UTILIZAÇÃO DO FORMALISMO PNP NA MODELAGEM COMPORTAMENTAL DE UM ROBÔ AUTÔNOMO PARA INSPEÇÃO DE TUBULAÇÕES

FRANCISCO E. SOUSA, CECÍLIA F. SILVA, CLAUIRTON A. SIEBRA

Laboratório de Sistemas Digitais, Centro de Informática, Universidade Federal da Paraíba
Centro de Informática, Rua dos Escoteiros, s/n, CEP. 58058-600, João Pessoa – PB – Brasil
{franciscoerberto,cecilia.flavia}@eng.ci.ufpb.br; clauirton@ci.ufpb.br

Abstract— The identification of problems, such as leaks, and the verification of the structural state of pipeline systems present several problems to the public administration of any city. Some works have proposed the use of robots as resource to carry out such tasks. However, these robots are controlled by human operators and their control is performed via communication cables. This means, robots do not usually present any level of autonomy. This approach brings up some operational problems due mainly to the use of cables and the total dependency associated with the human perception on problems in the pipe. Our work presents a behavioural model of an autonomous robot for pipelines inspections. This model is specified by means of the PNP (Petri Net Plans) formalism, which presents several advantages to our modelling, such as expressiveness and possibility to formal analysis of plans via Petri Net standard tools. Our experience in the use of PNP showed that this formalism is very intuitive, so that it facilitates the design of autonomous systems. Furthermore, the modelling of multi-robots systems is carried out in a natural way by means of special operators and this is an important resource for future extensions of this project.

Keywords— Planning description language, Autonomous robotics, Petri Net based plans.

Resumo— A identificação de problemas, como vazamentos, e a verificação do estado estrutural de um sistema de tubulações apresentam uma série de problemas para a administração pública de qualquer cidade. Alguns trabalhos apresentam a utilização de robôs como facilitadores de tais tarefas. Porém, estes robôs são controlados por operadores humanos e a interação é feita via cabos de comunicação. Ou seja, o robô não apresenta qualquer grau de autonomia. Essa abordagem trás alguns problemas operacionais, principalmente relacionados ao uso do cabo e a total dependência da percepção humana em relação aos possíveis problemas da tubulação. Nosso trabalho apresenta um modelo comportamental de um robô autônomo para a inspeção de tubulações. Este modelo é especificado através do formalismo PNP (*Petri Net Plans*), o qual apresenta diversas vantagens para a modelagem como expressividade e possibilidade de análises formais do plano por meio de ferramentas de Petri Net padrões. A experiência com o uso de PNP mostrou que o seu formalismo é bastante intuitivo, o que facilita o projeto de sistemas autônomos. Além disso, a modelagem de sistemas multirobôs é feita de forma natural via operadores especiais e este é um importante recurso para as extensões futuras deste projeto.

Palavras-chave— Linguagem para descrição de planos, Robótica autônoma, Planos baseados em Redes de Petri.

1 Introdução

A detecção de problemas em tubulações de fluidos, como água e esgoto, é uma tarefa de difícil realização devido a tais tubulações estarem sob uma densa camada sólida como pavimentos ou calçadas (Hunaidi et al., 2006). Desta forma, o problema só é detectado quando o mesmo já está em um estágio avançado. Ou seja, o fluido rompe tal camada e se espalha pela superfície. Enquanto o rompimento for parcial, o problema fica escondido por longos períodos, causando contaminação ou perda no volume útil do fluido. Esse é um problema sério em regiões que apresentam carências, como as cidades do interior de alguns estados que sofrem com a falta de água.

O grande problema é que não existem formas eficientes de se realizar um monitoramento do estado da tubulação (Chis e Saguna, 2007). De fato, um monitoramento periódico poderia identificar, por exemplo, rachaduras em tubulações de água ou entupimentos em um estágio inicial. Geralmente a perda de fluido entre duas seções de uma tubulação pode ser identificada pela perda de pressão. Mesmo sabendo qual é a seção do tubo com perda de pressão, o problema ainda é identificar onde está o ponto de evasão do líquido, caso ele não suba à superfície. Desta forma, seriam necessárias diversas escavações ao longo do percurso da tubulação.

Governos locais já estão à procura de soluções relacionadas à prevenção e monitoramento do estado das tubulações, uma vez que o investimento aplicado tende a ser menor do que a correção de problemas depois que os mesmos ocorrem. Neste cenário, a robótica pode ter um papel importante no monitoramento e identificação de problemas em tubulações (Tur e Garthwaite, 2010), uma vez que robôs especialmente especificados para este tipo de ambiente podem alcançar lugares impossíveis de serem observados por humanos sem um grande esforço (e.g. escavações). De fato, o custo do uso de robôs é extremamente baixo e a tecnologia atualmente disponível permite a especificação de soluções que vão desde a simples filmagem interna, até o completo mapeamento 3D da tubulação.

