Happy Feet - Plataforma Robótica para Espaço Inteligente baseado em comunicação por MQTT

Matheus R. Rocha * **Vitor S. Guerzet** * **Bruno N. Amigo** *
Fabricio B. Sá * **Leonardo A. Silva** *

* *Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo Coordenadoria de Eletrotécnica, Guarapari, ES, Brasil (e-mails: matheusrochareis@gmail.com, vitor7guerzet@gmail.com, bruno.amigo@ifes.edu.br, fabricio.sa@ifes.edu.br, leonardo.assis@ifes.edu.br).*

Abstract: The advancement of technology has increasingly brought people closer to robotics, making the concept of smart spaces and cities something conceivable and tangible. To provide these spaces a wide range of services, mobile robots are essential. Depending on the environment, these robots can be terrestrial, aerial and even aquatic. In this context, this work proposes a robotic platform, Happy Feet, for intelligent spaces based on MQTT protocol communication via Wifi for remote control. Interesting results are obtained through real tests done with an Arduino Mega and a NodeMCU embedded in the robot and a Raspberry Pi 3B as a remote station for the final position control.

Resumo: O avanço da tecnologia tem aproximado cada vez mais as pessoas da robótica, tornando o conceito de espaços e cidades inteligentes em algo concebível e palpável. Para prover a esses espaços uma grande diversidade de serviços, é fundamental a existência de robôs móveis. Dependendo do ambiente, esses robôs podem ser terrestres, aéreos e até aquáticos. Nesse contexto, este trabalho propõe uma plataforma robótica, o Happy Feet, para espaços inteligentes baseado em comunicação por protocolo MQTT via Wifi para controle remoto. Resultados interessantes são obtidos por meio de testes reais realizados com um Arduino Mega e um NodeMCU embarcados no robô e uma Raspberry Pi 3B como estação remota para o controle de posição final.

Keywords: Robotics; Arduino; Raspberry Pi; Intelligent Space; MQTT; Wifi.
Palavras-chaves: Robótica; Arduino; Raspberry Pi; Espaço Inteligente; MQTT; Wifi.

1. INTRODUÇÃO

Nos últimos anos, o avanço da tecnologia tem aproximado cada vez mais as pessoas da robótica. Robôs têm deixado de ser um instrumento de produção das indústrias e passado a ser parte do dia a dia dos cidadãos. O crescimento da internet também ajudou a conectar pessoas do mundo todo. Eletrodomésticos tornaram-se "inteligentes" e interconectados. *SmartTVs*, hoje em dia, possibilitam ao usuário assistir canais com a programação usual e também a assistir vídeos direto da internet.

Tudo isso torna o conceito de casas que se conectam com seus equipamentos "inteligentes" e permitem a interconexão entre eles algo concebível. Essa definição recebe o nome de "casa ou espaço inteligente" ou, em inglês, "*smart home*" ou "*intelligent space*". Nessas casas, câmeras, microfones, ultrassons, voltímetros e amperímetros, por exemplo, podem ser utilizados como sensores. As informações vindas dos sensores são processadas pelo sistema da *smart home*, produzindo comandos de ação em agentes atuadores espalhados pelo ambiente. (Rampinelli et al., 2014)

Esses atuadores, podendo ser motores, válvulas, pistões, relés, etc., servirão para movimentar uma cadeira de rodas, desligar um televisor, fechar a torneira da pia, ligar a luz de um cômodo ou o sistema de alarme da casa. Todos esses exemplos são serviços que podem ser inseridos num espaço inteligente de acordo com a demanda do usuário. (Queiroz et al., 2015; Rampinelli et al., 2014)

Para ampliar a diversidade dos serviços oferecidos ao usuário, é fundamental a existência de robôs móveis. Eles permitem a execução de tarefas não locais e diversas aplicações distintas, dependendo do atuador que possuir em sua estrutura. Dependendo do ambiente, robôs podem ter formas diferentes, sendo classificados como: terrestres, aquáticos e aéreos (Jones et al., 1998).

Em (Melo, 2015), um dos objetivos era que um robô terrestre navegasse por uma área delimitada, em busca de cilindros de metal, com o intuito de recolhê-los e conduzi-los até uma região de base pré-estabelecida. Para realizar essa tarefa, foram utilizados sensores de ultrassom. Já Shim and Cho (2015) utilizaram câmeras de segurança para localizar seu robô e obstáculos. Com a câmera, foi possível estimar a posição dos objetos e decidir a melhor

rota que atravessasse o corredor. Neste caso, a câmera já estava instalada no local, sendo possível aproveitar a infraestrutura do ambiente para executar a tarefa desejada.

