Remedino: Embedded Multi-Agent System on the CHON Platform for Assistive Medication Monitoring

Daniel Passos Soares, Caio Canto, Bruno Freitas, Nilson Lazarin

Bachelor's Degree in Electrical Engineering
Centro Federal de Eduação Tecnológica Celso Suckow da Fonseca (CEFET-RJ)
Nova Friburgo, RJ - Brasil

chon@grupo.cefet-rj.br

Abstract. *This work presents Remedino, an embedded multi-agent system for medication monitoring designed to support elderly patients and caregivers. The solution integrates a Raspberry Pi running BDI agents developed within the CHON framework (CHON OS) with an Arduino Uno responsible for audiovisual patient alerts via an LCD Shield and buzzer, using the Javino model for serial communication. The architecture utilizes IMAP and SMTP protocols to enable remote human-agent interaction. In operation, the AgentMailer monitors an email inbox for medication reminders and forwards them to the AgentARGO, which triggers the physical hardware alert. Upon the patient's physical button confirmation, a notification is automatically sent back to the caregiver. Experimental results validate the system's low cost, feasibility, and practical applicability in assistive healthcare scenarios.*

1. Introdução

O uso correto de medicamentos é um dos pilares do tratamento de doenças crônicas, e ainda assim continua sendo um dos elos mais frágeis do cuidado em saúde. Segundo a Organização Mundial da Saúde, a adesão média ao tratamento de longo prazo gira em torno de apenas 50% nos países desenvolvidos, sendo ainda menor nos países em desenvolvimento (Sabáté 2003), índice com consequências diretas sobre a eficácia clínica. O problema se agrava entre os idosos: com o avanço da idade é comum a polifarmácia, e uma revisão sistemática sobre idosos nessa condição vivendo em domicílio encontrou taxas de não adesão de 6% a 55%, associadas a declínio cognitivo, complexidade do esquema terapêutico e sobrecarga do cuidador (Zelko et al. 2016). Além do impacto individual, a não adesão se traduz em internações evitáveis e elevação dos custos de saúde.

Boa parte das estratégias de apoio ao paciente esbarra em uma limitação prática: o cuidador, familiar ou profissional, nem sempre está fisicamente presente no momento em que o remédio deve ser tomado. Há, portanto, uma lacuna entre quem precisa ser lembrado e quem deseja acompanhar se o cuidado foi de fato realizado, cuja solução requer um mecanismo que lembre o paciente de forma clara no ambiente em que ele vive e, ao mesmo tempo, ofereça ao cuidador remoto uma confirmação de que a medicação foi administrada.

Nesse cenário, os sistemas embarcados associados a técnicas de inteligência artificial surgem como alternativa promissora. Este trabalho apresenta o Remedino, um sistema multiagente embarcado para monitoramento de medicação que combina agentes deliberativos executados em um Raspberry Pi com um alerta físico controlado por um Arduino, utilizando o correio eletrônico como canal de interação entre humanos e agentes. Delimita-se, desde já, que o Remedino é concebido como uma prova de conceito simplificada, cujo propósito é introduzir o uso de agentes de IA no controle medicamentoso na forma aqui apresentada, sem a pretensão de esgotar as demais dimensões do problema de adesão terapêutica, tratadas neste texto como trabalhos futuros.

1.1. Definição do problema

As soluções disponíveis para lembrar o paciente de tomar a medicação apresentam limitações importantes sob a ótica do cuidado assistido. Os alarmes convencionais são facilmente silenciados e esquecidos, e não oferecem retorno sobre o cumprimento da tarefa. As caixas organizadoras dependem inteiramente da iniciativa do paciente e não informam o cuidador à distância. Os aplicativos de smartphone pressupõem que o usuário possua e saiba operar um aparelho com conectividade, condição nem sempre aplicável ao público idoso, e os dispensadores eletrônicos comerciais costumam ter custo elevado e operar sobre plataformas fechadas. Um ponto comum a essas abordagens é a ausência de um laço de confirmação acessível, e é justamente esse retorno que transforma um simples alarme em um instrumento de cuidado.