A limitação atual das abordagens que se utilizam de robôs é a falta de autonomia. Ou seja, os robôs são operados por humanos e os comandos são trocados através de uma interface a cabo. Este cabo ocasiona problemas operacionais e pode, inclusive, derrubar o

robô caso ele fique preso em algum obstáculo. Sem o uso do cabo, é necessário que o robô apresente um comportamento autônomo para que ele continue operacional em casos de falha no enlace sem fio. A autonomia também é importante para que o robô não fique tão dependente da percepção do controlador humano e também da sua perícia em controlar o robô em um ambiente hostil.

Este artigo apresenta a modelagem de um robô autônomo através do formalismo PNP (*Petri Nets Plan*). A principal vantagem deste formalismo é ser de simples uso e entendimento, mesmo sendo bastante expressivo para tratar, por exemplo, de ações concorrentes. Neste contexto, o restante do artigo está estruturado da seguinte forma. A seção 2 apresenta o estado da arte dos robôs de inspeção de tubulações, reforçando a importância do desenvolvimento de robôs autônomos. A seção 3 introduz os principais conceitos do formalismo PNP. A seção 4 apresenta a proposta de modelagem para um robô autônomo, o qual é baseado em diversos sensores para a identificação de rachaduras nas tubulações. A seção 5 faz uma análise do PNP gerado e o trabalho atual de transição para o protótipo físico. Finalmente, a seção 6 discute as conclusões deste trabalho e as pesquisas futuras.

2 Revisão da Literatura

A literatura em robótica discute alguns exemplos de robôs voltados à inspeção de tubulações. Uma revisão sobre tais trabalhos é discutida em (Tur and Garthwaite, 2010). Algumas conclusões podem ser tiradas da leitura de tais trabalhos. Primeiro, nenhum dos robôs foi projetado para atuar em tubulações em operação, uma vez que tal ambiente torna o projeto do robô bastante complexo. Existem robôs que atuam em tubulações de gás e, neste contexto, o sistema continua em operação pois o mesmo não trás grandes dificuldades para o controle e movimentação do robô. Um segundo aspecto interessante é que a maioria dos robôs utiliza rodas como forma de tração dentro das tubulações. Porém, a utilização das rodas é feita de maneira bem diversificada (Figura 1) com o objetivo do robô não perder sua estabilidade e virar dentro da tubulação (Mateos et al, 2012). Além das rodas, robôs com esteira também se apresentam como uma solução.

Figura 1. Exemplos de robôs de inspeções para tubulações

Existem também projetos mais elaborados, como os dois robôs à esquerda na Figura 1, os quais apresentam uma grande estabilidade. Contudo eles necessitam de mecanismos mais elaborados para manter a aderência. Deste modo, apesar de ganharem em estabilidade, perdem em simplicidade e robustez. A proposta apresentada neste trabalho tende a seguir uma abordagem similar à mostrada no canto superior direito.

Terceiro, pouquíssima atenção é dada a robôs com autonomia e que utilizam comunicação wireless (Song et al., 2016). Geralmente os robôs são controlados por operadores humanos e possuem comunicação via cabos. Para propósitos de praticidade e redução de custos, a questão da autonomia e conexão sem fio é importante (Chatzigeorgiou, 2010). Por exemplo, o uso de fios dificulta o movimento dos robôs e pode até mesmo vira-los dentro da tubulação caso fiquem presos em obstáculos. Já a autonomia permite que, mesmo sem comunicação, o robô possa realizar suas atribuições de forma independente.

A especificação de um comportamento autônomo para robôs é facilitada pela escolha da linguagem de modelagem. As opções atualmente disponíveis podem ser classificadas em dois grupos. O primeiro grupo concentra as linguagens baseadas em *Máquinas de Estados Finitos* (Gat, 1997), as quais são bastante intuitivas mas não possuem expressividade para a modelagem de ações concorrentes. Diferentemente, o segundo grupo, o qual se baseia nas Redes de Petri, permitem a representação de sistemas concorrentes e compartilhamento de recursos. Ou seja, elas são mais expressivas que as linguagens do primeiro grupo. Exemplos destas abordagens são apresentados em (Celaya et al., 2007; Costelha and Lima, 2007; Kuo e Chin, 2006).

PNP é uma linguagem que faz parte deste segundo grupo. Porém, em contraste com todos estes outros exemplos, PNP fornece um framework geral para descrição de comportamentos para robôs e sistemas multirobôs, melhor do que soluções específicas e direcionadas a problemas pontuais. Outro aspecto importante é que, mesmo sendo mais expressiva, PNP continua sendo intuitiva devido à caracterização explícita de estruturas atômicas as quais são diretamente interpretadas como ações e operadores para combinação de ações. Os principais conceitos de PNP são apresentados na próxima seção.