Queiroz et al. (2015) propõem a localização e controle de um robô em um espaço inteligente. Esse espaço fica em um laboratório da Universidade Federal do Espírito Santo, possuindo 4 câmeras em seu interior. O sistema detecta um grid dupla face de 4×6 LEDs infravermelhos fixado sobre a estrutura da plataforma robótica. Com o auxílio de técnicas de Visão Computacional, como homografia, é possível estimar a posição e a orientação do robô no laboratório e realizar o controle de movimentação.

Como pode-se notar, é bem comum a utilização de robôs terrestres em ambientes fechados. Apesar disso, atualmente, drones têm se popularizado bastante, inclusive em operações conjuntas com robôs terrestres. Muitas vezes, é possível realizar essas cooperações entre diferentes robôs para a realização de uma tarefa, visto que um robô pode não conseguir executá-la sozinho, ou a cooperação agrega as melhores características de cada robô para a tarefa.

Em (Sá et al., 2014), um robô aéreo (drone) auxilia um terrestre a identificar o melhor caminho para o deslocamento até um determinado local. Como o drone possui uma visão privilegiada do alto, é mais fácil perceber obstáculos no trajeto do robô terrestre, alertá-lo e definir uma nova rota. Para isso, o drone possui uma câmera, além de GPS, barômetro, acelerômetro e giroscópio. Esses últimos podem não estar diretamente relacionados à tarefa de detecção de obstáculo, mas possibilitam que o robô aéreo se localize e se movimente no ambiente.

Um drone também pode ser comandado para realizar missões de busca e seguimento de um alvo móvel até que sua base seja detectada (Salvador et al., 2017, 2019), ou ser usado como equipamento de filmagem em festas e eventos. De qualquer maneira, em todos os trabalhos, os autores tentavam estimar a postura do robô da forma mais precisa que dispunham, seja por câmeras, ultrassom ou odometria. Isso acontece porque as tarefas exigiam conhecer tais informações (e.g., se locomover até certo ponto, desviar de obstáculo, encontrar objetos, etc.).

Nesse contexto, este trabalho propõe o desenvolvimento de uma plataforma robótica, chamada Happy Feet, com sensores dentro do escopo de cidades digitais e espaços inteligentes, que possa ser controlada por Wifi para a realização de tarefas teleoperadas ou automáticas. Os autores estão desenvolvendo um espaço inteligente no IFES (Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo) Campus Guarapari, sendo esta plataforma o primeiro agente do sistema.

Sendo assim, o trabalho aqui proposto será apresentado em sete seções. A Seção 2 traz a descrição da plataforma desenvolvida neste trabalho. O modelo de robô uniciclo utilizado para as explanações do texto e implementação do controlador é detalhado na Seção 3. O processo de estimativa da odometria e o funcionamento do sistema de comunicação estão explicados nas Seções 4 e 5, respectivamente. Os resultados são apresentados e discutidos na Seção 6. A conclusão, propondo alguns trabalhos futuros, é abordada na Seção 7.

2. DESCRIÇÃO DA PLATAFORMA

Para a realização deste trabalho, uma plataforma robótica de tração diferencial foi desenvolvida pelo Laboratório Penguin do IFES Campus Guarapari, controlada por um Arduino Mega embarcado e uma Raspberry Pi Model 3B como estação remota. Essa plataforma foi chamada de Happy Feet e pode ser conferida na Figura 1.

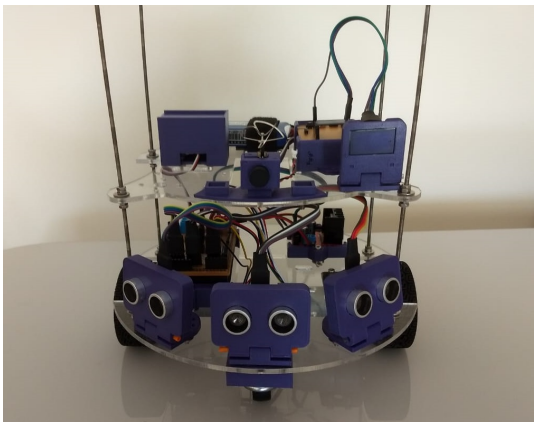

Figura 1. Happy Feet, a plataforma robótica desenvolvida.

