

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/364985543>

Desenvolvimento de Sistemas Multiagentes Embarcados

Presentation · October 2022

DOI: 10.13140/RG.2.2.25327.51362

CITATIONS

0

READS

168

2 authors:



Nilson Mori Lazarin

Federal Center for Technological Education Celso Suckow da Fonseca (CETESB)

94 PUBLICATIONS 134 CITATIONS

[SEE PROFILE](#)



Carlos Eduardo Pantoja

Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET-RJ)

139 PUBLICATIONS 280 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



LuBrAS [View project](#)



Projeto Turing [View project](#)

Desenvolvimento de Sistemas MultiAgentes Embarcados

**Carlos Eduardo Pantoja
Nilson Mori Lazarin**

Centro Federal de Educação Tecnológica (CEFET/RJ)

Universidade Federal Fluminense (UFF), Brasil



Introdução

- Agente

BORDINI, R.H.; HÜBNER, J.F.; WOOLDRIDGE, M. Programming Multi-Agent Systems in AgentSpeak using Jason. [S. l.]: Wiley, 2007(Wiley Series in Agent Technology).
HÜBNER, Jomi Fred. Um modelo de reorganização de sistemas multiagentes. 2003. Doutorado em Sistemas Digitais – Universidade de São Paulo, São Paulo, 2003. DOI 10.11606/T.3.2003.tde-17052004-151854.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. XII Escola de Informática da SBC, v. 2, p. 51–89, 2004.

Introdução

- Agente
 - Um sistema computacional capaz de **perceber e atuar em um ambiente através de sua própria deliberação, baseada em suas convicções e motivações** é denominado agente.

BORDINI, R.H.; HÜBNER, J.F.; WOOLDRIDGE, M. Programming Multi-Agent Systems in AgentSpeak using Jason. [S. l.]: Wiley, 2007(Wiley Series in Agent Technology).
HÜBNER, Jomi Fred. Um modelo de reorganização de sistemas multiagentes. 2003. Doutorado em Sistemas Digitais – Universidade de São Paulo, São Paulo, 2003. DOI 10.11606/T.3.2003.tde-17052004-151854.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. XII Escola de Informática da SBC, v. 2, p. 51–89, 2004.

Introdução

- Agente
 - Um sistema computacional capaz de **perceber e atuar em um ambiente através de sua própria deliberação, baseada em suas convicções e motivações** é denominado agente.
 - Esta entidade se difere de um sistema convencional por apresentar uma série de características adicionais.

BORDINI, R.H.; HÜBNER, J.F.; WOOLDRIDGE, M. Programming Multi-Agent Systems in AgentSpeak using Jason. [S. l.]: Wiley, 2007(Wiley Series in Agent Technology).
HÜBNER, Jomi Fred. Um modelo de reorganização de sistemas multiagentes. 2003. Doutorado em Sistemas Digitais – Universidade de São Paulo, São Paulo, 2003. DOI 10.11606/T.3.2003.tde-17052004-151854.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. XII Escola de Informática da SBC, v. 2, p. 51–89, 2004.

Introdução

- Agente
 - Um sistema computacional capaz de **perceber e atuar em um ambiente através de sua própria deliberação, baseada em suas convicções e motivações** é denominado agente.
 - Esta entidade se difere de um sistema convencional por apresentar uma série de características adicionais.
 - É **independente**, visto que seu funcionamento ou existência não depende de outros agentes.

BORDINI, R.H.; HÜBNER, J.F.; WOOLDRIDGE, M. Programming Multi-Agent Systems in AgentSpeak using Jason. [S. l.]: Wiley, 2007(Wiley Series in Agent Technology).
HÜBNER, Jomi Fred. Um modelo de reorganização de sistemas multiagentes. 2003. Doutorado em Sistemas Digitais – Universidade de São Paulo, São Paulo, 2003. DOI 10.11606/T.3.2003.tde-17052004-151854.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. XII Escola de Informática da SBC, v. 2, p. 51–89, 2004.

Introdução

- Agente
 - Um sistema computacional capaz de **perceber e atuar em um ambiente através de sua própria deliberação, baseada em suas convicções e motivações** é denominado agente.
 - Esta entidade se difere de um sistema convencional por apresentar uma série de características adicionais.
 - É **independente**, visto que seu funcionamento ou existência não depende de outros agentes.
 - É **ativo**, pois atua sobre o ambiente, por iniciativa própria para cumprir seus propósitos.

BORDINI, R.H.; HÜBNER, J.F.; WOOLDRIDGE, M. Programming Multi-Agent Systems in AgentSpeak using Jason. [S. l.]: Wiley, 2007(Wiley Series in Agent Technology).
HÜBNER, Jomi Fred. Um modelo de reorganização de sistemas multiagentes. 2003. Doutorado em Sistemas Digitais – Universidade de São Paulo, São Paulo, 2003. DOI 10.11606/T.3.2003.tde-17052004-151854.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. XII Escola de Informática da SBC, v. 2, p. 51–89, 2004.

Introdução

- Agente
 - Um sistema computacional capaz de **perceber e atuar em um ambiente através de sua própria deliberação, baseada em suas convicções e motivações** é denominado agente.
 - Esta entidade se difere de um sistema convencional por apresentar uma série de características adicionais.
 - É **independente**, visto que seu funcionamento ou existência não depende de outros agentes.
 - É **ativo**, pois atua sobre o ambiente, por iniciativa própria para cumprir seus propósitos.
 - É **colaborativo**, dado que se comunica com outros agentes para organizar suas ações.

BORDINI, R.H.; HÜBNER, J.F.; WOOLDRIDGE, M. Programming Multi-Agent Systems in AgentSpeak using Jason. [S. l.]: Wiley, 2007(Wiley Series in Agent Technology).
HÜBNER, Jomi Fred. Um modelo de reorganização de sistemas multiagentes. 2003. Doutorado em Sistemas Digitais – Universidade de São Paulo, São Paulo, 2003. DOI 10.11606/T.3.2003.tde-17052004-151854.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. XII Escola de Informática da SBC, v. 2, p. 51–89, 2004.

Introdução

- Agente
 - Um sistema computacional capaz de **perceber e atuar em um ambiente através de sua própria deliberação, baseada em suas convicções e motivações** é denominado agente.
 - Esta entidade se difere de um sistema convencional por apresentar uma série de características adicionais.
 - É **independente**, visto que seu funcionamento ou existência não depende de outros agentes.
 - É **ativo**, pois atua sobre o ambiente, por iniciativa própria para cumprir seus propósitos.
 - É **colaborativo**, dado que se comunica com outros agentes para organizar suas ações.
 - É **racional**, já que elabora planos de ação para atingir um objetivo.

BORDINI, R.H.; HÜBNER, J.F.; WOOLDRIDGE, M. Programming Multi-Agent Systems in AgentSpeak using Jason. [S. l.]: Wiley, 2007(Wiley Series in Agent Technology).
HÜBNER, Jomi Fred. Um modelo de reorganização de sistemas multiagentes. 2003. Doutorado em Sistemas Digitais – Universidade de São Paulo, São Paulo, 2003. DOI 10.11606/T.3.2003.tde-17052004-151854.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. XII Escola de Informática da SBC, v. 2, p. 51–89, 2004.

Introdução

- Agente
 - Um sistema computacional capaz de **perceber e atuar em um ambiente através de sua própria deliberação, baseada em suas convicções e motivações** é denominado agente.
 - Esta entidade se difere de um sistema convencional por apresentar uma série de características adicionais.
 - É **independente**, visto que seu funcionamento ou existência não depende de outros agentes.
 - É **ativo**, pois atua sobre o ambiente, por iniciativa própria para cumprir seus propósitos.
 - É **colaborativo**, dado que se comunica com outros agentes para organizar suas ações.
 - É **racional**, já que elabora planos de ação para atingir um objetivo.
 - Por fim, é **adaptável** porque em caso de insucesso, busca executar planos alternativos.

BORDINI, R.H.; HÜBNER, J.F.; WOOLDRIDGE, M. Programming Multi-Agent Systems in AgentSpeak using Jason. [S. l.]: Wiley, 2007(Wiley Series in Agent Technology).
HÜBNER, Jomi Fred. Um modelo de reorganização de sistemas multiagentes. 2003. Doutorado em Sistemas Digitais – Universidade de São Paulo, São Paulo, 2003. DOI 10.11606/T.3.2003.tde-17052004-151854.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. XII Escola de Informática da SBC, v. 2, p. 51–89, 2004.

Introdução

- Agente
- Sistemas Multiagentes (SMA)

BALAJI, P. G.; SRINIVASAN, D. An Introduction to Multi-Agent Systems. In: SRINIVASAN, Dipti; JAIN, Lakhmi C. (orgs.). Innovations in Multi-Agent Systems and Applications - 1. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 1–27. DOI 10.1007/978-3-642-14435-6_1. Disponível em: https://doi.org/10.1007/978-3-642-14435-6_1.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. XII Escola de Informática da SBC, v. 2, p. 51–89, 2004.

ALVARES, Luis Otávio; SICHMAN, Jaime Simão. Introdução aos sistemas multiagentes. 97., 1997. XVII Congresso da SBC-Anais JAI [...]. [S. I.: s. n.], 1997. v. 97, .

Introdução

- Agente
- Sistemas Multiagentes (SMA)
 - Formado por um **grupo de agentes autônomos fracamente conectados**, que atuam em um mesmo ambiente.

BALAJI, P. G.; SRINIVASAN, D. An Introduction to Multi-Agent Systems. In: SRINIVASAN, Dipti; JAIN, Lakhmi C. (orgs.). Innovations in Multi-Agent Systems and Applications - 1. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 1–27. DOI 10.1007/978-3-642-14435-6_1. Disponível em: https://doi.org/10.1007/978-3-642-14435-6_1.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. XII Escola de Informática da SBC, v. 2, p. 51–89, 2004.

ALVARES, Luis Otávio; SICHMAN, Jaime Simão. Introdução aos sistemas multiagentes. 97., 1997. XVII Congresso da SBC-Anais JAI [...]. [S. I.: s. n.], 1997. v. 97, .

Introdução

- Agente
- Sistemas Multiagentes (SMA)
 - Formado por um **grupo de agentes autônomos fracamente conectados**, que atuam em um mesmo ambiente.
 - **SMA investigam a coletividade** e não um único indivíduo, diferente de outros paradigmas da Inteligência Artificial (IA).

BALAJI, P. G.; SRINIVASAN, D. An Introduction to Multi-Agent Systems. In: SRINIVASAN, Dipti; JAIN, Lakhmi C. (orgs.). Innovations in Multi-Agent Systems and Applications - 1. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 1–27. DOI 10.1007/978-3-642-14435-6_1. Disponível em: https://doi.org/10.1007/978-3-642-14435-6_1.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. XII Escola de Informática da SBC, v. 2, p. 51–89, 2004.

ALVARES, Luis Otávio; SICHMAN, Jaime Simão. Introdução aos sistemas multiagentes. 97., 1997. XVII Congresso da SBC-Anais JAI [...]. [S. I.: s. n.], 1997. v. 97, .

Introdução

- Agente
- Sistemas Multiagentes (SMA)
 - Formado por um **grupo de agentes autônomos fracamente conectados**, que atuam em um mesmo ambiente.
 - **SMA investigam a coletividade** e não um único indivíduo, diferente de outros paradigmas da Inteligência Artificial (IA).
 - Busca assegurar meios para que os agentes desejem cooperar **para resolver um problema apresentado ao sistema**.

BALAJI, P. G.; SRINIVASAN, D. An Introduction to Multi-Agent Systems. In: SRINIVASAN, Dipti; JAIN, Lakhmi C. (orgs.). Innovations in Multi-Agent Systems and Applications - 1. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 1–27. DOI 10.1007/978-3-642-14435-6_1. Disponível em: https://doi.org/10.1007/978-3-642-14435-6_1.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. XII Escola de Informática da SBC, v. 2, p. 51–89, 2004.

ALVARES, Luis Otávio; SICHMAN, Jaime Simão. Introdução aos sistemas multiagentes. 97., 1997. XVII Congresso da SBC-Anais JAI [...]. [S. I.: s. n.], 1997. v. 97, .

Introdução

- Agente
- Sistemas Multiagentes (SMA)
- SMA Embarcado

PANTOJA, Carlos Eduardo; STABILE, Márcio Fernando; LAZARIN, Nilson Mori; SICHMAN, Jaime Simão. ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In: BALDONI, Matteo; MÜLLER, Jörg P.; NUNES, Ingrid; ZALILA-WENKSTERN, Rym (orgs.). Engineering Multi-Agent Systems. Lecture Notes in Computer Science. 10093. ed. Cham: Springer International Publishing, 2016. p. 136–155. Disponível em: https://link.springer.com/chapter/10.1007/978-3-319-50983-9_8.

Introdução

- Agente
- Sistemas Multiagentes (SMA)
- SMA Embarcado
 - é um sistema cognitivo baseado em:
 - **hardware**, responsável pelas percepção e atuação no ambiente real,
 - e **software**, responsável pelo raciocínio e controle do dispositivo.

PANTOJA, Carlos Eduardo; STABILE, Márcio Fernando; LAZARIN, Nilson Mori; SICHMAN, Jaime Simão. ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In: BALDONI, Matteo; MÜLLER, Jörg P.; NUNES, Ingrid; ZALILA-WENKSTERN, Rym (orgs.). Engineering Multi-Agent Systems. Lecture Notes in Computer Science. 10093. ed. Cham: Springer International Publishing, 2016. p. 136–155. Disponível em: https://link.springer.com/chapter/10.1007/978-3-319-50983-9_8.

Introdução

- Agente
- Sistemas Multiagentes (SMA)
- SMA Embarcado
 - é um sistema cognitivo baseado em:
 - **hardware**, responsável pelas percepção e atuação no ambiente real,
 - e **software**, responsável pelo raciocínio e controle do dispositivo.
 - **Não há controle remoto ou processamento externo.**

PANTOJA, Carlos Eduardo; STABILE, Márcio Fernando; LAZARIN, Nilson Mori; SICHMAN, Jaime Simão. ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In: BALDONI, Matteo; MÜLLER, Jörg P.; NUNES, Ingrid; ZALILA-WENKSTERN, Rym (orgs.). Engineering Multi-Agent Systems. Lecture Notes in Computer Science. 10093. ed. Cham: Springer International Publishing, 2016. p. 136–155. Disponível em: https://link.springer.com/chapter/10.1007/978-3-319-50983-9_8.

Introdução

Não há controle remoto ou processamento externo (comparação).

Introdução

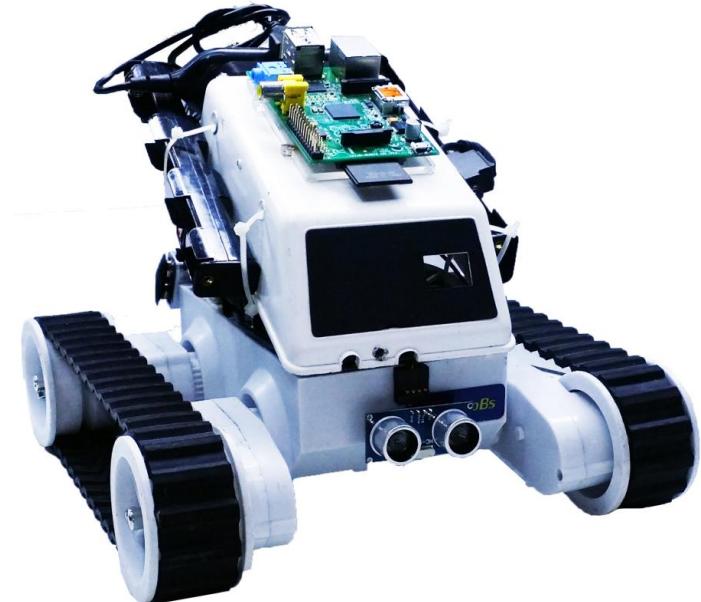
Não há controle remoto ou processamento externo (comparação).



Battlebots



RoboCup



SMA Embarcado

Introdução

Não há controle remoto ou processamento externo (comparação).

The screenshot shows the homepage of the CTC PUC-Rio website. At the top left is the PUC-Rio logo and the acronym CTC. To the right is a navigation menu icon. Below the header is a large text box containing the following statement:

Minotauro, da RioBotz/PUC-Rio, é vice-campeão da BattleBots, programa de TV dos EUA



Créditos: <https://www.ctc.puc-rio.br/minotauro-da-riobotz-puc-rio-e-vice-campeao-da-battlebots-programa-de-tv-dos-eua>

Conheça: <https://www.riobotz.com/>

Introdução

Não há controle remoto ou processamento externo (comparação).

CTC
CENTRO TÉCNICO CIENTÍFICO / PUC-RIO

Minotauro, da RioBotz/PUC-Rio, é vice-campeão da BattleBots, programa de TV dos EUA

Créditos: <https://www.ctc.puc-rio.br/minotauro-da-riobotz-puc-rio-e-vice-campeao-da-battlebots-programa-de-tv-dos-eua>

Conheça: <https://www.riobotz.com/>



Introdução

Não há controle remoto ou processamento externo (comparação).



COBERTURA ESPECIAL - TECNOLOGIA DISRUPTIVA - TERRESTRE

02 de Julho, 2021 - 08:00 (Brasília)

Instituto Militar de Engenharia é vice-campeão de maior competição de robótica do mundo



Créditos: <https://www.defesanet.com.br/tecdi/noticia/41217/Instituto-Militar-de-Engenharia-e-vice-campeao-de-maior-competicao-de-robotica-do-mundo-/>

Conheça: <http://roboime.com.br/>

Introdução

Não há controle remoto ou processamento externo (comparação).



COBERTURA ESPECIAL - TECNOLOGIA DISRUPTIVA - TERRESTRE
02 de Julho, 2021 - 08:00 (Brasília)

Instituto Militar de Engenharia é vice-campeão de maior competição de robótica do mundo



Créditos: <https://www.defesanet.com.br/tecdi/noticia/41217/Instituto-Militar-de-Engenharia-e-vice-campeao-de-maior-competicao-de-robotica-do-mundo-/>

Conheça: <http://roboime.com.br/>

Introdução

Não há controle remoto ou processamento externo (comparação).



VEÍCULOS AUTÔNOMOS MOVIDOS POR SMAS: CONSTRUÇÃO DE UM PROTÓTIPO USANDO ARDUINO E RASPBERRY

Orientadores: Nilson Mori Lazarin; Carlos Eduardo Pantoja

nilson.lazarin@cefet-rj.br; carlos.pantoja@cefet-rj.br

Alunos: Leonam Ramos Foli; Dayana da Silva Junger; João Victor Guinelli da Silva
ramosfoli@gmail.com; dayanacomputer@hotmail.com; joao.silva@cefet-rj.br

The image shows the theme banner for the "Semana de Extensão 2015" with the text "LUZ, CIÉNCIA E VIDA" overlaid on a background of green leaves and branches.



Introdução

Não há controle remoto ou processamento externo (comparação).

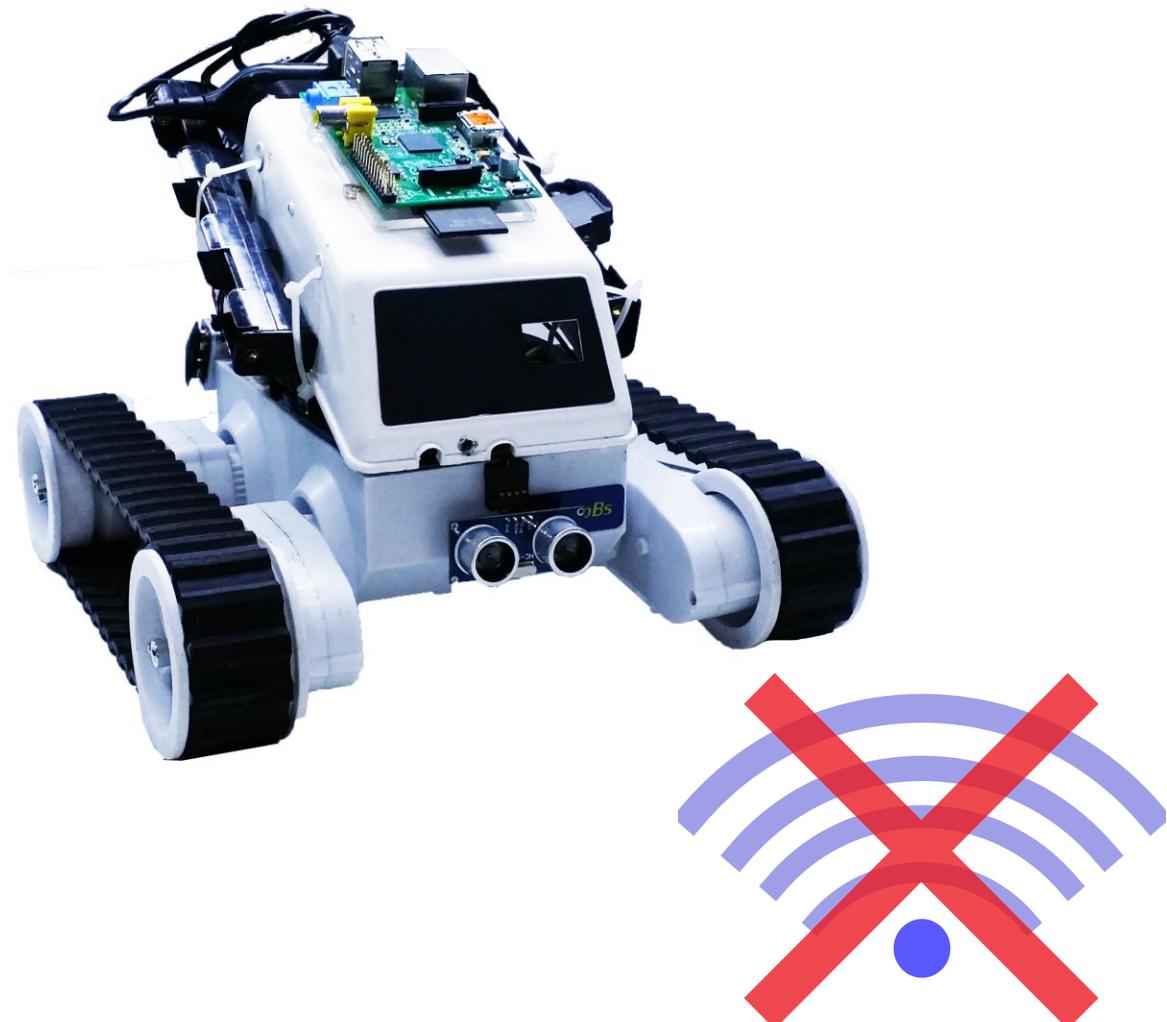


VEÍCULOS AUTÔNOMOS MOVIDOS POR SMAS: CONSTRUÇÃO DE UM PROTÓTIPO USANDO ARDUINO E RASPBERRY

Orientadores: Nilson Mori Lazarin; Carlos Eduardo Pantoja

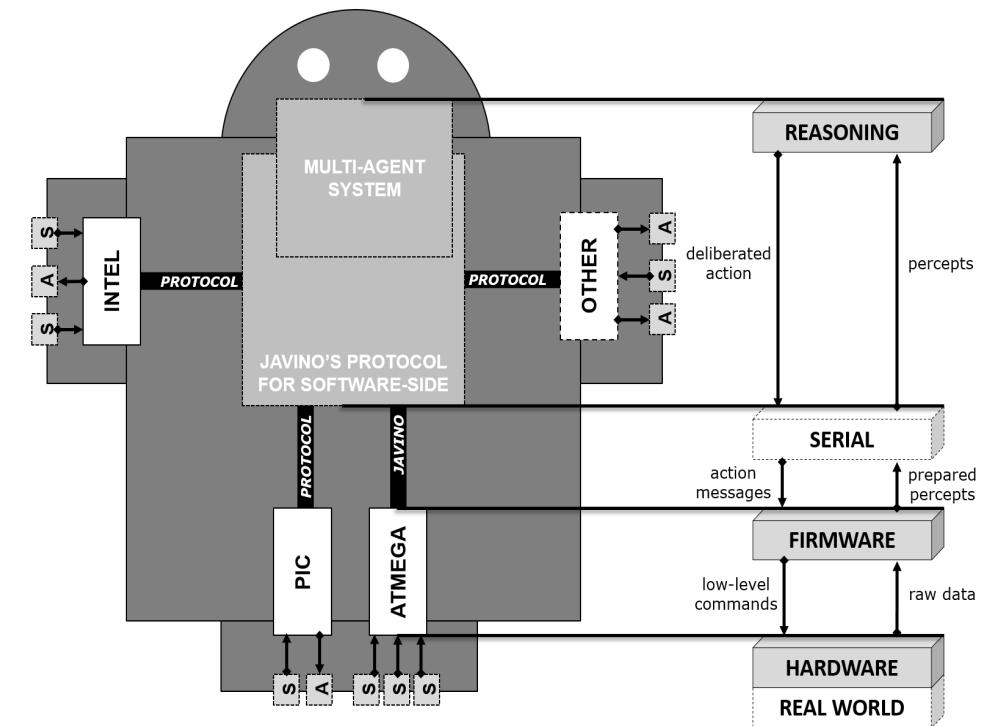
nilson.lazarin@cefet-rj.br; carlos.pantoja@cefet-rj.br

Alunos: Leonam Ramos Foli; Dayana da Silva Junger; João Victor Guinelli da Silva
ramosfoli@gmail.com; dayanacomputer@hotmail.com; joao.silva@cefet-rj.br



Introdução

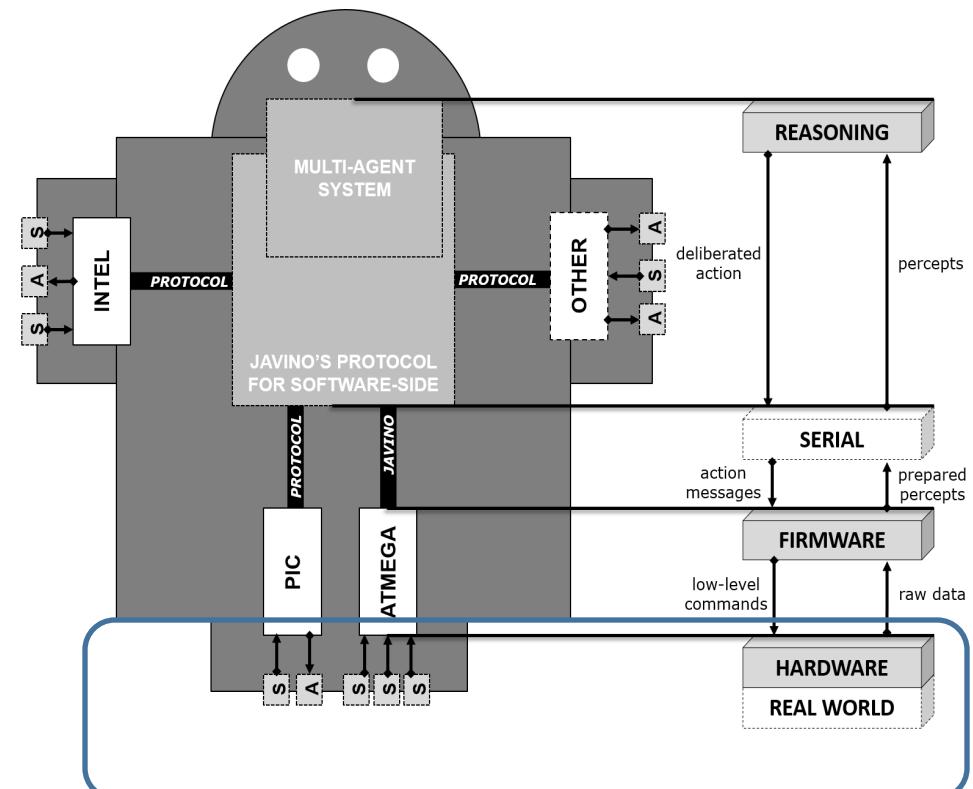
- Agentes
- Sistemas Multiagentes (SMA)
- SMA Embarcado
- Camadas do SMA Embarcado



PANTOJA, Carlos Eduardo; STABILE, Márcio Fernando; LAZARIN, Nilson Mori; SICHMAN, Jaime Simão. ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In: BALDONI, Matteo; MÜLLER, Jörg P.; NUNES, Ingrid; ZALILA-WENKSTERN, Rym (orgs.). Engineering Multi-Agent Systems. Lecture Notes in Computer Science. 10093. ed. Cham: Springer International Publishing, 2016. p. 136–155. Disponível em: https://link.springer.com/chapter/10.1007/978-3-319-50983-9_8.

Introdução

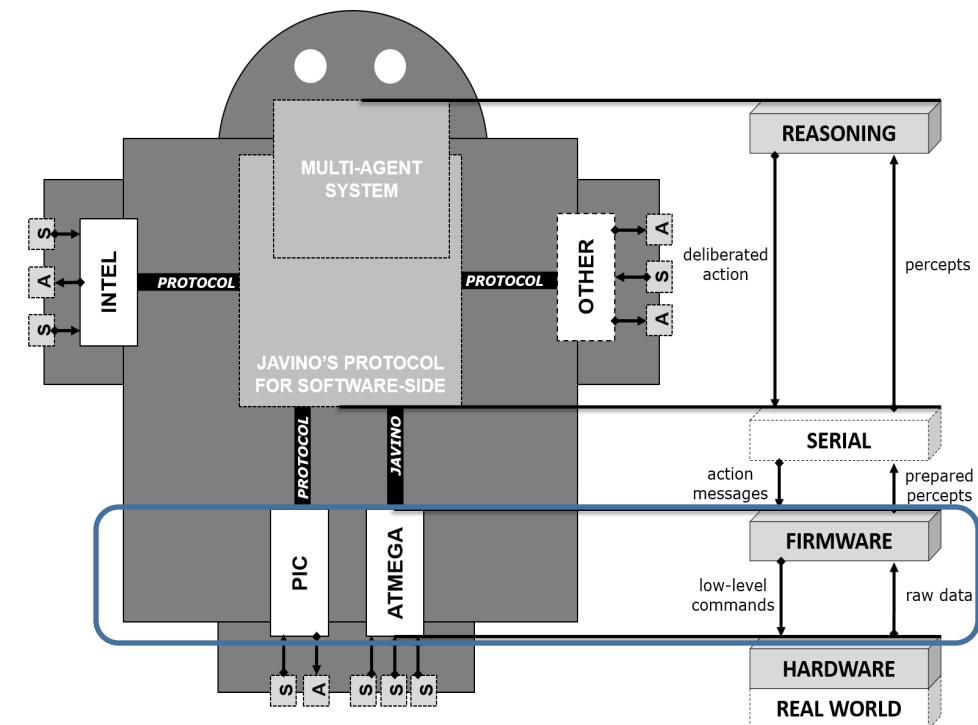
- Agentes
- Sistemas Multiagentes (SMA)
- SMA Embarcado
- Camadas do SMA Embarcado
 - **hardware:** conjunto de recursos que representam o ambiente do agente no mundo físico;



PANTOJA, Carlos Eduardo; STABILE, Márcio Fernando; LAZARIN, Nilson Mori; SICHMAN, Jaime Simão. ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In: BALDONI, Matteo; MÜLLER, Jörg P.; NUNES, Ingrid; ZALILA-WENKSTERN, Rym (orgs.). Engineering Multi-Agent Systems. Lecture Notes in Computer Science. 10093. ed. Cham: Springer International Publishing, 2016. p. 136–155. Disponível em: https://link.springer.com/chapter/10.1007/978-3-319-50983-9_8.

Introdução

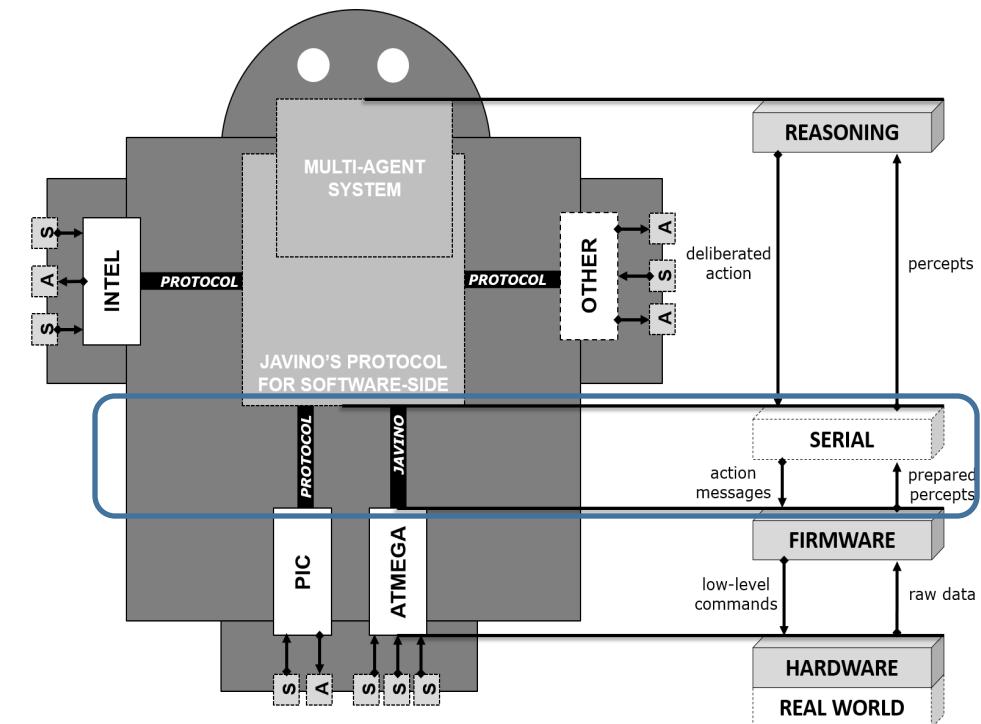
- Agentes
- Sistemas Multiagentes (SMA)
- SMA Embarcado
- Camadas do SMA Embarcado
 - **hardware:** conjunto de *recursos* que representam o ambiente do agente no mundo físico;
 - **firmware:** hospedada em um ou mais microcontroladores que manipulam a camada de hardware, conforme as deliberações do agente;



PANTOJA, Carlos Eduardo; STABILE, Márcio Fernando; LAZARIN, Nilson Mori; SICHMAN, Jaime Simão. ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In: BALDONI, Matteo; MÜLLER, Jörg P.; NUNES, Ingrid; ZALILA-WENKSTERN, Rym (orgs.). Engineering Multi-Agent Systems. Lecture Notes in Computer Science. 10093. ed. Cham: Springer International Publishing, 2016. p. 136–155. Disponível em: https://link.springer.com/chapter/10.1007/978-3-319-50983-9_8.

Introdução

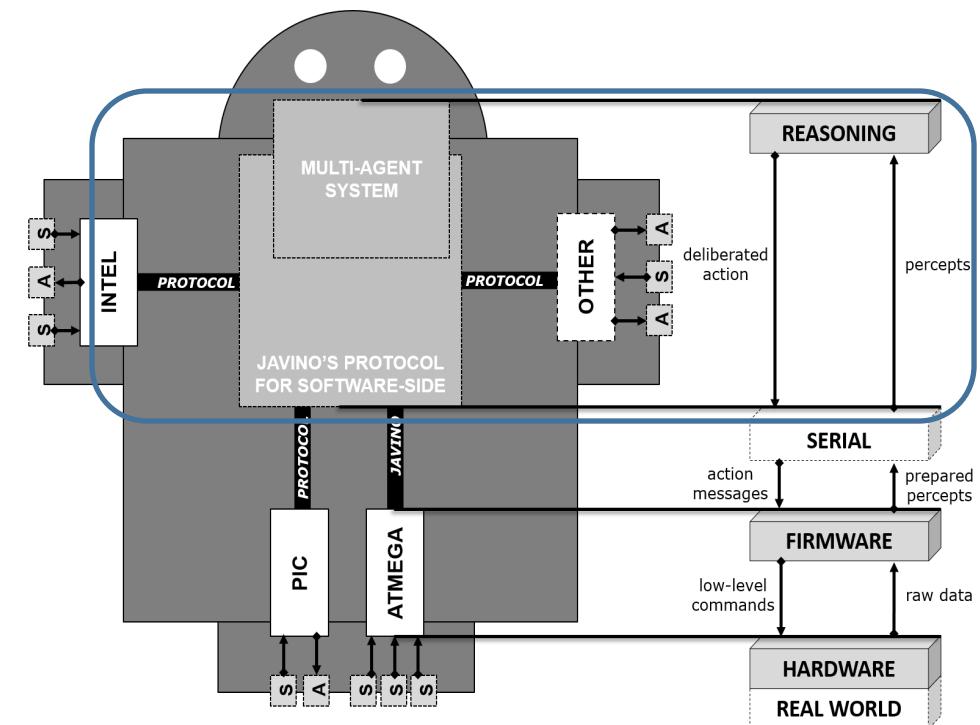
- Agentes
- Sistemas Multiagentes (SMA)
- SMA Embarcado
- Camadas do SMA Embarcado
 - **hardware**: conjunto de recursos que representam o ambiente do agente no mundo físico;
 - **firmware**: hospedada em um ou mais microcontroladores que manipulam a camada de hardware, conforme as deliberações do agente;
 - **interfaceamento**: permite a comunicação do agente com o microcontrolador;



PANTOJA, Carlos Eduardo; STABILE, Márcio Fernando; LAZARIN, Nilson Mori; SICHMAN, Jaime Simão. ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In: BALDONI, Matteo; MÜLLER, Jörg P.; NUNES, Ingrid; ZALILA-WENKSTERN, Rym (orgs.). Engineering Multi-Agent Systems. Lecture Notes in Computer Science. 10093. ed. Cham: Springer International Publishing, 2016. p. 136–155. Disponível em: https://link.springer.com/chapter/10.1007/978-3-319-50983-9_8.