3 A Linguagem PNP

A linguagem PNP (Ziparo et al., 2011) define um plano como uma Rede de Petri representada pela seguinte 5-tupla <P,T,F,W,Mo> onde

- P = $\{p_1,p_2,p_3,\ldots,p_m\}$ é um conjunto finito de posições, os quais representam o estado de execução do robô. Cada posição possui um *token* que pode ser entendido como uma li-

nha de execução (*thread*) associada a uma ação;

- T = $\{t_1,t_2,t_3,\ldots,t_n\}$ é um conjunto finito de transições, as quais modelam as condições de início T^I e término T^T de uma ação, além de outras operações de controle T^C;
- $F \subseteq (P \times T) \cup (T \times P)$ = conjunto de arcos ou relações de fluxo;
- $W: F \rightarrow \{1\}$ é a função peso dos arcos. Em PNP todos os arcos tem peso 1;
- $M_o: P \rightarrow \{0,1\}$ = é o estado inicial onde cada posição p_i terá um valor binário 0 ou 1. Quando p_i for 1, então tal posição está ativa.

As ações e operadores da linguagem são baseados nesta conceituação. Ações representam comportamentos primitivos dos robôs e são conceitos atômicos com os quais planos mais complexos são construídos. Dois tipos de ações são considerados na linguagem (Figura 2): ação ordinária e ação de percepção.

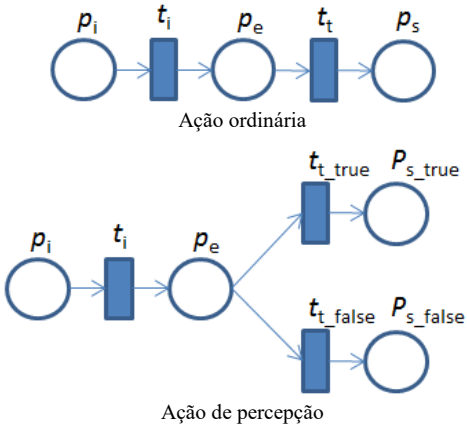

Figura 2. Tipos básicos de ação em PNP.

A ação ordinária é uma estrutura básica que modela uma ação determinística, explicitamente representando as ações como não instantâneas através da definição do seu evento de início (t_i), estado de execução (p_e) e evento de terminação (t_t). A ação de percepção implementa um tipo de não determinismo, onde a saída final da ação depende de alguma propriedade/variável cujo valor (*true* ou *false*) vai ser conhecido apenas em tempo de execução.

Os operadores (Figura 3) permitem a combinação de múltiplas redes para a criação de planos mais complexos. A forma mais simples de combinar ações é através do *merging*, o qual combina o p_s de uma rede com o p_i de outra rede quando os mesmos são iguais (Figura 3a). Essa operação cria ações sequenciais.

Outro operador importante é o *interrupt*, o qual permite monitorar a execução de um estado através da ligação de uma posição de execução com outra posição inicial através de uma transição de interrupção (Figura 3b) que, quando verificada, suspende a execução atual e dispara um plano alternativo de recuperação.

Os dois próximos operadores oferecem suporte à concorrência. Como cada token em um plano pode ser considerado um *thread*, o operador *Fork* (Figura 3c) gera múltiplos threads de um simple thread, enquanto o operador *Join* (Figura 3d) permite a sincronização de múltiplos threads. Ou seja, ele consome múltiplos *threads* simultaneamente e gera um simples thread sincronizado.

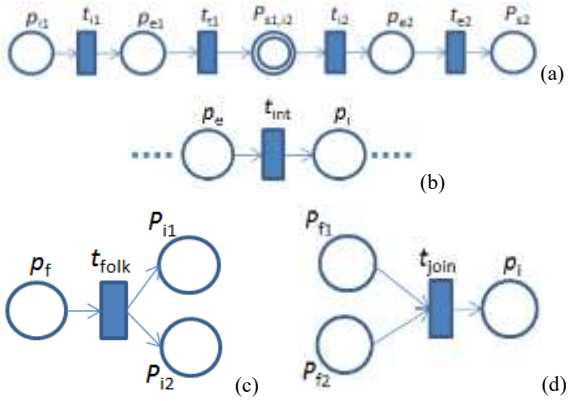

Figura 3. Operadores PNP.

Finalmente temos o pseudocódigo que auxilia no entendimento da execução do plano.

```
procedure execute (PNP<P,T,F,W,Mo,G>)
1.  estadoAtual ← Mo
2.  while estadoAtual ∉ G do
3.      for all t ∈ T do
4.          if ativa(t,estadoAtual) ∧ kb |= t.ϕ then
5.              executarTransição(t)
6.              estadoAtual ← atualizar(estadoAtual,t)
7.          end if
8.      end for
9.  end while
end procedure.
```

Nas linhas 3 e 4 vemos que, para todas as transições do plano, é testado se esta transição está ativa (o *thread* relacionado a cada uma das transições está em execução) e se a base de conhecimento *kb* contém as condições ϕ para a ativação da transição. Se tais condições forem verdadeiras, então a transição é executada e o estado atual do plano atualizado (linha 6).

Nossa intenção aqui não foi fazer uma revisão exaustiva da linguagem PNP, mas apenas apresentar os conceitos que são pertinentes ao entendimento da modelagem proposta. Uma discussão detalhada da linguagem pode ser encontrada em (Ziparo et al., 2011).