Como pode-se conferir na Figura 1, a estrutura de dois andares do robô foi confeccionada em acrílico transparente, sustentadas por quatro barras roscadas. As peças roxas que prendem os componentes eletrônicos ao corpo foram modeladas e impressas em PLA por uma impressora 3D.

Também estão embarcados no robô: um módulo Wifi ESP8266 12E (NodeMCU), três módulos ultrassom HC-SR04, um display OLED, uma câmera FPV (para Visão em Primeira Pessoa), um módulo de ponte H L298, dois motores CC de 6 V com encoder, e uma bateria Zippy de 1100 mAh, $2S$ e 6,6 V.

O Arduino Mega, estima a postura do robô (odometria) por meio das interrupções do encoder nos motores, calcula as distâncias de obstáculos pelos módulos ultrassom, comanda os motores com PID para cada roda, estima o nível de tensão da bateria e se comunica com o NodeMCU.

O NodeMCU conecta o robô à rede do laboratório e ao broker MQTT criado pela Raspberry Pi. Por ele, o robô publica informações gerais e de odometria (ponto central do robô (x, y), orientação ψ, nível de bateria, os sensores ultrassom us_E, us_D e us_F, e um identificador da mensagem id_msg), e obtém valores de velocidade linear (v) e angular (w) vindas do controlador da Raspberry Pi.

A Raspberry Pi usa a base de Linux (Raspberry Pi OS). O controlador utilizado para a tarefa descrita na Seção 6 foi programado em Python 3 e publica suas informações de velocidade em um broker MQTT Mosquitto (Mosquitto, 2019) com a biblioteca paho-mqtt (Light, 2019).

Demais informações sobre o projeto podem ser conferidas no link: `https://www.eletroifes.com.br/penguin/projetos/happy-feet` e no GitHub do laboratório: `https://github.com/Penguin-Lab`.

3. MODELO DO ROBÔ

Para definir o controle para um robô, é necessário um modelo de como o robô se comporta. No caso em questão, o Happy Feet é um robô com tração diferencial com duas rodas laterais e uma roda boba na frente. Dessa forma, é possível definir velocidades distintas para a roda direita (v_D em mm/s) e esquerda(v_E em mm/s), permitindo que a plataforma rode em torno de um ponto qualquer no mundo, ou siga em alguma direção em linha reta.

O modelo cinemático mais comum em robôs de tração diferencial é o modelo de robô uniciclo (Cruz and Carelli, 2006), que foi usado neste projeto. A Figura 2 apresenta este modelo com parâmetros construtivos e de controle.

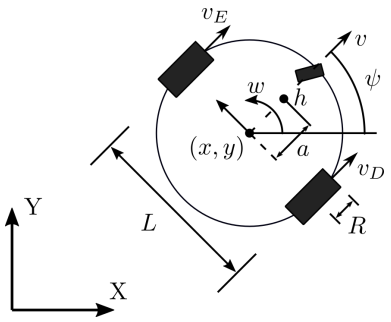

Figura 2. Parâmetros do modelo de robô uniciclo utilizado.

A postura do robô é definida pelo seu ponto central (x, y) e sua orientação ψ. R é o raio das rodas, v e w são as velocidades linear e angular do robô. v_D e v_E são as velocidades lineares das rodas direita e esquerda.

Com os valores de v e w recebidos do controlador, o robô calcula a velocidade de cada motor (Maximo, 2015) por:

$$v_D{}^* = \left(v + \frac{wL}{2}\right), \tag{1}$$

$$v_E{}^* = \left(v - \frac{wL}{2}\right), \tag{2}$$

sendo $v_D{}^*$ e $v_E{}^*$ as velocidades lineares desejadas para as rodas, e L a distância entre as duas rodas do robô.

Um PID executado no Arduino utiliza essas velocidades lineares como entrada desejada, as velocidades estimadas com os encoders (v_D e v_E) como entrada lida, e calcula um valor de PWM que controlará a tensão sobre os motores e, consequentemente, suas velocidades.

