Posicionamento Autónomo de Robot Móvel com Base na Potência de Sinal Wi-Fi

Marco Laia[1], Vasco Gaspar[1], João Caldeira[2], Vasco Soares[2]

[1] Instituto Politécnico de Castelo Branco, Portugal
{mlaia,vgaspar}@ipcbcampus.pt
[2] Instituto Politécnico de Castelo Branco, Instituto de Telecomunicações, Portugal
{jcaldeira,vasco.g.soares}@ipcb.pt

Resumo. O trabalho apresentado no contexto deste artigo pretende contribuir para a construção de uma rede sem fios composta por vários pontos de acesso móveis com a capacidade de se organizarem autonomamente de modo a cobrirem uma área definida. Nesse sentido, apresenta o desenho, implementação e avaliação de desempenho de um algoritmo que coordena a deslocação autónoma de um robô até um nó de destino, com base na avaliação do parâmetro Received Signal Strength Indication.

Palavras Chave: IoT, Redes de Sensores Móveis, Organização Autónoma.

1 Introdução

Nos dias de hoje, a conexão à Internet por parte de qualquer objeto físico está a aumentar em larga escala, o que tem dado enfâse à temática da Internet das Coisas (IoT). A IoT permite que uma rede com vários objetos interligados, não só recolha informações desse ambiente a partir dos seus sensores e realize interações com o mundo físico em termos de ações, comandos ou controlos, mas também utilize normas da Internet para fornecer serviços de, por exemplo, transferência de dados e comunicações. A introdução de IoT em robôs permite alargar a possibilidade de aumentar o número de projetos na área da robótica, com o objetivo de dar suporte a outras áreas da vida quotidiana, tais como na indústria, medicina, segurança, assistência pessoal e entretenimento.

Os robôs são usados para desempenhar tarefas autonomamente, que são projetadas recorrendo a uma linguagem de programação, que antes só podiam ser executadas recorrendo a ação humana. A construção e programação de robôs é possível devido à existência de vários tipos de microcontroladores, sendo um dos tipos mais utilizados o Arduino [1], uma plataforma eletrónica de código *open-source* baseada em hardware e software simples. A associação destas plataformas a módulos de comunicação Wi-Fi, possibilitam a obtenção de um indicador da intensidade do sinal da conexão entre dispositivos, denominada de *Received Signal Strength Indication* (RSSI). Este indicador é importante porque a partir deste é possível estimar a distância a que os dispositivos se encontram uns dos outros numa *Wireless Sensor Network* (WSN). Quanto

maior é o valor do RSSI, mais forte é o sinal de rádio e mais próximo estão os dispositivos. Porém, usando o RSSI não é possível obter a localização exata de um dispositivo, visto que o sinal transmitido é bastante instável devido a vários fatores, tais como os obstáculos que possam estar entre um transmissor e um recetor, a perda de trajetória (*path loss*) e o desvio da atenuação (*fading)*.

Neste artigo, apresenta-se o desenho, implementação e avaliação do desempenho de um algoritmo que coordena a deslocação autónoma de um robô até ao nó destino com base na avaliação do parâmetro RSSI. Este artigo encontra-se estruturado da seguinte forma. A Secção 2 apresenta o estado da arte. De seguida, a Secção 3 descreve a proposta de um algoritmo que coordena a deslocação autónoma de um robô até um nó fixo com base no RSSI. Posteriormente na Secção 4 é avaliado o desempenho deste algoritmo, discutindo-se os resultados obtidos. Finalmente, na Secção 5, apresentam-se as conclusões.

2 Estado da Arte

A utilização do parâmetro RSSI conjugada com sensores de suporte ao movimento, possibilita a realização de movimentos mais complexos e mais precisos, pois permite um maior controlo do robô. Num trabalho de investigação realizado por D. Reiser et al [2], o objetivo era avaliar a utilização de robôs para a aquisição de dados para sensores de wireless em localizações separadas em ambientes de agricultura. Através da utilização de sensores laser é possível fazer com que o dispositivo detete e se desvie de obstáculos que se encontrem no seu percurso, facilitando assim a sua deslocação. Ao longo da resolução do percurso vão sendo mapeadas as posições ideais para os nós, permitindo assim a criação de uma rota mais eficiente.