4 Modelagem Proposta

4.1 O Robô

O esquema seguinte (Figura 4) mostra a especificação física inicial do robô móvel, o qual foi utilizado como modelo para a construção do protótipo sob

controle da especificação PNP. Este protótipo foi desenvolvido para atuar em tubulações com diâmetro padrão de 38,1mm e serviu principalmente como uma plataforma de testes para os algoritmos de controle autônomo. Devido à quantidade de ruídos apresentada no primeiro protótipo, os dois motores da frente do robô, e consequentemente suas rodas, foram substituídas por uma roda central passiva do tipo pivô.

A principal decisão sobre a construção do robô está relacionada ao conjunto de sensores que será utilizado. No caso da detecção de rachaduras, as pesquisas ainda não chegaram a um consenso de qual seria o sensor mais adequado para esta tarefa. Os sensores acústicos se mostram os mais utilizados em pesquisas atuais. De acordo com tais pesquisas, existe uma clara característica que pode ser identificada no sinal acústico quando o mesmo atinge uma rachadura (Gao et al. 2004). Porém, devem ser observados outros componentes acústicos de fontes diversas, como o próprio motor do robô, que podem causar interferência no sinal de retorno (Galleher and Kurtz, 2008).

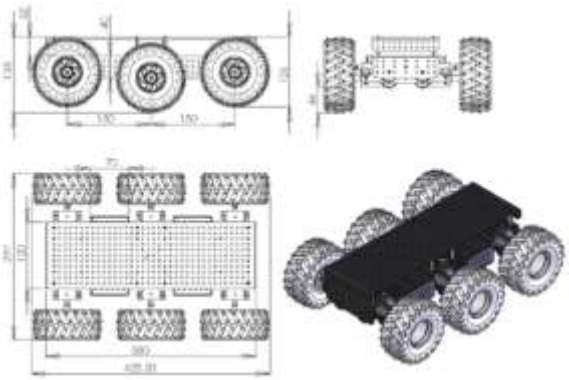

Figura 4. Especificação física do robô.

A questão da análise de imagens da câmera é outro aspecto que pode ser analisado. Sistemas de aprendizagem podem aprender o padrão de rachaduras e utilizar tal padrão para detecção das mesmas em fotos tiradas de forma periódica (Sinha and Karray, 2012). Uma terceira forma de investigação do problema é através de sensores de pressão. Neste caso a ideia é medir a pressão em diferentes setores da tubulação. Pesquisas iniciais mostram que existe um desvio da pressão em locais onde existe uma rachadura e tal diferença pode ser percebida mesmo quando a tubulação não apresenta o fluido (Choi, 2012). Por fim, também existe a possibilidade de utilizar a modificação da coloração da seção do tubo onde ocorreu a rachadura (Chen et al., 2012). Todas estas formas de percepção e uso de sensores podem ser utilizados de forma separada ou em conjunto, em uma abordagem híbrida, de forma a trazer uma maior segurança ao resultado encontrado. Tal definição é uma contribuição futura da continuidade deste trabalho.

4.2 Arquitetura Abstrata

A execução do algoritmo PNP que está sendo utilizado neste trabalho é baseada na arquitetura mostrada na Figura 5 (Ziparo et al., 2011). Tal arquitetura possui duas camadas principais, sendo elas:

- Camada simbólica: composta pela base de conhecimento (KB), a qual possui informações atualizadas sobre o domínio, e o módulo executor PNP, o qual funciona de acordo com o plano PNP de entrada. Basicamente este módulo implementa o algoritmo descrito anteriormente, de modo que ele realiza consultas à base de conhecimento, obtendo resultados que podem disparar as transações do modelo. Para o propósito do algoritmo de execução, a única consulta necessária é especificada por kb $\models t.\phi$ (ver linha 4 do algoritmo de execução);
- A camada numérica foge ao escopo deste artigo, mas sua função inicial é tratar todos os sinais de entrada (eliminação de ruídos, fusão de dados, etc.), convertendo os mesmos em descrições lógicas de acordo com o formalismo utilizado pela base de dados. Uma segunda função é executar as ações previamente descritas e mapeadas para os atuadores do robô (e.g. motores).

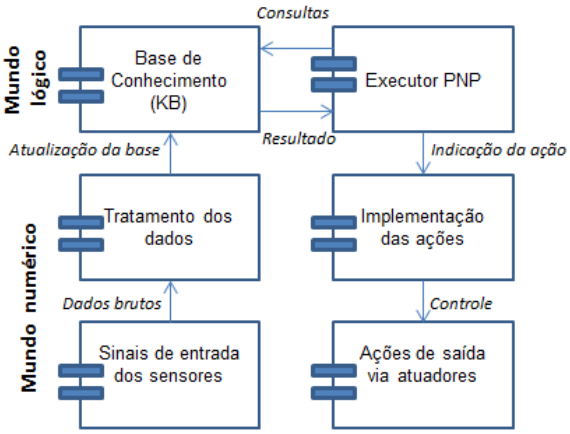

Figura 5. Arquitetura abstrata do robô.