Neste artigo, o ponto adotado para ser controlado como a variável de posiçao do robô será $h = (x_h, y_h)$. Esse ponto é igual a (x, y) deslocado na direção de v em a mm. Dessa forma, podemos calcular a variação da postura do ponto h como sendo:

$$\dot{x}_h = v\cos\psi - aw\,\mathrm{sen}(\psi), \tag{3}$$

$$\dot{y}_h = v\,\mathrm{sen}\psi + aw\cos(\psi), \tag{4}$$

$$\dot{\psi}_h = w, \tag{5}$$

sendo ψ_h a orientação do ponto h, que é igual a do robô. Rearranjando (3), (4) e (5) em matrizes, temos:

$$\begin{bmatrix}\dot{x}_h \\ \dot{y}_h\end{bmatrix} = \begin{bmatrix}\cos\psi & -a\,\mathrm{sen}\psi \\ \mathrm{sen}\psi & a\cos\psi\end{bmatrix}\begin{bmatrix}v \\ w\end{bmatrix} = A\begin{bmatrix}v \\ w\end{bmatrix}. \tag{6}$$

A matriz A é chamada de matriz de cinemática direta, sendo importante para definir o controlador implementado, que se baseia na cinemática inversa A^{-1} em suas leis de controle (Martins et al., 2008).

Por fim, todos os parâmetros, ou são calculados, ou são construtivos, exceto a, que é escolhido. Para que haja uma matriz de cinemática inversa A^{-1}, a precisa ser diferente de 0. No projeto, a ponta do robô foi escolhida como ponto h, consequentemente, $a = 9{,}5$ cm. Outros dados construtivos do robô são: $R = 3{,}35$ cm, e $L = 19{,}7$ cm.

4. ESTIMATIVA DA ODOMETRIA

Para estimar a posição (x, y) e orientação ψ do robô, a cada Δt de aproximadamente 20 ms, o Arduino Mega captura a quantidade de pulsos gerados pelos encoders da direita N_D e da esquerda N_E. Com esses valores, é possível estimar o quanto cada roda percorreu pela aproximação linear:

$$\Delta S_D = N_D\frac{2\pi R}{N_{TPR}}, \tag{7}$$

$$\Delta S_E = N_E\frac{2\pi R}{N_{TPR}}, \tag{8}$$

dado que ΔS_D [mm] e ΔS_E [mm] medem o quanto as rodas da direita e esquerda andaram em Δt [ms], e N_{TPR} é um dado do encoder de quantos pulsos há em uma revolução completa. O deslocamento total ΔS é igual a média de ΔS_D e ΔS_E:

$$\Delta S = \frac{\Delta S_D + \Delta S_E}{2}. \tag{9}$$

O ângulo $\Delta\theta$ andado pelo robô neste período é dado por:

$$\Delta\theta = \frac{(\Delta S_D - \Delta S_E)}{L}. \tag{10}$$

Com essas quantias, é possível atualizar os valores estimados de (x, y), ψ, v_D e v_E:

$$\psi_{n+1} = \psi_n + \Delta\theta, \tag{11}$$

$$x_{n+1} = x_n + \Delta S\cos\left(\psi_n + \frac{\Delta\theta}{2}\right), \tag{12}$$

$$y_{n+1} = y_n + \Delta S\,\mathrm{sen}\left(\psi_n + \frac{\Delta\theta}{2}\right), \tag{13}$$

$$v_D = \frac{1000\Delta S_D}{\Delta t}, \tag{14}$$

$$v_E = \frac{1000\Delta S_E}{\Delta t}, \tag{15}$$

para (x_{n+1}, y_{n+1}) e ψ_{n+1} os valores atualizados da postura estimada do robô, e (x_n, y_n) e ψ_n, seus valores anteriores.

Matematicamente, a odometria serve como uma forma de estimação da postura do robô. Porém, seu prejuízo é que o erro aumenta sem limites a menos que haja um referencial independente informando periodicamente a posição do robô, reduzindo esse erro (Borestein et al., 1996).

Além disso, utilizar somente o encoder para estimar a posição e orientação de um robô, leva a valores ruidosos que pioram bastante quando o robô executa muitas rotações. Isso acontece porque pequenos erros de orientação causam um constante crescimento no erro de posição nos eixos X e Y (Borestein et al., 1996). Esse fato inclusive poderá ser conferido na Seção 6.

Por isso, o objetivo é adicionar na próxima etapa do projeto uma IMU (Unidade de Medição Inercial), para ajudar na detecção e correção rápida desses erros de postura do robô, e câmeras no espaço inteligente do laboratório. Essas câmeras servirão como referenciais fixos independentes que ajudarão nessa estimativa, além de propiciar ao sistema diversos outros tipos de serviços.

5. SISTEMA DE COMUNICAÇÃO

Para que os valores de v e w calculados pela Raspberry Pi cheguem ao Arduino Mega, e as informações do robô (x, y, ψ, nível de bateria, us_E, us_F e us_D) cheguem na Raspberry Pi, foi utilizada a comunicação Wifi, sendo intermediada pelo módulo ESP8266 12E (Figura 3).