Há ainda um desafio técnico relacionado à comunicação dos sistemas embarcados. Muitos middlewares voltados para a Internet das Coisas adotam protocolos sem conexão e oferecem pouca garantia de segurança no transporte das mensagens, o que dificulta a interação confiável entre dispositivos distribuídos e entre máquinas e pessoas (Souza de Jesus et al. 2023). Para um cenário de saúde domiciliar, é desejável um canal de comunicação universal, seguro, assíncrono e que não exija instalação de software dedicado, combinação de requisitos que motiva a escolha do correio eletrônico como meio de interação no Remedino. Cabe delimitar que este trabalho trata apenas do lembrete, do alerta físico e da confirmação por botão como prova de conceito; a garantia efetiva de ingestão do medicamento é discutida na Seção 6 como trabalho futuro.

1.2. Interação humano-agente

Um agente é uma entidade capaz de perceber o ambiente por meio de sensores e de atuar sobre ele através de atuadores, buscando cumprir objetivos a partir de suas convicções e motivações (Russell and Norvig 2010). Entre os modelos cognitivos mais utilizados para descrever esse raciocínio está o modelo Crença, Desejo e Intenção (BDI), no qual crenças representam o conhecimento do agente sobre o mundo, desejos representam os objetivos que pretende alcançar e intenções correspondem aos planos que efetivamente se compromete a executar (Bratman et al. 1988).

A interação entre humanos e agentes ocupa papel central quando esses sistemas passam a fazer parte do cotidiano das pessoas. No monitoramento de medicação, o cuidador precisa de uma maneira simples de solicitar um lembrete e de receber a confirmação correspondente, sem dominar ferramentas complexas. O correio eletrônico atende bem a esse requisito, pois é um serviço amplamente difundido e dispensa aplicativos específicos. Trabalhos recentes demonstraram que é possível mapear mensagens entre agentes, expressas em KQML, para o formato de e-mail, viabilizando uma comunicação segura e simplificada entre humanos e agentes embarcados (Lazarin et al. 2024). O Remedino se apoia diretamente nessa ideia para construir seu canal de interação.

2. Fundamentação teórica e trabalhos relacionados

2.1. Sistemas multiagentes embarcados e o modelo BDI

A programação de agentes BDI conta com linguagens e plataformas consolidadas, como o Jason, interpretador da linguagem AgentSpeak que implementa o ciclo de raciocínio BDI em termos de crenças, objetivos e planos, e o JaCaMo, que amplia o escopo integrando agente, organização e ambiente (Bordini et al. 2007). Levar esse raciocínio para dispositivos embarcados exige mecanismos adicionais de integração com o hardware: a arquitetura Argo estende o agente Jason padrão para que ele controle microcontroladores, filtrando percepções e enviando ações a placas como o Arduino (Pantoja et al. 2016), e a comunicação entre a camada de raciocínio, no computador de placa única, e a camada física, no microcontrolador, é realizada pelo protocolo Javino, que troca mensagens pela

porta serial com verificação de integridade. Esse conjunto de ferramentas é distribuído pelo grupo de pesquisa CHON, que mantém um sistema operacional de propósito específico para sistemas multiagentes embarcados e o ambiente ChonIDE (Lazarin et al. 2023), tendo também tratado questões de comunicabilidade entre sistemas embarcados diante de falhas de conectividade (Souza de Jesus et al. 2023).

2.2. Comunicação de agentes por correio eletrônico

A comunicação entre agentes tradicionalmente se apoia em linguagens como o KQML, que representa, por meio de forças ilocucionárias como *tell* e *achieve*, a intenção do emissor ao enviar uma mensagem. A proposta de transportar essas mensagens sobre os protocolos de correio eletrônico deu origem à arquitetura de agente personalizada Mailer, que envia e recebe mensagens em KQML usando IMAP para a recepção e SMTP para o envio (Lazarin et al. 2024). Nesse mapeamento, o remetente do e-mail identifica a origem da mensagem, o assunto carrega a força ilocucionária e o corpo carrega o conteúdo proposicional; quando o destinatário é também um agente Mailer, a mensagem é retraduzida para o formato interno do Jason, e quando é um ser humano, a mesma estrutura pode ser lida e respondida em qualquer cliente de e-mail, tornando o correio eletrônico um canal assíncrono, seguro e independente de infraestrutura proprietária.