Nosaiba A. Sabto e Khalid Al Mutib [3] propuseram uma estratégia que consistia na utilização do RSSI com Radio Frequency Identification (RFID) e Back-Propagation Artificial Neural Network (BPANN), com o objetivo de permitir a deslocação autónoma de um robô sabendo a sua localização e a informação da sua orientação. O trabalho de Cory Q. Nguyen et al [4], teve como objetivo criar uma prova de conceito que permitisse às unidades de robôs móveis a capacidade de providenciar uma rede mesh wireless que fornece um serviço sem fios a utilizadores e também demonstra a capacidade de aumentar a taxa de transferência dessa rede mesh, reduzindo de forma autónoma a contagem de saltos necessária para o tráfego da rede. Este trabalho recorre à tecnologia de radiofrequência (RF).

Num trabalho de Joydeep Biswas e Manuela Veloso [5], o objetivo era a criação de um algoritmo que permitisse a localização e a navegação de um robô autonómo através da utilização de dados de odometria, recorrendo ao uso da tecnologia Wi-Fi. O trabalho de Eisuke Nakata et al [6] teve como objetivo a criação de um Access Point móvel (AP) que procura constantemente a melhor posição para conseguir fornecer o melhor sinal possível para todos os dispositivos ligados a este, recorrendo ao uso do *Wi-Fi*. Num trabalho de Nikolaus Correll et al [7] foi proposto um algoritmo que permite a implementação e manutenção de um backbone de comunicação móvel que fornece uma ligação *Wi-Fi*.

A abordagem apresentada neste artigo considera apenas o parâmetro RSSI, para coordenar o deslocamento autónomo de um robot móvel. Face às limitações que potencialmente daí advêm, por exemplo se o sinal estiver instável, facilmente pode transmitir valores errados, fazendo com que o dispositivo pense que o seu objetivo se encontra a uma distância mais curta quando, na realidade, se encontra mais longe. Este valor também não dá indicação em que direção é que o nó destino se encontra [8]. Assim, adotou-se uma abordagem de deslocação por tentativa e erro, que se descreve nas próximas secções.

3 Protótipo

Esta secção apresenta o desenho e implementação do protótipo de robot autónomo, que procura um nó de destino numa rede mesh com base no parâmetro RSSI. Esta secção encontra-se dividida em duas subsecções, na primeira é apresentado o *hardware* envolvido na construção do protótipo e na segunda é descrita a implementação de *software* do protótipo.

3.1 Componente de *hardware*

O protótipo é composto por 7 componentes essenciais, representados na Fig. 1. A alimentação de energia do robô é feita por uma bateria (1), neste caso, a partir de um *powerbank*, de forma a ser possível alimentar a ponte H (3). Esta ponte é um circuito elétrico que permite fornecer corrente elétrica em duas direções, em que cada direção do fluxo elétrico corresponde a uma direção de rotação dos motores. Existe esta necessidade devido à utilização da rotação dos motores em ambos os sentidos para movimentos diferentes. Os cabos de ligação da bateria (4) e o cabo de ligação dos motores (5) estão conetados na ponte H, na qual está integrada uma placa Arduino *NodeMCU 1.0* (2) que recebe o código para posteriormente ser executado. Os motores (direito (6) e esquerdo (7)) são utilizados para possibilitar o deslocamento do robô até ao nó destino. Estes motores são independentes, e para permitir que sejam realizados vários movimentos são usadas diferentes velocidades e tempos de ativação.

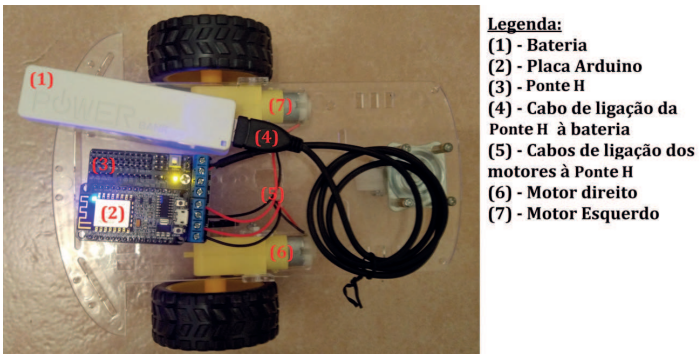

Fig. 1. Componentes do robô.