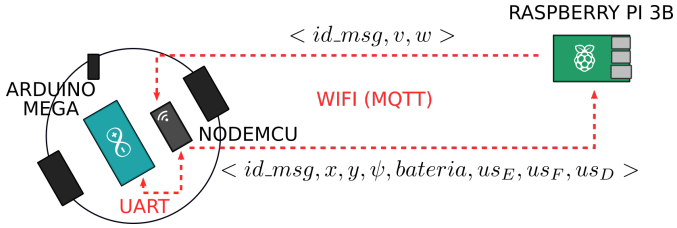

Figura 3. Esquema de comunicação entre o Arduino, o módulo Wifi e a Raspberry Pi.

Esse módulo Wifi e a Raspberry Pi se conectam em um roteador e se comunicam por um broker MQTT (Mosquitto, 2019) executado na Raspberry Pi. O protocolo MQTT de comunicação foi criado para Internet das Coisas. Ele foi desenvolvido para ser leve e simples, no modelo de publicação e inscrição. (Mosquitto, 2019) Qualquer nó conectado no broker MQTT (canal de comunicação do protocolo) pode publicar em um tópico (e.g., odometria) e se inscrever em outros (e.g., velocidades do controlador). Quando a informação é publicada em um tópico, ela fica disponível para qualquer nó inscrito naquele mesmo tópico consumir (obter) essa informação (MQTT.ORG, 2019). Esse protocolo tem se popularizado bastante atualmente, e requer baixa banda de rede.

Como ilustrado na Figura 3, a troca de dados entre o módulo Wifi e o Arduino Mega é realizada por serial UART. A biblioteca utilizada para facilitar a organização e controle de erros na serial foi a EasyTransfer (2016).

Além disso, foi implementada uma forma do robô não perder pacotes vindos do controlador devido a problemas de comunicação a nível de Wifi. Toda mensagem enviada pelo controlador possui um identificador de mensagem chamado *id_msg*, citado na Seção 2. Quando o módulo Wifi recebe uma informação da Raspberry, ele salva esse *id_msg*, e envia o mesmo na próxima publicação, no tópico de odometria. Isso permite que o controlador saiba qual foi a última mensagem que o Arduino recebeu. Caso não tenha recebido a última informação das velocidades publicadas, o controlador a envia novamente.

Caso o controlador perca comunicação completamente com o robô, o Arduino Mega possui um contador de espera de 1 s. Passado esse tempo, o robô para seus motores, se já não estiverem parados.

6. EXPERIMENTOS E RESULTADOS

Para testar e validar a plataforma, foi realizada uma tarefa *indoor* (em ambiente interno) de controle de posição final com um total de 4 pontos:

(1) o robô começa na origem $(0, 0)$ cm, orientado a 0^o;
(2) o usuário insere a posição do ponto $P1 = (200, 0)$ cm, e o robô segue para esse ponto;
(3) ao chegar dentro de um raio de convergência, o controlador para o robô e, em seguida, o usuário envia o robô para o ponto $P2 = (100, 0)$ cm, para que seja realizada uma rotação de 180^o;
(4) ao chegar dentro de um raio de convergência, o controlador pára o robô e, em seguida, o usuário envia o robô para o ponto $P3 = (100, 100)$ cm;
(5) ao chegar dentro de um raio de convergência, o controlador pára o robô e, em seguida, o usuário envia o robô para o ponto $P4 = (0, 0)$ cm;
(6) o experimento termina quando o robô chegar dentro do raio de convergência.

O raio de convergência foi definido como 5 cm.

A lei de controle utilizada segue o modelo para seguimento de trajetória proposto em (Martins et al., 2008):

$$v = l_x \tanh\left(\frac{K_x}{l_x}\tilde{x}\right)\cos(\psi) + l_y \tanh\left(\frac{K_y}{l_y}\tilde{y}\right)\text{sen}(\psi), \quad (16)$$

$$w = -\frac{1}{a}l_x \tanh\left(\frac{K_x}{l_x}\tilde{x}\right)\text{sen}(\psi) + \frac{1}{a}l_y \tanh\left(\frac{K_y}{l_y}\tilde{y}\right)\cos(\psi), \quad (17)$$

em que $l_x = l_y = 250$ mm/s, $K_x = K_y = 1$, v e w são as velocidades linear e angular de saída do controlador, $a = 9{,}5$ cm e $\tilde{x}$ e $\tilde{y}$ são calculados por:

$$\tilde{x} = x_d - x_h, \quad (18)$$

$$\tilde{y} = y_d - y_h, \quad (19)$$

sendo (x_d, y_d) o ponto desejado.