2.3. Sistemas de lembrete de medicação

A literatura registra diferentes abordagens para o lembrete de medicação, de alarmes simples e caixas organizadoras a aplicativos de smartphone e dispensadores automatizados, que em geral priorizam o aviso ao paciente mas tratam de forma limitada o retorno ao cuidador e o acionamento remoto independente de plataformas fechadas. O Remedino se diferencia ao reunir, sobre hardware embarcado de baixo custo, a decisão distribuída entre agentes BDI especializados, o acionamento e o retorno por correio eletrônico como canal universal e o fechamento do laço de cuidado por meio de confirmação física devolvida automaticamente ao solicitante, instanciando em um caso de uso de saúde a arquitetura de comunicação por e-mail proposta na literatura (Lazarin et al. 2024).

3. Proposta

O objetivo do Remedino é demonstrar, de forma prática e delimitada, que um sistema multiagente embarcado pode receber um pedido de medicação por e-mail, lembrar o paciente por meio de um alerta físico claro e devolver ao cuidador a confirmação de que o botão de tomada foi acionado. Para isso, o sistema foi organizado em duas camadas de software cooperantes, a camada de raciocínio e a camada física, e em um canal de comunicação baseado em correio eletrônico.

3.1. Justificativa da escolha por agentes

A decisão de emitir o alerta, repeti-lo em intervalos e encerrá-lo mediante a confirmação do paciente poderia, em princípio, ser implementada por uma estrutura condicional simples (*if-else*) ou por um script sequencial sem qualquer capacidade de raciocínio autônomo, alternativa que reduziria a complexidade inicial de implementação. Essa solução, no entanto, fixaria o comportamento do dispositivo a um conjunto estático de regras, incapaz de incorporar novas percepções, negociar com outros componentes do sistema ou aprender com o histórico de uso do paciente. A opção por agentes BDI, ainda que introduza uma sobrecarga adicional de projeto em relação a uma lógica condicional, preserva a extensibilidade necessária para que trabalhos futuros incorporem percepções mais ricas, coordenação entre agentes e adaptação ao comportamento observado, sem exigir uma reformulação da arquitetura. A escolha por agentes se justifica, portanto, menos pela complexidade da decisão tomada nesta prova de conceito e mais pela possibilidade prática de aprimoramentos futuros que essa arquitetura viabiliza.

3.2. Visão geral da arquitetura

A arquitetura do Remedino é composta por dois agentes deliberativos que residem em um Raspberry Pi e por um firmware que executa no Arduino. O agente `agentMailer` é responsável pela comunicação com o mundo externo via e-mail, enquanto o agente `agentARGO` é responsável pelo controle do hardware. Os dois agentes cooperam por meio de troca de mensagens internas, e o agente de hardware se comunica com o Arduino pelo protocolo Javino. A Figura 1 apresenta essa organização e os fluxos de informação entre as partes.

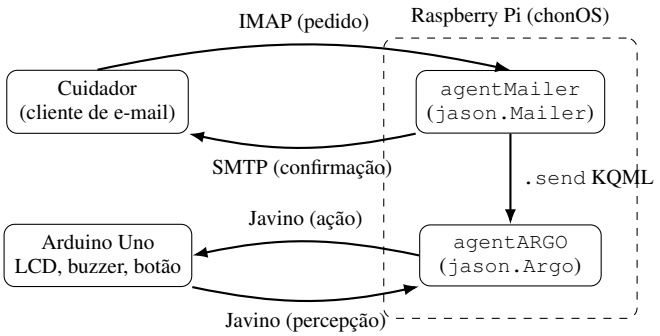

Figure 1. Arquitetura do Remedino e os fluxos de comunicação entre cuidador, agentes e hardware.