3.2 Componente de *software*

De modo a conseguir estabelecer a comunicação entre o robô e o nó de destino, foi utilizada a biblioteca *painlessMesh* [9] pois esta permite a criação de uma rede mesh Wi-Fi utilizando o módulo ESP8266. Devido à utilização desta biblioteca, não existe a necessidade de estar constantemente a vigiar e a gerir a rede. É através desta ligação que o robô obtém o valor do RSSI correspondente à ligação estabelecida.

Inicialmente, tentou-se descobrir as condições a utilizar nos movimentos para a frente, trás, esquerda e direita. A determinação da condição do andamento para a frente e para trás foi alcançada, sendo esta a verificação do valor do RSSI obtido, em que se o valor fosse inferior ao valor inferior pré-definido (*rssiInnerThreshold*), o robô iria afastar-se do nó destino. Por outro lado, se o valor fosse superior ao valor superior pré-definido (*rssiOuterThreshold*) o robô iria aproximar-se do nó destino. No entanto, ao seguir esta abordagem, foi encontrado o problema da identificação da condição para determinação dos movimentos para a esquerda ou para a direita. Por este motivo, foi decidido abordar o problema utilizando um método de tentativa e erro. Assim, nesta nova abordagem, foi decidido que o robô iria efetuar oito movimentos teste. Para tal, é obtido o valor do RSSI na posição atual. Caso o valor do RSSI obtido esteja dentro dos valores limites, inferior (*rssiInnerThreshold*) e superior (*rssiOuterThreshold*), pré-definidos, o robô não realiza qualquer movimento. Caso o valor do RSSI obtido esteja fora dos limites pré-definidos é iniciada a sequência dos oito movimentos teste. Cada um dos movimentos teste corresponde a um movimento que o robô consegue realizar. Entre cada movimento teste, o robô retorna sempre à posição inicial. Após o final de cada movimento teste é registado o valor do RSSI, entre o robô e o nó destino, num *array*. Após todos os movimentos teste terem sido realizados, é executada uma rotina, pelo controlador do robô, que percorre todo o *array* para identificar o índice onde se encontra o melhor valor de RSSI. No caso de existirem valores iguais, é escolhido sempre o que tiver o índice inferior, ou seja, o primeiro a aparecer no *array*. Após o melhor valor de RSSI ser encontrado, o robô efetua o movimento correspondente, identificado pelo índice do *array*. De seguida, o robô define como posição inicial a posição final da decisão do movimento anterior e realiza novamente o teste sobre o valor do RSSI. Em caso de necessidade (valor do RSSI fora dos limites pré-definidos), o robô realiza os oito movimentos teste para decisão do próximo movimento a realizar. Assim, descreve-se de seguida o algoritmo adotado.

```
Algoritmo
  Se RSSI atual < rssiOuterThreshold então
       Efetua os oito testes de movimento registando o valor do RSSI
       no novo local e retorna à posição inicial entre cada teste
       Encontra o índice onde se encontra o melhor valor do RSSI
          Seleciona o movimento a efetuar de acordo com o índice
          encontrado no ponto anterior

  Se RSSI atual se encontrar entre o rssiOuterThreshold e o rssiIn-
  nerThreshold então
     Permanece no mesmo local

  Se RSSI atual > rssiInnerThreshold então
     Afasta-se do nó destino
```