Por este controlador possuir estabilidade por Lyapunov (Martins et al., 2008) para os valores escolhidos, é possível focar na validação da plataforma.

As variáveis analisadas foram: o tempo médio que demora um ciclo de controle no robô ($\bar{t_c}$), o erro médio absoluto de estimativa de posição do robô ($\bar{e_e}$) e a distância total média percorrida no final da tarefa ($\bar{d_T}$). Para se obter um resultado menos ruidoso, a tarefa de 4 pontos foi realizada dez vezes.

Para calcular $\bar{t_c}$, o Arduino Mega enviou para a Raspberry Pi o tempo gasto para se completar um ciclo de controle

durante todo o experimento. Esses valores foram salvos e, ao final das dez repetições, foi calculada uma média simples deles.

Para calcular $\bar{e_e}$, o controlador salvou os valores estimados pelo robô nos quatro pontos, nas dez repetições (um total de 40 valores diferentes de x_h e 40 de y_h). No final do processo, esses valores foram subtraídos do valor medido pelos autores in loco, e calculado uma média simples dos módulos das dez repetições, gerando quatro valores médios absolutos de erro (um para cada ponto final da tarefa).

$\bar{d_T}$ foi calculada como a média das dez distâncias totais percorridas pelo robô da origem até o último ponto ($P4$). Cada distância total foi encontrada pelo fato de que todas as posições enviadas pelo robô foram salvas em um arquivo na Raspberry Pi durante as dez repetições da tarefa. Dessa forma, é possível estimar por uma distância euclidiana o deslocamente em cada instante.

Na Tabela 1, estão apresentados os resultados de $\bar{e_e}$ no eixo X (e_{eX}), no eixo Y (e_{eY}), e euclidiano (calculado pela distância euclidiana, $erro = \sqrt{(e_{eX}^2 + e_{eY}^2)}$), para os experimentos realizados. Para esses experimentos, $\bar{t_c}$ foi de 56,8 ms e $\bar{d_T}$ de 5,42 m.

Tabela 1. Resultados de $\bar{e_e}$ em cm no eixo X, no eixo Y e euclidiano para as dez repetições da tarefa.

Erro	$P1$	$P2$	$P3$	$P4$
X	12,1	6,2	49,7	**18,9**
Y	8,4	21,4	16,0	**54,9**
Euclidiano	14,7	22,2	52,2	**58,1**

Como era de se esperar em uma estimativa de odometria utilizando apenas a integração de velocidade lida no encoder, o erro aumenta conforme o robô anda. Entretanto, após andar em média por 5,42 m, o robô acumulou apenas 58,1 cm de erro euclidiano médio. Mesmo com a manobra de 180^o, o valor euclidiano máximo anotado no ponto $P2$ foi de 34,1 cm.

Como já comentado, na próxima etapa da pesquisa, serão adicionadas câmeras ao espaço inteligente, permitindo uma melhor estimativa por Visão Computacional. Além disso, para ajudar a detectar e corrigir os erros de orientação, será adicionada uma IMU ao robô.

Outro ponto válido no trabalho é que $\bar{t_c}$ está abaixo de 57 ms, um valor aceitável para um ciclo de controle. Controle esse que, inclusive, é realizado parte em um microcomputador remoto, não embarcado no robô.

É importante destacar que esse tempo também está atrelado ao hardware utilizado. Vale lembrar que o experimento foi realizado com o controlador sendo executado em uma Raspberry Pi 3B. Portanto, pode-se inferir que na utilização de um equipamento mais potente, como um servidor dedicado para o espaço inteligente, que não limitasse o processamento do sistema, esses valores seriam diminuídos.

É importante também ressaltar que a sintonia do controlador não é o foco neste experimento. Como comprovado em (Martins et al., 2008), por Lyapunov, o controlador utilizado é estável para os ganhos utilizados, i.e., mesmo com um caminho não ideal, ele conduzirá o robô para a posição desejada. Para este trabalho, o objetivo dos experimentos é testar a viabilidade de utilizar esse robô no espaço inteligente que está sendo desenvolvido.

Para uma melhor visualização do experimento, uma das repetições foi gravada e pode ser conferida no link: `https://youtu.be/wi2uiv8qxOs`. A Figura 4 apresenta um resumo das posturas do robô ao alcançar os quatro pontos durante o experimento gravado.