3.3. Camada física

A camada física é constituída por um Arduino Uno acoplado a um *LCD Keypad Shield*, que reúne um display de cristal líquido de duas linhas por dezesseis colunas e botões lidos por uma única entrada analógica, ao qual foram adicionados um sinalizador sonoro, responsável pelo apito de chamada de atenção, e um indicador luminoso. O Arduino se conecta ao Raspberry Pi por um cabo USB, que serve tanto para alimentação quanto para a troca de mensagens seriais segundo o protocolo Javino, que encapsula as mensagens com verificação de erro, garantindo que comandos e percepções trafeguem de forma íntegra. O firmware embarcado tem três responsabilidades: apresentar ao paciente a mensagem de que está na hora de tomar o remédio, com o nome do medicamento e a instrução de confirmação; capturar o acionamento do botão de confirmação; e informar ao agente, sob demanda, o estado atual da medicação, isto é, se a dose já foi confirmada ou se o sistema ainda aguarda a ação do paciente.

3.4. Camada de raciocínio

A camada de raciocínio é composta pelos dois agentes BDI. O `agentMailer` utiliza a arquitetura Mailer e mantém, como crenças iniciais, as credenciais da conta de e-mail do robô e a configuração dos servidores de recepção e envio. Ao iniciar, ele estabelece o serviço de correio eletrônico e passa a verificar a caixa de entrada de forma periódica, integrada ao seu ciclo de raciocínio. Quando um pedido de medicação chega, o agente registra qual remédio foi solicitado e quem fez a solicitação, e em seguida envia ao agente de hardware uma mensagem solicitando o início do alerta.

O `agentARGO` utiliza a arquitetura Argo e, ao iniciar, conecta-se à porta serial do Arduino e abre o canal de percepções. Ao receber a solicitação de alerta, ele entra em um laço que exibe o nome do medicamento no display, aciona o sinalizador sonoro e luminoso por alguns segundos e repete o ciclo em intervalos regulares, mantendo a chamada ativa até que a confirmação ocorra. Quando o Arduino informa, por meio de uma percepção, que o paciente apertou o botão, o agente encerra o alerta e comunica ao `agentMailer` que a medicação foi confirmada, o que dispara o envio do e-mail de retorno ao cuidador.

3.5. Comunicação por e-mail: KQML sobre IMAP e SMTP

O canal de interação com o cuidador foi construído sobre o mapeamento entre mensagens KQML e o formato de e-mail. A Tabela 1 resume essa correspondência, na qual o assunto do e-mail carrega a força ilocucionária e o corpo carrega o termo a ser interpretado pelo agente. Para acionar o robô, o cuidador envia um e-mail cujo assunto é a força *tell* e cujo corpo é o termo que descreve a medicação, por exemplo `tomarRemedio("Dipirona 500mg")`. Ao confirmar a tomada, o robô responde ao remetente original com uma mensagem informando que o medicamento foi tomado.

Table 1. Mapeamento entre a mensagem KQML e o e-mail adotado no Remedino.

Elemento KQML	Campo do e-mail	Exemplo no Remedino
Remetente (origem)	De (*From*)	`cuidador@exemplo.com`
Força ilocucionária	Assunto (*Subject*)	`tell`
Conteúdo proposicional	Corpo (*Body*)	`tomarRemedio("Dipirona 500mg")`

Uma característica importante dessa abordagem é a natureza assíncrona do correio eletrônico. Caso o sistema esteja temporariamente fora do ar, a mensagem permanece armazenada no servidor e é processada quando o agente volta a verificar a caixa de entrada. Além disso, a verificação periódica é incorporada ao próprio ciclo de raciocínio do agente Mailer, de modo que a recepção de e-mails ocorre de forma transparente, sem a necessidade de um laço explícito de espera no código do agente.

4. Prova de conceito

Para validar a proposta, o Remedino foi implementado e executado sobre a plataforma do grupo CHON. Esta seção descreve o ambiente de desenvolvimento, o ciclo de funcionamento observado, a implementação dos agentes e do firmware e os desafios técnicos enfrentados durante a construção do protótipo.