O problema que este projeto visa resolver consiste num robô autónomo conseguir alcançar o seu nó destino utilizando a menor quantidade de recursos possíveis. No entanto, devido à ausência de sensores no robô, o mesmo só tem o valor do RSSI para se guiar. Este facto faz com que existam dificuldades acrescidas na deslocação do robô, uma vez que este não sabe a sua posição nem a do nó destino, assim como também não sabe a distância até ao destino, sabendo apenas a intensidade do sinal. Uma vez que as placas Arduíno não suportam *Multithreading*, para se poder efetuar movimentos mais complexos é necessário recorrer à utilização da disparidade da velocidade dos dois motores do robô. Isto permite a execução de movimentos mais complexos. A ausência de *Multithreading* causa problemas na programação de funções em simultâneo, em particular na necessidade de obtenção do valor do RSSI ao mesmo tempo que o robô se desloca de modo a permitir um funcionamento fluído. Para além disto, também cria problemas na programação de movimentos que necessitem de mais do que uma função de movimento. Para a resolução destes problemas, foi utilizado um escalonador que permite através da especificação de intervalos temporais escalonar tarefas de forma sequencial, permitindo "simular" processamento paralelo, neste caso, por exemplo, para obtenção do valor do RSSI e para controlar o movimento do robô. À passagem dos intervalos temporais especificados no escalonador, este despoleta as funções que se lhe encontram associadas. Os intervalos temporais de escalonamento podem ser alterados de modo a permitir um melhor desempenho e compatibilidade entre as diferentes funções existentes no programa. No caso da solução desenvolvida foram criadas três funções escalonadas periodicamente – *ConnectionTaskScheduler, RSSITaskScheduler* e *MovingRobotTaskScheduler*.

A função *ConnectionTaskScheduler* permite realizar um *reset* ao controlador caso este fique inativo por um determinado período de tempo pré-definido. Esta função garante que se por alguma razão a execução do *firmware* do controlador bloquear em algum ponto, por mais do que o período de tempo pré-definido, o controlador é reiniciado. Esta funcionalidade traz robustez ao algoritmo pois evita que este possa deixar de responder devido a situações não controladas. A função *RSSITaskScheduler* é responsável por monitorizar o valor do RSSI usado nas comunicações e desencadear os movimentos do robô dependendo do valor RSSI obtido. A função *MovingRobotTaskScheduler* é utilizada para controlar os movimentos, realizados pelo robô. Devido às limitações do controlador usado, os movimentos do robô têm de ser controlados através do controlo da passagem do tempo após ativação adequada dos motores. Ou seja, os movimentos são conseguidos ativando os motores por um período de tempo.

Para controlar a passagem do tempo, não podemos bloquear o controlador à espera que esse tempo se esgote, devido às outras ações que, entretanto, necessitam ser executadas. Assim, esta função é escalonada em intervalos de tempo curtos, para controlar a passagem do tempo quando os motores são ativados para realizar determinado movimento. Caso o tempo de movimento tenha sido atingido, numa das execuções desta função, é dada ordem aos motores para parar, terminando assim o movimento que estava em execução. Todos os movimentos realizados pelo robô são controlados pela ativação dos motores no sentido correto e pela passagem do tempo, nomeadamente, movimento para a frente, para trás, para a direita e para a esquerda. Assim, os

valores temporais usados para definir o momento de desligar os motores vão determinar as distâncias percorridas para a frente e para trás e no caso das viragens os ângulos das mesmas.

4 Avaliação de Desempenho

Nesta secção são apresentados dois testes que foram executados para avaliação de desempenho do algoritmo proposto neste artigo. Para este efeito, foi construída uma *testbed* composta por um nó de destino fixo e um robot móvel. Os parâmetros usados nos testes e os seus respetivos valores, são apresentados na Tabela 1. Nos dois testes realizados, apenas a distância entre o robô e nó destino varia.

Tabela 1. Parâmetros usados nos testes.

Parâmetros usados	**Valores Usados**
Zona de RSSI Válido	Entre -30 (rssiInnerThreshold) e -40 (rssiOuterThreshold)
Valor temporal *ConnectionTaskScheduler*	10 segundos
Valor Temporal *RSSITaskScheduler*	1,5 segundos
Valor Temporal *MovingRobotTaskScheduler*	0,5 segundos

4.1 Testes

Teste 1

Foi realizado um primeiro teste, ilustrado na Fig. 2, assumindo uma distância de 1,50m entre o robô e o nó de destino.

Fig. 2. Distância entre nó e robô (Teste 1).