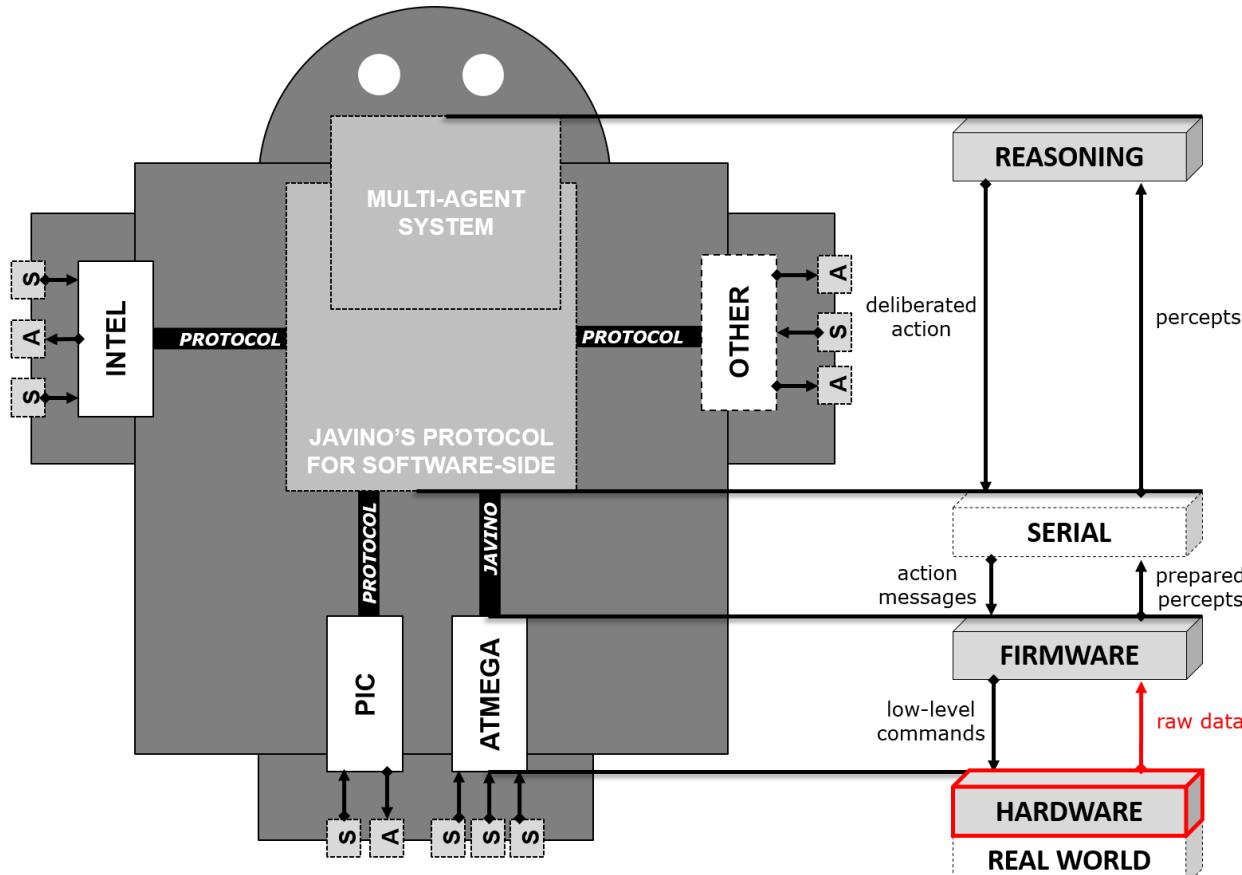
Introdução

- Agentes
- Sistemas Multiagentes (SMA)
- SMA Embarcado
- Camadas do SMA Embarcado
 - **hardware**: conjunto de recursos que representam o ambiente do agente no mundo físico;
 - **firmware**: hospedada em um ou mais microcontroladores que manipulam a camada de hardware, conforme as deliberações do agente;
 - **interfaceamento**: permite a comunicação do agente com o microcontrolador;
 - **raciocínio**: é um SMA hospedado em um computador que executa o controle do dispositivo onde estiver embarcado.



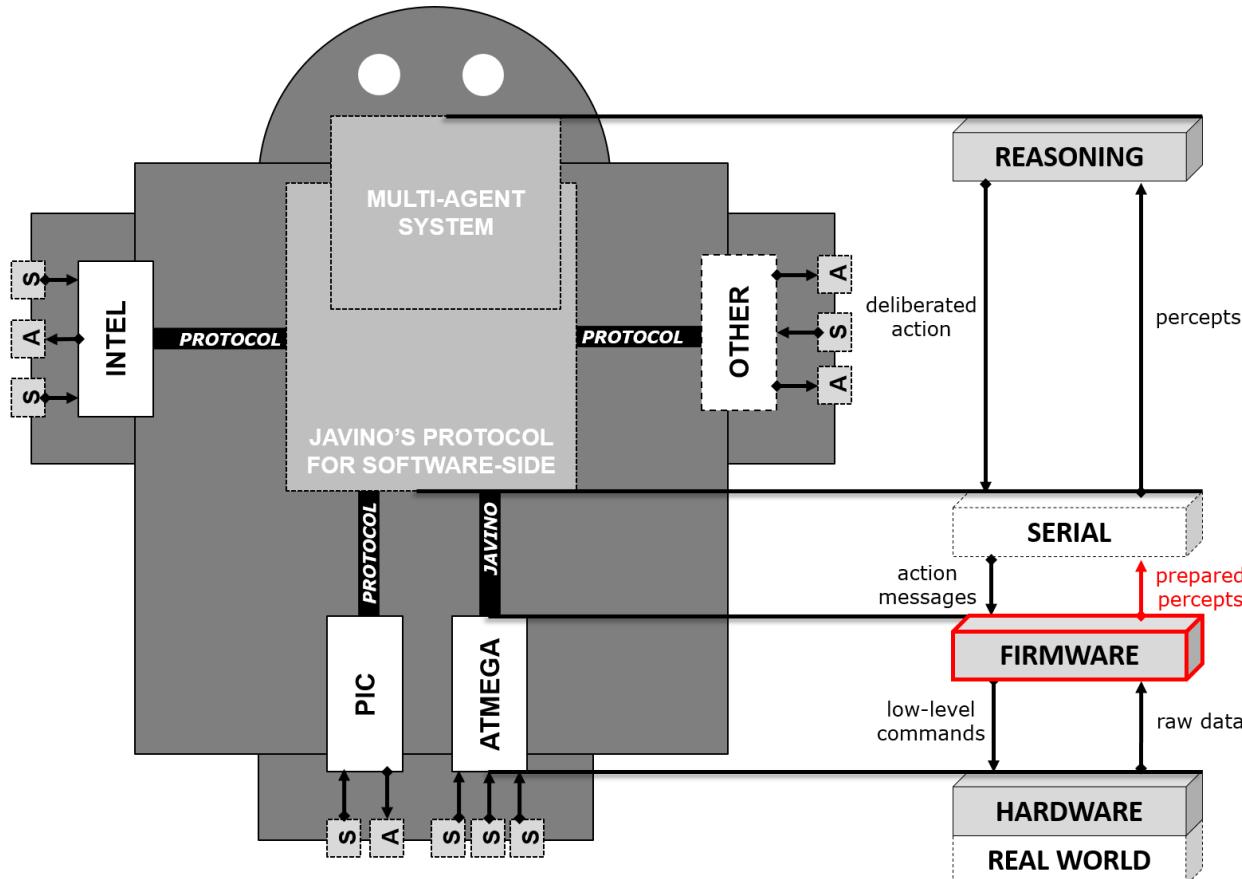
PANTOJA, Carlos Eduardo; STABILE, Márcio Fernando; LAZARIN, Nilson Mori; SICHMAN, Jaime Simão. ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In: BALDONI, Matteo; MÜLLER, Jörg P.; NUNES, Ingrid; ZALILA-WENKSTERN, Rym (orgs.). Engineering Multi-Agent Systems. Lecture Notes in Computer Science. 10093. ed. Cham: Springer International Publishing, 2016. p. 136–155. Disponível em: https://link.springer.com/chapter/10.1007/978-3-319-50983-9_8.

Arquitetura Física e Lógica: Fluxo da Mensagem



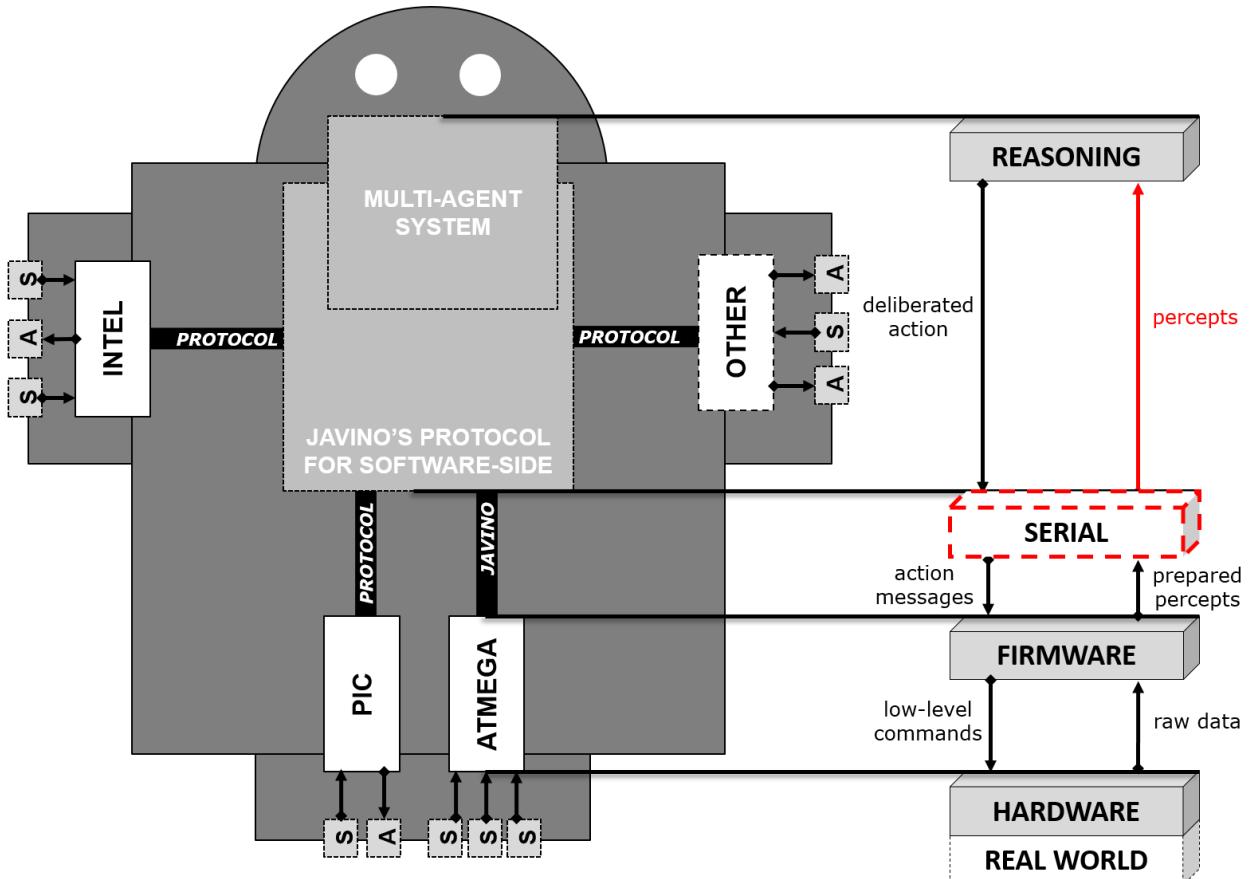
Sensores capturam dados brutos do mundo real e os enviam para um dos microcontroladores escolhido para o projeto.

Arquitetura Física e Lógica: Fluxo da Mensagem



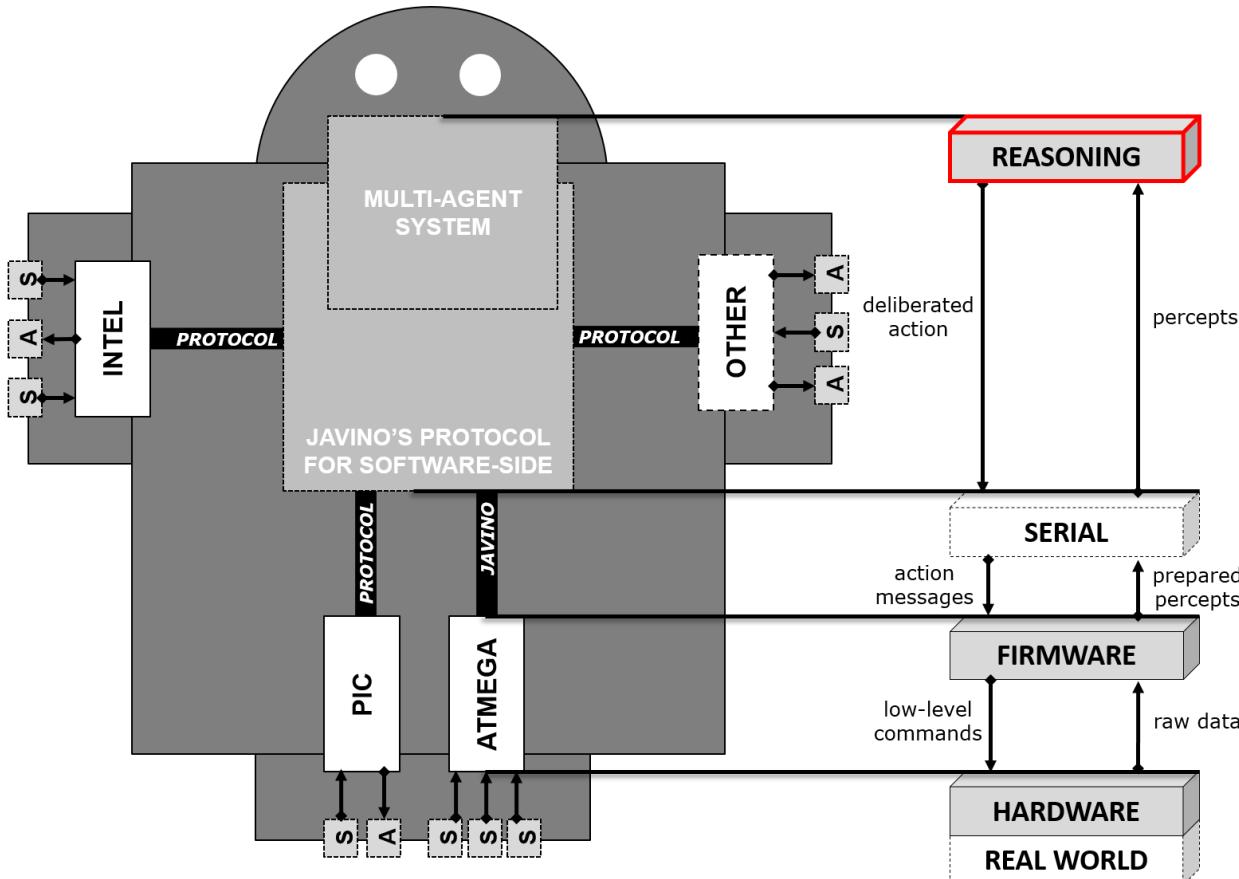
Na programação do microcontrolador, os dados brutos são transformados em percepções baseado na linguagem de programação orientada a agentes escolhida.

Arquitetura Física e Lógica: Fluxo da Mensagem



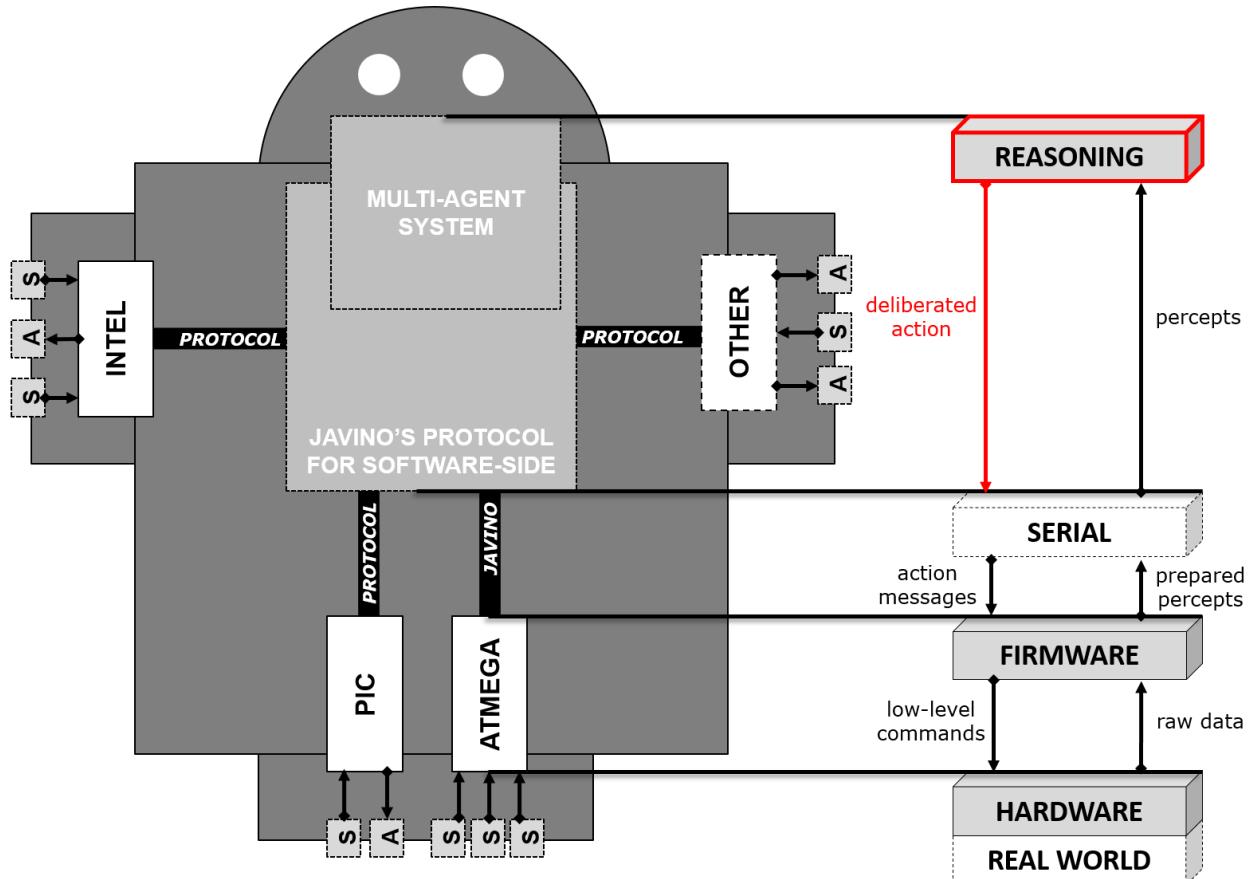
É responsável por enviar as percepções para a camada de raciocínio usando a comunicação serial.

Arquitetura Física e Lógica: Fluxo da Mensagem



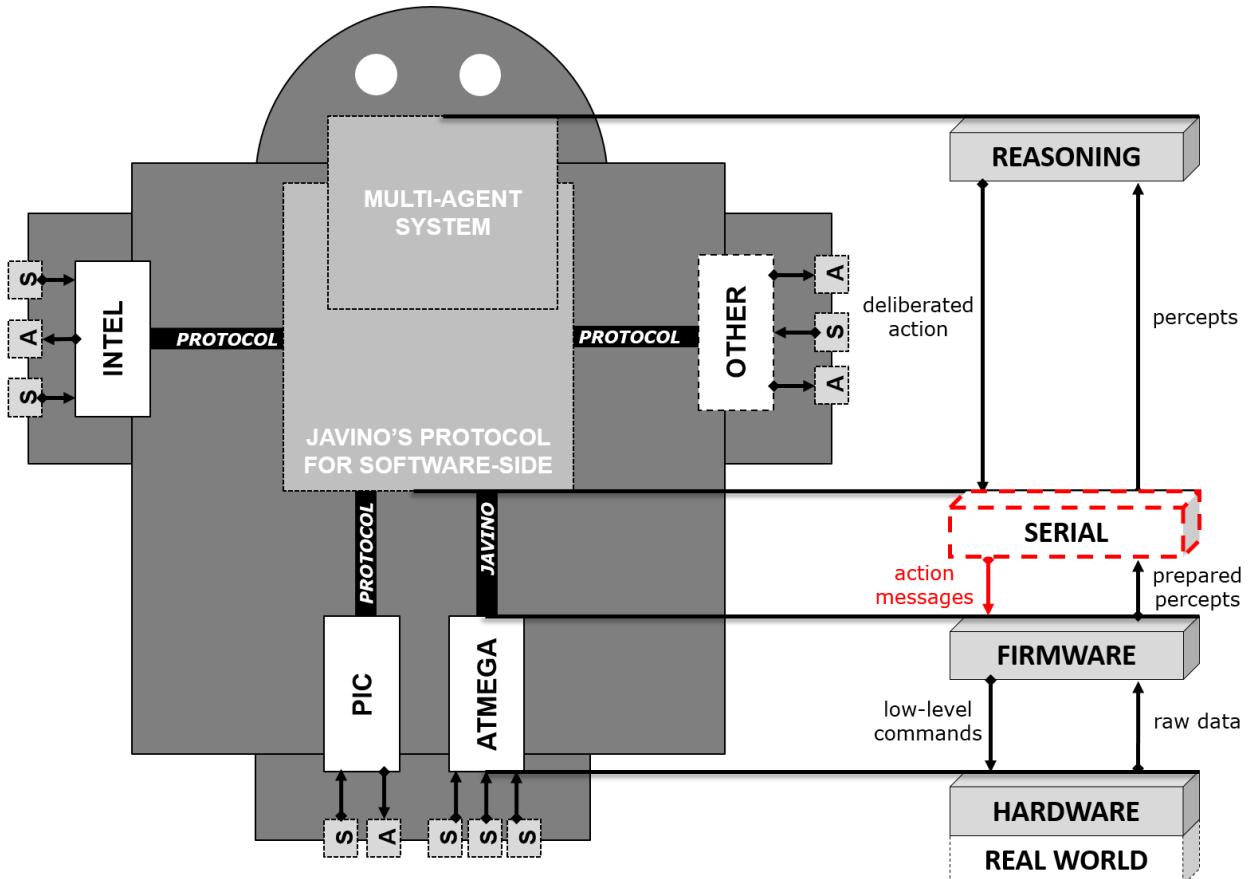
O agente é capaz de raciocinar com as percepções que vem diretamente do mundo real.

Arquitetura Física e Lógica: Fluxo da Mensagem



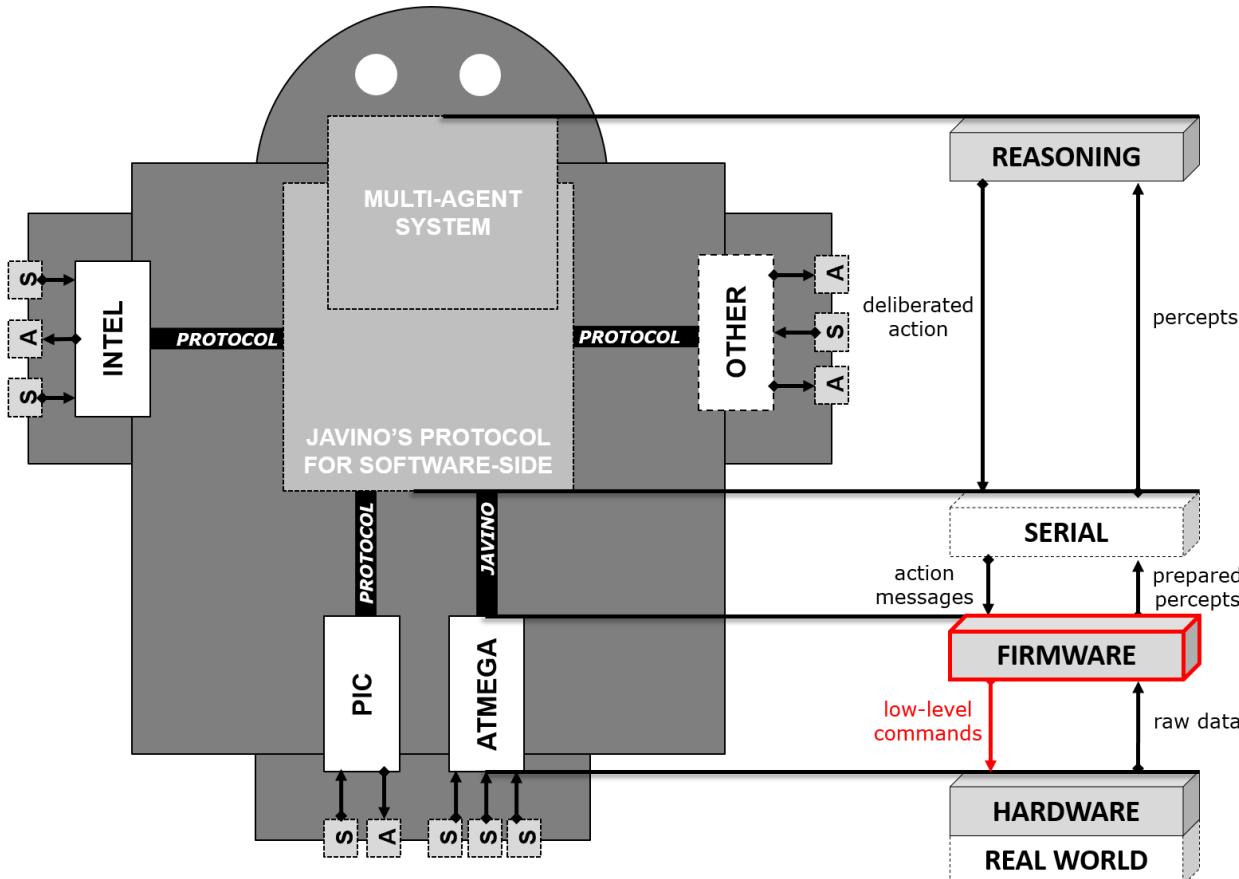
Então, o agente delibera e se alguma ação precisar ser executada. Neste caso, uma mensagem é enviada a camada serial.

Arquitetura Física e Lógica: Fluxo da Mensagem



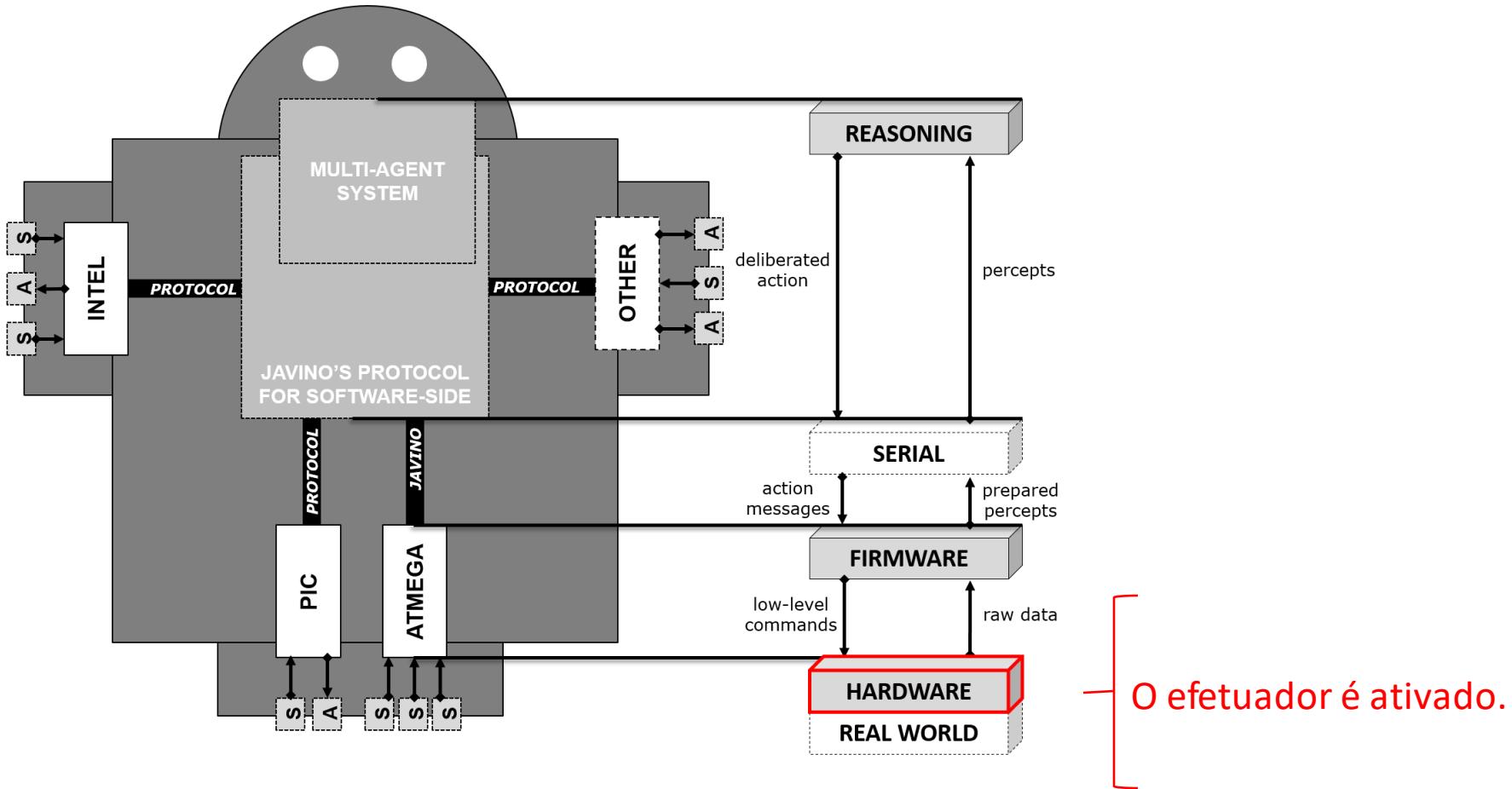
Redireciona as mensagens de ações para o microcontrolador que está conectado na porta USB identificado na mensagem.

Arquitetura Física e Lógica: Fluxo da Mensagem



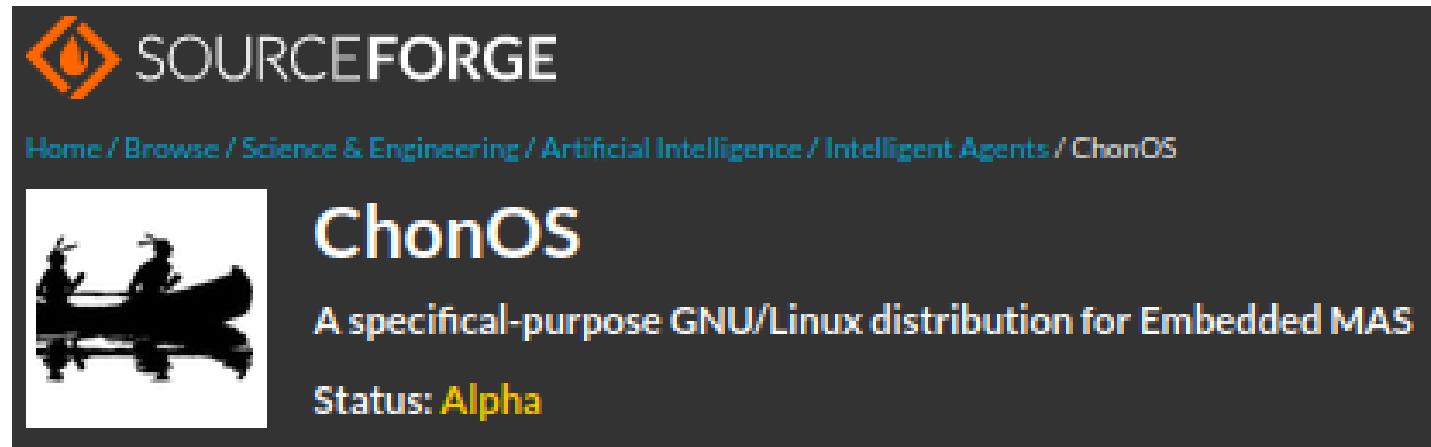
Todas as funções possíveis dos atuadores são programadas para serem executadas em resposta às mensagens vinda da porta serial.

Arquitetura Física e Lógica: Fluxo da Mensagem



ChonOS

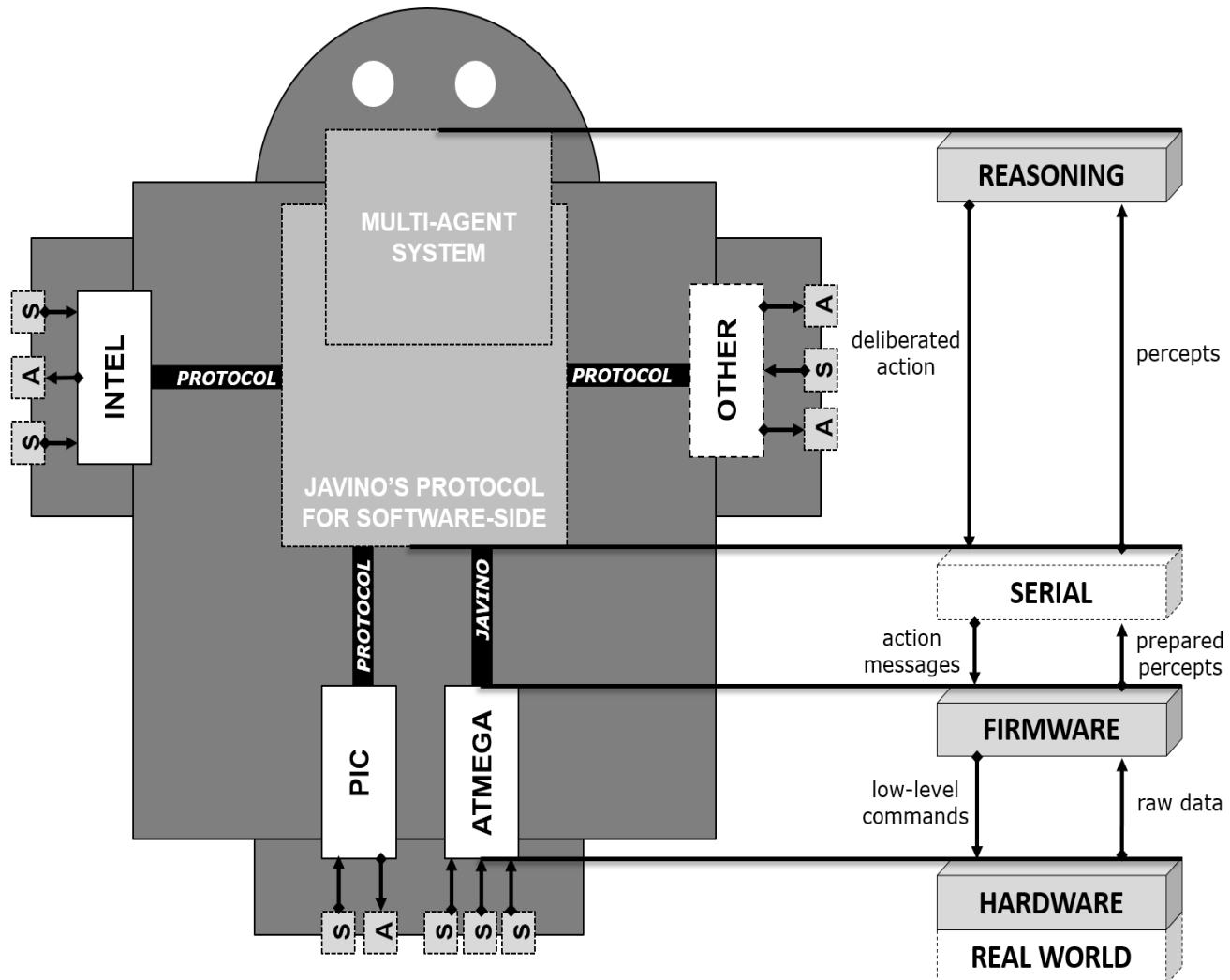
(Cognitive Hardware on Network - Operational System)



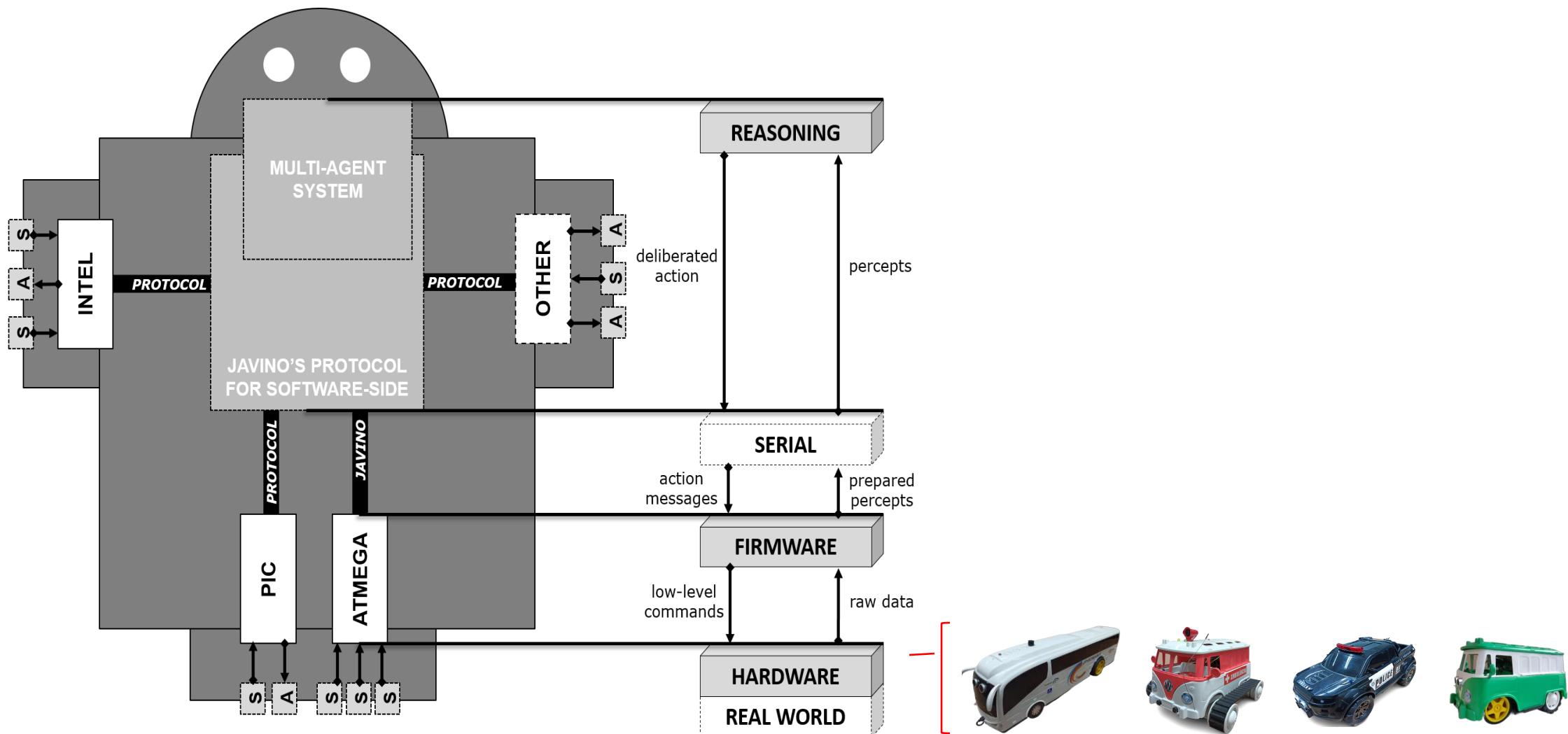
Motivação para o desenvolvimento de uma distro?