4.1. Ambiente de desenvolvimento

O sistema foi desenvolvido e executado no ChonIDE, ambiente integrado mantido pelo grupo CHON, hospedado sobre o sistema operacional embarcado chonOS em um Raspberry Pi. O projeto é descrito em um único arquivo, que reúne os códigos dos agentes, suas arquiteturas, e o firmware do Arduino. A execução é realizada por um interpretador embarcado do Jason, que integra nativamente as arquiteturas necessárias, incluindo controle de hardware e comunicação por correio eletrônico, fator decisivo para que os dois agentes coexistissem em um mesmo sistema multiagente.

4.2. Ciclo de funcionamento

O funcionamento do sistema pode ser descrito como uma sequência bem definida de etapas, apresentada a seguir. O ciclo começa com o robô em repouso, exibindo no display uma tela de espera animada, e termina com o retorno automático a esse mesmo estado, pronto para um novo pedido.

1. O cuidador envia um e-mail com assunto `tell` e corpo `tomarRemedio("...")`.
2. O `agentMailer` verifica a caixa, traduz o e-mail para KQML e registra o solicitante.
3. O `agentMailer` solicita ao `agentARGO` o início do alerta.
4. O `agentARGO` exibe o medicamento no LCD, aciona o apito e o LED e repete em intervalos regulares.
5. O paciente aperta o botão de confirmação no Arduino.
6. O Arduino percebe a confirmação e o `agentARGO` encerra o alerta.
7. O `agentMailer` responde ao cuidador informando que o remédio foi tomado.

8. Após alguns segundos, o display retorna à tela de espera, pronto para um novo ciclo.

Durante a operação, o display assume três estados distintos. No estado de repouso, ele exibe o nome do sistema e uma animação de espera, formada por reticências que se completam de forma cíclica para indicar que o robô está ativo e aguardando. No estado de alerta, ele exibe o nome do medicamento na primeira linha e a instrução de confirmação na segunda, com a luz de fundo piscando de forma contínua para reforçar o chamado. No estado de confirmação, ele exibe uma mensagem de agradecimento por alguns segundos antes de retornar automaticamente ao repouso.

4.3. Implementação dos agentes e do firmware

A camada de raciocínio foi escrita em AgentSpeak. O Código 1 apresenta o trecho central do `agentARGO`, responsável pelo laço de alerta e pela reação à confirmação do paciente. O laço exibe o nome do medicamento, aciona o sinalizador e aguarda o intervalo configurado antes de repetir, enquanto um plano separado, sensível à percepção de confirmação, interrompe o ciclo e notifica o agente de e-mail.

```
/* O agentMailer pede o alerta, ja com o nome do remedio */
+!alertar(Nome) : alertando(false) <-
   -+alertando(true);
   -+remedio(Nome);
   .print("Iniciando alerta para: ",Nome);
   !apitar.
/* Laco de alerta: mostra o remedio, apita e repete a cada intervalo */
+!apitar : alertando(true) & intervalo(T) & remedio(Nome) <-
   .argo.act(Nome);          // exibe o nome do remedio no LCD
   .wait(1500);              // pausa para o Javino exibir antes do proximo comando
   .argo.act(buzzerAlert);   // aciona o apito e o LED
   .wait(3000);
   .argo.act(alertOff);
   .wait(T);
   !apitar.
+!apitar : alertando(false) <-
   .print("Alerta encerrado.").
/* Percepcao do Arduino: o paciente apertou o botao de confirmacao */
+medicacao(tomada) : alertando(true) <-
   -+alertando(false);
   .argo.act(alertOff);
   .send(agentMailer, achieve, medicacaoConfirmada).
```

Código 1. Raciocínio do agentARGO para o alerta e a confirmação.