Esta abordagem assume que existe disponível um conjunto de ações já implementadas A = $\{a_1,a_2,...,a_k\}$, de modo que cada ação vai rodar em um thread separado, dentro de uma posição; e o seu início, término ou interrupção vai ser controlado por uma transação quando a mesma for executada (ver linha 5 do algoritmo de execução).

4.3 Modelagem PNP

Um robô autônomo para verificação de tubulações tem como principal função a detecção de furos e rachaduras. Tal função pode ser modelada através de uma ação de percepção (Figura 6).

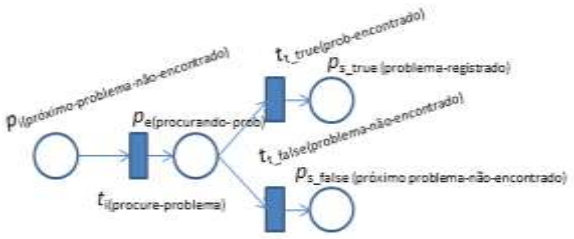

Figura 6. Ação de percepção para detecção de problemas.

De acordo com a figura, a ação (p_i) se inicia com o objetivo de se achar um problema. Se p_i estiver ativa e nenhuma localização foi identificada no ciclo atual, então p_e passa a ser ativo e o robô executa o processo de procura. Se o ciclo for terminado e nenhum problema encontrado, p_{s_false} é acionado. Caso contrário, p_{s_true} é acionado.

A procura acima é realiza em uma seção reta do cano dentro de um tempo pré-definido (ciclo). Porém, uma curva pode ser encontrada e uma função de mudança de sentido deve ser acionada. Esta função é modelada pela ação ordinária seguinte (Figura 7).

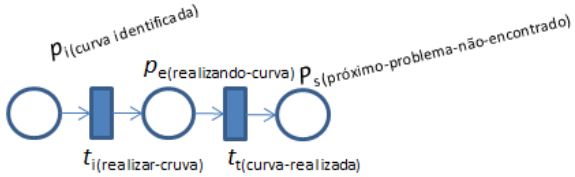

Figura 7. Ação ordinária para mudança de sentido.

A função de desvio de obstáculos deve ser acionada quando o robô identifica que algum objeto está bloqueando o seu caminho. Esta função também pode ser modelada como uma ação ordinária (Figura 8).

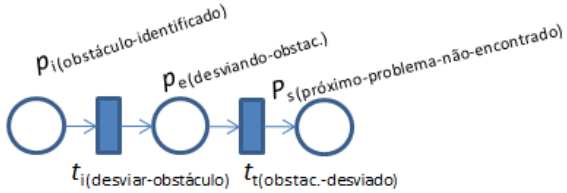

Figura 8. Ação ordinária para desvio de obstáculo.

Durante o percurso do robô, ele pode receber algum evento ou identificar alguma situação na qual ele deva retornar para a posição de início. Essa função é bastante importante para garantir a autonomia do robô. A ação ordinária da Figura 9 representa tal função.

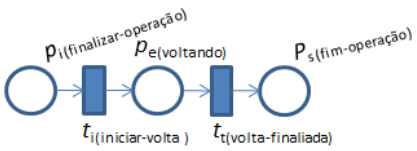

Figura 9. Ação ordinária para retorno.

Em algumas situações, o robô deve registrar a sua posição de forma que ele possa, por exemplo, retornar a esta posição. Essa função é complexa porque o robô não pode utilizar o GPS, uma vez que ele se encontra a alguns metros abaixo da superfície. Desta forma, também precisamos de uma ação ordinária para calcular esta posição (Figura 10).

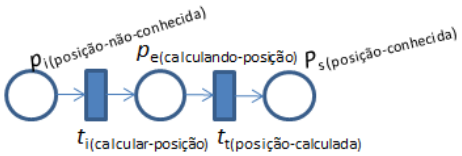

Figura 10. Ação ordinária para cálculo da posição do robô.

Outra ação importante é o envio de uma imagem ou vídeo para a central. Isso pode ser feito, por exemplo, quando o robô tiver dúvidas sobre uma anormalidade identificada na tubulação. A própria existência de um obstáculo, por exemplo, pode ser considerada uma anormalidade dentro de um cano de água. Esta função é modelada pela ação ordinária seguinte (Figura 11).

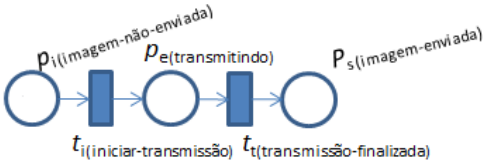

Figura 11. Ação ordinária para envio de mídia.

Uma última função é a identificação do nível crítico da bateria. Esta função também não é trivial porque este nível deve permitir que o agente volte para sua posição de início. Ou seja, ela depende da distância do robô da sua posição inicial. Tal função é representada por uma ação de percepção (Figura 12).