- Integração e configuração das diversas tecnologias

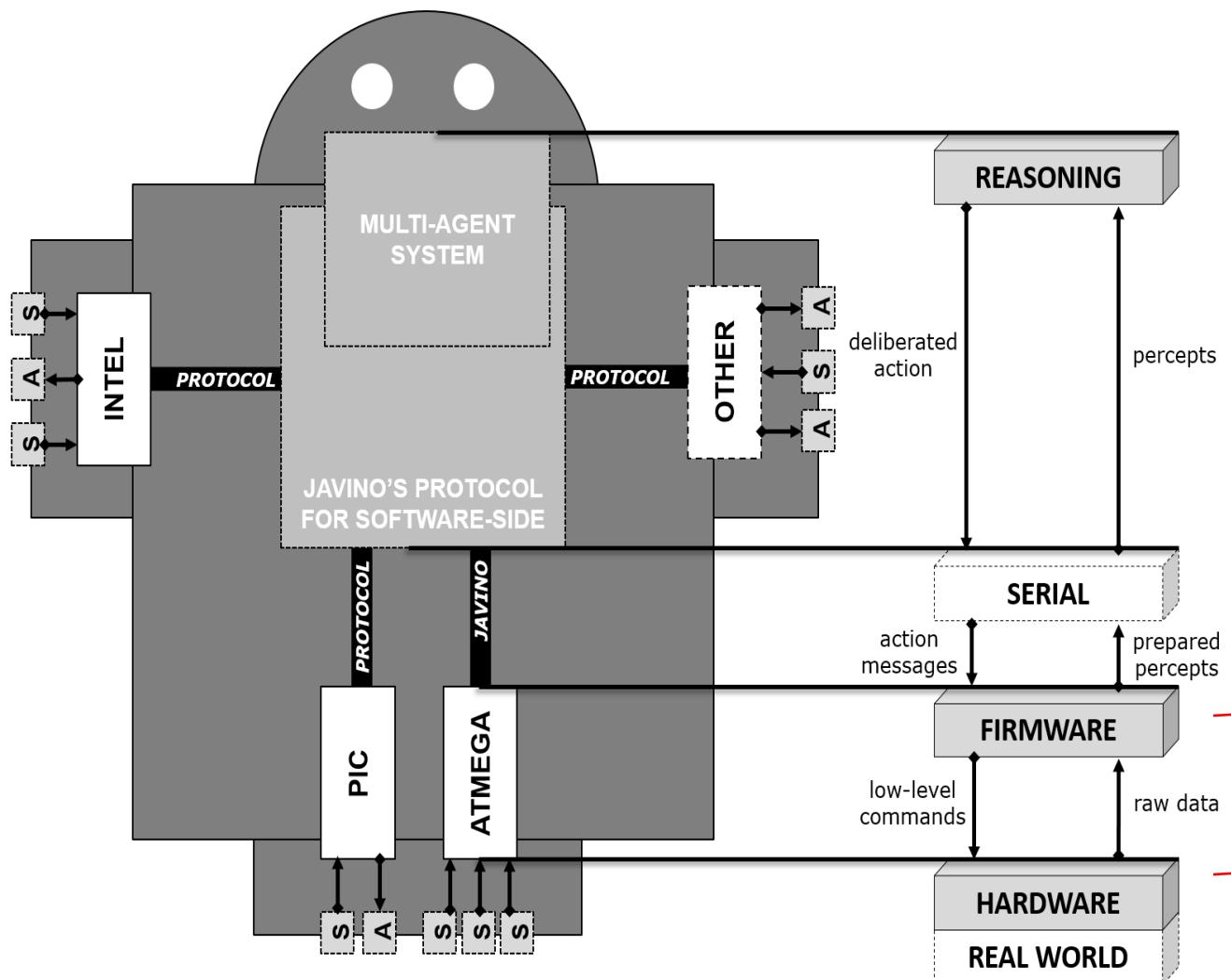
Tecnologias de Desenvolvimento: Visão geral



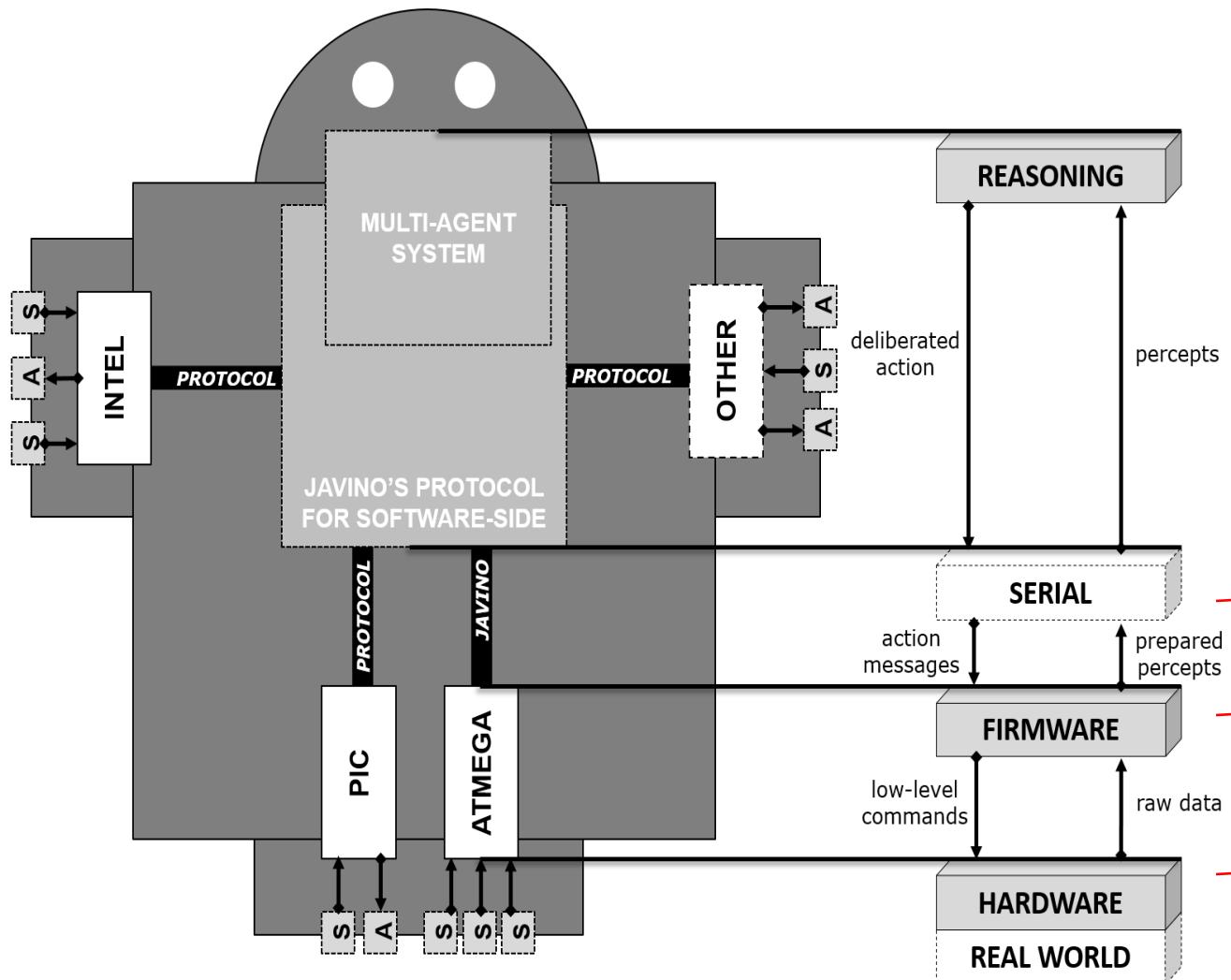
Tecnologias de Desenvolvimento: Visão geral



Tecnologias de Desenvolvimento: Visão geral



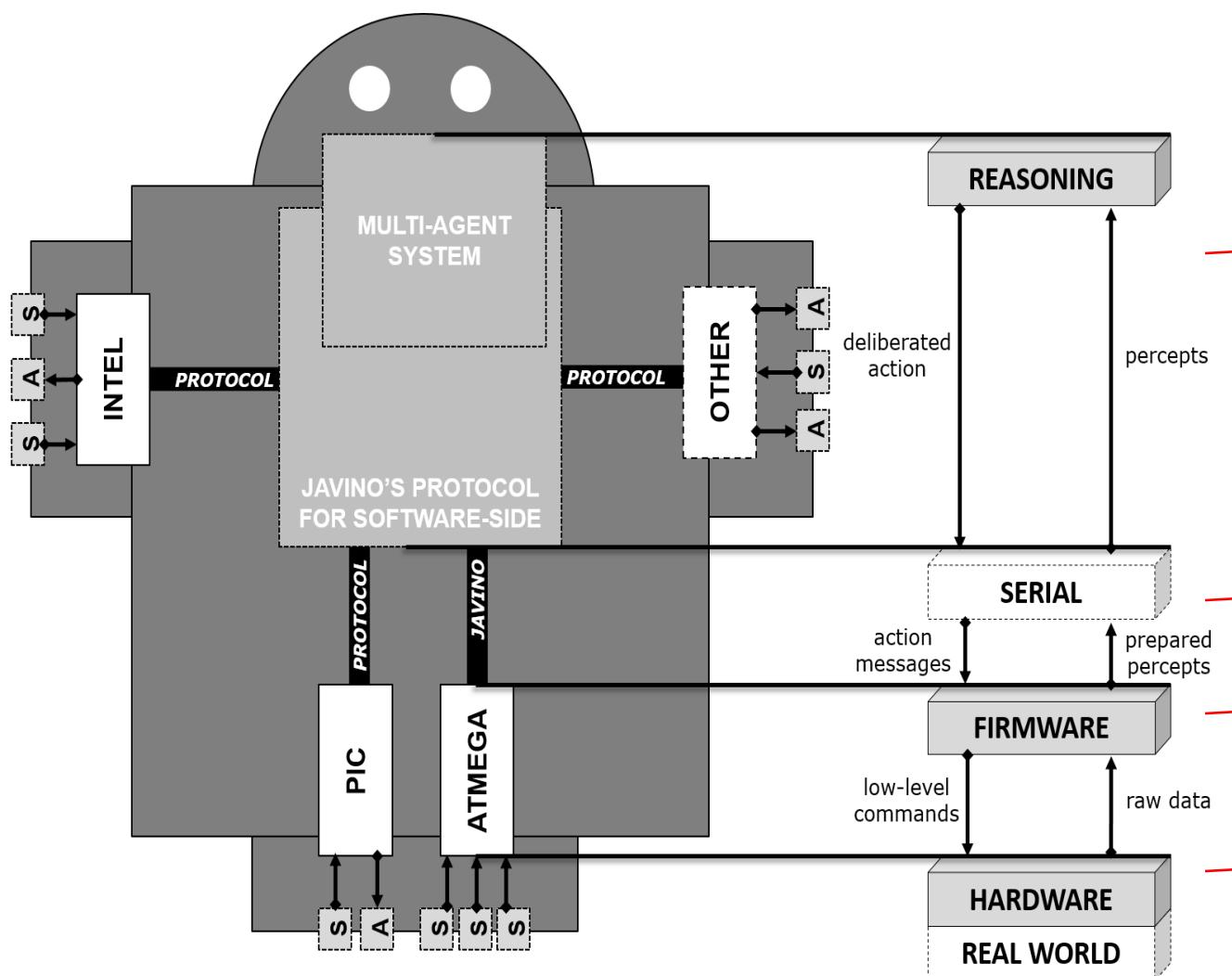
Tecnologias de Desenvolvimento: Visão geral



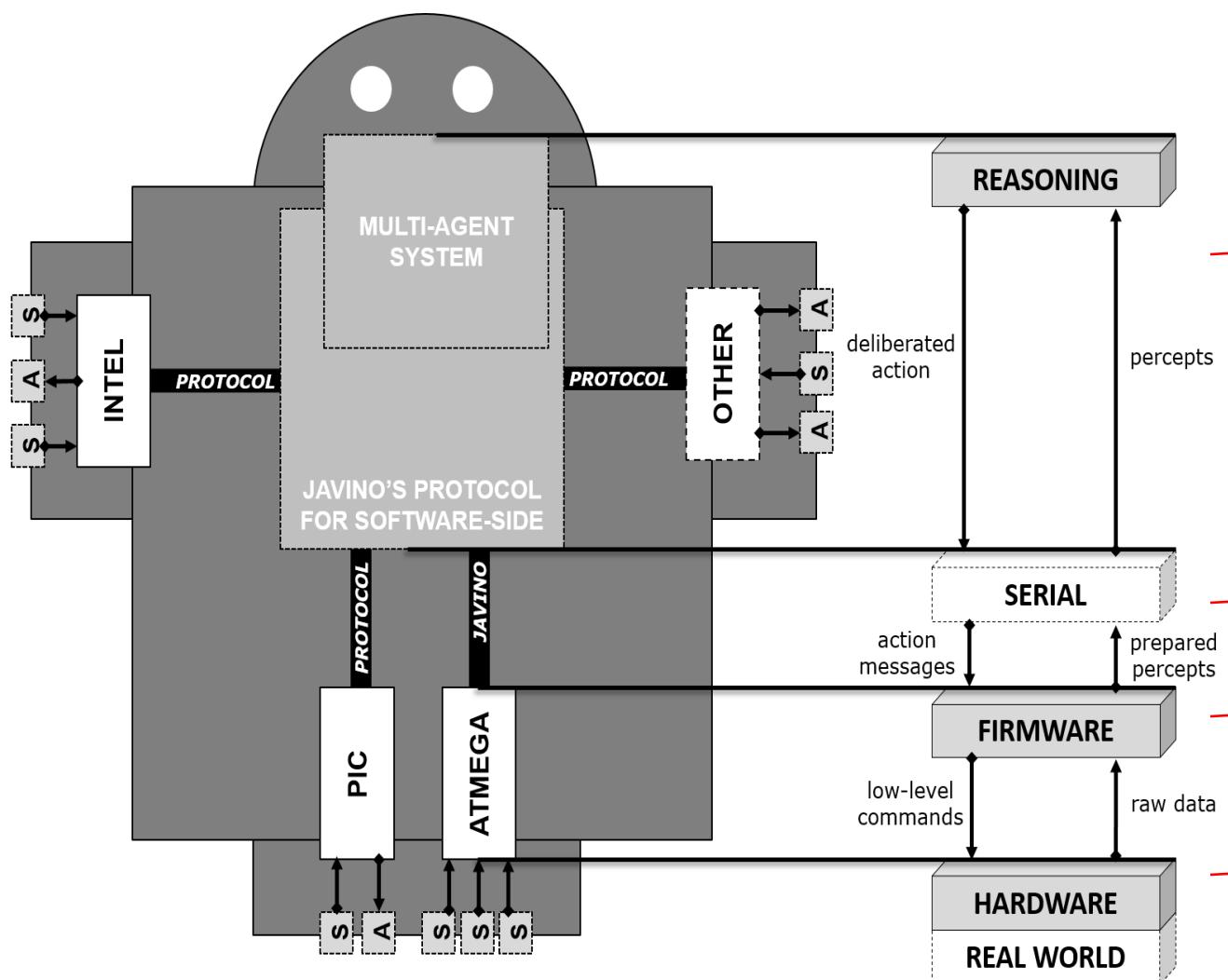
JavINO
— + ™
ARDUINO



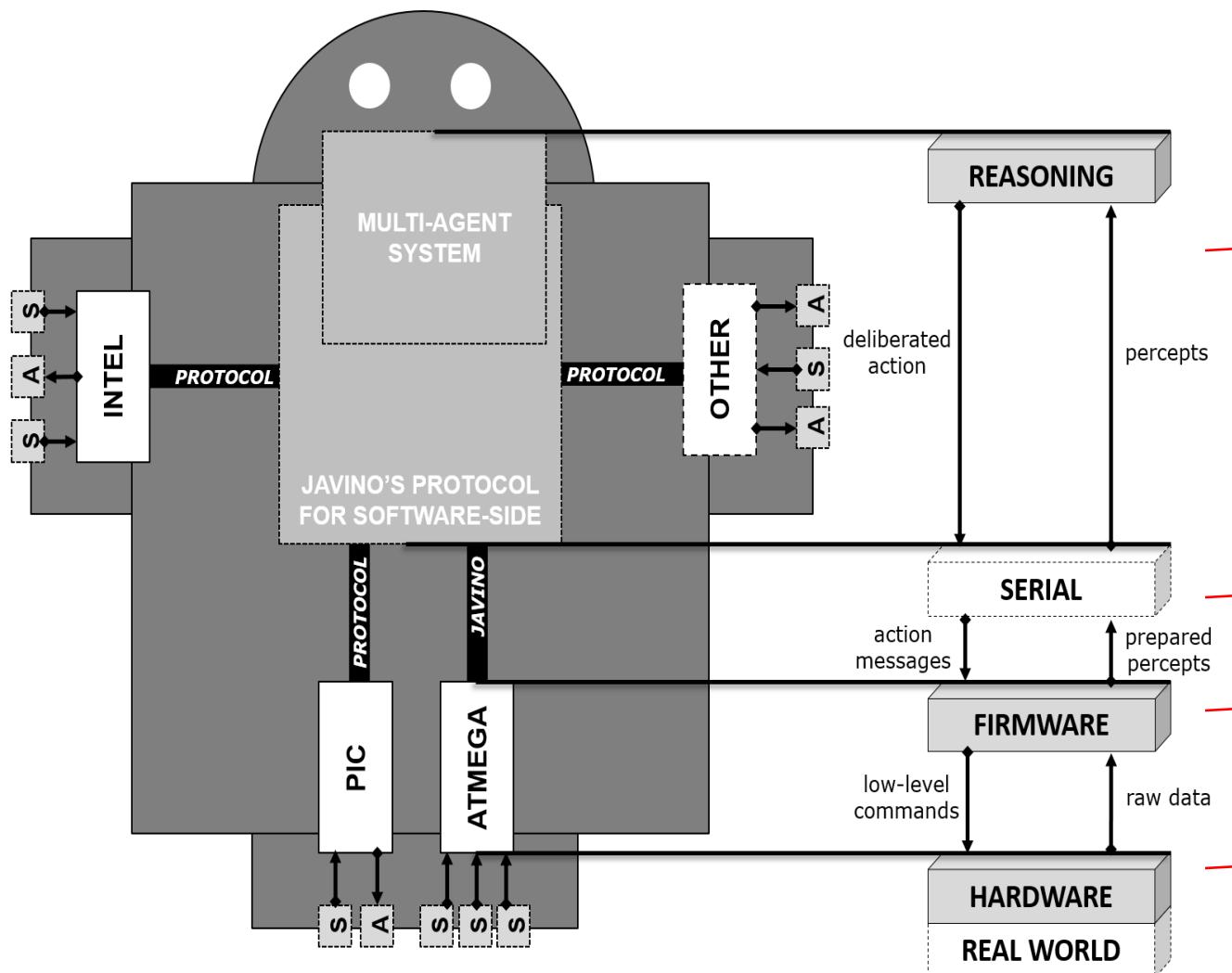
Tecnologias de Desenvolvimento: Visão geral



Tecnologias de Desenvolvimento: Visão geral



Tecnologias de Desenvolvimento: Visão geral



Abordagem Proposta

Abordagem Proposta

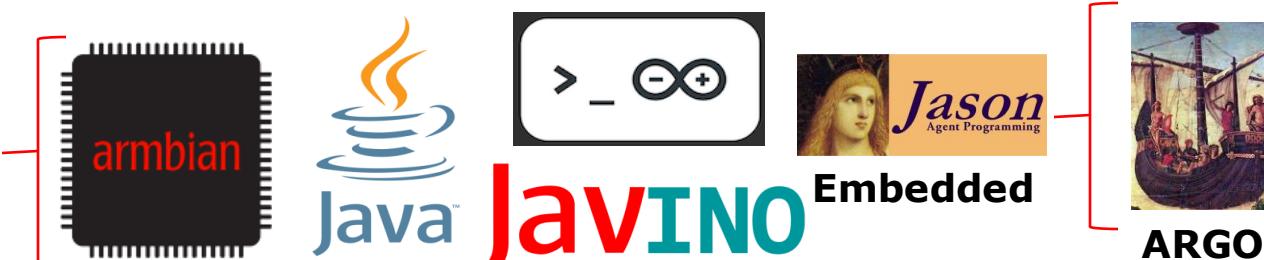


Embedded

Abordagem Proposta



Abordagem Proposta



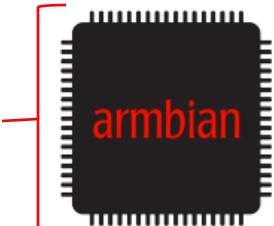
Comunicadores
+
Protocolos de
Transferências

Desenvolvimento de Sistemas MultiAgentes Embarcados

Abordagem Proposta



ChonOS



armbian



Java™



JavINO

Embedded
Jason
Agent Programming



ARGO

Comunicadores
+
Protocolos de Transferências

Abordagem Proposta



Desenvolvimento de Sistemas MultiAgentes Embarcados

Abordagem Proposta

The screenshot shows the chonIDE interface. The top menu bar includes: chonIDE, Redes, Nome do bot, Mind Inspector, Logs do SMA, Iniciar SMA, Parar SMA, and navigation icons. The left sidebar displays a project structure under 'mas_project': 'Sistema multiagente' (expanded) containing 'Agentes' (expanded) with 'Argo' selected, and 'Firmwares' (expanded) containing 'C++ sketch 1'. Other sections include 'Bibliotecas' and component names like 'HCSR04_ultrasonic_sensor' and 'Javino'. The main workspace shows a code editor for 'sketch 1' with the following C++ code:

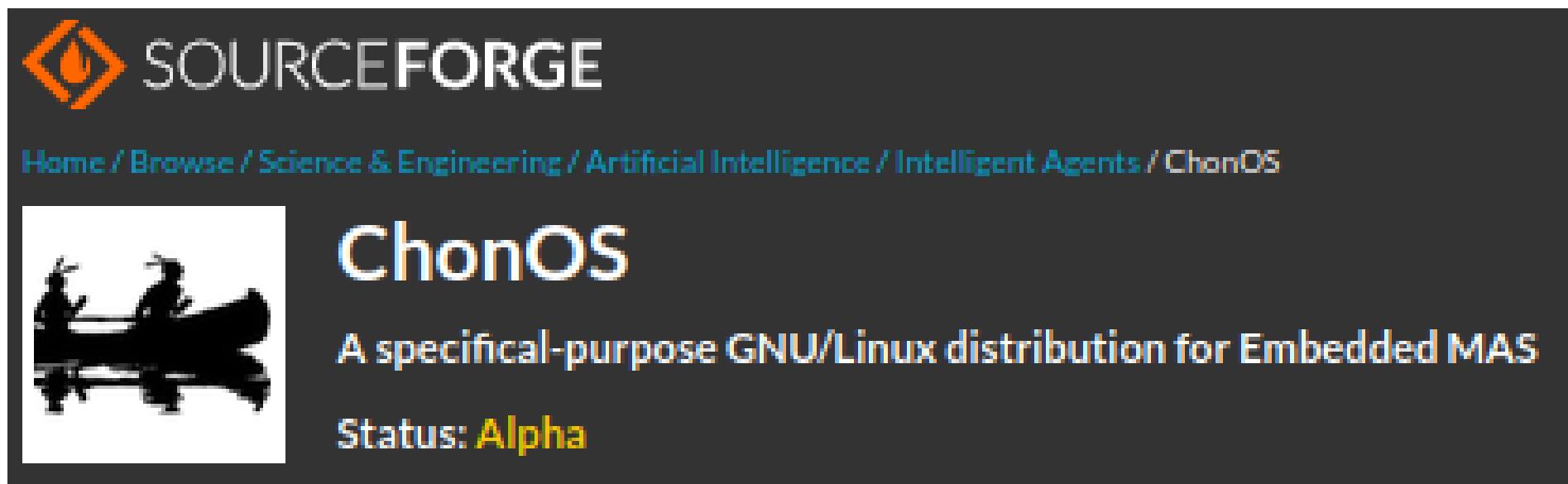
```
void setup() {  
    // put your setup code  
    // here, to run once:  
}  
  
void loop() {  
    // put your main code  
    // here, to run repeatedly:  
}
```

The right sidebar lists available boards: Arduino Uno, arduino:avr:uno, and /dev/ttyACM0.



ChonOS

(Cognitive Hardware on Network - Operational System)



<http://chonos.sf.net>

Baixar

Home / Browse / Science & Engineering / Artificial Intelligence / Intelligent Agents / ChonOS / Files

ChonOS Files
A specifical-purpose GNU/Linux distribution for Embedded MAS
Status: Alpha Brought to you by: kadupantoja, nilsonmori

Summary Files Reviews

Download Latest Version chonos-alfa3-BPI-M2-Zero.zip [2]

Home / alfa4

Name	Modified	Size
Parent folder		
chonos-alfa4-BPI-M2-Zero.zip	2022-09-22	1.2 GB
Totals: 1 Item		1.2 GB

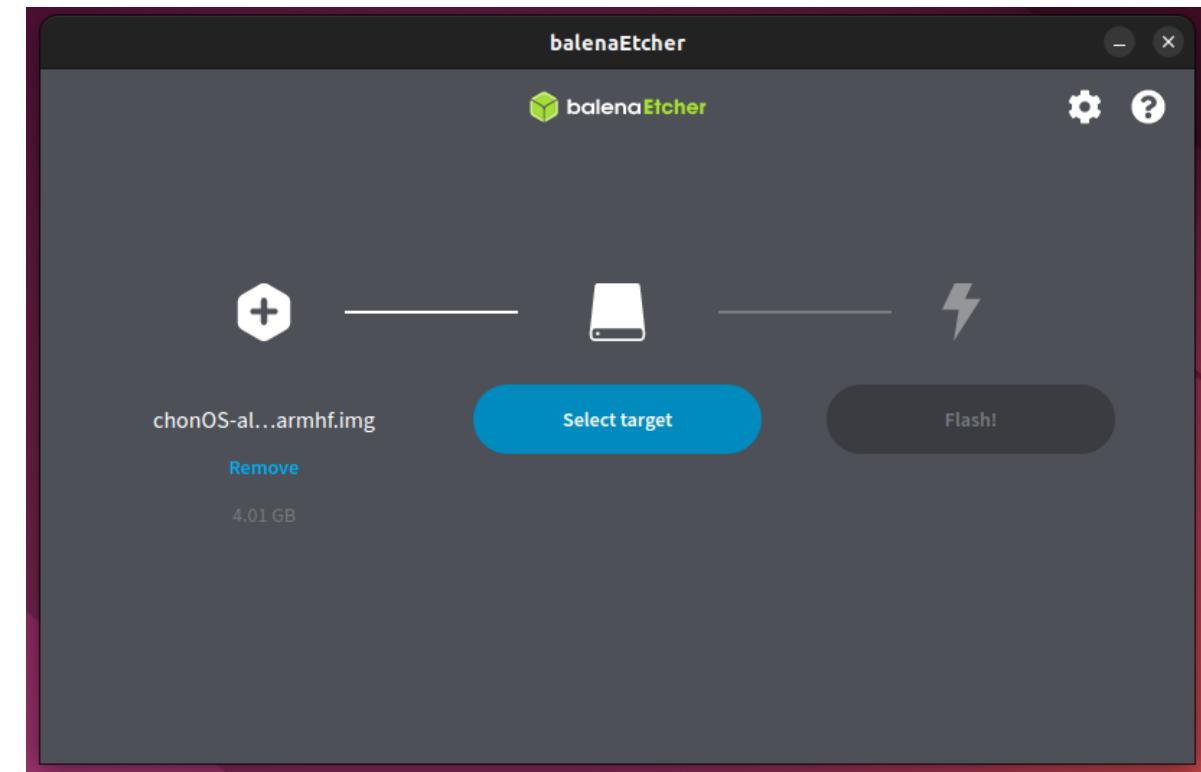
Home / alfa4 / BananaPi

Name	Modified	Size
Parent folder		
chonos-alfa4-BPI-M2-Zero.zip	2022-09-22	1.2 GB
Totals: 1 Item		1.2 GB

Home / alfa4 / RaspberryPi

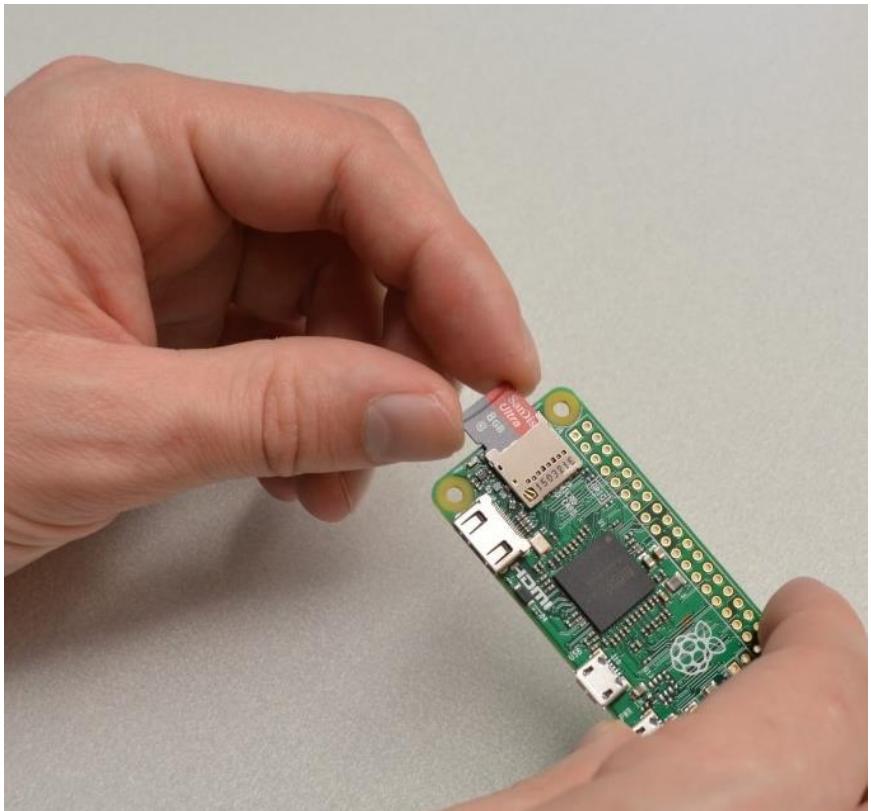
Name	Modified	Size
Parent folder		
chonos-alfa4-RPI-3-Model-B.zip	2022-09-22	2.2 GB
chonos-alfa4-RaspberryPi-1-Model-B-2012.zip	2022-09-22	1.2 GB
chonos-alfa4-RPI-Zero-W.zip	2022-09-22	1.3 GB
Totals: 3 Items		4.7 GB

Gravar

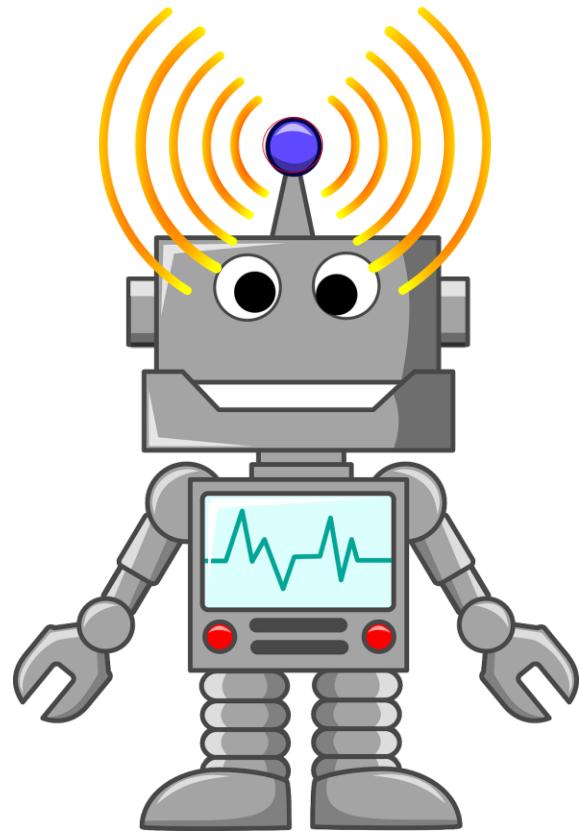


<https://balena.io/etcher>

Ligar

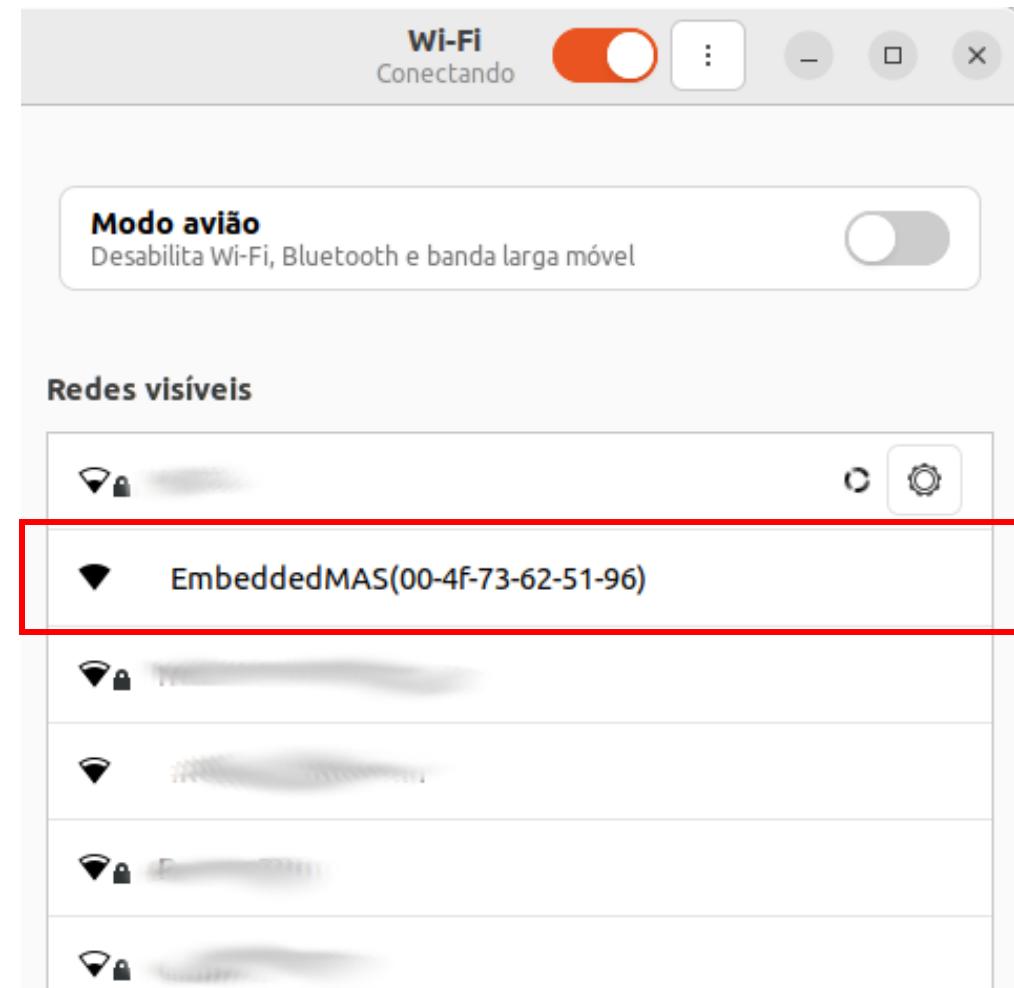
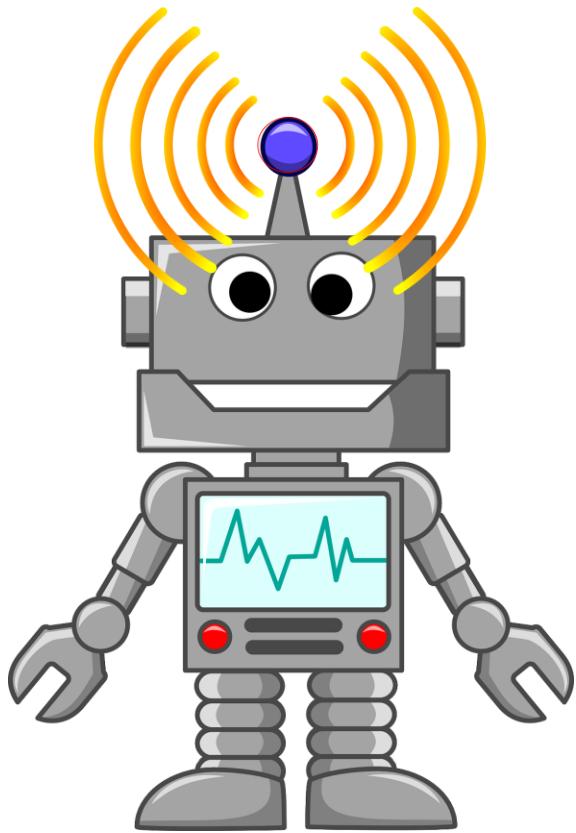