No `agentMailer`, a recepção do pedido e a resposta ao cuidador são tratadas por planos específicos. Ao receber a crença correspondente ao pedido, o agente guarda o remetente e o medicamento e solicita o alerta ao agente de hardware. Quando recebe a notificação de confirmação, ele monta uma mensagem legível e a envia de volta ao solicitante, conforme ilustrado no Código 2.

```
+tomarRemedio(Nome)[source(Remetente)] <-
   -+solicitante(Remetente);
   -+remedioAtual(Nome);
   .send(agentARGO, achieve, alertar(Nome)).

+!medicacaoConfirmada : solicitante(Email) & remedioAtual(Nome) <-
   .concat("O paciente confirmou que tomou ",Nome,".",Corpo);
   .mailer.sendEMail(Email,"Remedio confirmado",Corpo).
```

Código 2. Recepção do pedido e resposta ao cuidador no agentMailer.

Do lado do Arduino, o firmware mantém variáveis de estado que indicam se há um alerta na tela, se a confirmação ocorreu e em que instante. A cada iteração do laço principal, ele verifica o botão, decide qual tela exibir e atende aos comandos recebidos do agente. O Código 3 mostra o trecho que gera as percepções enviadas ao agente, informando o estado da medicação.

```
void getExogenousPerceptions(){
  javino.addPercept("device(arduinoWithLCDKeypadShield)");
  if(confirmado) javino.addPercept("medicacao(tomada)");
  else           javino.addPercept("medicacao(aguardando)");
  javino.sendPercepts();
}
```

Código 3. Geração de percepções no firmware do Arduino.

4.4. Desafios de implementação

A construção do protótipo exigiu tratar desafios técnicos determinantes para a confiabilidade do sistema. O primeiro foi a coexistência de duas arquiteturas de agente distintas em um mesmo sistema multiagente, viabilizada pelo interpretador embarcado que as integra nativamente. O segundo surgiu na execução em modo não interativo, sem interface gráfica, condição natural de um dispositivo sem monitor: a configuração padrão tentava abrir um console gráfico e falhava, sendo a solução redirecionar o registro de log dos agentes para o terminal de texto.

Um terceiro desafio envolveu a recepção de e-mails em provedores de grande porte, cuja estrutura de pastas inclui um diretório que serve apenas como contêiner e não armazena mensagens, provocando um erro ao percorrer as pastas; o problema foi contornado restringindo, na configuração da conta, quais pastas ficam visíveis ao protocolo de recepção. Por fim, na comunicação serial, o envio de dois comandos em sequência muito próxima fazia com que o protocolo descartasse o primeiro, resolvido com uma pequena pausa entre comandos. A restrição de dezesseis colunas do display exigiu decisões de interface como a exibição compacta do nome e da dose do medicamento e o realce do alerta pelo piscar da luz de fundo enquanto a confirmação não ocorre.

5. Resultados e discussão

O sistema foi exercitado de ponta a ponta, desde o envio do e-mail de pedido até o recebimento, pelo cuidador, da confirmação automática. O agente de e-mail autenticou-se no servidor, leu a caixa de entrada e interpretou a mensagem, traduzindo-a em uma crença que disparou o encadeamento de ações previsto, e o agente de hardware acionou o alerta no Arduino, que exibiu o nome do medicamento e manteve o chamado ativo, com o display piscando e o apito emitido em intervalos regulares. Ao apertar o botão de confirmação, observou-se a transição imediata para a tela de agradecimento e o retorno automático à tela de espera, enquanto a confirmação era percebida pelo agente de hardware, que encerrou o alerta e notificou o agente de e-mail, o qual respondeu ao remetente original em linguagem legível, fechando o laço proposto neste escopo delimitado.

Quanto ao comportamento temporal, a recepção de pedidos depende da verificação periódica da caixa de entrada incorporada ao ciclo do agente de e-mail; esse intervalo, da ordem de um minuto, mostrou-se adequado ao caso de uso, e a natureza assíncrona do correio eletrônico revelou-se benéfica, pois pedidos enviados enquanto o sistema reiniciava permaneceram no servidor e foram processados na verificação seguinte. De maneira geral, os resultados confirmam a viabilidade da proposta, construída sobre hardware de baixo custo e ferramentas abertas, sem exigir aplicativo dedicado por parte do cuidador. A distribuição do comportamento entre agentes especializados conferiu modularidade ao sistema, de modo que o tratamento do e-mail e o controle do hardware permanecem separados e podem evoluir de forma independente.