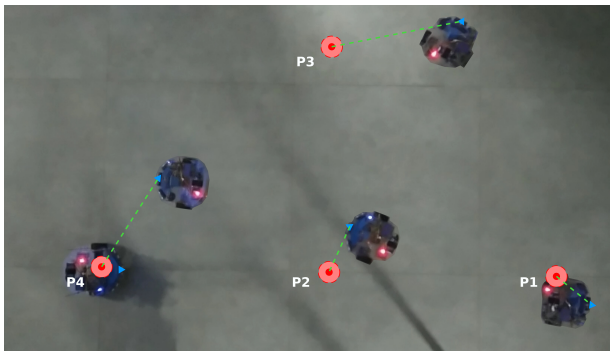

Figura 4. Resumo dos pontos alcançados pelo robô no experimento gravado.

É possível ver em vermelho os pontos desejados ($P1$, $P2$, $P3$ e $P4$) com o raio de convergência ilustrado com um círculo rosa. Os triângulos em azul representam a ponta do robô, o ponto controlado no sistema, e sua orientação. Para os quatro momentos, há uma linha tracejada em verde, destacando a distância entre a posição real do robô e o ponto desejado.

A Figura 5 apresenta o caminho percorrido pelo robô neste experimento gravado.

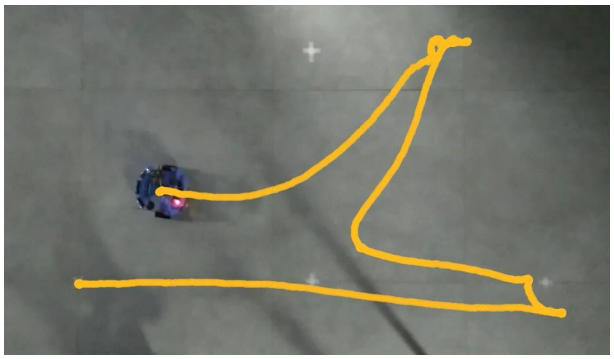

Figura 5. Caminho percorrido pelo robô para alcançar os quatro pontos.

Como é possível notar pelas Figuras 4 e 5, o robô apresentou os erros de estimativa de postura já comentados, visto que foram utilizados apenas dados de odometria. É válido ressaltar que, para o controlador e para o robô, o caminho foi realizado com sucesso, sem erros, parando sempre dentro dos círculos de convergência adotados.

7. CONCLUSÃO E TRABALHOS FUTUROS

Neste artigo foi proposta uma plataforma robótica para espaços inteligentes, o Happy Feet. A intenção é que o robô seja um agente que permita a realização de diversas tarefas neste espaço. Para isso, essa plataforma estima sua posição

e recebe comandos de velocidade via protocolo MQTT de um controlador executado remotamente, no caso, em uma Raspberry Pi.

O robô proposto foi testado em um ambiente interno e apresentou resultados satisfatórios, mesmo ao realizar manobras com variações bruscas de orientação. A estimativa de postura se mostrou aceitável por atingir um erro médio euclidiano de 58,1 cm, mesmo depois de percorrer 5,42 m.

Além disso, o tempo dos ciclos de controle foi de apenas 56,8 ms, mesmo com o controle sendo realizado remotamente. Esse valor também deve diminuir com a utilização de um computador mais potente dedicado para o espaço inteligente, que não limite o processamento do sistema.

Como esse projeto utiliza componentes eletrônicos populares e de fácil aquisição, ele é facilmente replicável e adaptável para sua inclusão em um projeto com mais agentes robóticos ou para servir de aprendizado em escolas de nível fundamental e médio. Tais resultados podem ser considerados promissores e sugerem que o robô Happy Feet proposto está adequado para ser utilizado como agente no espaço inteligente desenvolvido no IFES Campus Guarapari.

Além disso, com a adição de uma IMU ao robô e câmeras no espaço inteligente, no avançar do projeto, os erros de postura do Happy Feet serão diminuídos. As câmeras também possibilitarão diversos outros serviços e tarefas para o sistema, como: o seguimento de um alvo móvel, a detecção de pessoas e objetos, reconstrução tridimensional do local, mapeamento, etc.

Muitas dessas tarefas e serviços demandam um volume grande de poder de processamento, principalmente as que envolvem imagens. Nesse contexto, é possível identificar novamente a necessidade futura de um computador mais potente que a Raspberry Pi 3B utilizada neste trabalho.