Acessar

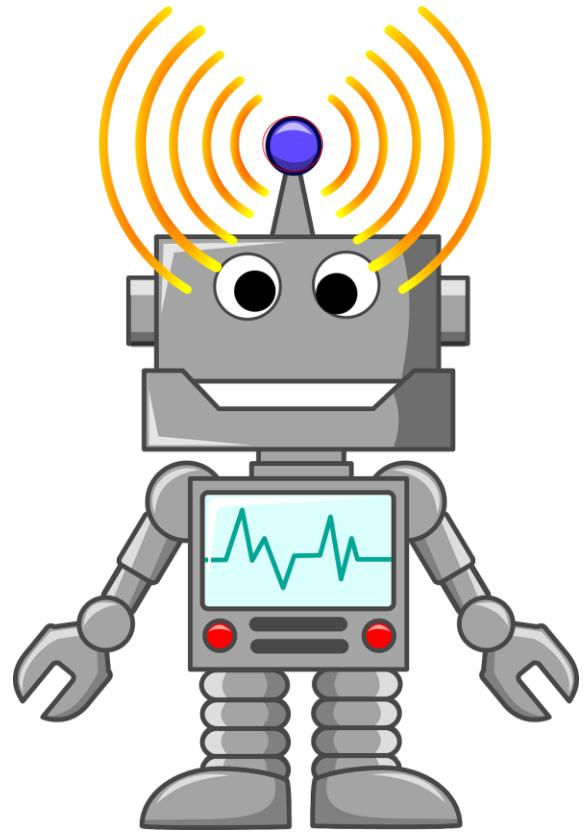


Desenvolvimento de Sistemas MultiAgentes Embarcados

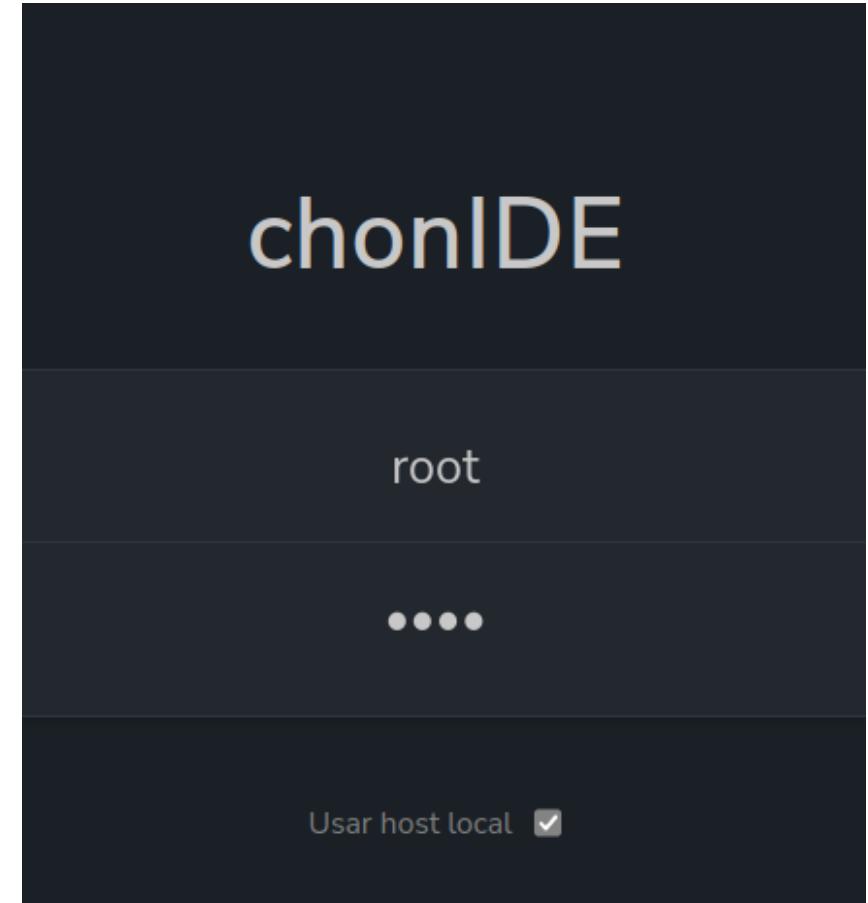
Acessar



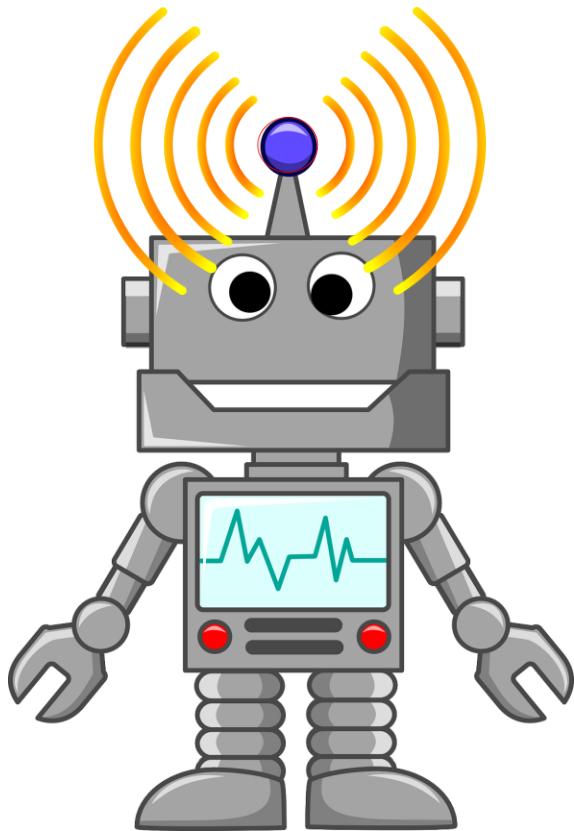
Acessar



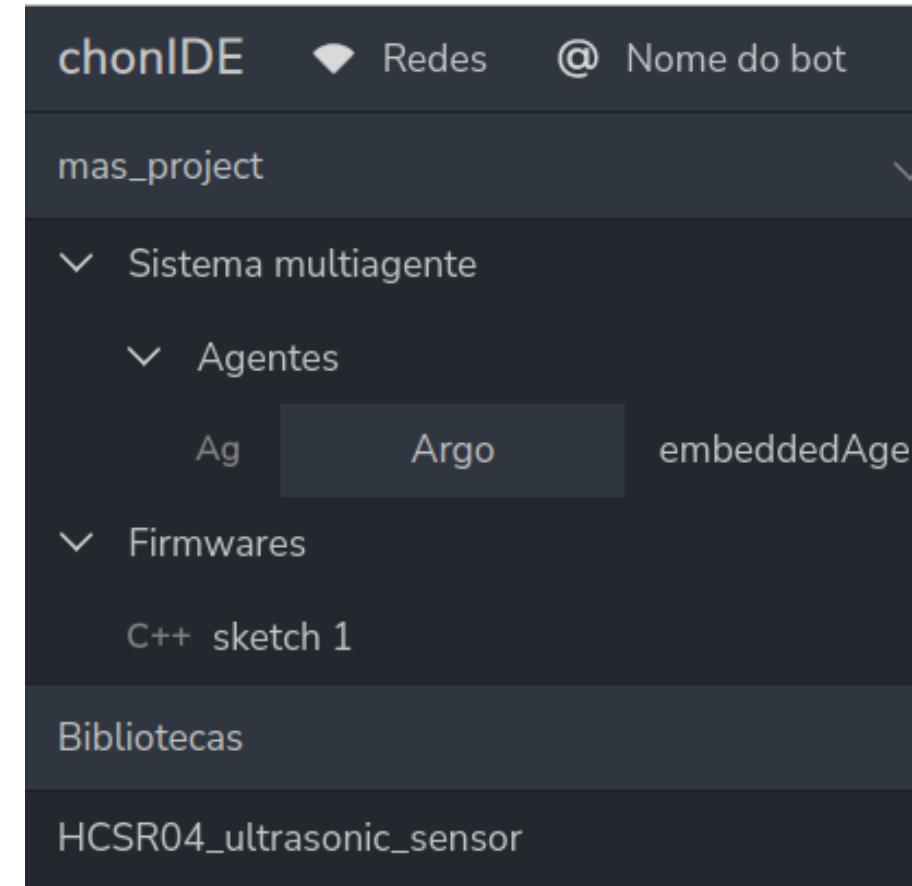
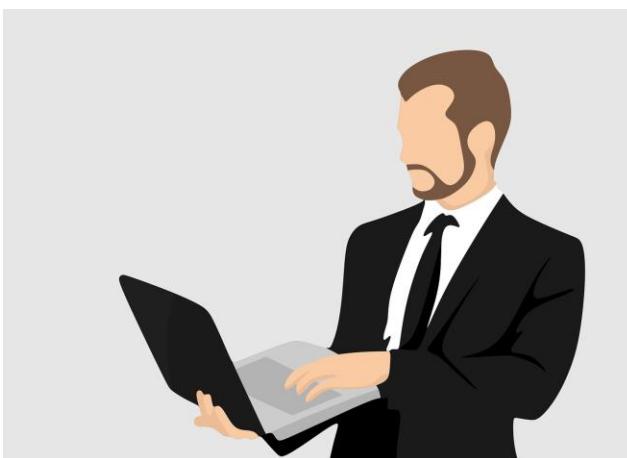
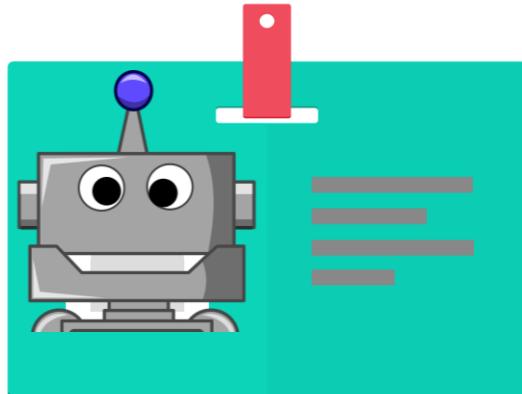
<https://local.bot.chon.group:3270/>



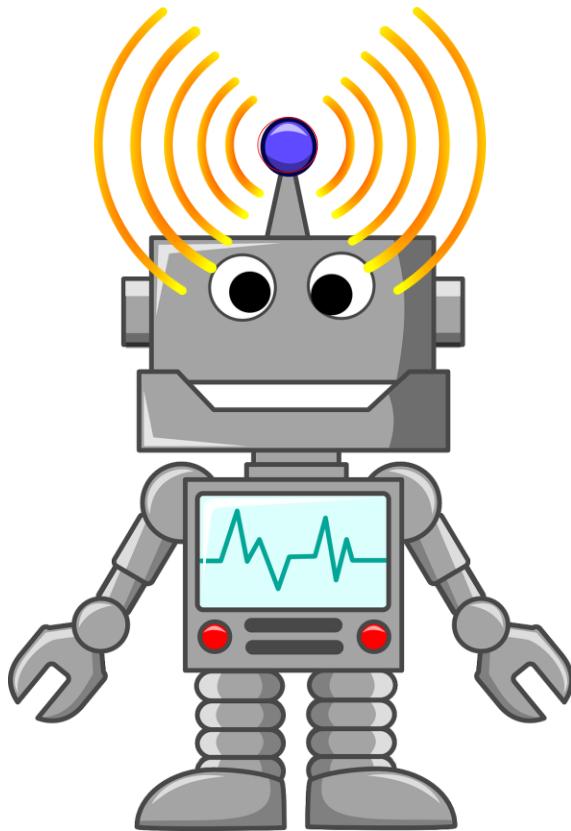
Nomear



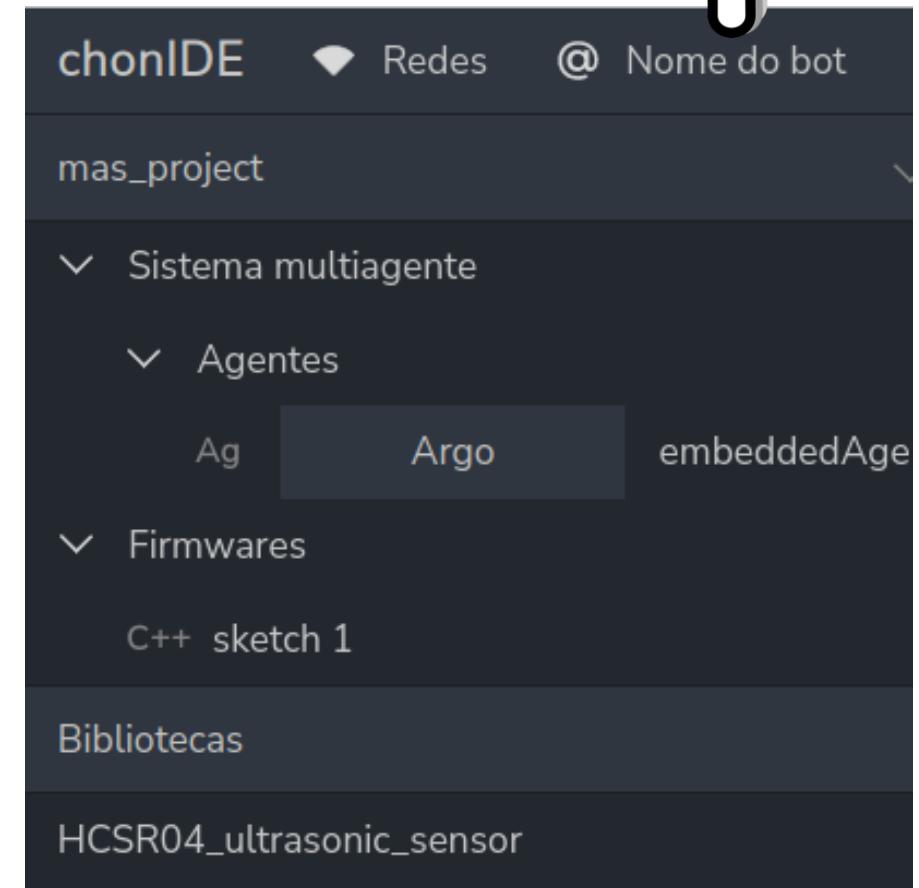
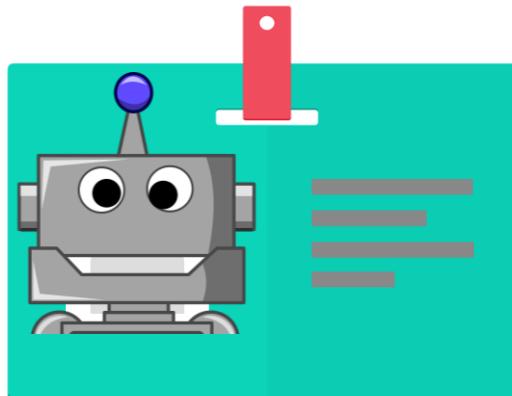
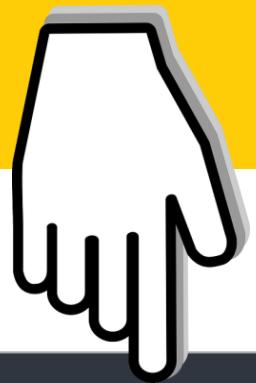
<https://local.bot.chon.group:3270/>



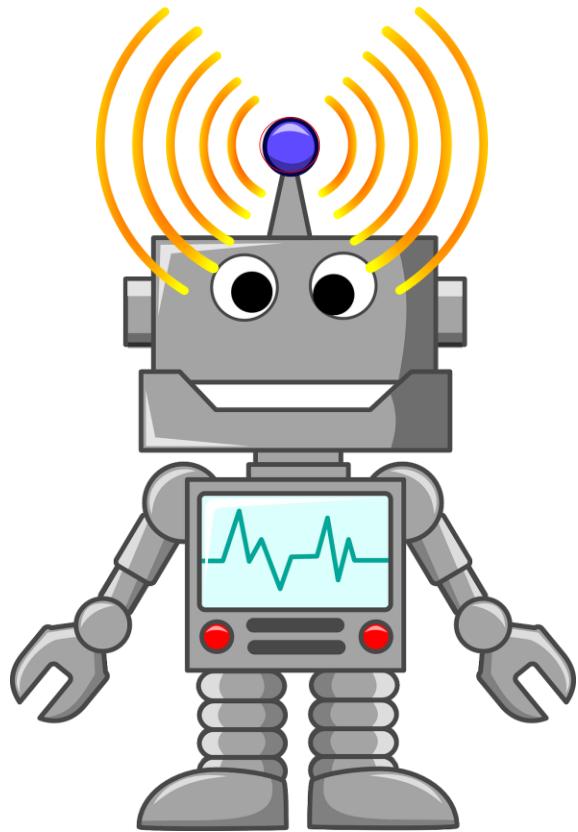
Nomear



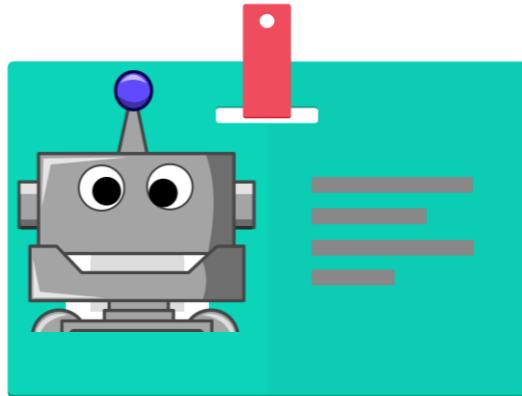
<https://local.bot.chon.group:3270/>



Nomear



<https://local.bot.chon.group:3270/>



Informe o nome do seu bot

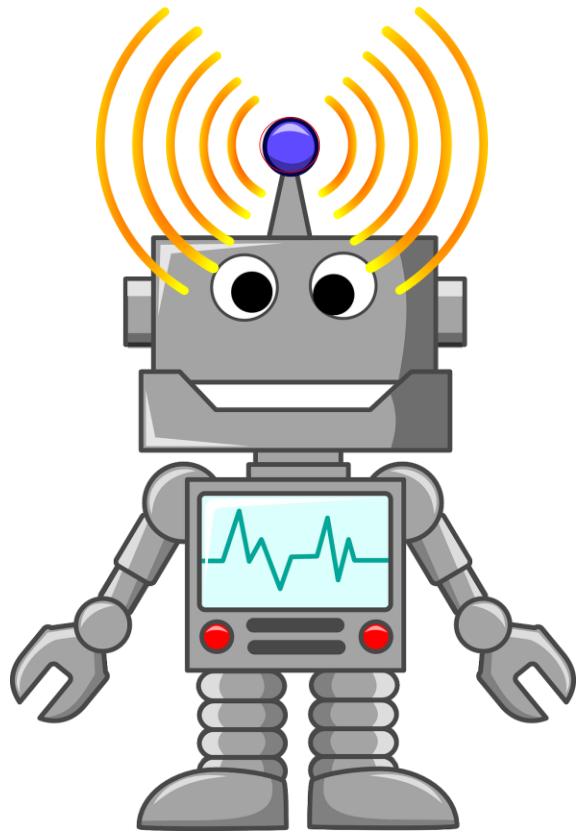
me, você será capaz de acessar a chonide via: t-800.bot.chon.gro

Salvar nome

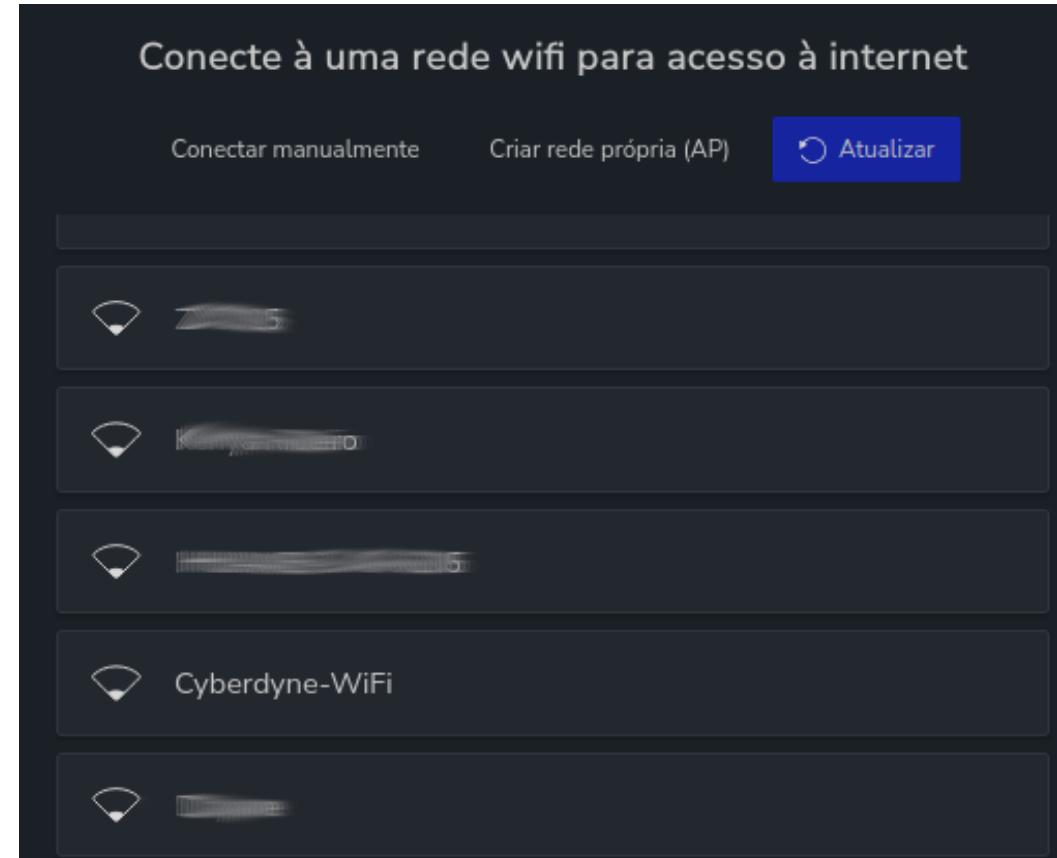
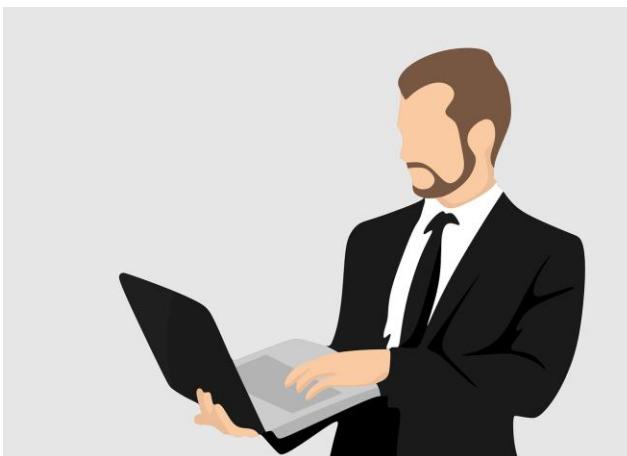
Voltar para codador

A screenshot of a web interface for naming a bot. The title is "Informe o nome do seu bot". Below it, a message says "me, você será capaz de acessar a chonide via: t-800.bot.chon.gro". There is a text input field containing "t-800". Below the input field are two buttons: "Salvar nome" (Save name) and "Voltar para codador" (Return to codifier).

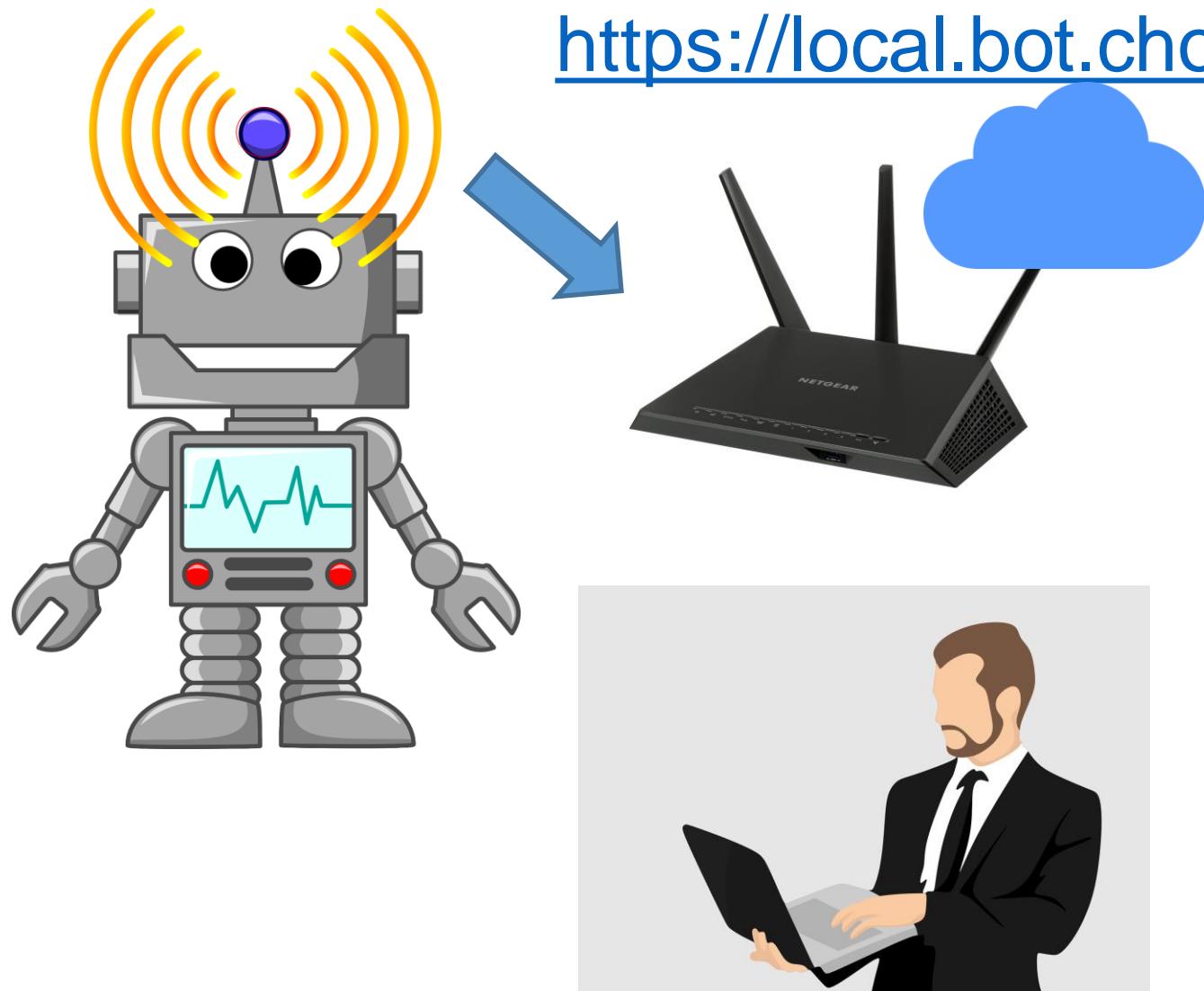
Coneectar



<https://local.bot.chon.group:3270/>



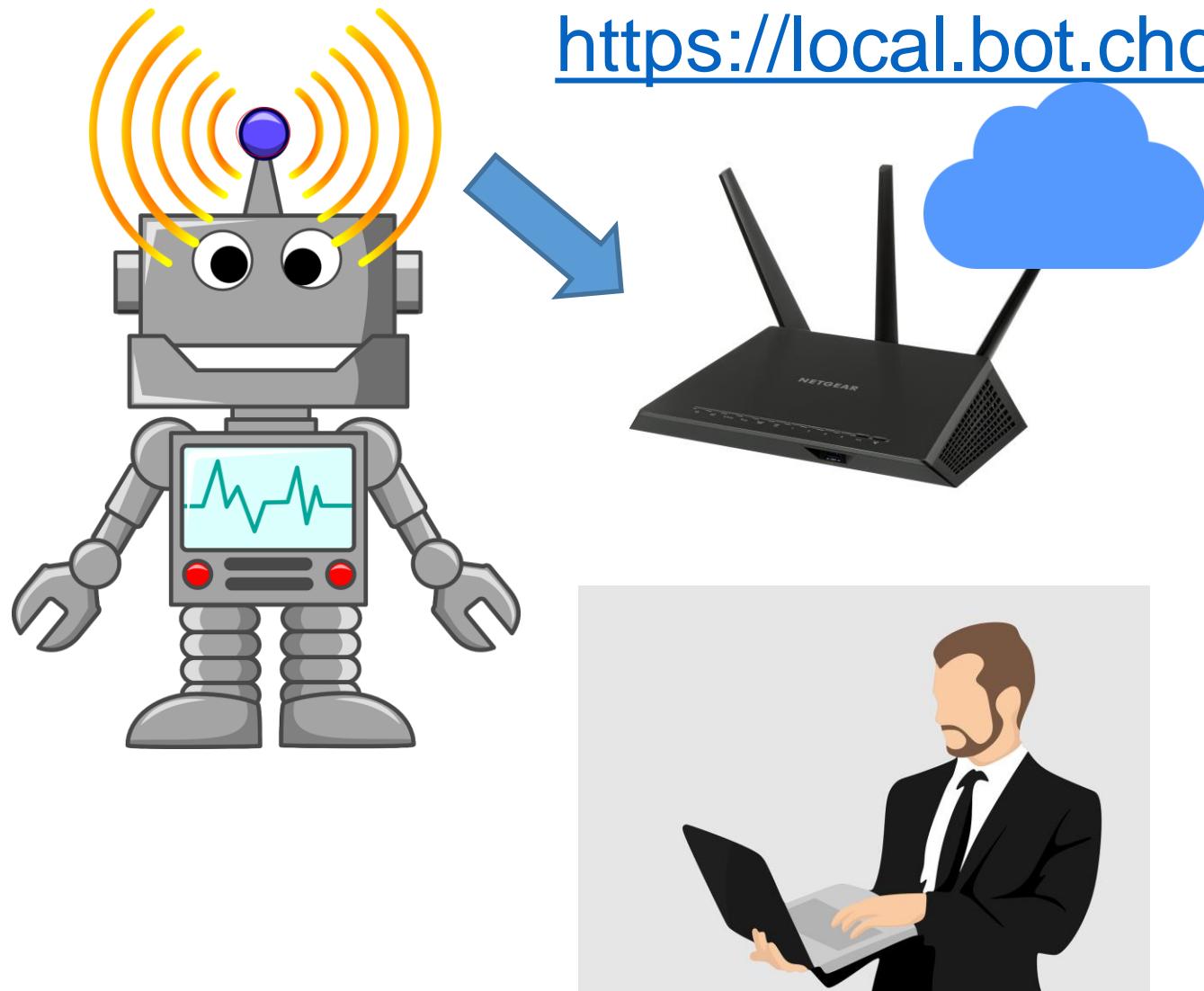
Coneectar



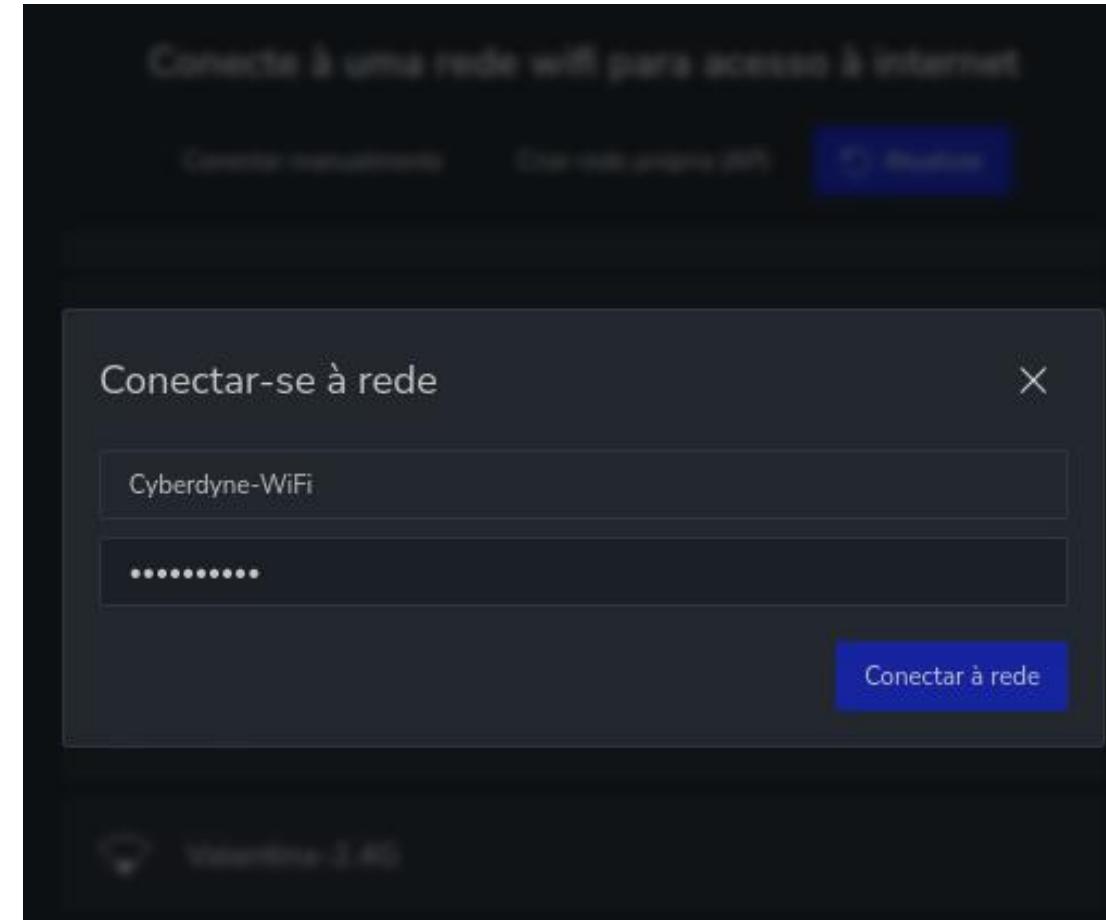
<https://local.bot.chon.group:3270/>



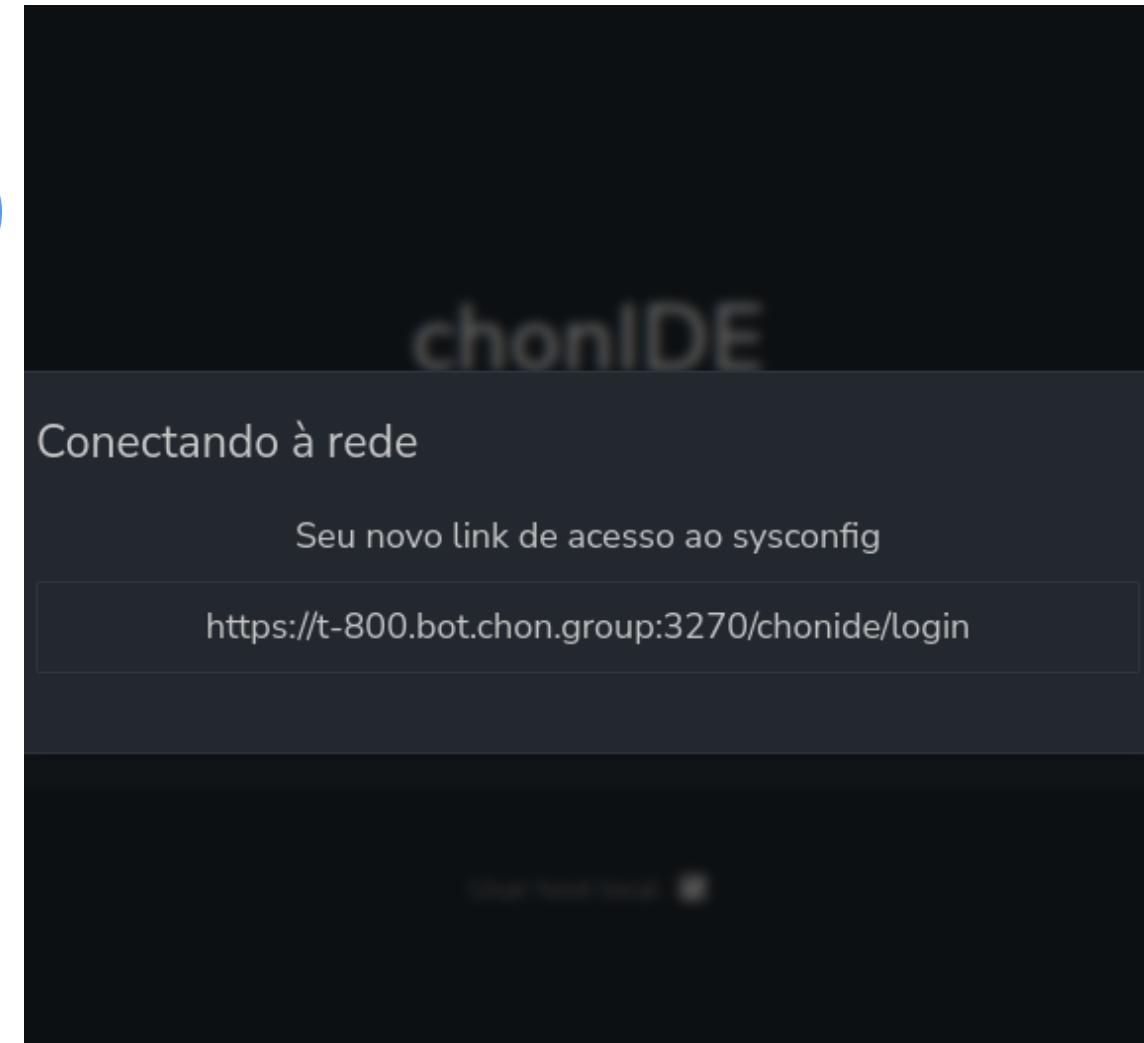
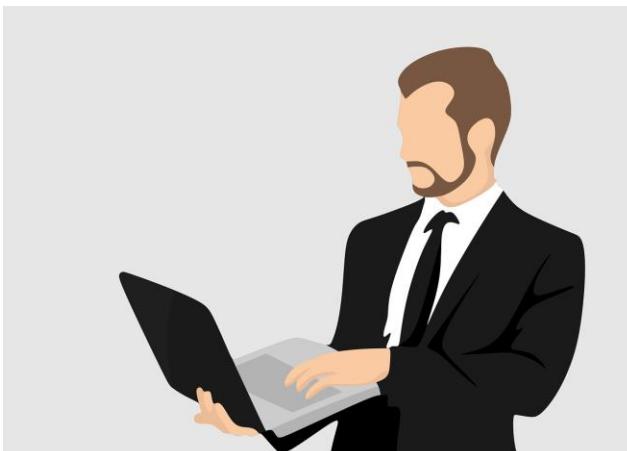
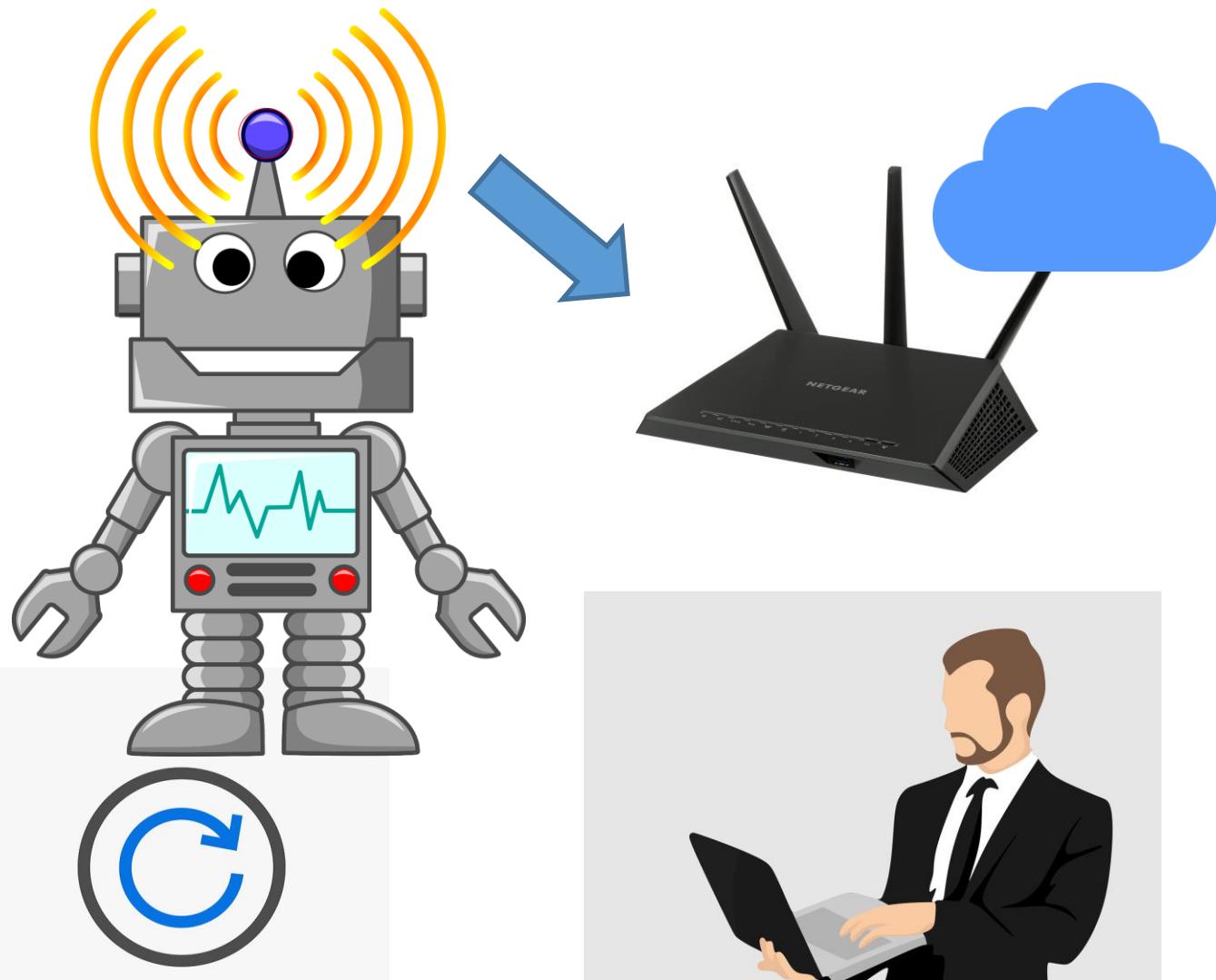
Coneectar