Após a primeira iteração, os valores de RSSI foram guardados num *array*. Os valores, correspondentes a cada movimento feito pelo robô, estão representados na Fig. 3.

```
15:14:07.633 -> Escolha do movimento com o melhor valor!
15:14:07.633 -> -49 (0)
15:14:07.669 -> -55 (1)
15:14:07.669 -> -49 (2)
15:14:07.669 -> -55 (3)
15:14:07.669 -> -61 (4)
15:14:07.669 -> -63 (5)
15:14:07.669 -> -68 (6)
15:14:07.669 -> -64 (7)
15:14:07.707 -> Case 0 - Movement Forward
```

Legenda:
(0) - Movement Forward
(1) - Mov. Right Forward
(2) - Turn Right
(3) - Mov. Right Backward
(4) - Movement Backward
(5) - Mov. Left Backward
(6) - Turn Left
(7) - Mov. Left Backward

Fig. 3. Valores de RSSI no array, após a 1ª iteração.

A partir destes valores, o algoritmo escolheu a ação *Movement Forward*. Este movimento corresponde ao índice 0, onde se encontra o valor -49. Note-se que este valor foi obtido em dois movimentos, *Movement Forward* e *Turn Right*. No caso de existirem valores iguais, o algoritmo escolhe sempre o primeiro valor.

Após ser escolhido o melhor valor de RSSI, o robô efetua o movimento correspondente a esse valor e começa uma segunda iteração na qual foram obtidos os seguintes valores: -41, -47, -41, -47, -52, -42, -35 (ver Fig. 4, 2ª iteração).

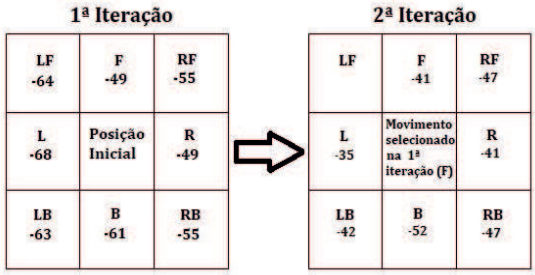

Fig. 4. Iterações (Teste 1).

O motivo pelo qual não se realizaram todos os movimentos foi o facto de o valor -35 ter sido obtido durante o teste. Como este valor se encontra dentro dos valores limites do *rssiInnerThreshold* (-30) e *rssiOuterThreshold* (-40), não houve necessidade de finalizar os restantes testes, significando que o robô está perto do nó destino.

Neste teste, verificou-se que a deslocação do robô até ao nó destino demorou 2 minutos e 45 segundos.

Teste 2

No segundo teste, ilustrado na Fig. 5, assumiu-se uma distância de 3 metros entre o robô e o nó de destino.

Fig. 5. Distância entre nó e robô (Teste 2).

Após a primeira iteração, os valores de RSSI foram guardados num *array*. Os valores, correspondentes a cada movimento feito pelo robô, estão representados na Fig. 6.

```
15:33:33.034 -> Escolha do movimento com o melhor valor!
15:33:33.034 -> -55 (0)
15:33:33.034 -> -67 (1)
15:33:33.034 -> -54 (2)
15:33:33.034 -> -58 (3)
15:33:33.034 -> -56 (4)
15:33:33.034 -> -54 (5)
15:33:33.034 -> -54 (6)
15:33:33.034 -> -56 (7)
15:33:33.034 -> Case 2 - Turn Right
```

Legenda:
(0) - Movement Forward
(1) - Mov. Right Forward
(2) - Turn Right
(3) - Mov. Right Backward
(4) - Movement Backward
(5) - Mov. Left Backward
(6) - Turn Left
(7) - Mov. Left Backward

Fig. 6. Valores de RSSI no *array*, após a 1ª iteração (Teste 2).

A partir destes valores, o algoritmo escolheu a ação *Turn Right*. Este movimento corresponde ao índice 2, onde se encontra o valor -54. Note-se que este valor foi obtido em três movimentos, *Turn Right*, Movement Left Backward e *Turn Left*. Como mencionado anteriormente, no caso de existirem valores iguais, o algoritmo escolhe sempre o primeiro valor.

