dos valores destas variáveis enquanto as mesmas já estiverem sendo utilizadas e garantindo assim o funcionamento em paralelo adequado.

6 Conclusão

O andamento do projeto está ocorrendo de maneira satisfatória, dentro dos prazos estipulados. A empresa, que está em constante contato com a equipe, considera positivo o trabalho até o momento. Como passos futuros tem-se as seguintes tarefas:

- Movimentar o robô;
- Implementar o sensoriamento;
- Realizar a comunicação entre as interfaces do sistema;

Dentro destas tarefas estão inseridas várias outras menores, tais como o acionamento dos servo motores, a implementação do protocolo de comunição JSON, dentre outros. Cada uma destas sub tarefas demandará dedicação da equipe para que seja feito de forma satisfatória.

Os resultados obtidos até o presente momento demostram que o projeto está em processo de conclusão e os desafios estão sendo solucionados com perfeição.