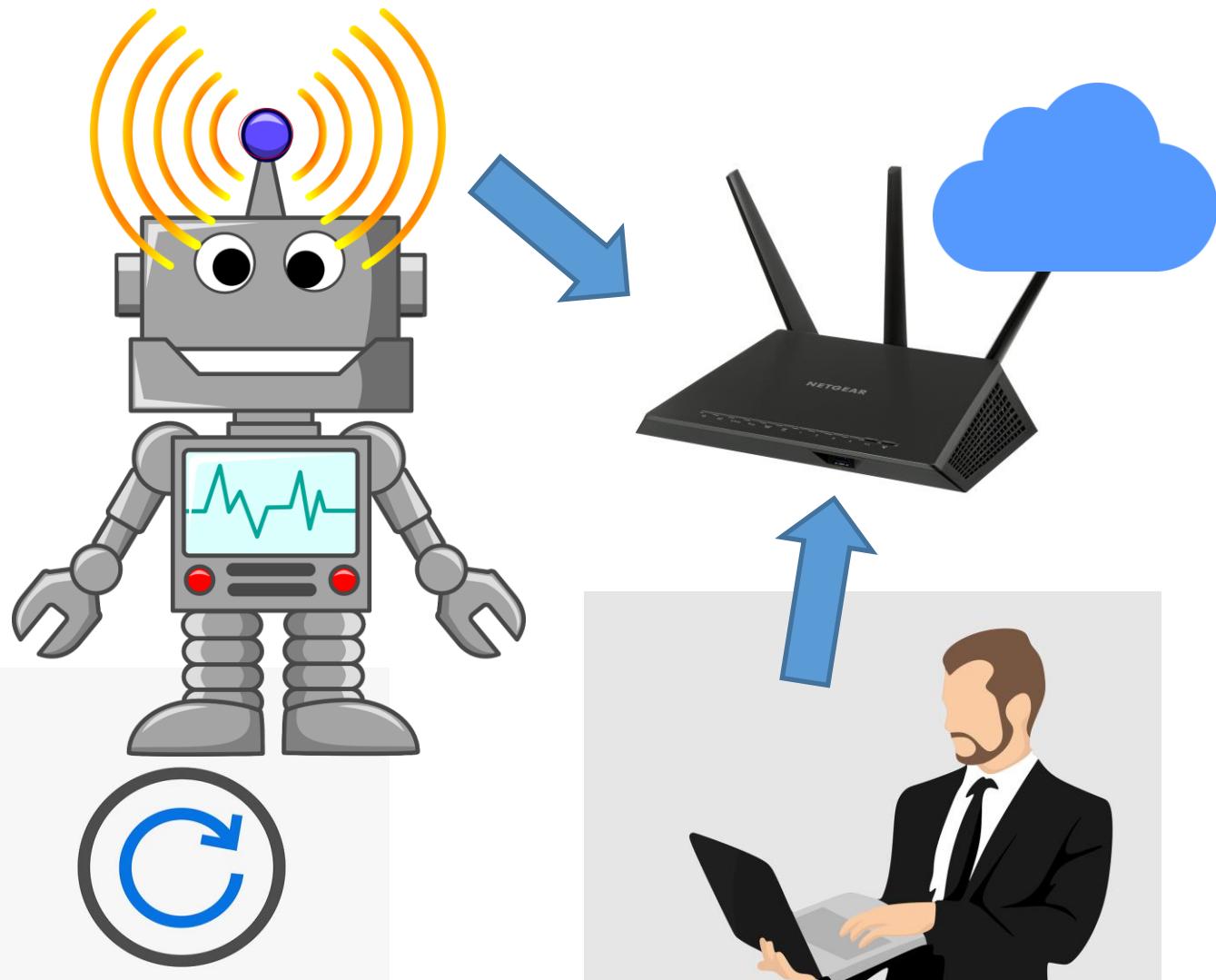
<https://local.bot.chon.group:3270/>



Coneectar

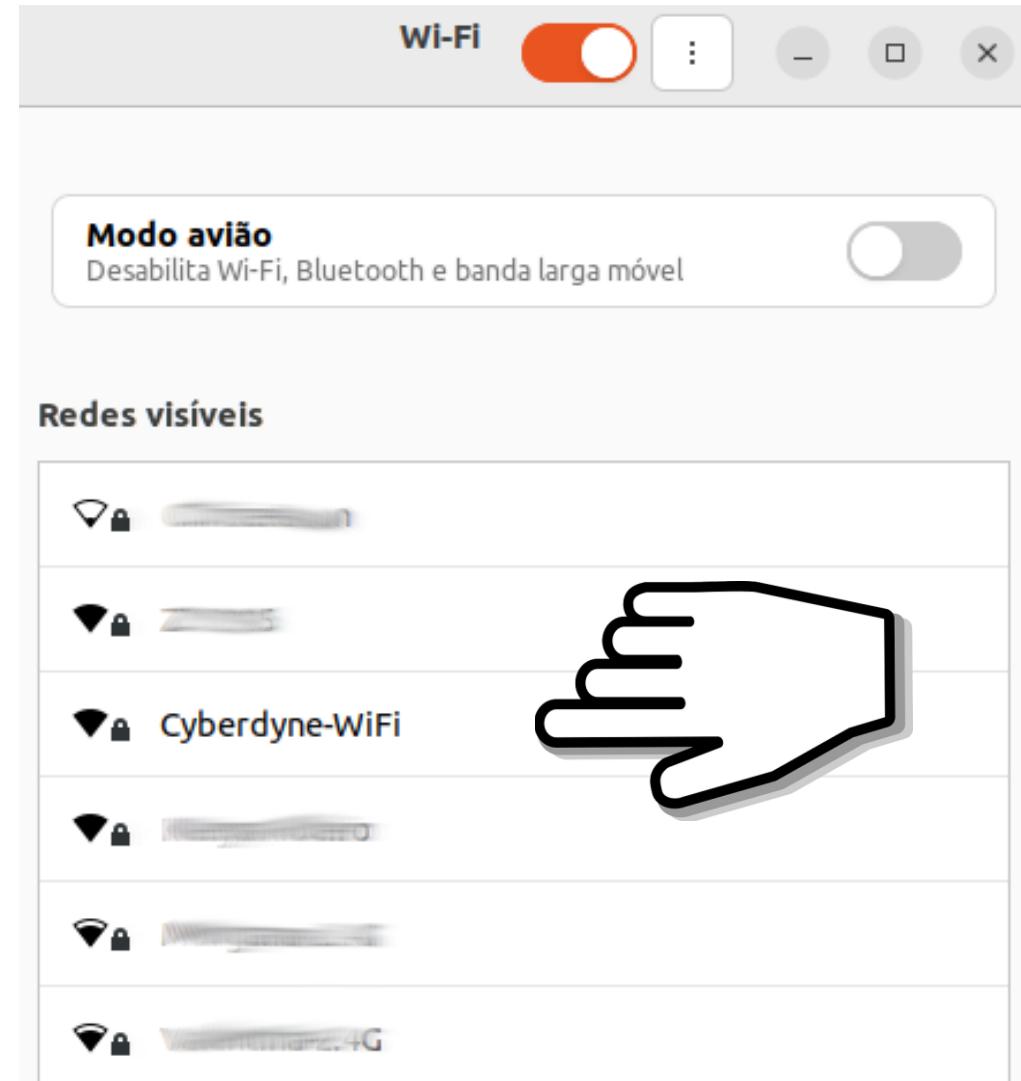
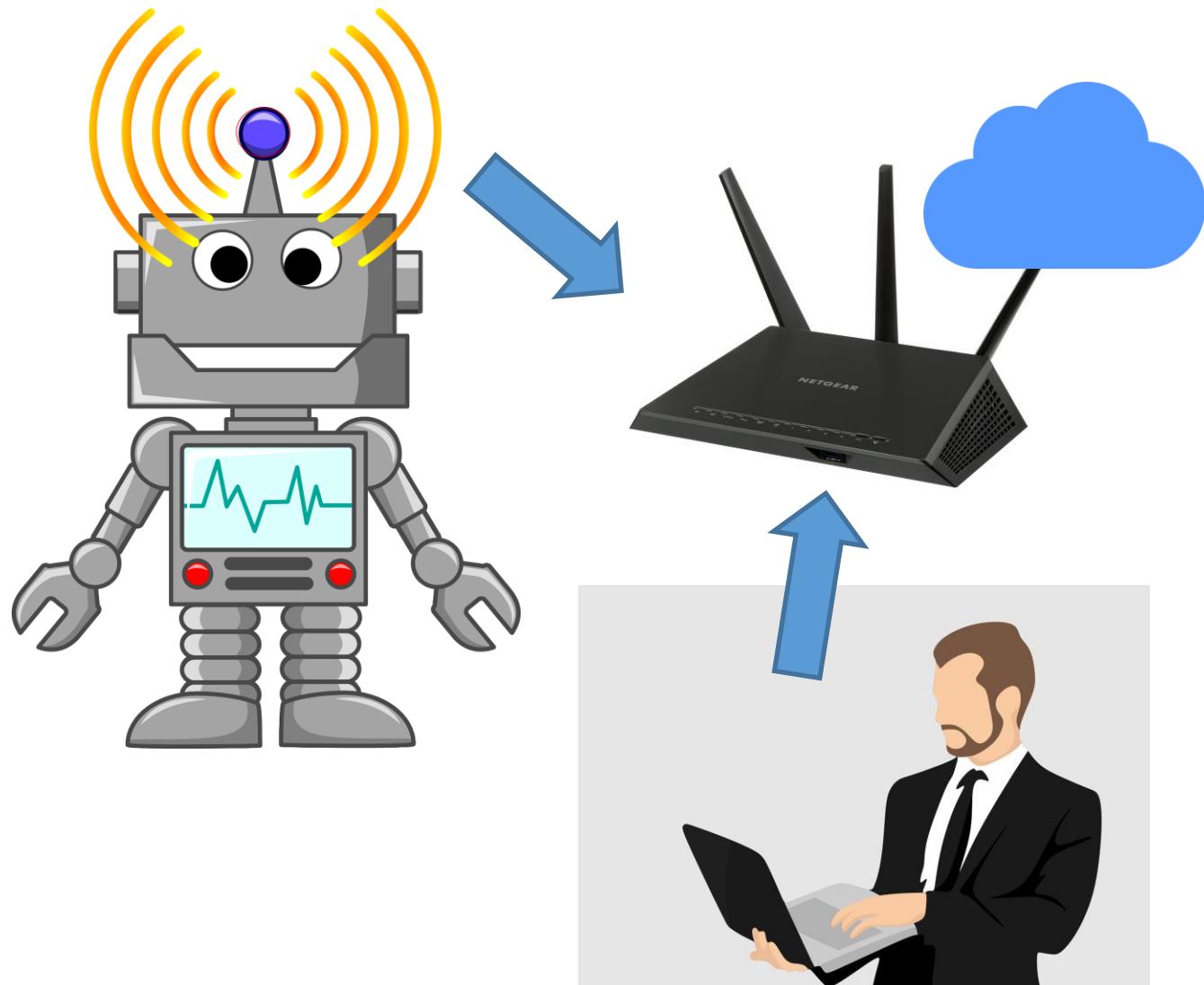


Coneectar

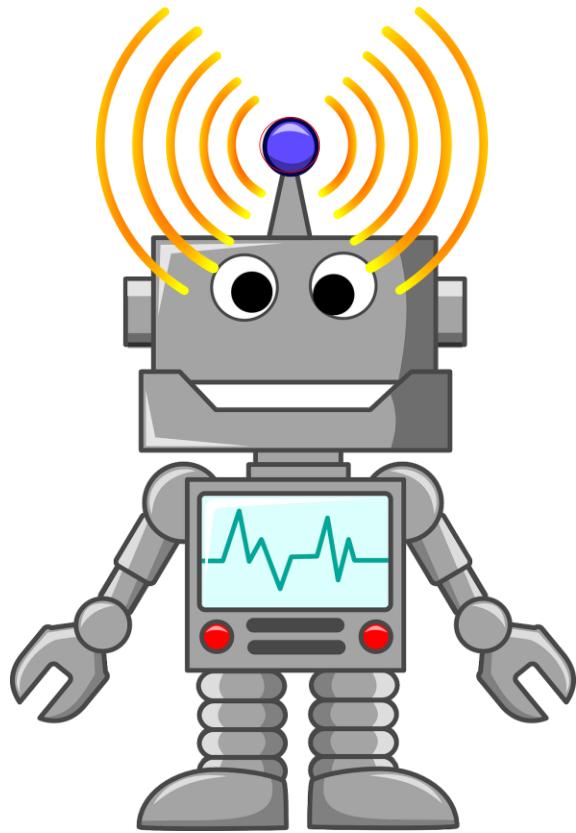


Desenvolvimento de Sistemas MultiAgentes Embarcados

Coneectar



Acessar



<https://t-800.bot.chon.group:3270/>



NOME ESCOLHIDO para o seu bot

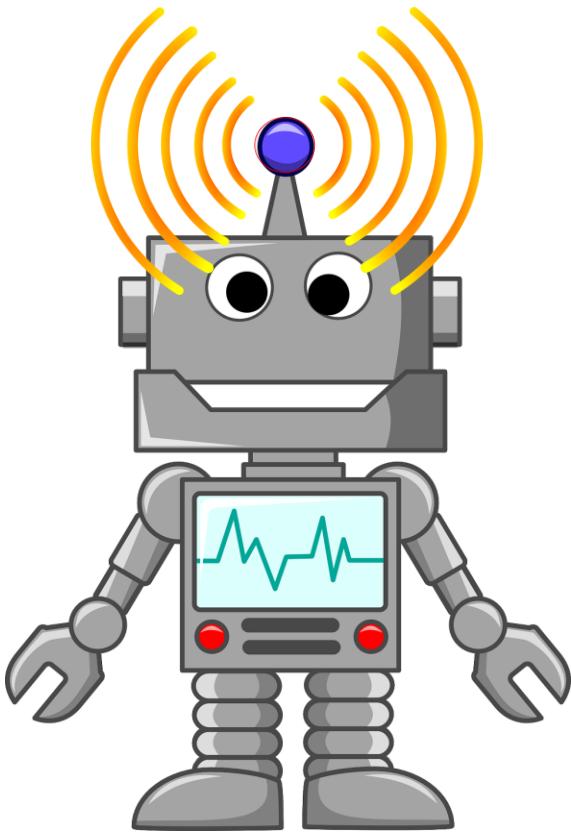
Atenção: O nome do bot é gerenciado por um serviço de DDNS.

Atualmente não há qualquer reserva de nome para o bot.

Futuramente poderemos implementar um cadastro para reserva de nome de bot.

Mais informações em <http://ddns.chon.group/>

Acessar



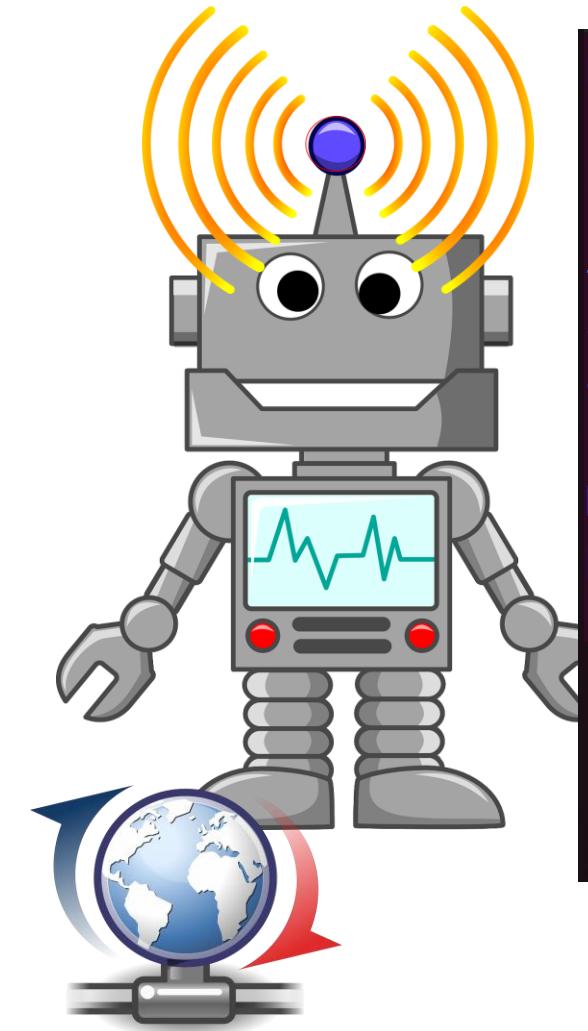
<https://t-800.bot.chon.group:3270/>

← → C https://t-800.bot.chon.group:3270/chonide/coder

chonIDE ⚡ Redes @ Nome do bot

mas_project	✓ Projeto salvo	embeddedAgent1
✓ Sistema multiagente		
✓ Agentes	+	1 /* Initial beliefs and rules */
Ag Argo embeddedAgent1		2
✓ Firmwares	+	3 /* Initial goals */
C++ sketch 1		4
		5 !start.
		6
		7 /* Plans */
		8
		9 +!start <- .print("Hello world!");

Atualizar



```
nilson@nilsonpc:~$ ssh root@t-800.bot.chon.group  
root@t-800.bot.chon.group's password:  
Linux t-800 5.15.61+ #1579 Fri Aug 26 11:08:59 BST 2022 armv6l
```

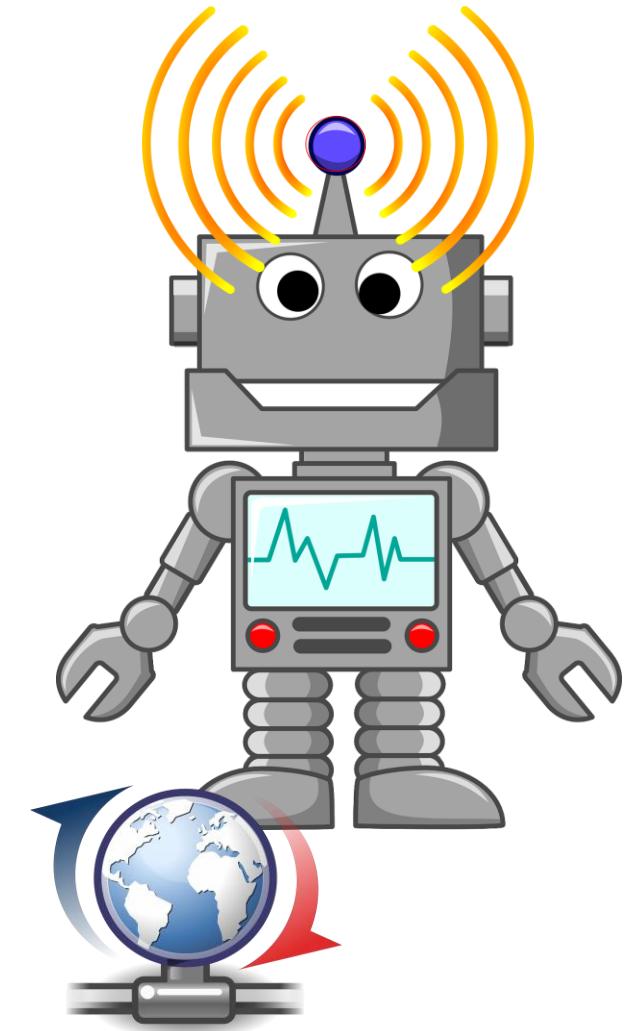
The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

Last login: Sun Oct 16 00:35:33 2022 from 192.168.1.7

root@t-800:~# chonosUpdate

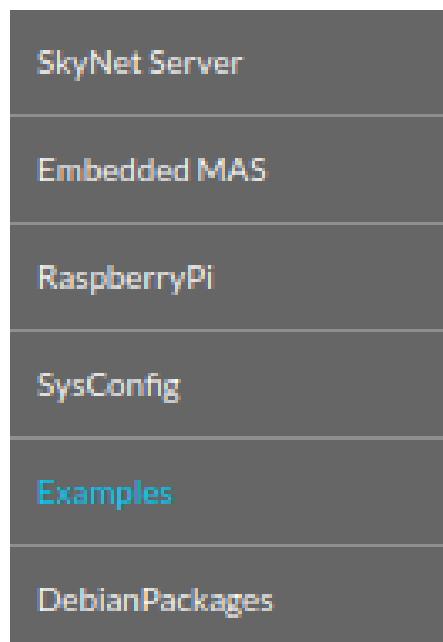
Atualizar



```
root@t-800:~# chonosUpdate
Ign:1 http://packages.chon.group chonos InRelease
Hit:2 http://archive.raspberrypi.org/debian bullseye InRelease
Ign:3 http://packages.chon.group chonos Release
Hit:4 http://raspbian.raspberrypi.org/raspbian bullseye InRelease
Ign:5 http://packages.chon.group chonos/main armhf Packages
Ign:6 http://packages.chon.group chonos/main all Packages
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
17 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
chonos is already the newest version (0.3.14).
chonos-ddnsmng is already the newest version (0.3.14).
chonos-embeddedmas is already the newest version (0.3.10).
chonos-firmwaremng is already the newest version (0.3.11).
chonos-log is already the newest version (0.3.11).
chonos-netmng is already the newest version (0.3.14).
chonos-nettest is already the newest version (0.3.11).
chonos-sysconfig is already the newest version (0.3.13).
chonos-task is already the newest version (0.3.10).
javino is already the newest version (1.3.10).
javino set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 17 not upgraded.
root@t-800:~#
```



(Cognitive Hardware on Network - Operational System)



Hello World!

<https://nome-do-seu-bot.bot.chon.group:3270/>

<https://sourceforge.net/p/chonos/examples/ci/master/tree/00-helloWorld/helloWorldExample01/>

ChonIDE

The screenshot shows the ChonIDE interface with a dark theme. At the top, there is a navigation bar with the following items: chonIDE, Redes, Nome do bot, Mind Inspector, Logs do SMA, Iniciar SMA (with a green play icon), and Parar SMA (with a red square icon). Below the navigation bar is a sidebar containing the project structure:

- mas_project (highlighted with a red box)
- Sistema multiagente
 - Agentes
 - Ag (selected, highlighted with a grey box)
 - Jason
 - embeddedAgent1
- Firmwares
 - C++ sketch 1
- Bibliotecas
- HCSR04_ultrasonic_sensor
- Javino

To the right of the sidebar, there is a message "✓ Projeto salvo" (Project saved) with a checkmark icon. Below the sidebar is a code editor window titled "embeddedAgent1". The code is as follows:

```
1 /* Crenças */
2
3 /* Objetivos */
4
5 !start.
6
7 /* Planos */
8
9 +!start <-
10 .print("Hasta la vista, baby").
```

chonIDE WiFi Redes @ Nome do bot Mind Inspector Logs do SMA ▶ Iniciar SMA □ Parar SMA

mas_project ✓ Projeto salvo embeddedAgent1

✗ Sistema multiagente

✗ Agentes

Ag Jason embeddedAgent1

+ +

✗ Firmwares

C++ sketch 1

Bibliotecas + ⌂

HCSR04_ultrasonic_sensor

Javino

```
1 /* Crenças */
2
3 /* Objetivos */
4
5 !start.
6
7 /* Planos */
8
9 +!start <-
10 .print("Hasta la vista, baby").
```

ChonIDE

The screenshot shows the ChonIDE interface with the following elements:

- Top Bar:** chonIDE, Redes, Nome do bot, Mind Inspector, Logs do SMA, Iniciar SMA, Parar SMA.
- Project Structure:** mas_project (selected), Sistema multiagente, Firmwares, Bibliotecas, HCSR04_ultrasonic_sensor, Javino.
- Current Agent:** Jason (highlighted in grey).
- Agent Editor:** Shows embeddedAgent1 with the following code:

```
/* Crenças */
/* Objetivos */
!start.
/* Planos */
+!start <-
    .print("Hasta la vista, baby").
```
- Code Editor:** A red box highlights the "C++ sketch 1" section in the Firmwares list.

The screenshot shows the ChonIDE interface for developing multi-agent systems. The top bar includes tabs for 'Mind Inspector', 'Logs do SMA', 'Iniciar SMA' (Start SMA), and 'Parar SMA' (Stop SMA). The main workspace displays a project structure under 'mas_project':

- Sistema multiagente**:
 - Agentes**: A list containing 'Ag' and 'Jason'. 'Jason' is currently selected, highlighted with a grey background.
 - embeddedAgent1**: A new agent entry with a '+' icon to its right.
- Firmwares**: A list containing 'C++ sketch 1'.
- Bibliotecas**: A list containing 'HCSR04_ultrasonic_sensor'.
- Javino**: A list item.

A red box highlights the code editor window for 'embeddedAgent1'. The code is as follows:

```
1  /* Crenças */  
2  
3  /* Objetivos */  
4  
5  !start.  
6  
7  /* Planos */  
8  
9  +!start <-  
10 .print("Hasta la vista, baby").
```

chonIDE Redes @ Nome do bot Mind Inspector Logs do SMA Iniciar SMA Parar SMA

mas_project ✓ Projeto salvo embeddedAgent1

Sistema multiagente

Agents

Ag Jason embeddedAgent1

Firmwares

C++ sketch 1

Bibliotecas

HCSR04_ultrasonic_sensor

Javino

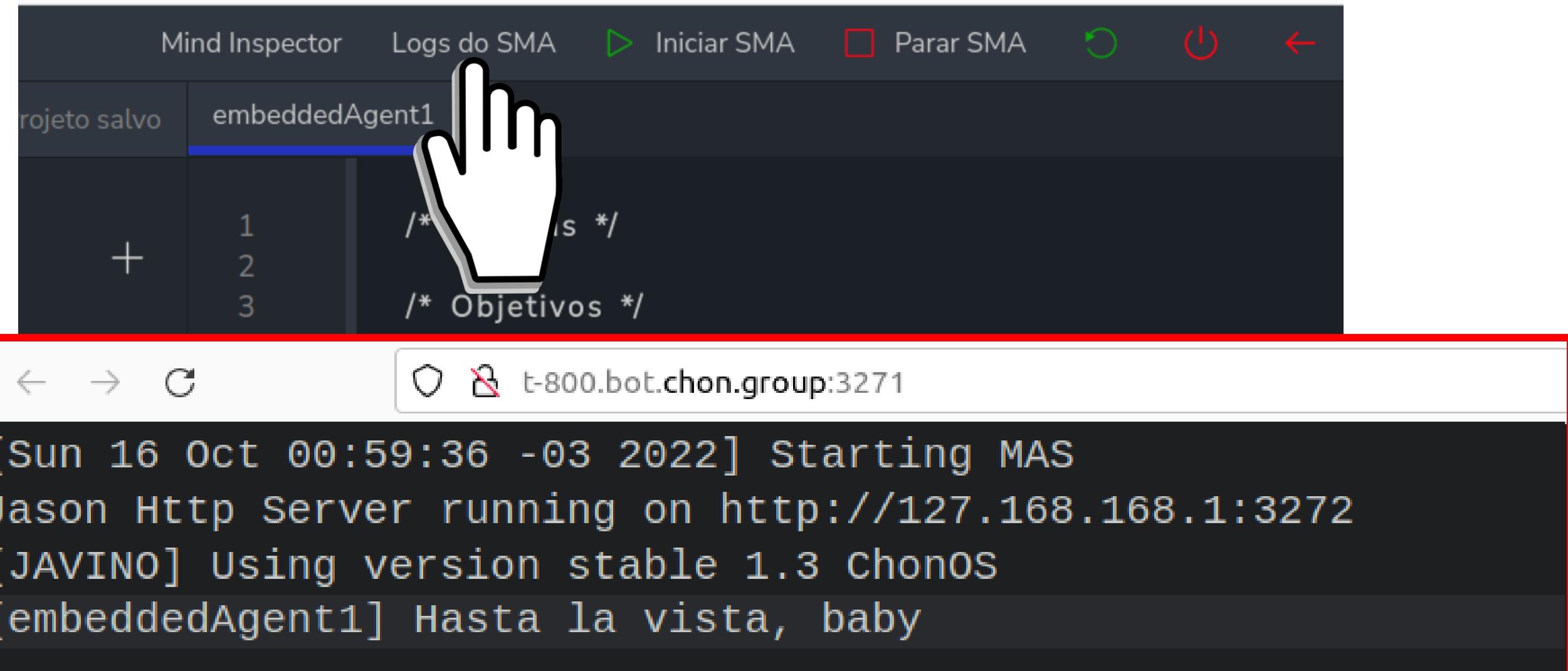
1 /* Crenças */
2
3 /* Objetivos */
4
5 !start.
6
7 /* Planos */
8
9 +!start <-
10 .print("Hasta la vista, baby").