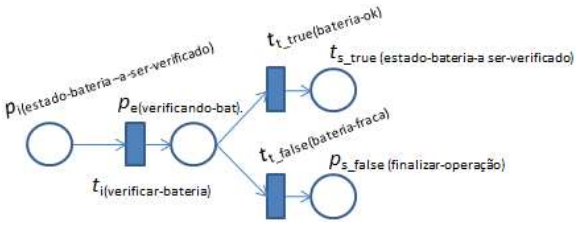

Figura 12. Ação de percepção para nível crítico da bateria.

Por fim, todas estas ações atômicas podem ser integradas através de combinação de posições e dos operadores t_{int}, t_{folk} e t_{join}. O resultado final é o plano (PNP) de controle do robô (Figura 13).

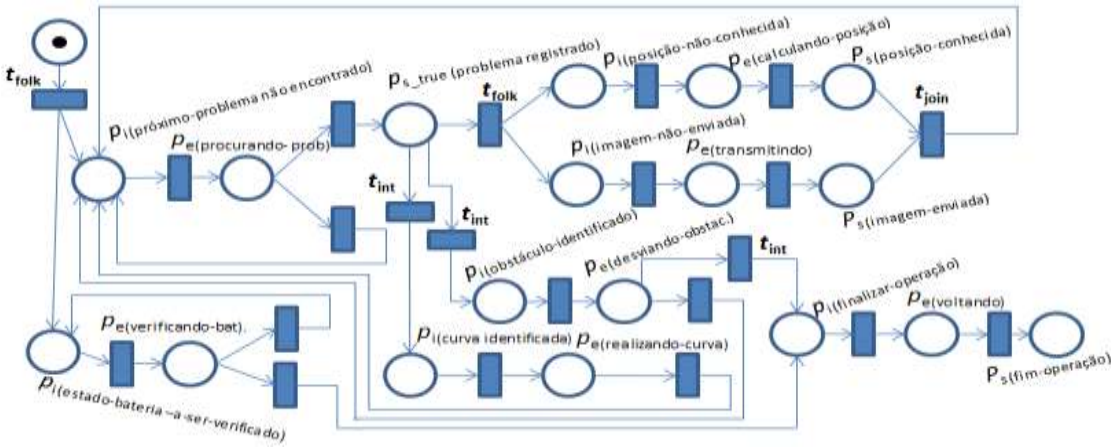

Figura 13. PNP para robô de inspeção para tubulações (alguns rótulos foram retirados para efeito de simplicidade).

Nesta modelagem pode ser observada a expressividade de PNP, principalmente para lidar com ações concorrentes. Logo no início do plano existe um t_{folk} que ativa duas posições p_i: a de procura por problemas e a de verificação da bateria. Cada uma dessas ações de percepção também implementa naturalmente um loop que controla o comportamento do robô. Outro uso de t_{folk} acontece quando duas ações diferentes são disparadas no momento que um problema é identificado. Estas ações são a de definir a posição dentro da tubulação para futuras referências e a de enviar imagens do problema para a central. Ao final temos um t_{join} que sincroniza a finalização destas duas ações antes de continuar com a execução do plano. Outro recurso interessante são os t_{int}, os quais permitem o monitoramento da execução das ações em um p_e. Por exemplo, quando o robô está à procura de um problema, podem acontecer dois fatores que interrompem esta procura: o aparecimento de uma junção de desvio de 90 graus na tubulação, ou algum tipo de obstáculo. Deste modo, a procura é interrompida e tais ações são realizadas. Ao final, a ação de procura é novamente inicializada. Uma t_{int} também é disparada caso o obstáculo não possa ser contornado. Neste caso, o plano é encerrado com a volta do robô à sua posição inicial. .

5 Análise e Discussão

A especificação PNP gerada foi submetida à análise da ferramenta APO - *Analysis of Petri Nets and Transitions Systems* (Best and Schlachter, 2015), a qual implementa vários métodos de verificação para redes de Petri. De acordo com esta ferramenta, as seguintes características foram destacadas:

- A especificação não é *output_nonbranching*, ou seja, ela possui posições com saídas para mais do que uma transição.

Essa característica é principalmente forçada pelos operadores de percepção que possuem no mínimo duas transições para sua posição de execução. Esta estrutura da especificação também força outra característica detalhada a seguir:

- A especificação não é *persistente*, uma vez que existem situações em que duas transições distintas são simultaneamente habilitadas, mas a ocorrência de uma faz com que a outra seja automaticamente desabilitada.

O que se deve evitar em tais situações é a presença de não determinismo em tempo de execução. Ou seja, situações que podem gerar conflitos. A ferramenta de análise, por exemplo, indica a seguinte característica da especificação:

- A especificação não é livre de conflitos de comportamento (*behaviourally conflict-free*) porque ela apresenta situações onde duas ou mais transições distintas, que são simultaneamente habilitadas, compartilham uma posição (p_i) em comum.