Na Fig. 7, estão representadas as 4 primeiras iterações deste teste. Até à 3ª iteração, o robô obtém sempre valores para cada movimento e é sempre feita a escolha do movimento que tem o melhor valor de RSSI. Na 4ª iteração, o robô obtém o valor -53 na posição inicial que, comparado com os restantes valores obtidos nos outros movimentos, é o melhor valor de acordo com o algoritmo criado. Isto sucede-se quando o valor da posição inicial é melhor que os restantes valores obtidos em cada movimento efetuado. Neste caso, o valor da posição inicial é escolhido e o robô permanece no mesmo sítio. De seguida, o robô recomeça os testes.

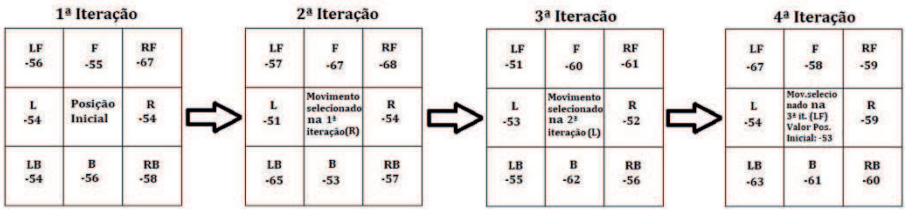

Fig. 7. 1ª, 2ª, 3ª e 4ª iterações (Teste 2).

Na Fig. 8, estão representados os valores de RSSI obtidos em todos os movimentos da 5ª até à 8ª iteração. Na 8ª iteração repete-se a mesma situação ocorrida na 4ª iteração. O valor obtido na posição inicial é melhor que os valores obtidos em todos os movimentos efetuados pelo robô, logo, este permanece no mesmo local até que seja iniciada outra iteração.

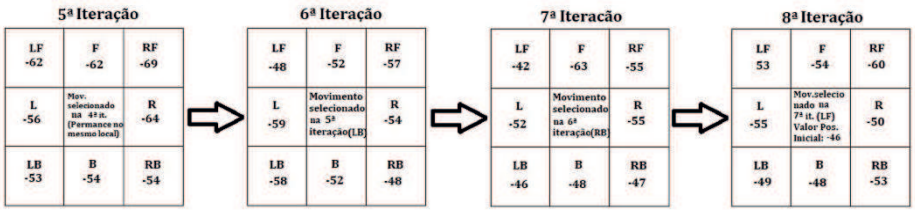

Fig. 8. 5ª, 6ª, 7ª e 8ª iterações (Teste 2).

Depois de feita a 8ª iteração, o robô faz mais 3 iterações, representadas pela Fig. 9, sendo que a última não efetua os testes todos devido a ter obtido um valor de RSSI dentro dos limites *rssiInnerThreshold* (-30) e *rssiOuterThreshold* (-40).

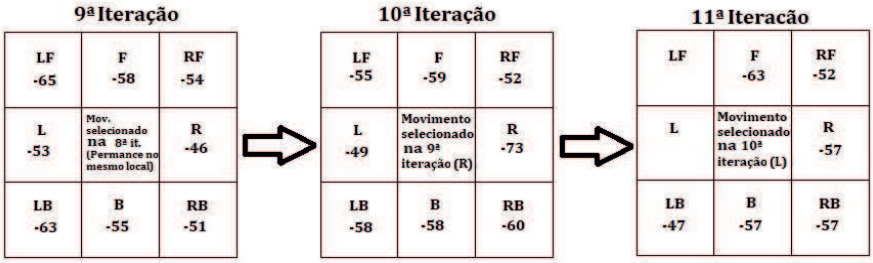

Fig. 9. 9ª, 10ª e 11ª Iterações (Teste 2).

Neste teste, verificou-se que a deslocação do robô até ao nó destino demorou 13 minutos e 47 segundos.