Mind Inspector Logs do SMA Iniciar SMA Parar SMA

Executando SMA, processo 2468

```
+ 1      /* Crenças */  
+ 2  
+ 3      /* Objetivos */  
+ 4  
+ 5      !start.  
+ 6  
+ 7      /* Planos */  
+ 8  
+ 9      +!start <-  
+ 10     .print("Hasta la vista, baby").
```



Mind Inspector Logs do SMA Iniciar SMA Parar SMA

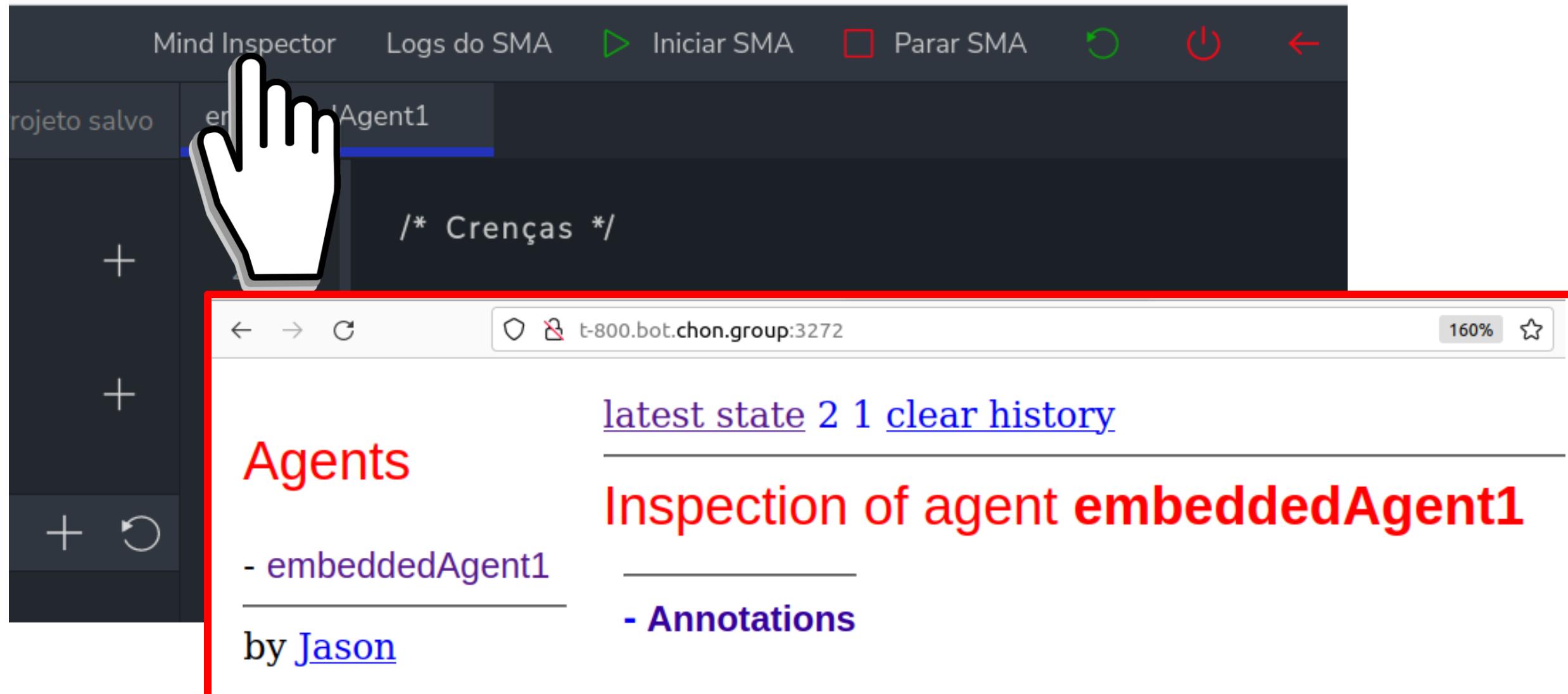
projeto salvo | embeddedAgent1

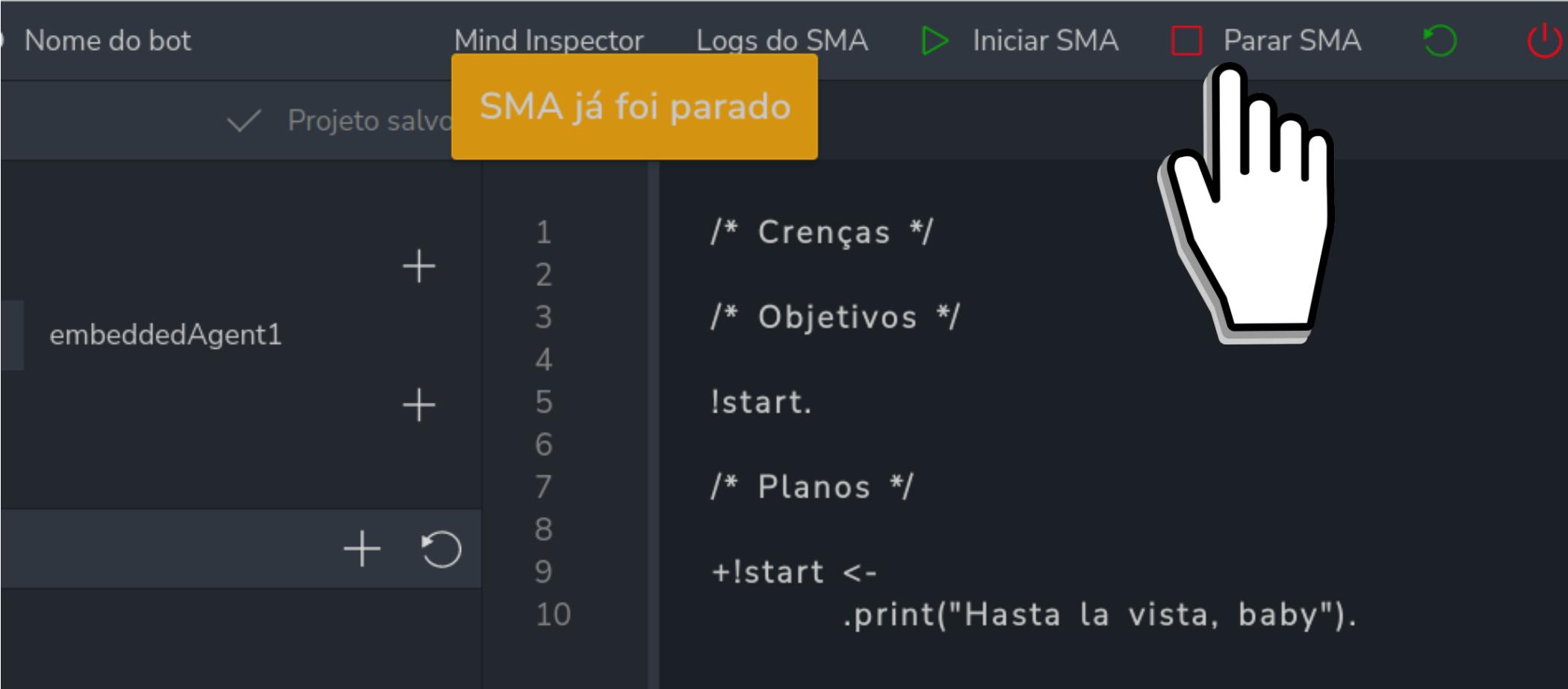
+

	1	/* Objetivos */
	2	
	3	/* Objetivos */

t-800.bot.chon.group:3271

```
[Sun 16 Oct 00:59:36 -03 2022] Starting MAS
Jason Http Server running on http://127.168.168.1:3272
[JAVINO] Using version stable 1.3 chonOS
[embeddedAgent1] Hasta la vista, baby
```



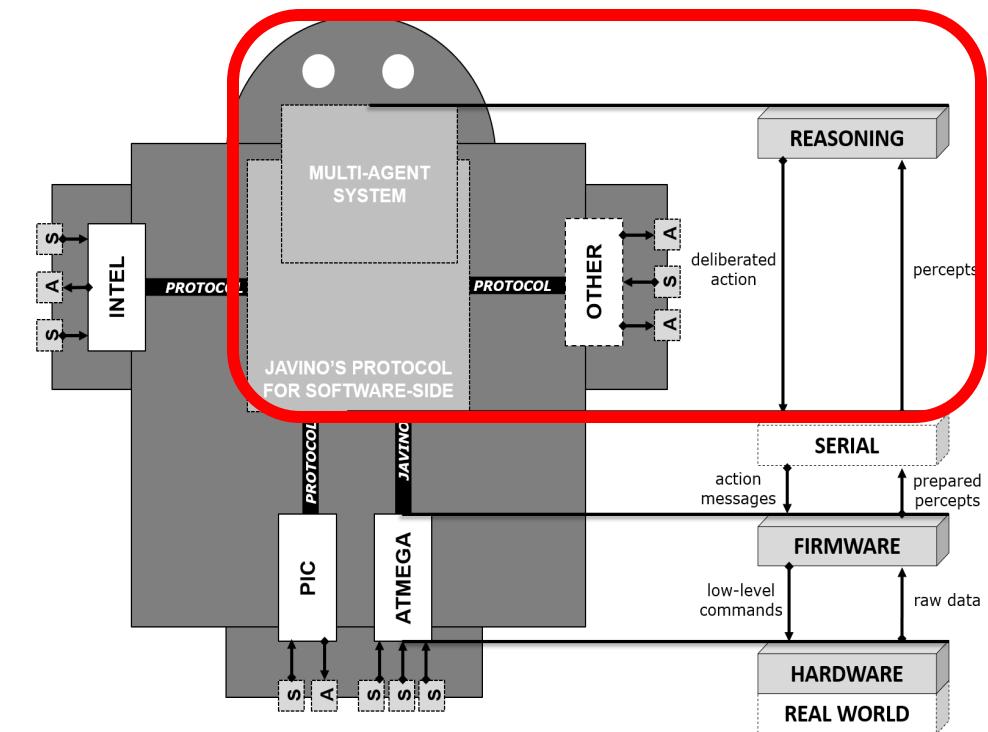


The screenshot shows the ChonIDE interface. At the top, there's a toolbar with buttons for 'Nome do bot' (Name the bot), 'Mind Inspector', 'Logs do SMA', 'Iniciar SMA' (Start SMA) with a green play button, 'Parar SMA' (Stop SMA) with a red square button, a refresh icon, and a power icon. A message box in the center says 'SMA já foi parado' (SMA has been stopped). On the left, there's a sidebar with a checkmark and 'Projeto salvo' (Project saved). Below it, there are two sections: 'embeddedAgent1' and another unnamed section starting with '+'. The main area is a code editor with the following content:

```
1  /* Crenças */  
2  
3  /* Objetivos */  
4  
5  !start.  
6  
7  /* Planos */  
8  
9  +!start <-  
10   .print("Hasta la vista, baby").
```

A large white hand cursor icon is positioned in the upper right quadrant of the code editor area.

CAMADA DE RACIOCÍNIO



- A principal arquitetura para o desenvolvimento de agentes cognitivos é o modelo **BDI** – *belief-desire-intention*.

MICHEL, Fabien; FERBER, Jacques; DROGOUL, Alexis. Multi-Agent Systems and Simulation: A Survey from the Agent Community's Perspective. **Multi-Agent systems: Simulation and applications**. [S. l.]: CRC Press, 2009.

WOOLDRIDGE, Michael. Intelligent Agents. **Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence**. Cambridge, MA, USA: MIT Press, 1999. p. 27–77.

- A principal arquitetura para o desenvolvimento de agentes cognitivos é o modelo **BDI** – *belief-desire-intention*.
- Este modelo está fundamentado no entendimento do raciocínio prático humano que decide, momento a momento, qual ação realizar para alcançar nossos objetivos.

MICHEL, Fabien; FERBER, Jacques; DROGOUL, Alexis. Multi-Agent Systems and Simulation: A Survey from the Agent Community's Perspective. **Multi-Agent systems: Simulation and applications**. [S. l.]: CRC Press, 2009.

WOOLDRIDGE, Michael. Intelligent Agents. **Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence**. Cambridge, MA, USA: MIT Press, 1999. p. 27–77.

O BDI possibilita a implementação de atitudes mentais (crenças, desejos e intenções) em agentes cognitivos.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. **XII Escola de Informática da SBC**, v. 2, p. 51–89, 2004.

O BDI possibilita a implementação de atitudes mentais (crenças, desejos e intenções) em agentes cognitivos.

- Crença é uma informação, ou seja, aquilo que o agente sabe sobre o ambiente, si mesmo ou outros agentes.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. **XII Escola de Informática da SBC**, v. 2, p. 51–89, 2004.

O BDI possibilita a implementação de atitudes mentais (crenças, desejos e intenções) em agentes cognitivos.

- Crença é uma informação, ou seja, aquilo que o agente sabe sobre o ambiente, si mesmo ou outros agentes.
- Desejo é um propósito que o agente busca tornar realidade.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. **XII Escola de Informática da SBC**, v. 2, p. 51–89, 2004.

O BDI possibilita a implementação de atitudes mentais (crenças, desejos e intenções) em agentes cognitivos.

- Crença é uma informação, ou seja, aquilo que o agente sabe sobre o ambiente, si mesmo ou outros agentes.
- Desejo é um propósito que o agente busca tornar realidade.
- Intenção é uma ação que o agente faz para alcançar um desejo.

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. **XII Escola de Informática da SBC**, v. 2, p. 51–89, 2004.

AgentSpeak(L)

A linguagem de programação **AgentSpeak(L)** é uma extensão natural e elegante de programação em lógica para a arquitetura de agentes BDI, que representa um modelo abstrato para a **programação de agentes cognitivos.**

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. **XII Escola de Informática da SBC**, v. 2, p. 51–89, 2004.

AgentSpeak(L)

Um agente AgentSpeak(L) corresponde à especificação de:

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. **XII Escola de Informática da SBC**, v. 2, p. 51–89, 2004.

AgentSpeak(L)

Um agente AgentSpeak(L) corresponde à especificação de:

- Uma base de crenças;

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. **XII Escola de Informática da SBC**, v. 2, p. 51–89, 2004.

AgentSpeak(L)

Um agente AgentSpeak(L) corresponde à especificação de:

- Uma base de crenças;
- Uma lista de objetivos;

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. **XII Escola de Informática da SBC**, v. 2, p. 51–89, 2004.

AgentSpeak(L)

Um agente AgentSpeak(L) corresponde à especificação de:

- Uma base de crenças;
- Uma lista de objetivos;
- Uma biblioteca de planos;

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. **XII Escola de Informática da SBC**, v. 2, p. 51–89, 2004.



Jason | a Java-based interpreter for an extended version of AgentSpeak

<http://jason.sf.net>

HÜBNER, Jomi Fred; BORDINI, Rafael Heitor; VIEIRA, Renata. Introdução ao desenvolvimento de sistemas multiagentes com Jason. **XII Escola de Informática da SBC**, v. 2, p. 51–89, 2004.

Jason Framework: Crenças

- **Beliefs**

Em Jason, um agente armazena as **informações** percebidas do ambiente; as informações internas; e informações de comunicação através de crenças.

As **crenças** são armazenadas em uma **Base de Crenças** (Belief Base).

As crenças são representadas como predicados da **lógica tradicional**. Os **predicados** representam propriedades particulares.

Jason Framework: Crenças

- **Tipos**

1. **Percepções do Ambiente (Percepts)**

Informações coletadas pelo agente que são relativas ao sensoriamento constante do ambiente.

2. **Notas Mentais (Mental Notes)**

- Informações adicionadas na base de crenças pelo próprio agente.
 - coisas que aconteceram no passado;
 - promessas;
 - execução de um plano;
 - constante do ambiente.

3. **Comunicação**

Informações obtidas pelo agente através da interação com outros agentes.

Jason Framework: Crenças

- **Exemplos: Crenças Iniciais**

salario(5000).

paulo(alto).

missionStarted.

carro(ferrari, kadu).

OBS. 1: Toda **crença inicial** em Jason deve terminar com **.**

OBS. 2: Toda **crença** deve começar com letra **minúscula**.

Jason Framework: Crenças

- **Exemplos: Strong Negation**

~missionCompleted.

~alto(carlos).

~dia.

OBS.: Toda **strong negation** em Jason deve começar com **~**

Jason Framework: Objetivos

- **Goals**

Em Jason, os **goals** (objetivos) representam os estados do mundo em que o agente deseja atingir.

- **Tipos**

1. **Objetivo de realização (!)**

- É um objetivo para atingir determinado estado desejado pelo agente.

2. **Objetivo de Teste (?)**

- É um objetivo que tem basicamente a finalidade de resgatar informações da base de crenças do agente.

Jason Framework: Objetivos

- **Exemplos: Objetivos Iniciais**

!start.
!bark.

OBS. 1: Todo **objetivo inicial** em Jason deve ser um

Objetivo de realização: começar com **!**; e terminar com **.**

OBS. 2: Todo **objetivo** deve começar com letra **minúscula**.

Jason Framework: Planos e Ações

- **Formato de um Plano**

Triggering_event : context <- body.

1. **Triggering Event**

Um agente pode ter diversos objetivos. Os planos são ativados baseados nos eventos que podem ser ativados em determinado momento.

2. **Context**

São as condições para a ativação de um plano dentro vários eventos.

3. **Body**

É o corpo do plano. Uma sequência de ações a ser executada pelo agente.

Jason Framework: Planos e Ações

- **Tipos de Triggering Events**

Adição (+)

Eventos ativadores podem **adicionar** atitudes mentais (crenças ou objetivos)

Exemplo de Adição de Plano de Objetivo

`!bark.`

```
+!bark : true <-
    .print("Au Au Au!") .
```

Jason Framework: Planos e Ações

- **Tipos de Triggering Events**

Remoção (-)

Eventos ativadores podem **remover** atitudes mentais
(crenças ou objetivos)

`!bark.`

`+!bark : dog(unknow) <-
.print("Au Au Au!");`

`-!bark <-
.print("sniff sniff!");
!bark.`

Funciona como um **"tratamento de erros"** para planos que não possuem ativação.

Exemplo de Remoção de Plano de Objetivo

Jason Framework: Planos e Ações

- **Tipos de Planos**

1. Achievement Goal

`!bark.`

Exemplo de Plano de Realização

```
+!bark : true <-
    .print("Au Au Au!") .
```

É **um objetivo** que o agente se compromete em **atingir**.

Jason Framework: Planos e Ações

- **Tipos de Planos**

2. Test Goal

São planos que são ativados quando se **recuperam informações** da base de crenças.

!bark.

Exemplo de Plano de Teste

```
+!bark : dog(unknow) <-
    .print("Au Au Au!") .
```

```
-!bark <-
    .print("sniff sniff!") ;
!sniff.
```

```
+!sniff <-
    .print("Is it bob?") ;
?dog(X) ;
    .print(X) .
```

```
+?dog(X) <-
    X = bob;
+dog(X) .
```

Jason Framework: Planos e Ações

- **Tipos de Planos**

3. Belief

São planos ativados quando o agente **adiciona** ou **remove** uma **crença** da sua base de crenças

Exemplo de Plano de Crença

```
!sniff.
```

```
+!sniff <-
    .print("Is it bob?");
+dog(bob).
```

```
+dog(bob) <-
    .print("sniff sniff!").
```

Jason Framework: Planos e Ações

- **Ações de um Plano**

Achievement e Test Goals

São as chamadas para execução de um plano.

```
!bark.  
  
+!bark <-  
    .print("sniff!");  
    !sniff;  
    .print("sniff!");
```

```
+!sniff <-  
    .print("Is it bob?");  
    ?dog(X);  
    .print(X).
```

```
+?dog(X) <-  
    X = bob;  
    +dog(X);  
    .print("I found X").
```

```
!bark.  
  
+!bark <-  
    .print("sniff!");  
    !!sniff;  
    .print("sniff!");
```

```
+!sniff <-  
    .print("Is it bob?");  
    ?dog(X);  
    .print(X).
```

```
+?dog(X) <-  
    X = bob;  
    +dog(X);  
    .print("I found X").
```

Jason Framework: Planos e Ações

- **Ações de um Plano**

Mental Notes

São ações que **adicionam**,
removem ou **atualizam**
uma **crença** na base de
crenças do agente.

```
hungry.  
food(100).  
stomach(0).
```

```
!eat.
```

```
+!eat: hungry & food(F) & stomach(S) & S<=50 <-  
.print("Eating...");  
-+food(F-1);  
-+stomach(S+1);  
.print(F);  
!eat.
```

```
+!eat: stomach(S) & S>50 <-  
.print("I'm Satisfied.");  
-hungry.
```

Jason Framework: Planos e Ações

• Ações de um Plano

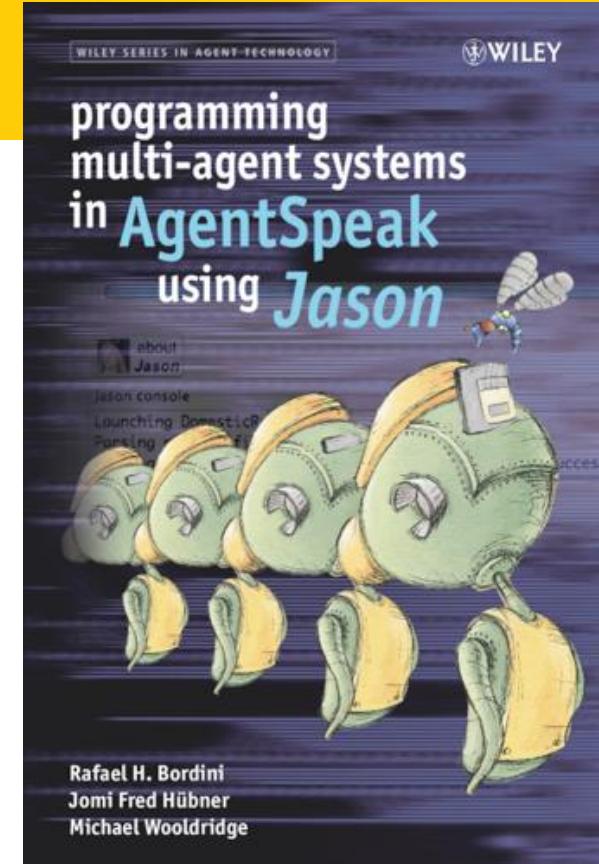
Internal Action

São ações pré-definidas **executadas** internamente no agente.

.print	.max	.create_agent
.send	.nth	.date
.broadcast	.sort	.wait
.drop_all_desires	.substring	.random
.my_name	.drop_all_events	.kill_agent
.concat	.abolish	.time
.length	.string	.perceive
.min	.count	.stopMAS

Consulte páginas: 237-255

BORDINI, R.H.; HÜBNER, J.F.; WOOLDRIDGE, M. **Programming Multi-Agent Systems in AgentSpeak using Jason**. [S. l.]: Wiley, 2007(Wiley Series in Agent Technology). Disponível em: https://www.google.com.br/books/edition/Programming_Multi_Agent_Systems_in_Agent/AJHD4GkIQs0C



Jason Framework: Planos e Ações

- **Ações de um Plano**

External Action

São ações **disponíveis no ambiente.**

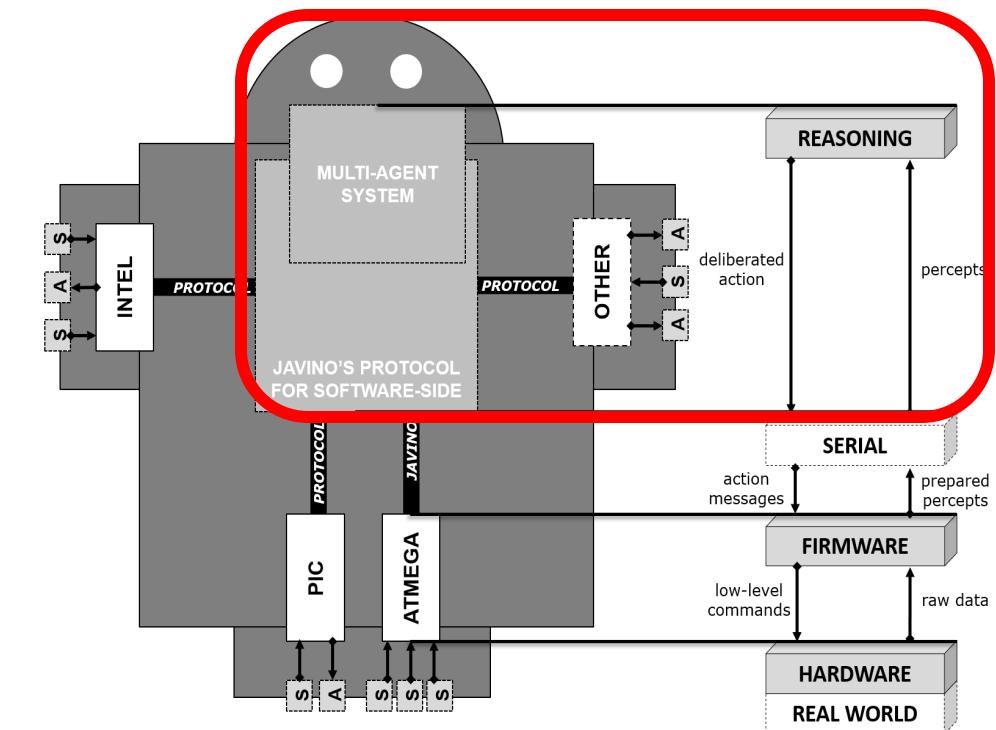
**Fora do escopo
deste curso!**

```
+!move : distance(X) & X >= 50 & X < 1000 <-
    .print("Moving ahead.");
    moveFront;
    .wait(500);
    !refresh.

+!move : distance(X) & X < 50 <-
    .print("Turning left.");
    moveLeft;
    .wait(500);
    !refresh.

+!move : distance(X) & X >= 1000 <-
    .print("No communication! Stopped!");
    stop;
    .wait(500);
    !refresh.
```

Programando um SingleAgent System



SingleAgent System: Exemplo



```
singleAgentEx01

1  /* Crenças Iniciais */
2  sintoFome.
3  estoqueComida(20).
4  energia(0).

5
6  /* Objetivos Iniciais */
7  !comer.

8
9  /* Planos */
10 +!comer: sintoFome & estoqueComida(C) & energia(E) & E<=10 <-
11   .print("Comendo");
12   .wait(200);
13   -+estoqueComida(C-1);
14   -+energia(E+2);
15   !comer.

16
17 +!comer: energia(E) & E>10  <-
18   .print("Barriga cheia");
19   -sintoFome.|
```

<https://sourceforge.net/p/chonos/examples/ci/master/tree/01-singleAgent/singleAgentExample01/>

SingleAgent System: Exemplo 2

```
singleAgent

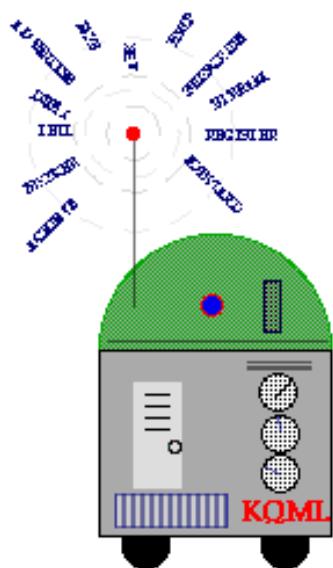
1  /* Crenças Iniciais */
2  estoqueComida(20).
3  energia(0).
4
5  /* Objetivos Iniciais */
6  !curtir.
7
8  /* Planos */
9  +!comer: sintoFome & estoqueComida(C) & C>3 & energia(E) & E<=10 <-
10    -+estoqueComida(C-3);      -+energia(E+3);
11    .print("Com fome..... [Estoque de comida=",C,"] [Energia=",E,"]");
12    .wait(1000);   !comer.
13
14  +!comer: energia(E) & E>10  <-
15    .print("Satisfeito..... [Energia=",E,"]");
16    -sintoFome;   .wait(2000);   !curtir.
17
18  +!comer <-
19    ?estoqueComida(X);
20    .print("Acabou a comida. [Estoque de comida=",X,"]");
21    .wait(2000);   !trabalhar.
22
23  +!curtir: energia(E) & E>5 <-
24    .print("Curtindo....");   .wait(2000);
25    -+energia(E-3);         -sintoFome;   !curtir.
26
27  +!curtir: energia(E) & E<=5<-
28    ?energia(X);   .print("Sem Energia..... [Energia=",X,"]");
29    +sintoFome;   .wait(2000);   !comer.
30
31  +!trabalhar <-
32    .print("Trabalhando");   .wait(1000);   ?estoqueComida(C);
33    -+estoqueComida(C+3);   ?energia(E);   -+energia(E-1);
34    +sintoFome;           !comer;        !curtir.
```

<https://sourceforge.net/p/chonos/examples/ci/master/tree/01-singleAgent/singleAgentExample02/>

Comunicação Entre Agentes

No início de cada ciclo de raciocínio, o agente verifica mensagens que ele possa ter recebido de outros agentes

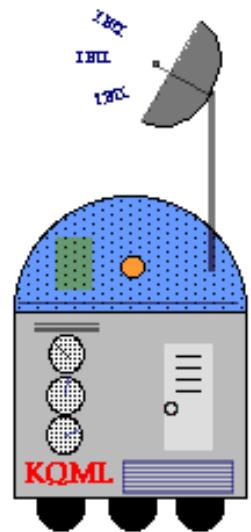
Baseada na teoria dos Atos de Fala e *KQML*



KQML (Knowledge Query and Manipulation Language) é um protocolo de comunicação para sistemas baseados em conhecimento.

KQML

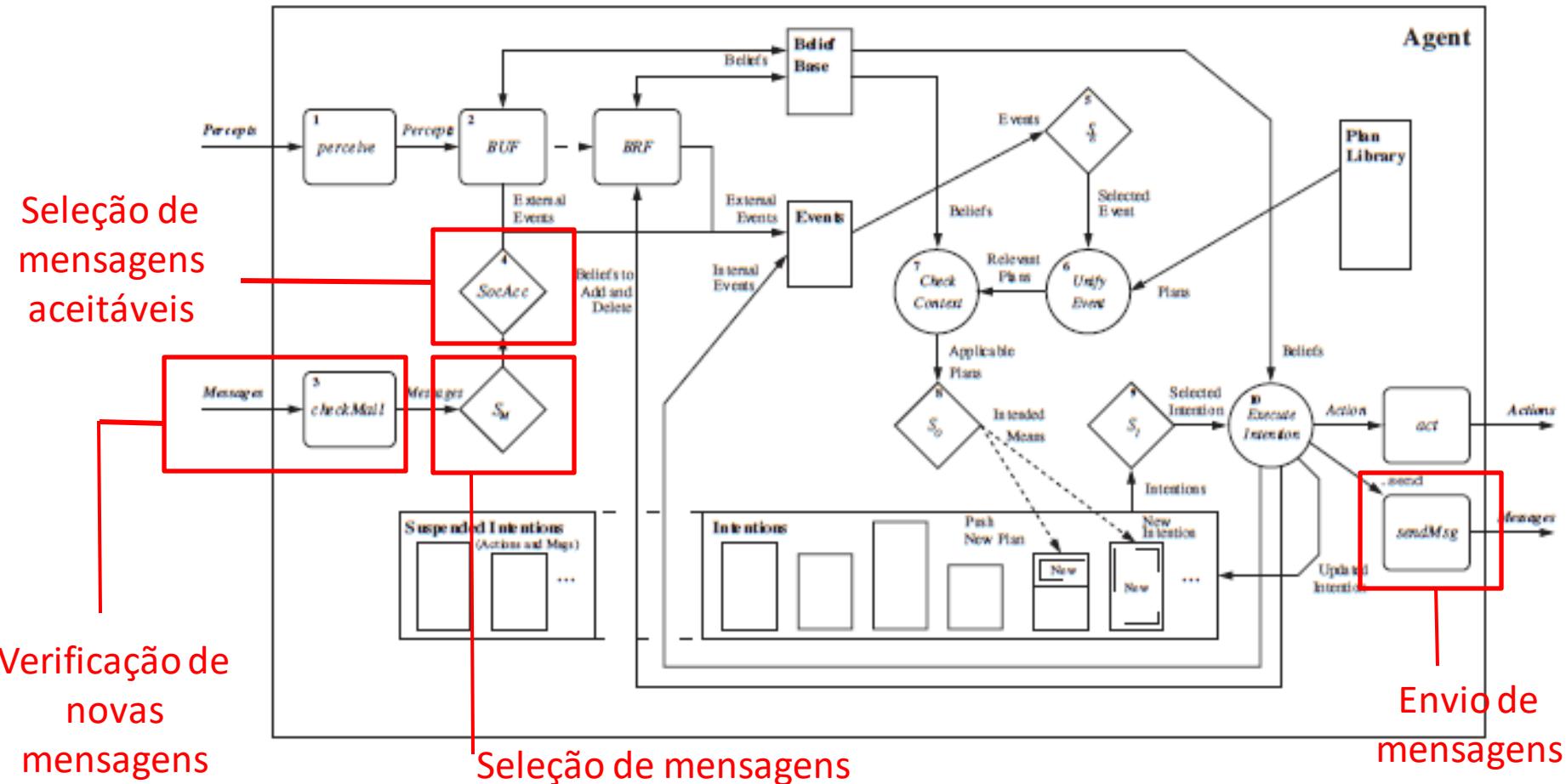
Knowledge Query and Manipulation Language



<https://redirect.cs.umbc.edu/csee/research/kqml/kqml.html>

Jason Framework: Comunicação

- **Reasoning Cycle**



Jason Framework: Comunicação

- **Estrutura**
.send(receiver; ilf; message; answer; timeout)
.broadcast(ilf; message)
- **receiver**
 - Nome do agente destinatário da mensagem (Ou lista de destinatários)
- **ilf**
 - Força ilocucionária do ato de fala (KQML)
- **message**
 - Conteúdo da mensagem
- **answer**
 - Um termo qualquer que irá armazenar a resposta (campo opcional)
- **timeout**
 - Tempo limite em milisegundos para receber uma resposta (campo opcional)

Jason Framework: Comunicação

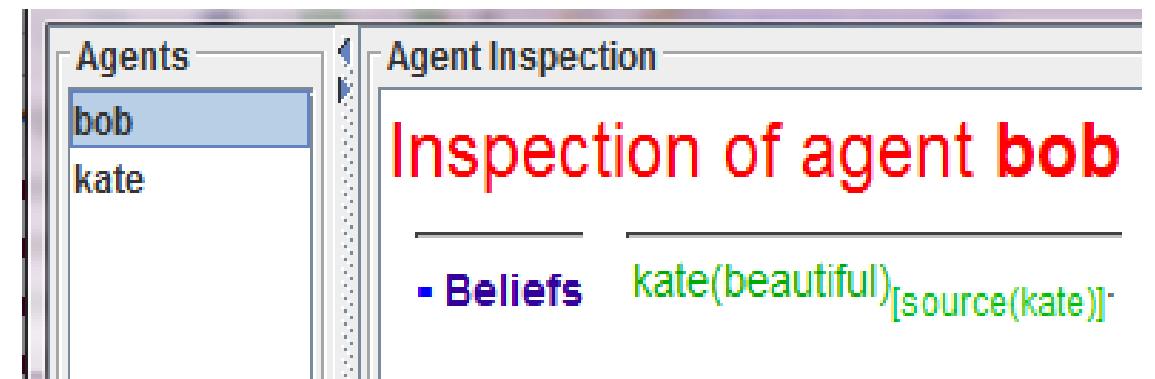
- **Performativas Implementadas**

1. tell

O agente remetente pretende que o receptor **acredite** que o conteúdo enviado é verdadeiro de acordo com as crenças do remetente.

Agente Kate

```
!talkTo.  
  
+!talkTo : true <-  
    .print("I'm beautiful.");  
    .send(bob, tell, kate(beautiful)).
```



Jason Framework: Comunicação

- **Performativas Implementadas**

2. untell

O agente remetente pretende que o receptor **não acredite** que o conteúdo enviado é verdadeiro de acordo com as crenças do remetente.

Agente Kate

```
!talkTo.  
  
+!talkTo : true <-  
  .print("Hi Bob, I'm Beautiful!");  
  .send(bob, tell, kate(beautiful)).  
  
+~kate(beautiful) [source(bob)] <-  
  .print("Sorry.");  
  .send(bob, untell, kate(beautiful)).
```

Agente Bob

```
+kate(beautiful) <-  
+~kate(beautiful);  
.print("No, You Don't!");  
.send(kate, tell, ~kate(beautiful)).
```

Jason Framework: Comunicação

- **Performativas Implementadas**

3. achieve

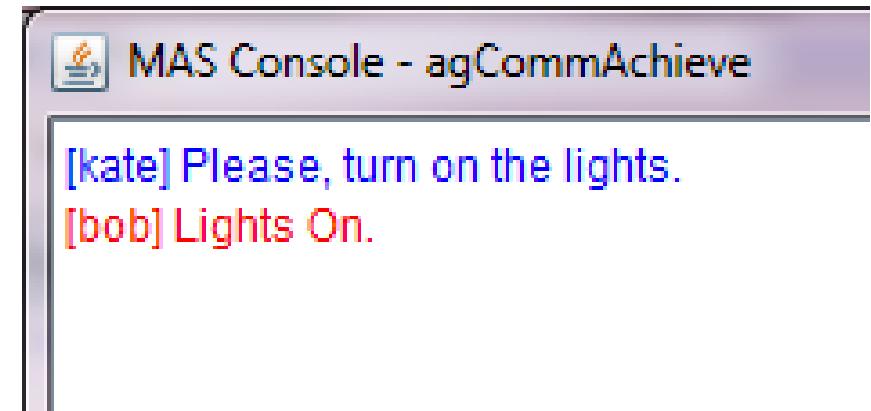
O agente remetente pede que o receptor **tente atingir um objetivo** de estado verdadeiro de acordo com conteúdo enviado.

Agente Kate

```
!talkTo.  
  
+!talkTo : true <-  
    .print("Please, turn on the lights.");  
    .send(bob, achieve, turn(on)).
```

Agente Bob

```
+!turn(on) <-  
    .print("Lights On.");
```



Jason Framework: Comunicação

- **Performativas Implementadas**

4. unachieve

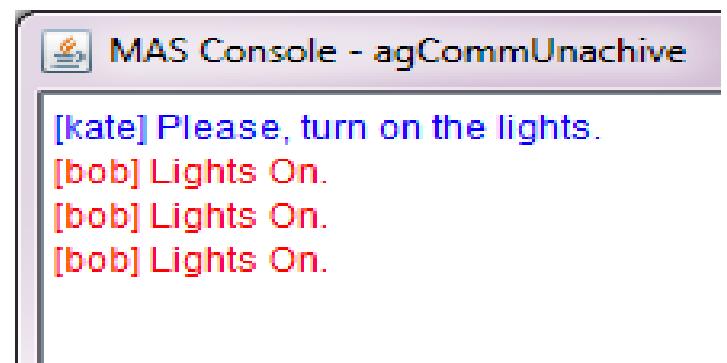
O agente remetente pede que o receptor **deixe de tentar atingir um objetivo** de estado verdadeiro de acordo com conteúdo enviado.

Agente Kate

```
!talkTo.  
  
+!talkTo : true <-  
    .print("Please, turn on the lights.");  
    .send(bob, achieve, turn(on)).  
  
+light(on) <-  
    .send(bob, unachieve, turn(on)).
```

Agente Bob

```
+!turn(on) <-  
    .print("Lights On.");  
    .send(kate, tell, light(on));  
    !turn(on).
```



Jason Framework: Comunicação

- **Performativas Implementadas**

5. askOne

O agente remetente deseja saber se a resposta do receptor para determinada questão é verdadeira.

Agente Kate

```
!talkTo.  
  
+!talkTo : true`<-  
    .print("What's your name?");  
    .send(bob, askOne, name(Name), Reply);  
+Reply.
```

Agente Bob

```
name(bob).
```



Jason Framework: Comunicação

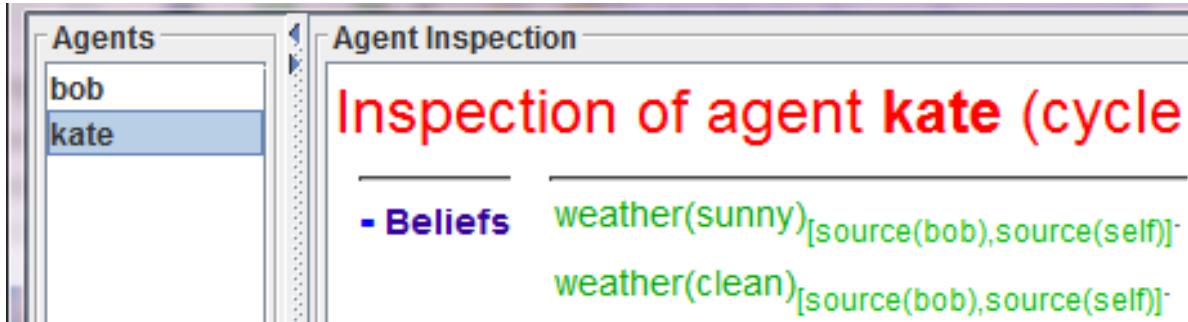
- **Performativas Implementadas**

6. askAll

O agente remetente deseja saber todas as respostas do receptor sobre uma questão.

Agente Bob

```
weather(clean).  
weather(sunny).
```



The screenshot shows the Jason Agent Inspection interface. On the left, there is a tree view labeled 'Agents' with two nodes: 'bob' and 'kate'. The node 'kate' is selected and highlighted in blue. To the right, under 'Agent Inspection', the title 'Inspection of agent kate (cycle)' is displayed in red. Below it, the section '- Beliefs' is shown in blue, listing two entries: 'weather(sunny)[source(bob),source(self)]' and 'weather(clean)[source(bob),source(self)]'.