Desta forma, transições que possuem uma p_i em comum devem apresentar condições disjuntas de forma a evitar o conflito e, consequentemente, o não determinismo. A análise também destaca a importante característica abaixo:

- A especificação é *simply live*, uma vez que existe ao menos um caminho da configuração inicial para a configuração final. Isso mostra que o sistema não é feito para ser executado de forma infinita, tendo condições para a finalização do mesmo.

A ferramenta também destacou o número máximo de tokens que podem estar em execução, e quais são os processos relacionados a tal execução (Figura 14). Esta informação é importante para indicar a carga máxima de processamento que pode ser requerida do sistema físico (microcontrolador do robô). De acordo com a análise, em um dado momento *t*, o seguinte conjunto de posições pode estar ativo:

$$(p_2 \vee p_{22}) \wedge (p_{10} \vee p_{12} \vee p_{18}) \wedge (p_{11} \vee p_{13} \vee p_{19})$$

Porém, destas posições, as que realmente executam processamento são p_{22} (verificação do estado da bateria), p_{12} (cálculo do posicionamento do robô) e p_{13} (transmissão de imagem).

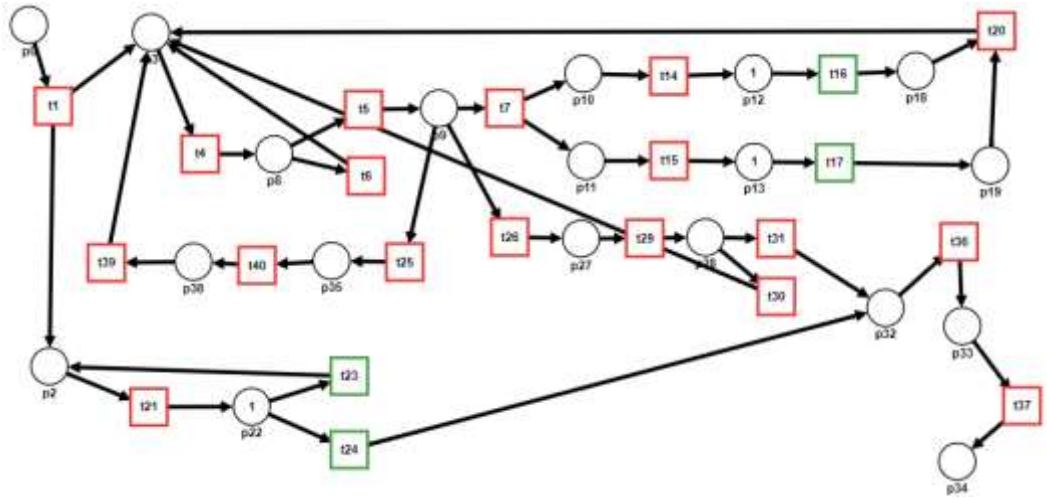

Figura 14. Exemplo de uma das análises feitas com a especificação PNP, mostrando o número máximo de posições de execução ativas.

As figuras seguintes mostram a implementação da parte numérica da arquitetura (ver Figura 5), a qual é parcialmente realizada em hardware. O ambiente de testes foi simplificado para uma seção de cano reto (Figura 15) devido a restrições físicas (a seção transversal do cano precisaria ser maior, ou o robô menor para que o mesmo fosse capaz de realizar uma curva de 90°). Algumas variações de fissuras e pequenos pontos de vazamentos foram introduzidos no cano, e a maior parte do cano foi colocada dentro de uma caixa de areia de modo a simular um ambiente mais real e isolado de ruídos externos.

Figura 15. Ambiente de testes.

A posição de execução, referente à "procura de problemas" foi implementada através de dois sensores do tipo microfone de alta sensibilidade (Figura 16); enquanto que a identificação de um obstáculo é realizada por um sensor do tipo ultrassônico. O nível de água acima de um limite de segurança suportado pelo robô também foi considerado um obstáculo. Desta forma, um sensor de nível de água (Figura 17) também foi utilizado para gerar um sinal para a interrupção da execução normal do robô. Por fim, o uso da câmera não foi avaliado neste protótipo. Desta forma, o robô apenas envia um sinal wireless para a central de dados, indicando que um vazamento foi encontrado (Figura 18). A análise da distância máxima de transmissão obtida por tal sinal dentro de uma tubulação em diversas configurações, de forma que o robô não perca a sua conexão com a central de dados, é um dos trabalhos futuros deste projeto.

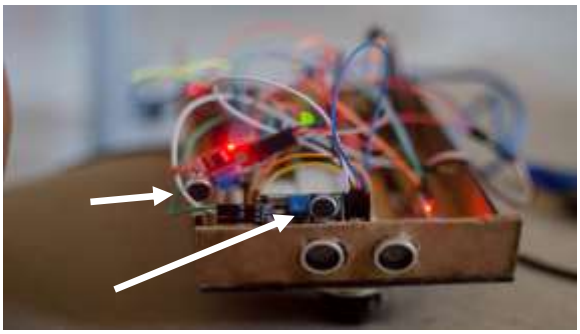

Figura 16. Sensores do tipo microfone.

Figura 17. Sensor para captação do nível de água.