4.2 Discussão dos Resultados

A solução proposta cumpriu o objetivo estabelecido: fazer o robô mover-se de forma a conseguir encontrar o ponto (nó destino). Porém, esta solução apresenta algumas limitações ao nível de eficiência: dado que o robô realiza muitos movimentos, isto provoca uma diminuição na eficiência do algoritmo de procura do nó; no entanto, existe um ganho de precisão nos movimentos e na deslocação do robô até ao nó destino. A eficiência do algoritmo também é posta em causa dado o tempo que o robô demora até chegar ao nó destino, isto porque o valor do RSSI transmitido é inconstante e por serem vários os movimentos a efetuar pelo robô.

Essencialmente, o valor do RSSI obtido pelo robô tem impacto em todas tarefas que o robô tem de executar. Como já foi referido anteriormente, a instabilidade deste valor pode prejudicar a decisão do robô no movimento a ser feito e na deslocação do robô até ao nó.

Por vezes, o robô faz movimentos diferentes dos movimentos programados prejudicando assim a procura pelo nó destino. Não foi encontrado nenhum motivo para a ocorrência deste problema, mas suspeita-se que isto ocorra devido a sincronias nos tempos de escalonamento dos movimentos a efetuar pelo robô e na obtenção dos valores do RSSI. Outro motivo poderá ser a quantidade de movimentos existentes (oito), que podem dificultar a sincronização entre estes e as ações (obtenção do valor do RSSI) que o robô terá que fazer.

5 Conclusões

O artigo apresentado resulta de um estudo para proposta e avaliação do desempenho de um algoritmo que coordene a deslocação autónoma de um robô até um nó destino, conhecendo apenas o parâmetro RSSI da comunicação entre ambos.

Foram realizados testes de validação ao algoritmo proposto para avaliar a eficiência e rapidez. Os testes realizados permitem concluir sobre algumas limitações desta abordagem, tais como a inconsistência dos valores de RSSI obtidos, o robô efetuar movimentos diferentes dos programados e principalmente o tempo que o robô requer na definição do caminho até ao nó destino.

Para trabalho futuro, pretende-se solucionar os problemas identificados, otimizando o algoritmo, e testar esta proposta em ambientes com múltiplos robôs e nós de destino.

Agradecimentos

Os autores expressam o seu agradecimento à empresa InspiringSci, Lda pelo interesse e contribuição, determinantes para a concretização deste trabalho.

Referências

[1] "Arduino." [Online]. Available: https://www.arduino.cc/. [Accessed: 05-Jul-2019].

[2] D. Reiser, D. S. Paraforos, M. T. Khan, H. W. Griepentrog, and M. Vázquez-Arellano, "Autonomous field navigation, data acquisition and node location in wireless sensor networks," *Precis. Agric.*, vol. 18, no. 3, pp. 279–292, 2017.

[3] N. A. Sabto and K. Al Mutib, "Autonomous mobile robot localization based on RSSI measurements using an RFID sensor and neural network BPANN," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 25, no. 2, pp. 137–143, 2013.

[4] C. Q. Nguyen, B. C. Min, E. T. Matson, A. H. Smith, J. E. Dietz, and D. Kim, "Using mobile robots to establish mobile wireless mesh networks and increase network throughput," *Int. J. Distrib. Sens. Networks*, vol. 2012, 2012.

[5] J. Biswas and M. Veloso, "WiFi localization and navigation for autonomous indoor mobile robots," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 4379–4384, 2010.

[6] E. Nakata, T. Ebihara, and K. Mizutani, "Mobile robotic access point for transitional optimization of wireless access point positioning," *2014 IEEE 3rd Glob. Conf. Consum. Electron. GCCE 2014*, pp. 439–441, 2014.

[7] N. Correll, J. Bachrach, D. Vickery, and D. Rus, "Ad-hoc wireless network coverage with networked robots that cannot localize," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 3878–3885, 2009.

[8] K. Heurtefeux and F. Valois, "Is RSSI a good choice for localization in wireless sensor network?," *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, no. March, pp. 732–739, 2012.

[9] "GitHub - gmag11/painlessMesh." [Online]. Available: https://github.com/gmag11/painlessMesh?fbclid=IwAR208SvoRHyYLnLMgoB9XfnLNPwiLy2cTQWF5D3BGOhs4vQp6ZIxu7rYaC4. [Accessed: 01-Jun-2019].