Agente Kate

```
!goToBeach.  
  
+!goToBeach <-  
  !talkTo;  
  !analyze.  
  
+!talkTo : true <-  
  .print("What is the weather prevision?");  
  .send(bob, askAll, weather(Name)).
```

Jason Framework: Comunicação

- **Performativas Implementadas**

7. askHow

O agente remetente deseja saber todas implementações de planos do receptor para determinado plano.

```
MAS Console - agCommAskHow
[kate] Please, Can you teach me how to turn on the lights?
[kate] I don't know how to turn on the lights.
[kate] I don't know how to turn on the lights.
[kate] Lights On.
```

Agente Kate

```
!talkTo.  
!turn(on).  
  
+!talkTo : true <-  
    .print("Please, Can you teach me how to turn on the lights?");  
    .wait(5000);  
    .send(bob, askHow, "+!turn(on)").
```

Agente Bob

```
+!turn(on) <-  
.print("Lights On.").
```

- **Performativas Implementadas**

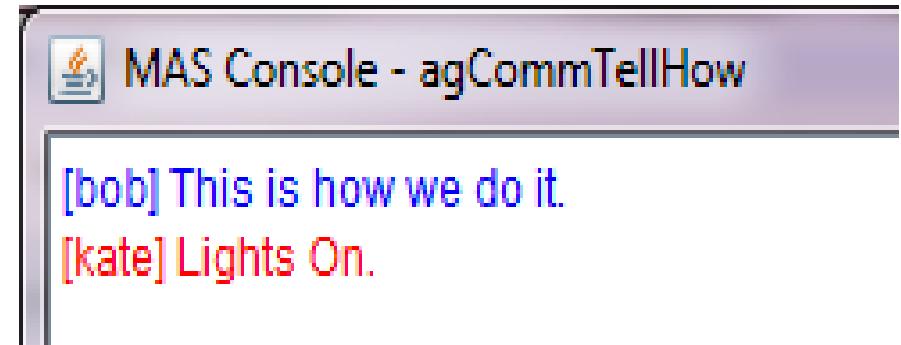
8. tellHow

O agente remetente informa ao agente receptor a implementação de um plano.

```
!teach(kate).                                Agente Bob

+!teach(kate) <-
  .print("This is how we do it.");
  .send(kate, tellHow, "+!turn(on) <- .print(\"Lights On.\")");
  .wait(3000);
  .send(kate, achieve, turn(on));

+!turn(on) <-
  .print("Lights On.");
```



- **Performativas Implementadas**

9. untellHow

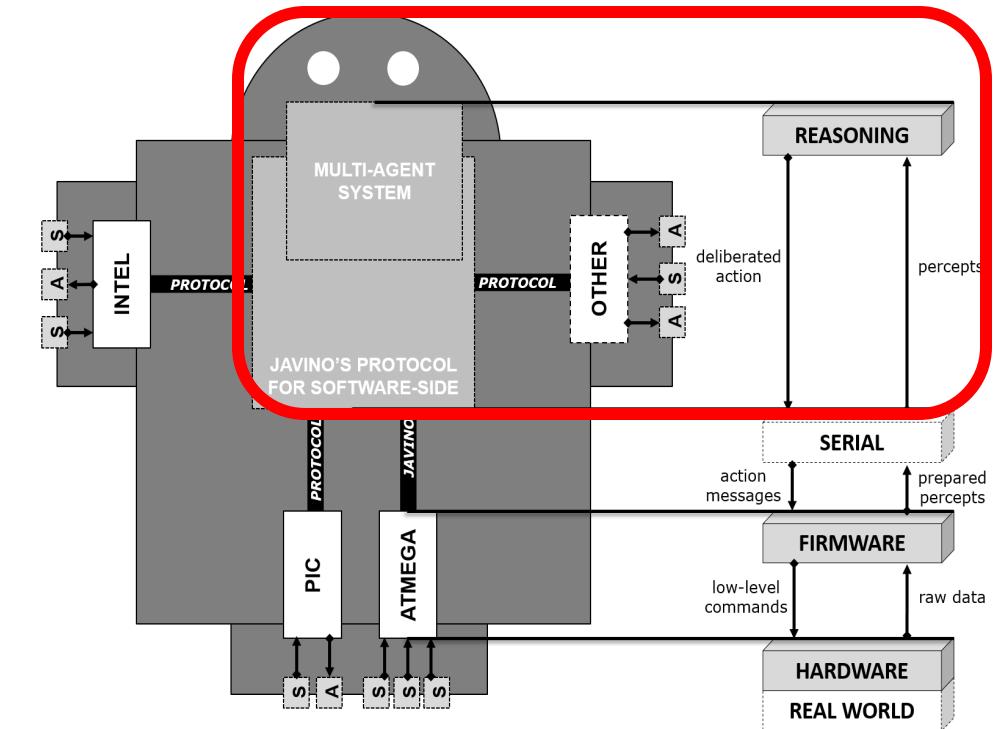
O agente remetente solicita ao agente destinatário a remoção da implementação de um plano da biblioteca de planos do receptor.

```
!teach(kate).          Agente Bob

+!teach(kate) <-
    .print("This is how we dance.");
    .wait(2000);
    .send(kate, tellHow, "@d +!dance : true <- .print(\"I'm dancing with myself!\")");
    .wait(1000); !dance.
)
.wait(2000);
.send(kate, untellHow, "@d");

@d
+!dance : true <-
    .print("I'm dancing with myself!");
    .wait(1000);
    !dance.
```

Programando um MultiAgent System



MultiAgent System: Exemplo

cozinheiro

```
1  /* Crenças Iniciais */
2  ultimoPedido(0).
3
4  /* Objetivos Iniciais */
5  !aguardarPedidos.
6
7  /* Planos */
8  +!aguardarPedidos: not preparandoPedido <-
9      .print("Aguardando Pedidos");
10     .wait(5000);
11     !aguardarPedidos.
12
13 +!aguardarPedidos: preparandoPedido <-
14     .wait(10000);
15     !aguardarPedidos.
16
17 +!pedido(Product,Qtd)[source(Cliente)] <-
18     +preparandoPedido;
19     ?ultimoPedido(N);
20     NrPedido=N+1;
21     -+ultimoPedido(NrPedido);
22     .print("Preparando o pedido Nr ",NrPedido);
23     .wait(2000);
24     .send(Cliente,tell,entregaComida(Product,Qtd));
25     -preparandoPedido.
```

comilao

```
1  /* Crenças Iniciais */
2  estoqueComida(0). energia(0).
3  /* Objetivos Iniciais */
4  !curtir.
5  /* Planos */
6  +!comer: sintoFome & estoqueComida(C) & C>3 & energia(E) & E<=10 <-
7      -+estoqueComida(C-3); -+energia(E+3);
8      .print("Comendo..... [Estoque de comida=",C-3,"] [Energia=",E+3,"]");
9      .wait(1000);
10     !comer.
11
12 +!comer: energia(E) & E>10 <-
13     .print("Satisfeito..... [Energia=",E,"]");
14     -sintoFome; .wait(2000); !curtir.
15
16 +!comer <-
17     ?estoqueComida(X);
18     .print("Preciso pedir comida..... [Estoque de comida=",X,"]");
19     .wait(2000); !pedirComida.
20
21 +!curtir: energia(E) & E>5 <-
22     .print("Curtindo...."); .wait(2000); -+energia(E-3);
23     -sintoFome; !curtir.
24
25 +!curtir: energia(E) & E<=5 <-
26     ?energia(X); .print("Sem Energia..... [Energia=",X,"]");
27     +sintoFome; .wait(2000); !comer.
28
29 +!pedirComida <-
30     .print("Pedindo Comida");
31     .send(cozinheiro,achieve,pedido(lanche,5));
32     .wait(5000); !comer.
33
34 +entregaComida(Product,Qtd)[source(Entregador)] <-
35     ?estoqueComida(X);
36     -+estoqueComida(X+Qtd);
37     .print("Oba! Chegou comida!!!!");
38     -entregaComida(Product,Qtd)[source(Entregador)].
```

<https://sourceforge.net/p/chonos/examples/ci/master/tree/02-multiAgent/multiAgentExample01/>

MultiAgent System: Exemplo 2

banco	cozinheiro	comilao
<pre> 1 /* Crenças Iniciais */ 2 saldo(comilao,20). 3 saldo(cozinheiro,10). 4 5 /* Objetivos Iniciais */ 6 7 /* Planos */ 8 +!pix(Destino,Valor)[source(Origem)]: saldo(Origem,Saldo) 9 & Saldo >= Valor <- 10 NovoSaldo = Saldo - Valor; 11 -saldo(Origem,Saldo); 12 +saldo(Origem,NovoSaldo); 13 14 ?saldo(Destino,SaldoDestino); 15 NovoSaldoDestino = SaldoDestino + Valor; 16 -saldo(Destino,SaldoDestino); 17 +saldo(Destino,NovoSaldoDestino); 18 19 .random(NrRegistro); 20 +operacao(NrRegistro,Origem,Destino,Valor); 21 .print("Pix realizado de [",Origem, 22 "] para [",Destino, 23 "] valor [",Valor, 24 "] código [",NrRegistro,"]"); 25 .send(Origem,tell,operacao(NrRegistro,Origem,Destino,Valor)). 26 27 +!pix(Destino,Valor)[source(Origem)]: saldo(Origem,Saldo) 28 & Saldo < Valor <- 29 .print("Saldo insuficiente... saldo=",Saldo); 30 .send(Origem,tell,semDinheiro). </pre>	<pre> 1 /* Crenças Iniciais */ 2 ultimoPedido(0). 3 /* Objetivos Iniciais */ 4 !aguardarPedidos. 5 6 /* Planos */ 7 +!aguardarPedidos: not preparandoPedido <- 8 .print("Aguardando Pedidos"); 9 .wait(5000); 10 !aguardarPedidos. 11 12 +!aguardarPedidos: preparandoPedido<- 13 .wait(10000); 14 !aguardarPedidos. 15 16 +!pedido(Product,Qtd,Pix)[source(Cliente)] <- 17 .send(banco,askOne,operacao(Pix,Cliente,cozinheiro,Qtd),Reply); 18 +Reply; 19 ?operacao(Registro,Cliente,cozinheiro,Qtd); 20 +preparandoPedido; 21 !prepararPedido(Product,Qtd,Pix,Registro,Cliente); 22 +atendido(Pix); 23 -preparandoPedido. 24 25 +!prepararPedido(Product,Qtd,Pix,Validacao,Cliente): Pix=Validacao 26 & not atendido(Pix)<- 27 ?ultimoPedido(N); 28 NrPedido=N+1; 29 +ultimoPedido(NrPedido); 30 .print("Preparando o pedido Nr ",NrPedido); 31 .wait(2000); 32 .send(Cliente,tell,entregaComida(Product,Qtd)). 33 34 -!prepararPedido(Product,Qtd,Pix,Validacao,Cliente) <- 35 .print("Pix inválido ou já atendido"). </pre>	<pre> 1 /* Crenças Iniciais */ 2 estoqueComida(0). 3 energia(0). 4 /* Objetivos Iniciais */ 5 !curtir. 6 7 /* Planos */ 8 +!comer: sintoFome & estoqueComida(C) & C>3 9 & energia(E) & E<=10 <- 10 +estoqueComida(C-3); +energia(E+3); 11 .print("Comendo..... [Geladeira=",C-3,"] [Energia=",E+3,"]"); 12 .wait(1000); !comer. 13 14 +!comer: energia(E) & E>10 <- 15 .print("Satisfeito..... [Energia=",E,"]"); 16 -sintoFome; .wait(2000); !curtir. 17 18 +!comer <- 19 ?estoqueComida(X); 20 .print("Preciso pedir comida.... [Geladeira=",X,"]"); 21 .wait(2000); !pedirComida. 22 23 +!curtir: energia(E) & E>5 <- 24 .print("Curtindo...."); .wait(2000); -+energia(E-3); 25 -sintoFome; !curtir. 26 27 +!curtir: energia(E) & E<=5<- 28 ?energia(X); .print("Sem Energia..... [Energia=",X,"]"); 29 +sintoFome; .wait(2000); !comer. 30 31 +pedirComida <- 32 .print("Pedindo Comida"); 33 .send(banco,achieve,pix(cozinheiro,5)); 34 .wait(1000); 35 ?operacao(CodPix,comilao,cozinheiro,5); 36 .send(cozinheiro,achieve,pedido(lanche,5,CodPix)); 37 -operacao(CodPix,comilao,cozinheiro,5); 38 .wait(5000); !comer. 39 40 +entregaComida(Product,Qtd)[source(Entregador)] <- 41 ?estoqueComida(X); 42 +estoqueComida(X+Qtd); 43 .print("Oba! Chegou comida!!!!"); 44 -entregaComida(Product,Qtd)[source(Entregador)]. 45 46 +semDinheiro[source(banco)] <- 47 .drop_desire(curir); 48 .drop_desire(pedirComida); 49 .print("Ababou a festa :("). </pre>

<https://sourceforge.net/p/chonos/examples/ci/master/tree/02-multiAgent/multiAgentExample02/>

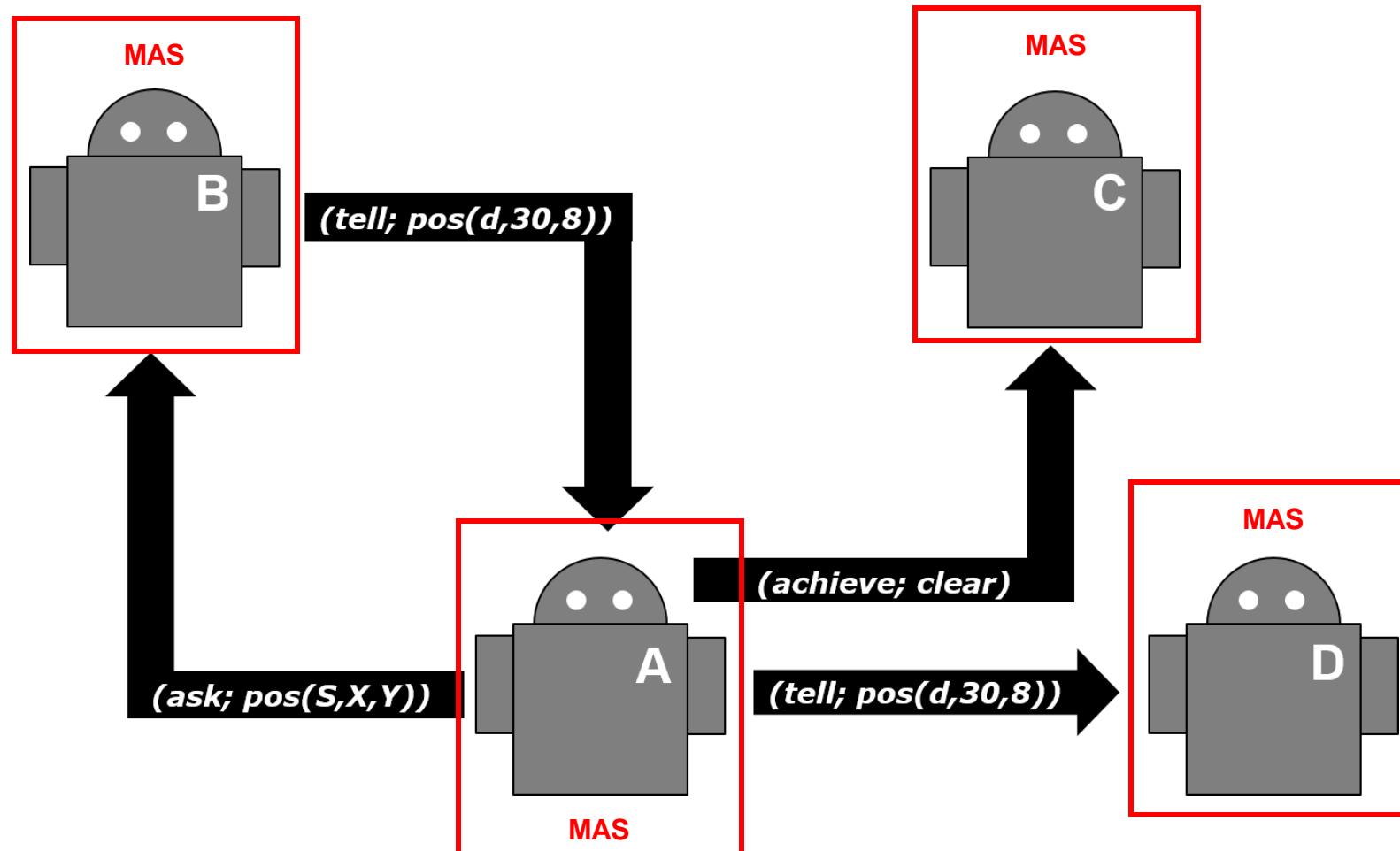
COMUNICAÇÃO ENTRE SMAs

Open MultiAgent System

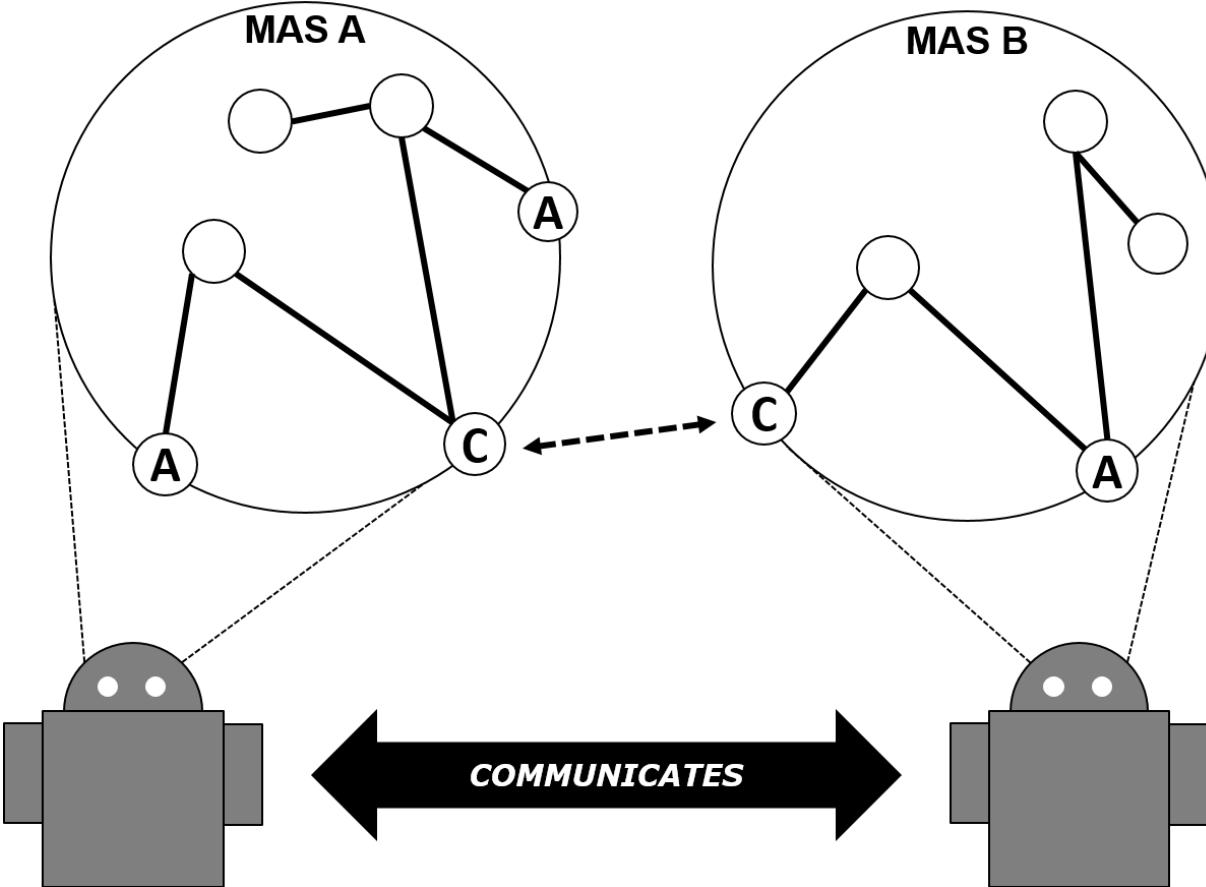
Um SMA pode ser Aberto, dessa forma é possível:

- **troca de mensagens** com agentes estrangeiros;
- **entrada ou saída** de agentes no SMA.

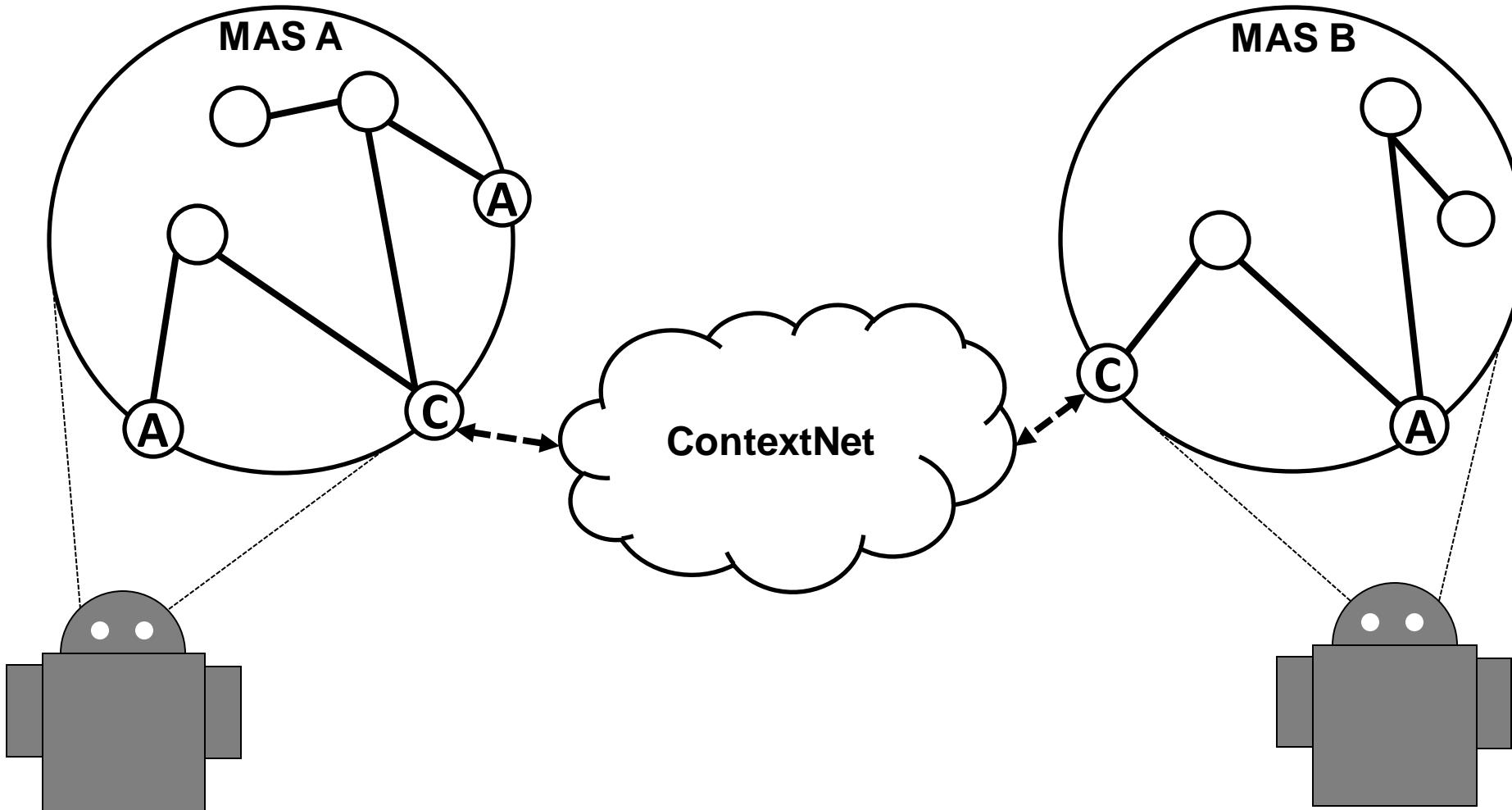
Open MultiAgent System



Agentes Comunicadores



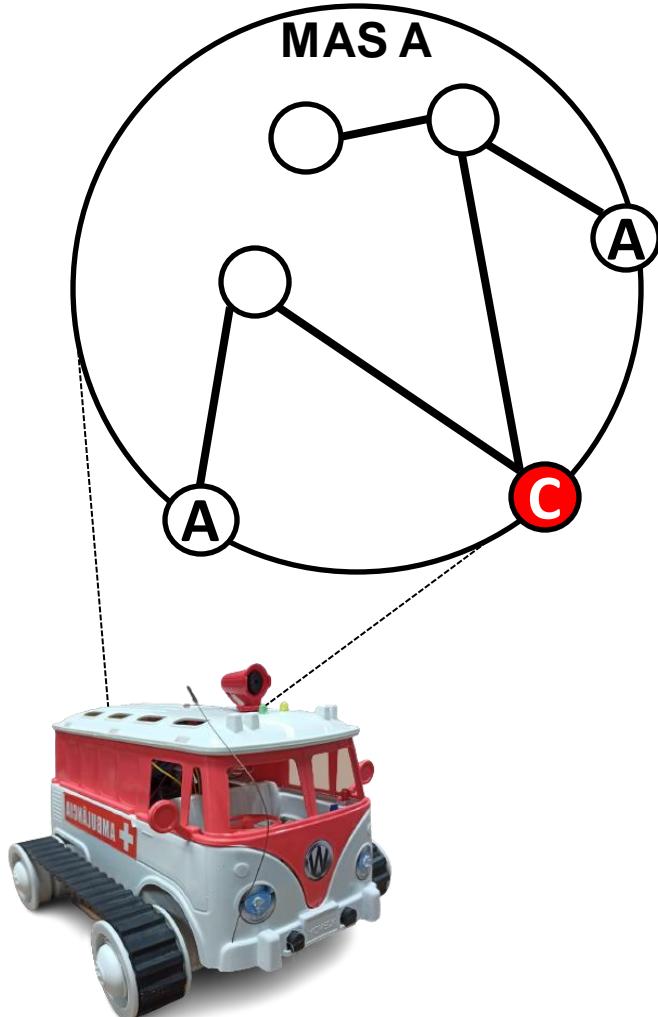
Agentes Comunicadores: Middleware IoT



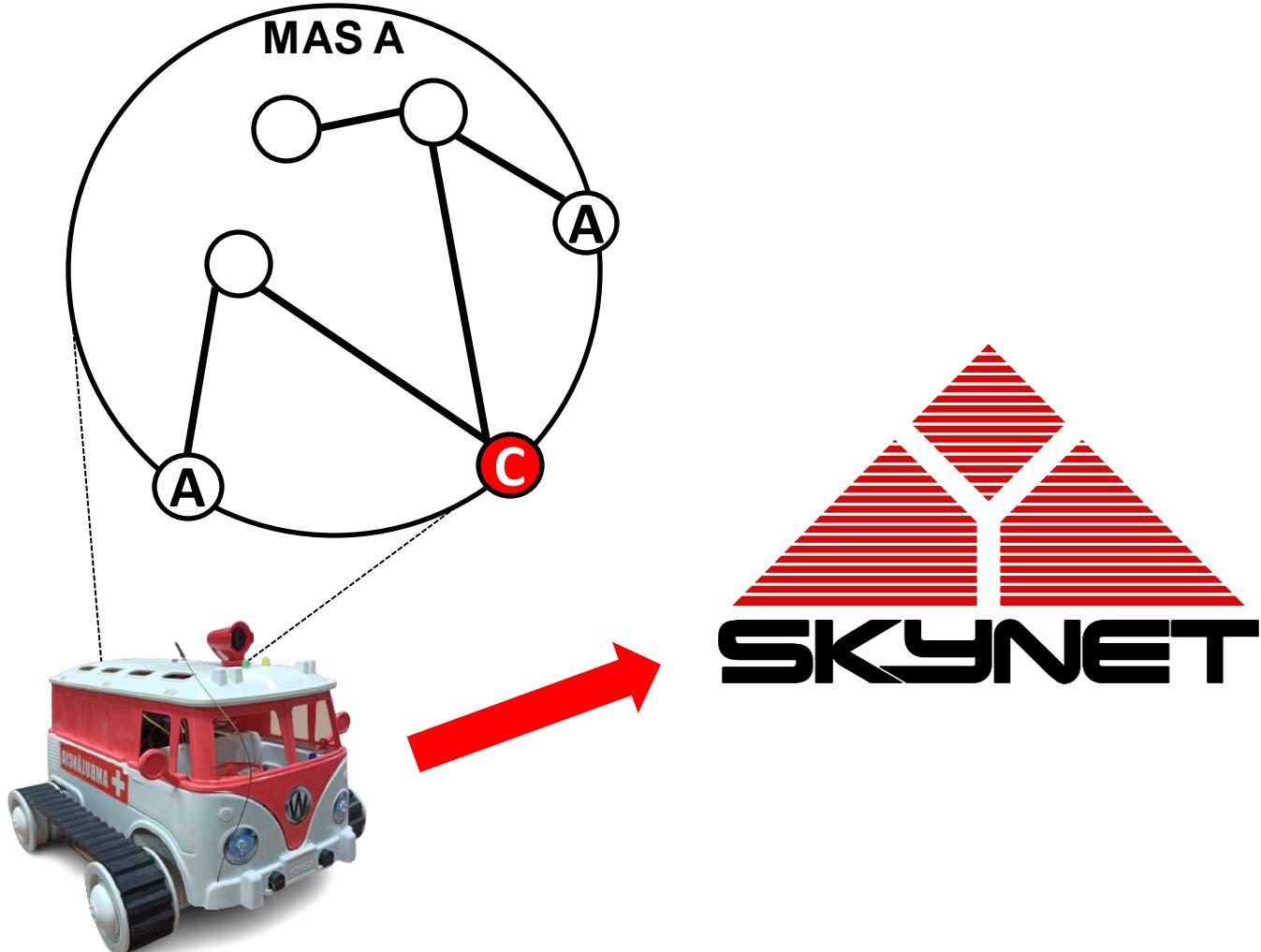
Agentes Comunicadores: Middleware IoT

O ContextNet é um *middleware* que visa a **aplicações colaborativas** abrangentes de pequena e grande escala, como **monitoramento on-line** ou **coordenações de atividades** de **entidades móveis** e **compartilhamento de informações**.