Alguns estudos também são interessantes de serem realizados, como o desempenho do robô utilizando-se uma ESP32 no lugar do Arduino Mega e do NodeMCU, e o desenvolvimento de robôs com a estrutura física de um modelo Ackermann, que se assemelha aos carros. Isso permitiria o início da pesquisa dos agentes em cidades inteligentes, no controle de carros autônomos.

Além disso, seria interessante analisar a inserção do ROS (Sistema Operacional de Robótica) ao sistema, visto que o ROS possibilita a fácil adição e utilização de diversos algoritmos e bibliotecas voltadas para a área de robótica. Ele também permite uma comunicação semelhante ao MQTT, facilitando a troca de informações entre os agentes do sistema.

Por fim, também como trabalhos futuros, serão implementadas tarefas mais complexas que o simples controle de posição final, como: seguir uma trajetória/ alvo, voltar para a base de carregamento de bateria quando ela estiver baixa, mapear o ambiente utilizando os sensores ultrassônicos, dentre outras.

AGRADECIMENTOS

Os autores agradecem ao CNPq e ao Ifes pelo apoio e suporte financeiro.

REFERÊNCIAS

Borestein, J., Everetta, H.R., and Feng, L. (1996). Where am i, sensors and methods for mobile robot positioning. *Prepared by the University of Michigan*.

Cruz, C.D.L. and Carelli, R. (2006). Dynamic modeling and centralized formation control of mobile robots. In *IECON 2006-32nd Annual Conference on IEEE Industrial Electronics*, 3880–3885. IEEE.

EasyTransfer (2016). Easytransfer. `https://github.com/madsci1016/Arduino-EasyTransfer`.

Jones, J.L., Seiger, B.A., and Flynn, A.M. (1998). *Mobile robots: inspiration to implementation*. CRC Press.

Light, R. (2019). Paho-mqtt. `https://pypi.org/project/paho-mqtt/`.

Martins, F.N., Celeste, W.C., Carelli, R., Sarcinelli-Filho, M., and Bastos-Filho, T.F. (2008). An adaptive dynamic controller for autonomous mobile robot trajectory tracking. *Control Engineering Practice*, 16(11), 1354–1363.

Maximo, M.R.O.A. (2015). Model predictive controller for trajectory tracking by differential drive robot with actuation constraints. *Anais do XII Simpósio Brasileiro de Automaćão Inteligente (SBAI'2015)*.

Melo, R.C. (2015). Navegação autônoma de robô uniciclo: estudos de caso em uma abordagem baseada em comportamento. *Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) - Departamento de Engenharia Elétrica, Centro Tecnológico. Universidade Federal do Espírito Santo*.

Mosquitto, E. (2019). Eclipse mosquitto. `https://mosquitto.org/`.

MQTT.ORG (2019). Oasis mqtt. `http://mqtt.org/`.

Queiroz, F.M., Covre, V.B., Rampinelli, M., Pereira, F.G., Vassalo, R.F., and Bastos-Filho, T.F. (2015). Localização e Guiagem de um Robô Móvel em um Espaço Inteligente. In *Anais do XII Simpósio Brasileiro de Automaćão Inteligente (SBAI'2015)*.

Rampinelli, M., Covre, V.B., de Queiroz, F.M., Vassallo, R.F., Bastos-Filho, T.F., and Mazo, M. (2014). An intelligent space for mobile robot localization using a multi-camera system. *Sensors*, 14(8), 15039–15064.

Salvador, R.M., de Assis Silva Vitor Henrique de Moraes Escalfoni, L., , and Vassallo, R.F. (2019). A quadrotor flight mode for tracking based on computer vision using ros. *Anais do XIV Simpósio Brasileiro de Automação Inteligente (SBAI'2019)*.

Salvador, R.M., de Moraes Escalfoni, V.H., de Assis Silva, L., and Vassallo, R.F. (2017). Módulo de voo para quadrimotor baseado em visão computacional e ros para seguimento de padrão móvel. *Anais do XIII Simpósio Brasileiro de Automação Inteligente (SBAI'2017)*.

Shim, J.H. and Cho, Y.I. (2015). A mobile robot localization using external surveillance cameras at indoor. *Procedia Computer Science*, 56, 502–507.

Sá, F.B., Cypriano, M.F., Queiroz, F.M., Vassalo, R.F., Pereira, F.G., and Neto, A.F. (2014). Planejamento de trajetória para um robô móvel usando imagens capturadas por um vant: conceitos e resultados preliminares. In *Anais do XX Congresso Brasileiro de Automática (CBA'2014)*.