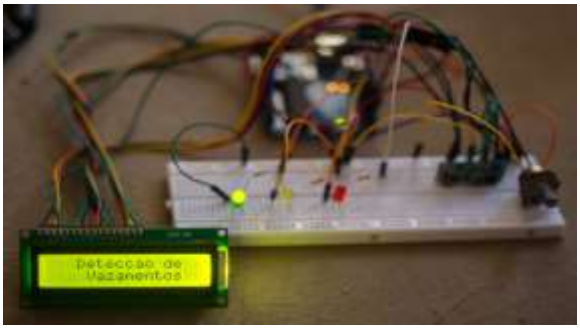

Figura 18. Central de dados.

6 Conclusão e Trabalhos Futuros

Este trabalho apresenta duas contribuições principais. Primeiro é mostrada uma especificação comportamental completa de um robô autônomo para inspeção de tubulações. A literatura atual não apresenta tais descrições uma vez que os robôs que realizam inspeções são na sua grande maioria controlados por operadores humanos. Segundo, esta especificação demonstra a expressividade da linguagem PNP na descrição dos planos, enquanto sua leitura e uso continuam simples devido ao modelo explícito para ações e operadores.

Atualmente o projeto está na fase de implementação da camada numérica do robô, de modo que nossas ações futuras estão voltadas para a integração do processo lógico com tal camada (sensores e atuadores). Projetos futuros pretendem realizar uma avaliação de diversos tipos de sensores, junto com o pré-processamento dos sinais gerados (e.g. filtragem e caracterização), de modo a identificar qual seria a melhor abordagem para este domínio.

Referências Bibliográficas

Best, E. and Schlachter, U (2015). Analysis of Petri Nets and Transition Systems. Proceedings of the 8th Interaction and Concurrency Experience, pp.53-67.

Celaya, J. R.; Desrochers, A. A. and Graves, R. J (2007). Modeling and analysis of multi-agent systems using petri nets. IEEE International Conference on Systems, Man and Cybernetics, pp. 1439–1444.

Chatzigeorgiou, D. M (2010). Analysis and Design of an In-Pipe System for Water Leak Detection. Mater Thesis, MIT.

Chen, Z.; Temitayo, M. and Isa, D (2012). Pipe Flaws Detection by Using the Mindstorm Robot. International Journal of Engineering Research and Applications, Vol.2, No. 6; pp. 569-574.

Chis, T. and Saguna, A (2007). Pipeline Leak Detection Techniques. Annals Computer Science Series, 5th Tome 1st fasc.

Choi, C (2012). Robot Design for Leak Detection in Water-Pipe Systems. Master Thesis, MIT.

Costelha, H. and Lima, P (2007). Modelling, analysis and execution of robotic tasks using petri nets. Proceeding of Interantional Conference on Intelligent Robots and Systems, pp. 1449–1454.

Galleher, J. and Kurtz, D. W (2008). Evaluation of an Un-Tethered Free-Swimming Acoustic Leak Detection Technology. Journal of Water Resources Planning and Management, Vol. 58.

Gao, Y. et al (2004). A model of the correlation function of leak noise in buried plastic pipes, Journal of Sound and Vibration, vol. 277, pp. 133-148.

Gat, E (1997) ESL: A language for supporting robust plan execution in embedded autonomous agents. Proceedings of the IEEE Aerospace Conference, Vol. 1, pp. 319–324.

Hunaidi, O.; Wang, A. and Guan, W (2006). A new system for locating leaks in urban water distribution pipes. Management of Environmental Quality: An International Journal, Vol. 17, No. 4; pp. 450-466.

Kuo, C. and Lin, I (2006). Modeling and control of autonomous soccer robots using distributed agent oriented petri nets. IEEE International Conference on Systems, Man and Cybernetics, Vol. 5; pp. 4090–4095.

Mateos, A. L.; Zhou, K. and Vincze, M (2012). Towards efficient pipe maintenance: DeWaLoP in-pipe robot stability controller. IEEE International Conference on Mechatronics and Automation, Chengdu, pp. 1-6.

Sinha, S. K. and Karray, F (2012). Classification of underground pipe scanned images using feature extraction and neuro-fuzzy algorithm. IEEE Transactions on Neural Networks, Vol. 13, No. 2; pp. 393–401.

Song, Z.; Ren, H.; Zhang, J. and Ge, S. K (2016). Kinematic Analysis and Motion Control of Wheeled Mobile Robots in Cylindrical Workspaces. IEEE Transactions on Automation Science and Engineering, Vol. 13, No. 2; pp. 1207-1214.

Tur, J. M. and Garthwaite, W (2010). Robotic Devices for Water Main In-Pipe Inspection: A Survey, Journal of Field Robotics, Vol. 27, No. 4; pp. 491-508.

Ziparo, V. A.; Iocchi, L.; Lima, P.; Nardi, D. and Palamara, P (2011). Petri Net Plans - A framework for collaboration and coordination. in multi-robot systems. Autonomous Agents and Multi-Agent Systems, Vol. 23, No. 3; pp. 344-383.