6. Conclusão

Este trabalho apresentou o Remedino, uma prova de conceito simplificada de sistema multiagente embarcado para introdução do uso de IA no controle medicamentoso, voltada ao apoio de pacientes idosos e seus cuidadores, sem a pretensão de resolver a totalidade dos desafios da adesão terapêutica. A solução integrou agentes BDI executados em um Raspberry Pi a um Arduino responsável pelo alerta audiovisual, empregando o protocolo Javino para a comunicação serial e o correio eletrônico, por meio dos protocolos IMAP e SMTP, como canal de interação humano-agente, tal como aqui delimitado. A prova de conceito demonstrou que o sistema é capaz de receber um pedido por e-mail, lembrar o paciente de maneira clara, capturar o acionamento do botão de confirmação e devolver essa informação ao cuidador de forma automática.

Os resultados obtidos evidenciam que a combinação entre raciocínio deliberativo distribuído, hardware embarcado de baixo custo e comunicação por correio eletrônico constitui uma alternativa promissora para aplicações de saúde assistida, com arquitetura modular, segura e independente de aplicativos proprietários. É importante ressaltar que a confirmação por botão indica apenas a interação do paciente com o dispositivo, não garantindo, por si só, que a medicação tenha sido efetivamente ingerida; essa limitação, própria do escopo simplificado adotado, é justamente uma evidência a favor do uso de agentes de IA, cuja capacidade de aprender com as circunstâncias e com o histórico de uso pode ser explorada em trabalhos futuros para aumentar a confiabilidade dessa confirmação, por exemplo por meio de sensores complementares ou do reconhecimento de padrões de comportamento do paciente. Como demais trabalhos futuros, pretende-se incorporar uma agenda de lembretes programados, permitir a especificação de remetentes autorizados, ampliar as informações exibidas ao paciente e investigar o suporte a múltiplos pacientes e medicamentos, ampliando a autonomia do sistema e seu potencial de uso real em contextos de cuidado continuado.

References

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons Ltd.

Bratman, M. E., Israel, D. J., and Pollack, M. E. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(3):349–355.

Lazarin, N. M., Pantoja, C. E., and Viterbo, J. (2023). Towards a toolkit for robotic-agents supported AI: Proposal and first impressions. In *Proceedings of the XXXI Workshop sobre Educação em Computação (WEI)*, pages 20–29, Porto Alegre, RS, Brasil. SBC.

Lazarin, N. M., Souza, J. P. B., Pantoja, C. E., Alexandre, T., and Viterbo, J. (2024). Human and BDI-agent interaction via KQML messages over IMAP and SMTP. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Digital Twins: The PAAMS Collection*, volume 15157 of *Lecture Notes in Computer Science*, pages 172–183. Springer Nature Switzerland, Cham.

Pantoja, C. E., Stabile, M. F., Lazarin, N. M., and Sichman, J. S. (2016). ARGO: An extended jason architecture that facilitates embedded robotic agents programming. In *Engineering Multi-Agent Systems (EMAS 2016)*, volume 10093 of *Lecture Notes in Computer Science*, pages 136–155. Springer International Publishing, Cham.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition.

Sabáté, E. (2003). *Adherence to Long-Term Therapies: Evidence for Action*. World Health Organization, Geneva.

Souza de Jesus, V., Lazarin, N. M., Pantoja, C. E., Manoel, F. C. P. B., Alves, G. V., and Viterbo, J. (2023). A middleware for providing communicability to embedded MAS based on the lack of connectivity. *Artificial Intelligence Review*, 56(S3):2971–3001.

Zelko, E., Klemenc-Ketis, Z., and Tusek-Bunc, K. (2016). Medication adherence in elderly with polypharmacy living at home: A systematic review of existing studies. *Materia Socio-Medica*, 28(2):129–132.