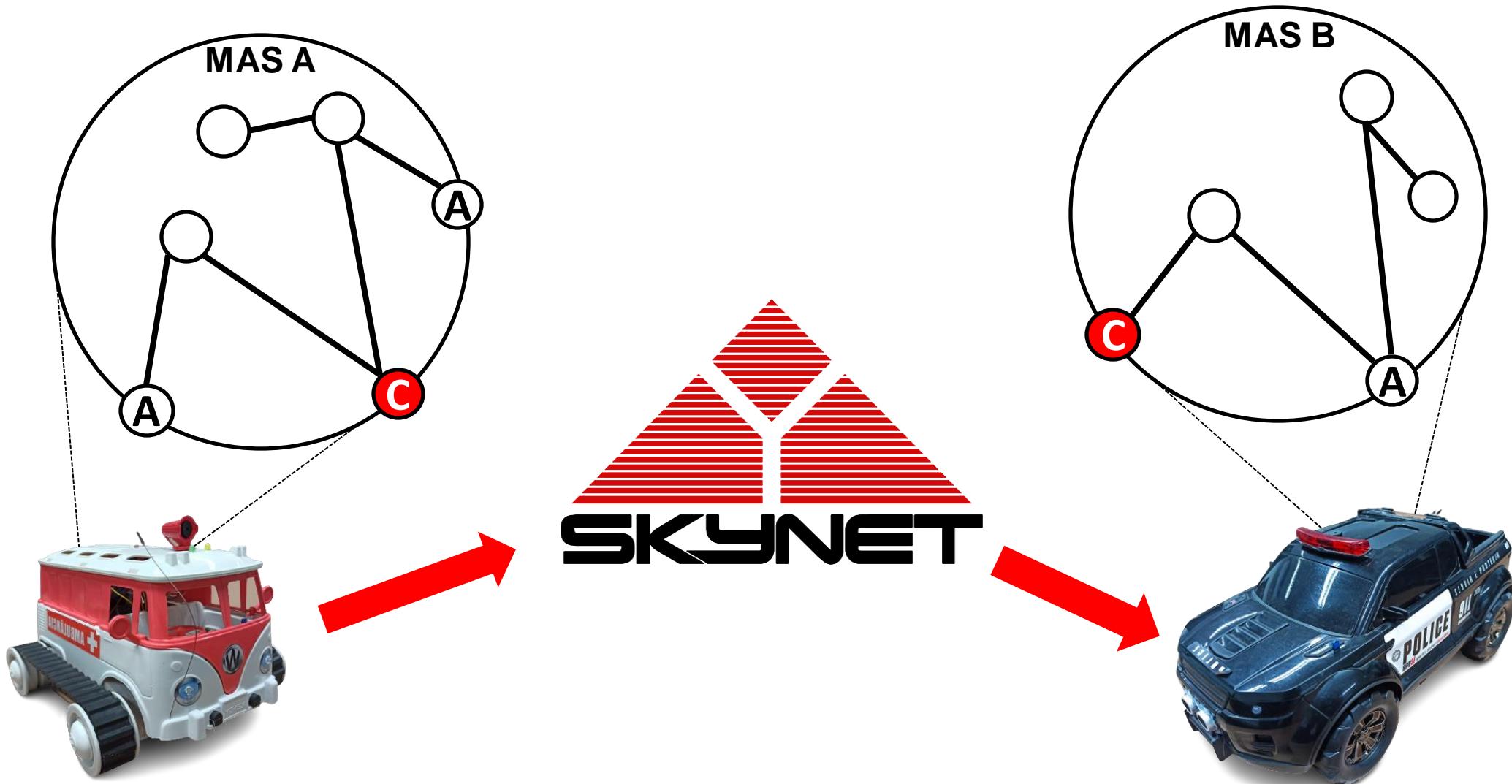
Agentes Comunicadores: Middleware IoT



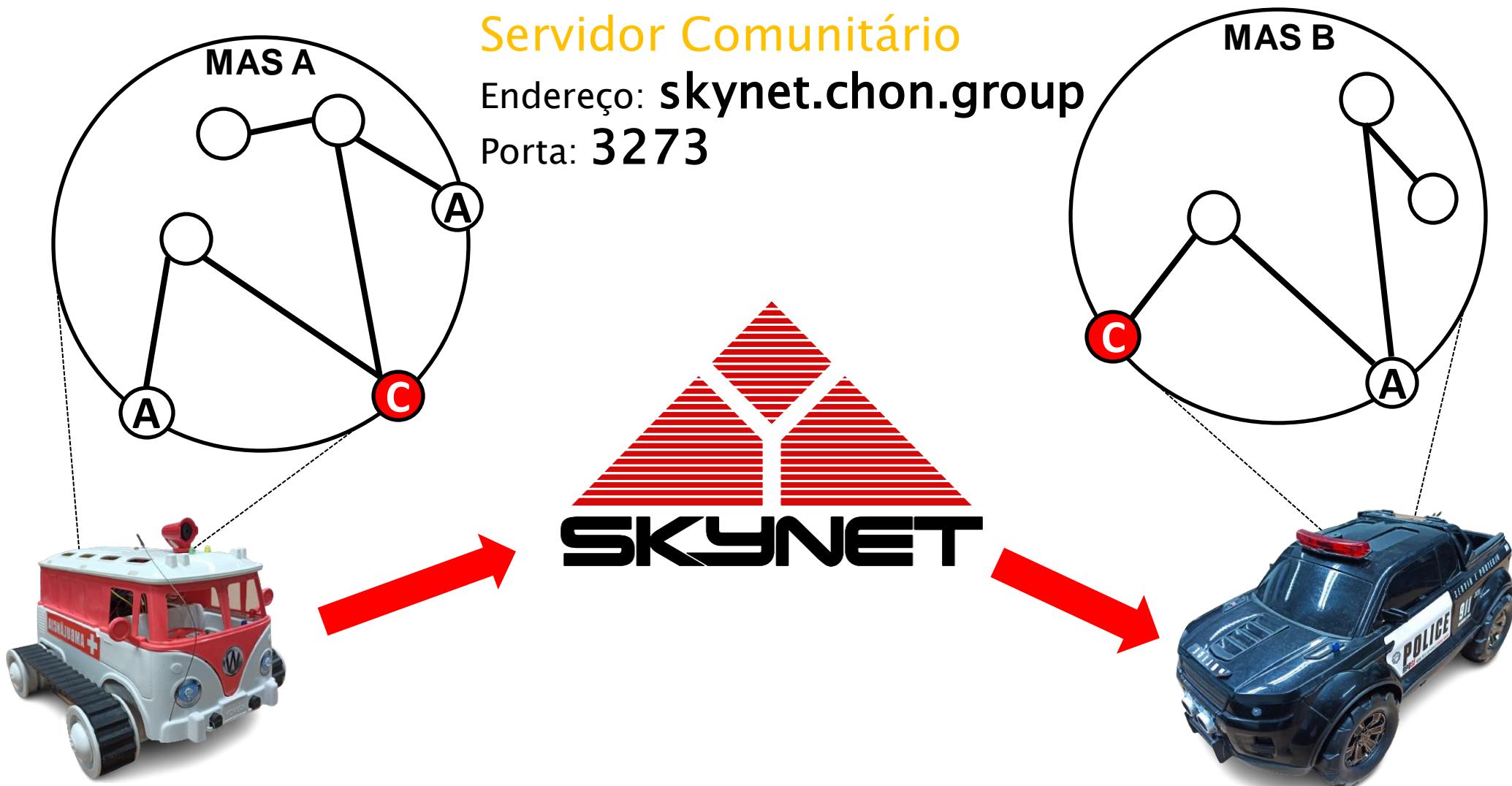
Agentes Comunicadores: Middleware IoT



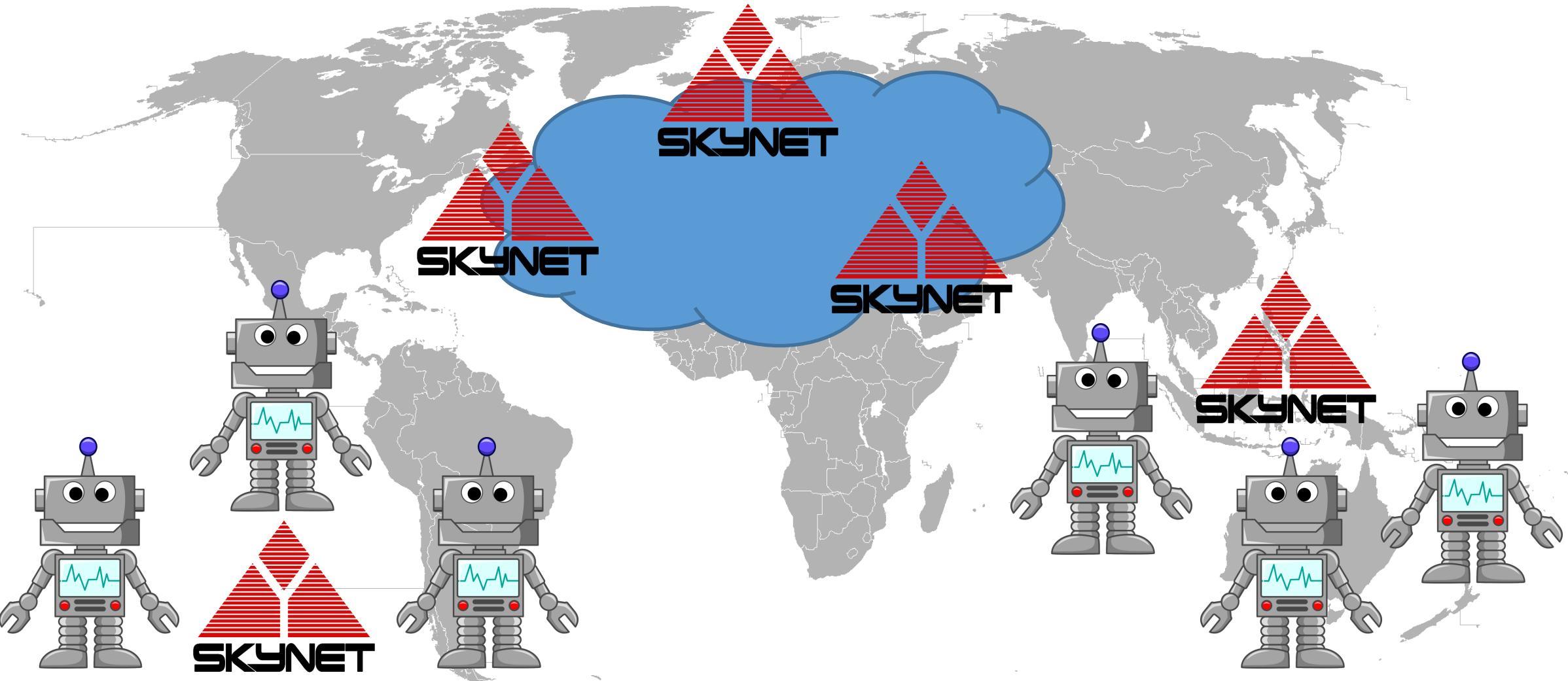
Agentes Comunicadores: Middleware IoT



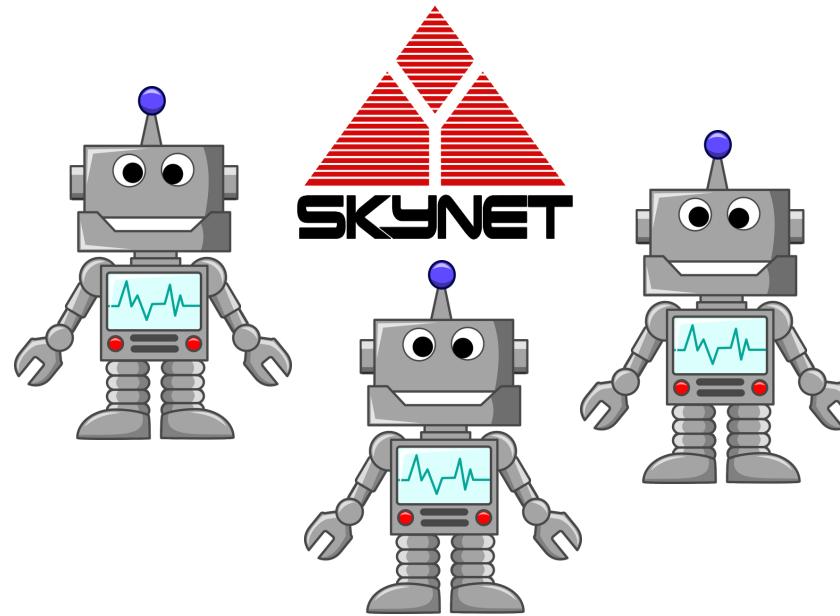
Agentes Comunicadores: Middleware IoT



SkynetServer: Geograficamente Distribuídos



SkynetServer: Local



**Depende do
DEBIAN 10!**

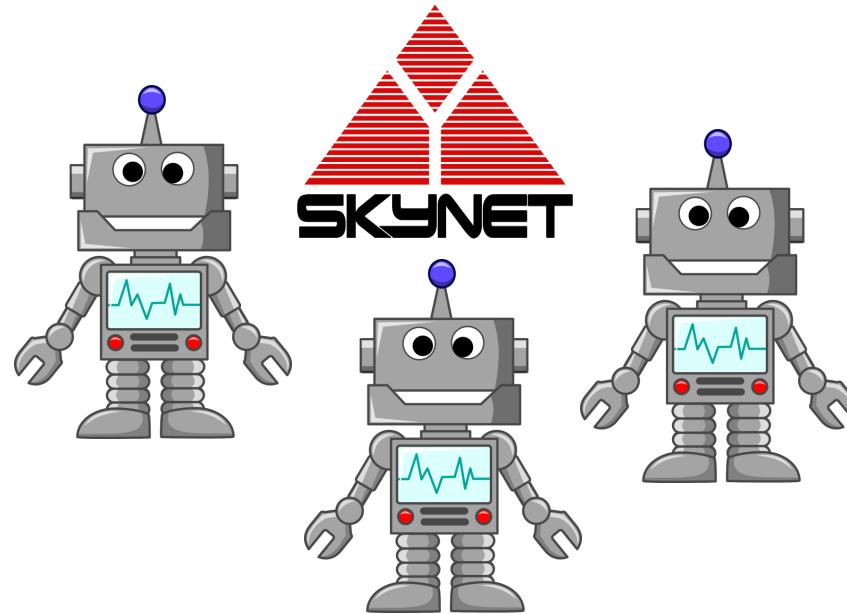
How to Install?

```
apt update; apt install git
git -C /opt clone git://git.code.sf.net/p/chonos/skynet chonos-skynet
/opt/chonos-skynet/bin/install.sh
reboot
```

<https://sourceforge.net/p/chonos/skynet/>

A screenshot of a SourceForge project page. At the top, there's a navigation bar with links for "Open Source Software", "Business Software", and "Resources". Below the navigation bar, the page title is "ChonOS SkyNet Server" with the subtitle "A specific-purpose GNU/Linux distribution for Embedded MAS". It shows the status as "Alpha" and credits "Brought to you by: kadupantoja, nilsonmori". On the left, there's a sidebar with links for "Summary", "Files", and "Git". The "Git" section is expanded, showing a tree view with a commit hash "4762e2" and a master branch. It also includes "HTTPS" and "git://" access options. On the right, there's a sidebar with links for "SkyNet Server", "Embedded MAS", "RaspberryPi", "SysConfig", "Examples", and "Debian Packages".

SkynetServer: VMLocal



Depende do <https://www.virtualbox.org/>

Home / Browse / Science & Engineering / Artificial Intelligence / Intelligent Agents / ChonOS / Files



ChonOS Files

A specifical-purpose GNU/Linux distribution for Embedded MAS

Status: Alpha Brought to you by: [kadupantoja](#), [nilsonmori](#)

Summary

Files

Reviews

Sup

[Home](#) / vms

Name

Modified

Size

Parent folder

[skynet-virtualbox.ova](#)

< 2 hours ago

1.2 GB

<https://sourceforge.net/projects/chonos/files/vms/>

```
skynetLocal [Executando] - Oracle VM VirtualBox
Arquivo Máquina Visualizar Entrada Dispositivos Ajuda
Iniciando Skynet...
Utilize o seguinte comando para conectar --> .connectCN(192.168.0.103 ,3273,SEU-UUID)
Use Ctrl+C para parar
Working Directory = /root
Gateway started...
Gateway MR-UDP IP: 192.168.0.103:3273
ReliableSocket: new utils pool size = 3
```

Agentes Comunicadores: Ações Internas

- **Communicator** Internal Actions:

- **.sendOut(agentUuid, force, message)**

envia uma mensagem de um agente comunicador para outro comunicador de um SMA distinto.

- **.connectCN(servidor, porta, UUID)**

se conecta ao servidor IoT com um id específico para o agente.

- **.disconnectCN**

desconecta do servidor IoT atualmente conectado.

Exemplo: Comunicação entre SMA Embarcados

Exemplo: Comunicação entre SMA Embarcados



Embedded MAS
enterprise

Exemplo: Comunicação entre SMA Embarcados

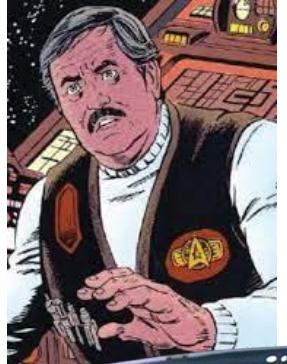
agent scott



Embedded MAS
enterprise

Exemplo: Comunicação entre SMA Embarcados

agent scott



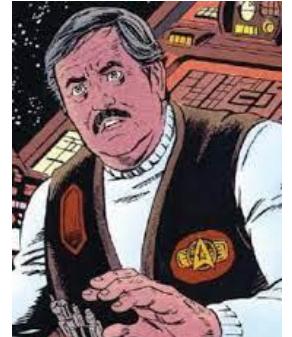
agent uhura



Embedded MAS
enterprise

Exemplo: Comunicação entre SMA Embarcados

agent scott

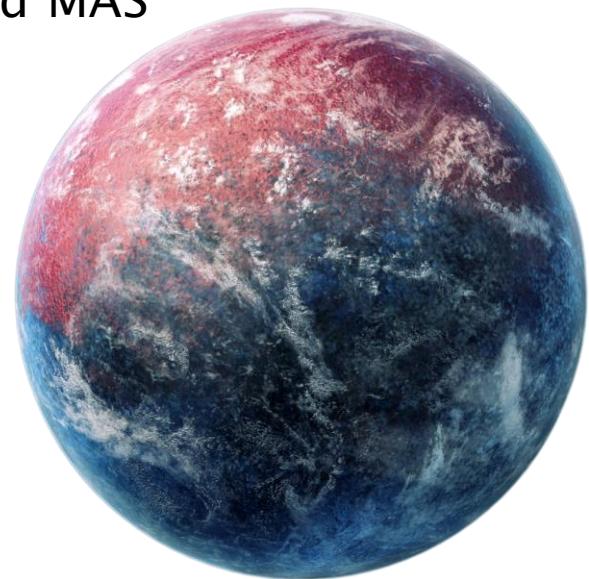


agent uhura



Embedded MAS
enterprise

Embedded MAS
bajor



Exemplo: Comunicação entre SMA Embarcados

agent scott

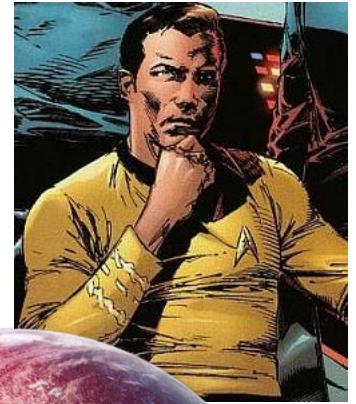


agent uhura



Embedded MAS
bajor

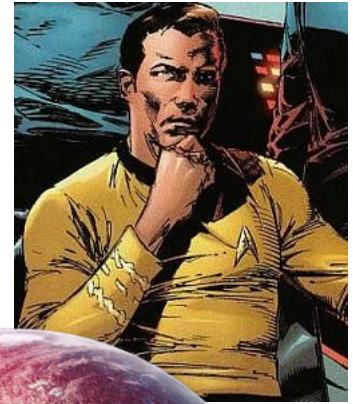
agent kirk



Exemplo: Comunicação entre SMA Embarcados



This is Lieutenant
Nyota Uhura,
Communications
officer.

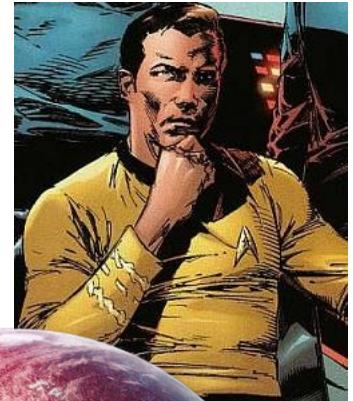


Exemplo: Comunicação entre SMA Embarcados



This is Lieutenant
Nyota Uhura,
Communications
officer.

Kirk to Enterprise...
Damage report!



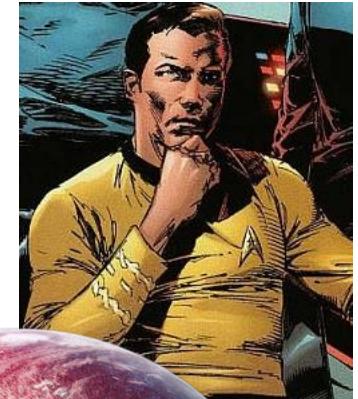
Exemplo: Comunicação entre SMA Embarcados



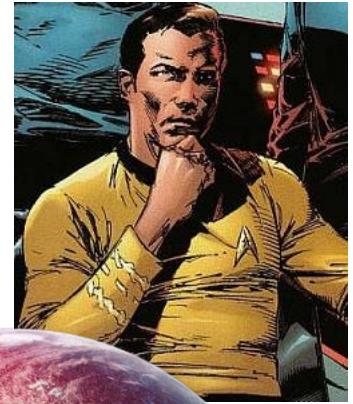
This is Lieutenant
Nyota Uhura,
Communications
officer.

Kirk to Enterprise...
Damage report!

Commander Kirk,
Deck 2 compromised!



Exemplo: Comunicação entre SMA Embarcados



Exemplo: Comunicação entre SMA Embarcados



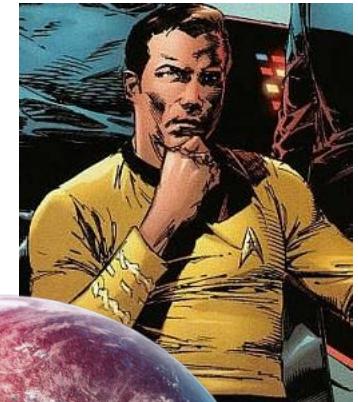
```
1 /* Initial beliefs and rules */
2
3 /* Initial goals */
4
5 /* Plans */
6 +redAlert[source(X)] <-
7   .print("Computer, Commander Montgomery Scott, Chief Engineering Office");
8   .print("Entreprise, Red Alert!");
9   -redAlert[source(X)].
10
```

```
1 /* Crenças Iniciais */
2 // Gerar UUID aleatório ---> https://www.uuidgenerator.net/
3 myUUID("b4bec4b9-221a-455c-9f3b-9b49bd9e8440").
4
5 //public SkynetServer
6 //skyNetAddress(skynet.chon.group).
7
8 //local IP of SkynetServer
9 skyNetAddress("192.168.0.103").
10 skyNetPort(3273).
11
12 /* Objetivos Iniciais */
13 !conf.
14
15 /* Planos */
16 +!conf : myUUID(ID) & skyNetAddress(Server) & skyNetPort(Port) <-
17   .print("This is Lieutenant Nyota Uhura, Communications officer");
18   .connectCN(Server,Port,ID).
19
20 +!damageReport[source(X)] <-
21   .print("Commander Kirk, Deck 2 compromised!");
22   .sendOut(X,tell,report("Deck 2)).
23
24 +communication(trying)[source(X)] <-
25   .print("Entreprise Listen ",X);
26   .sendOut(X,tell,communication(ok));
27   -communication(trying)[source(X)].
28
29 +retransmit(Dest,Msg)[source(X)] <-
30   .send(Dest,tell,Msg);
31   -retransmit(Dest,Msg)[source(X)].
32
```

https://sourceforge.net/p/chonos/examples/ci/master/tree/03-limitedOpenMultiAgent/limitedOpenExample01/ncc_1701/

Exemplo: Comunicação entre SMA Embarcados

```
1 /* Crenças Iniciais */
2 uhuraUUID("b4bec4b9-221a-455c-9f3b-9b49bd9e8440"). //https://www.uuidge
3 myUUID("cdc19c19-5616-4fc5-8d54-372631dd8eff").
4
5 //skyNetAddress(skynet.chon.group).           //public SkynetServer
6 skyNetAddress("192.168.0.103").               //local IP of Your SkynetSer
7 skyNetPort(3273).
8
9 /* Objetivo Inicial */
10 !contact.
11
12 /* Planos */
13
14 +!conf : myUUID(ID) & skyNetAddress(Server) & skyNetPort(Port) <-
15   .print("This is Commander James T. Kirk, Enterprise's Captain");
16   .connectCN(Server,Port,ID);
17   +connected;
18   !testComm.
19
20 +!contact: connected & uhuraUUID(Uhura) & communication(ok)<-
21   .print("Damage report!");
22   .sendOut(Uhura, achieve, damageReport).
23
24 +!contact: not connected <-
25   .print("Without Communication!");
26   !conf.
27
28 +report(Damages)[source(X)] <-
29   .print("Damages in ", Damages);
30   .print("Stay in standard orbit and wait for instructions.");
31   .sendOut(X, tell, retransmit(scott, redAlert));
32   .wait(2000);
33   .disconnectCN.
34
35 +!testComm : connected & uhuraUUID(Uhura) & not communication(ok)<-
36   .print("Kirk to Enterprise...");
37   .sendOut(Uhura, tell, communication(trying));
38   .wait(2500);
39   !testComm.
40
41 +!testComm: communication(ok) <- !contact.
42
```



<https://sourceforge.net/p/chonos/examples/ci/master/tree/03-limitedOpenMultiAgent/limitedOpenExample01/bajor/>

Transferência de Agentes entre SMAs

Protocolo de Transferência de Agentes

Os agentes móveis são agentes especiais **capazes de transcender** seu SMA podendo mover-se, por exemplo, para outro SMA. Os agentes móveis também são **capazes de interagir** com agentes de outros SMA e também transferir-se para um ambiente chamado de ambiente aberto, onde agentes de diferentes SMA podem interagir e trocar informações.

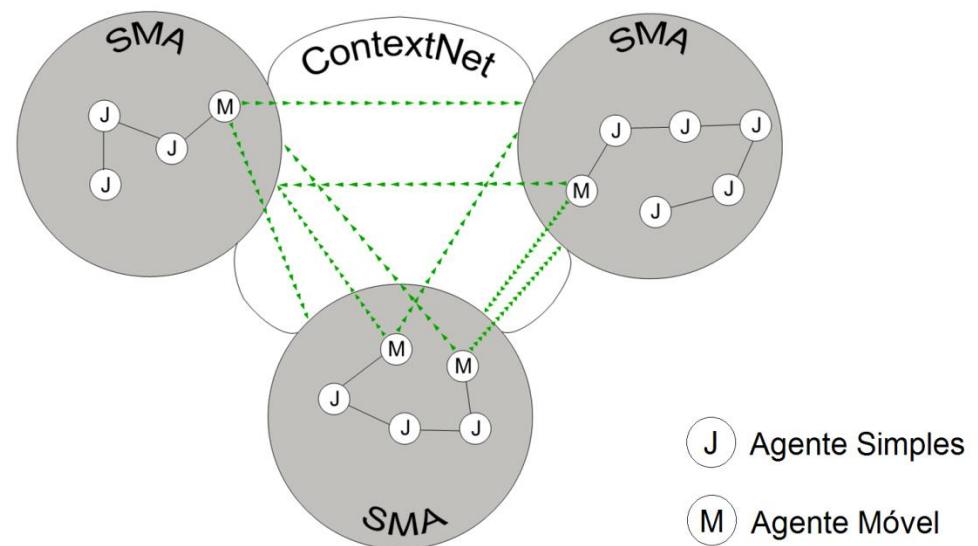


Figura 1. Transferência de agentes móveis

Protocolo de Transferência de Agentes

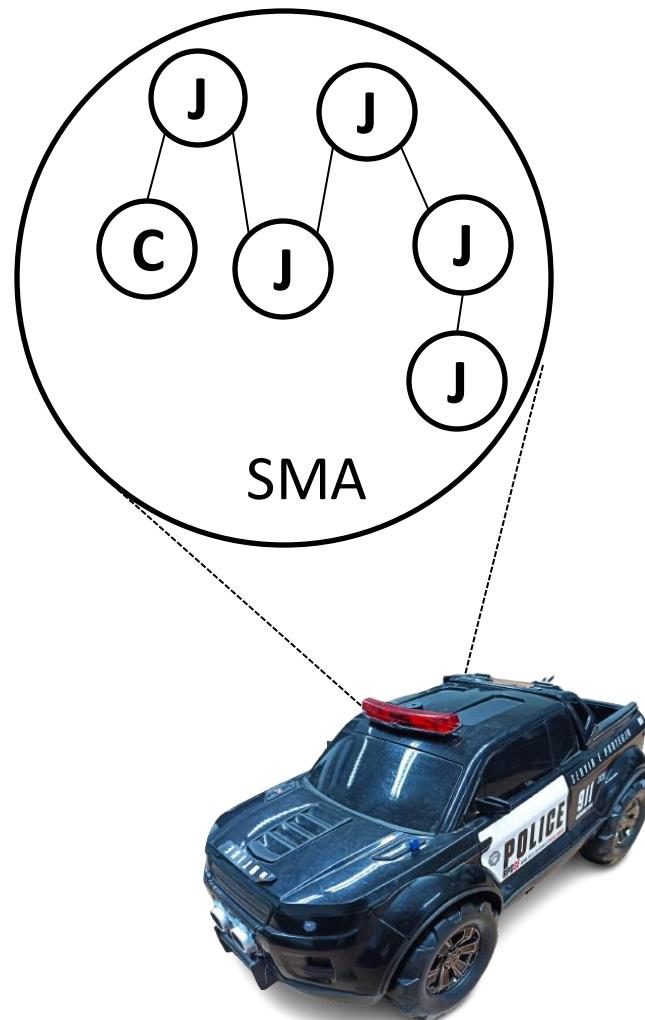
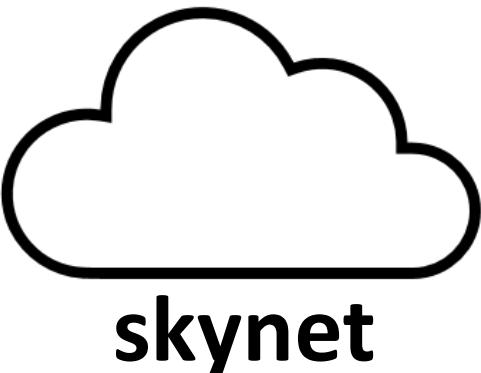
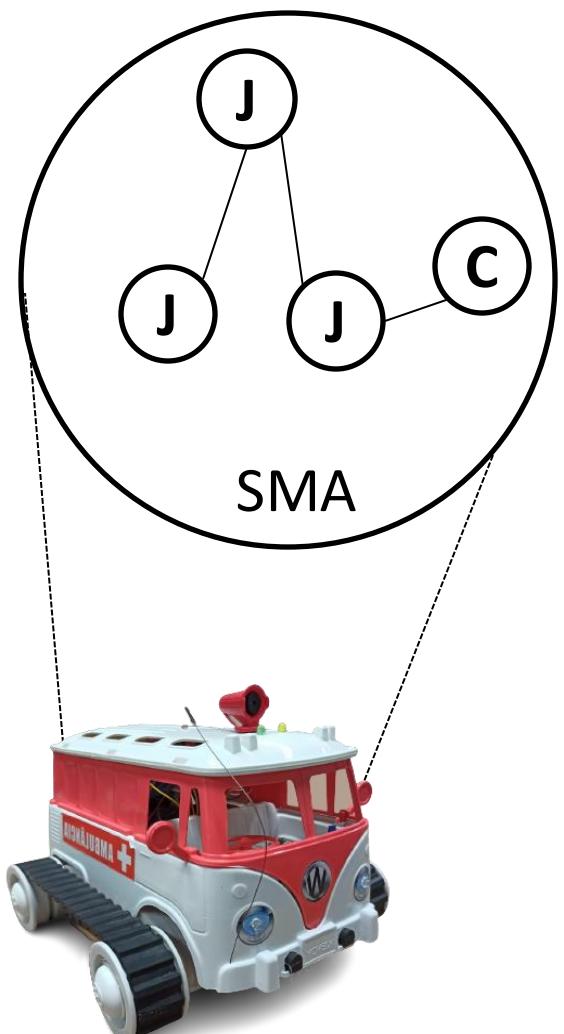
Um **agente cognitivo** está situado em um **SMA** qualquer e ambientado em um **dispositivo**, este agente fica “preso” ao SMA e ao dispositivo, e caso o dispositivo seja danificado o agente cognitivo **não consegue se transferir** para outro SMA.

Protocolo de Transferência de Agentes

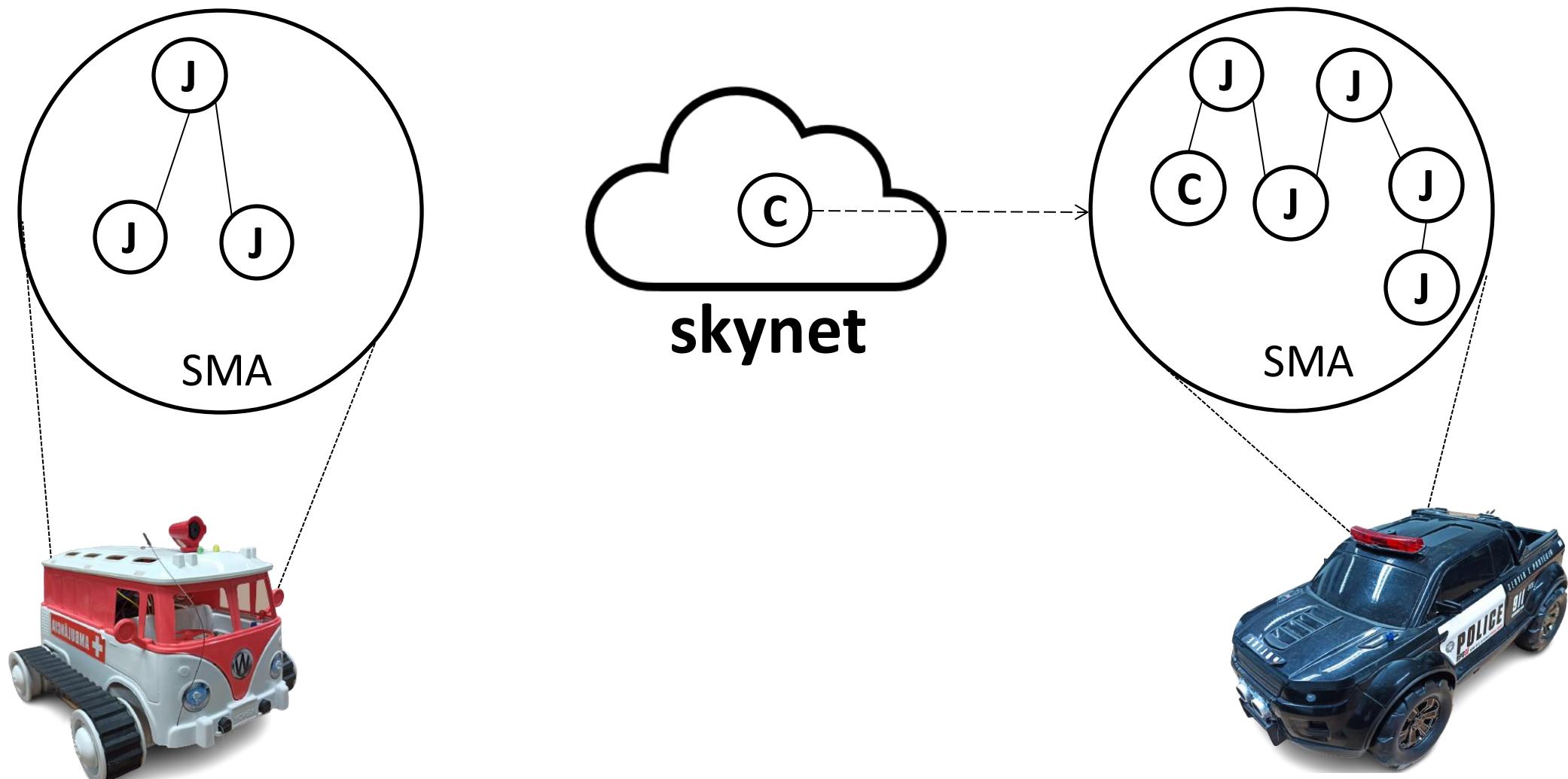
O protocolo de transferência de agentes prevê três possíveis relações entre o agente móvel com o novo SMA.

- Mutualismo
- Inquilinismo
- Predatismo

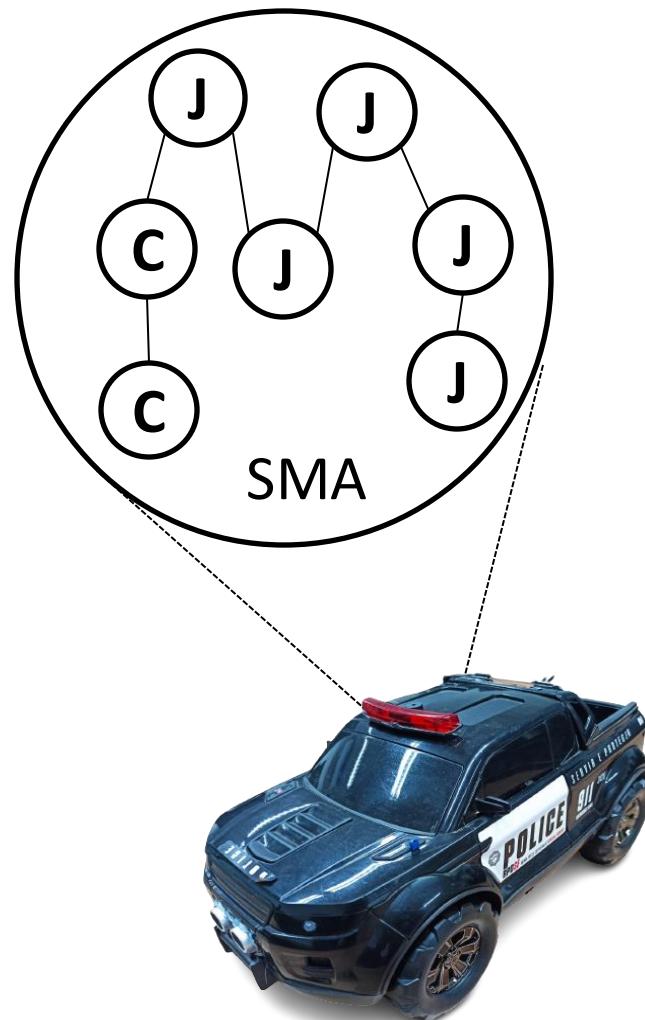
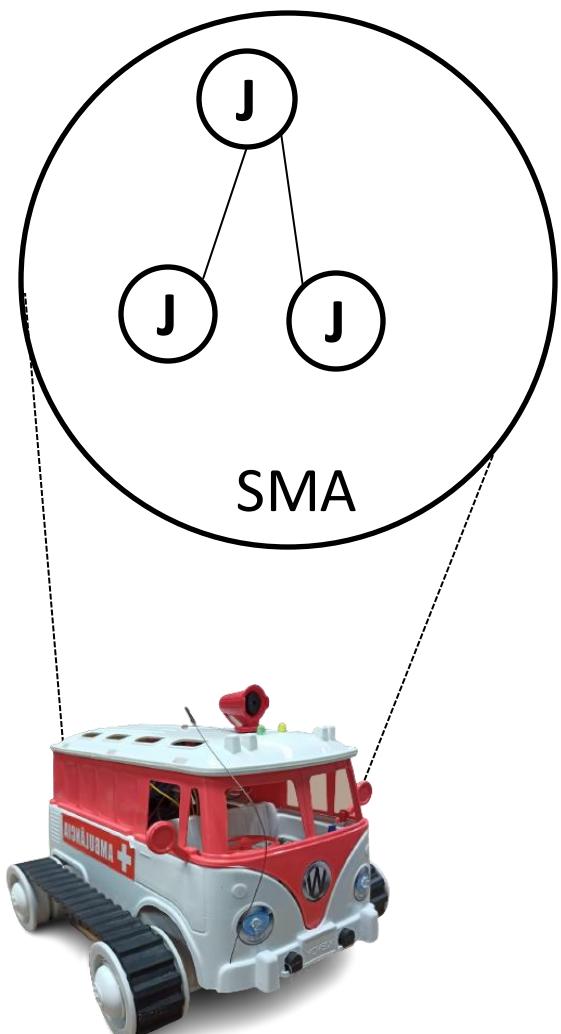
Bio-Inspired: Mutualismo



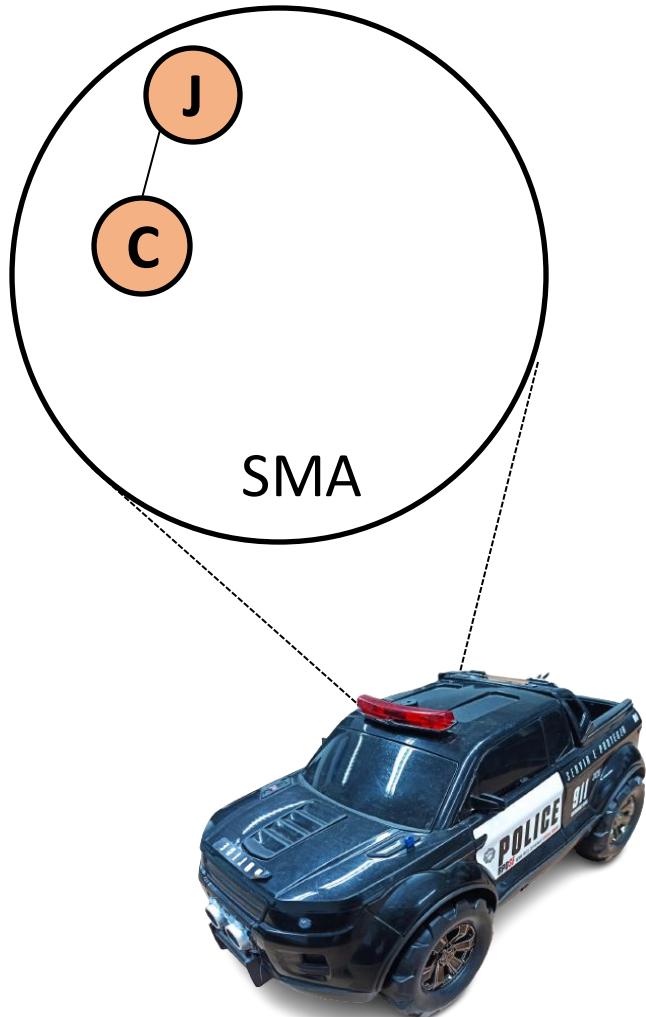
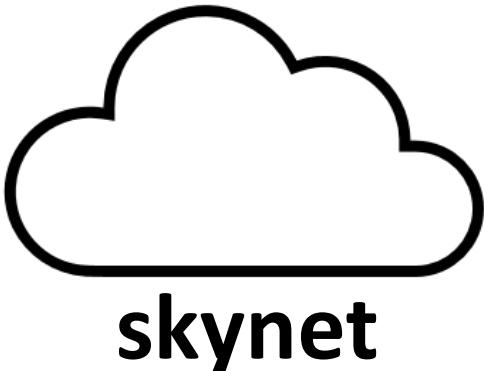
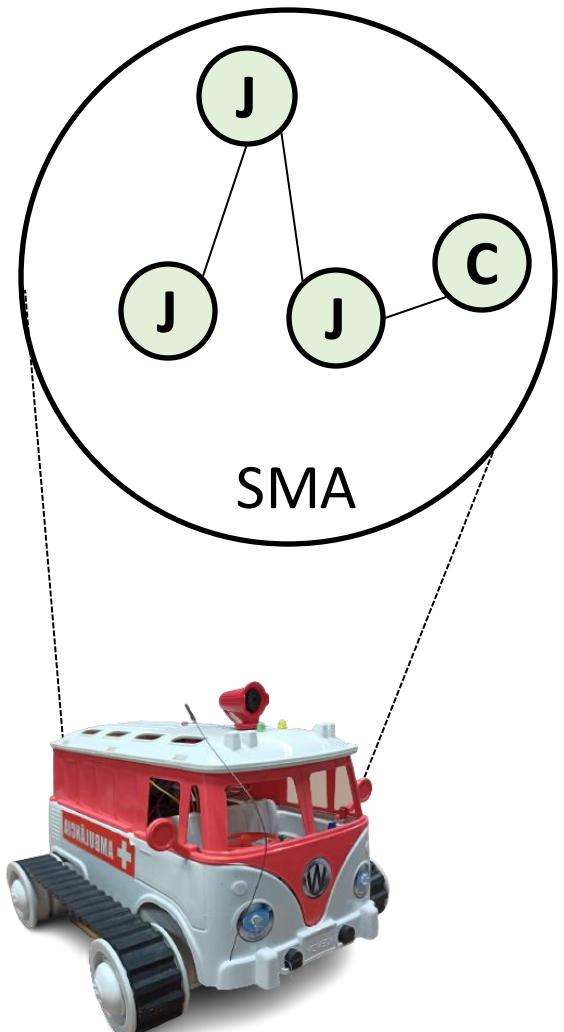
Bio-Inspired: Mutualismo



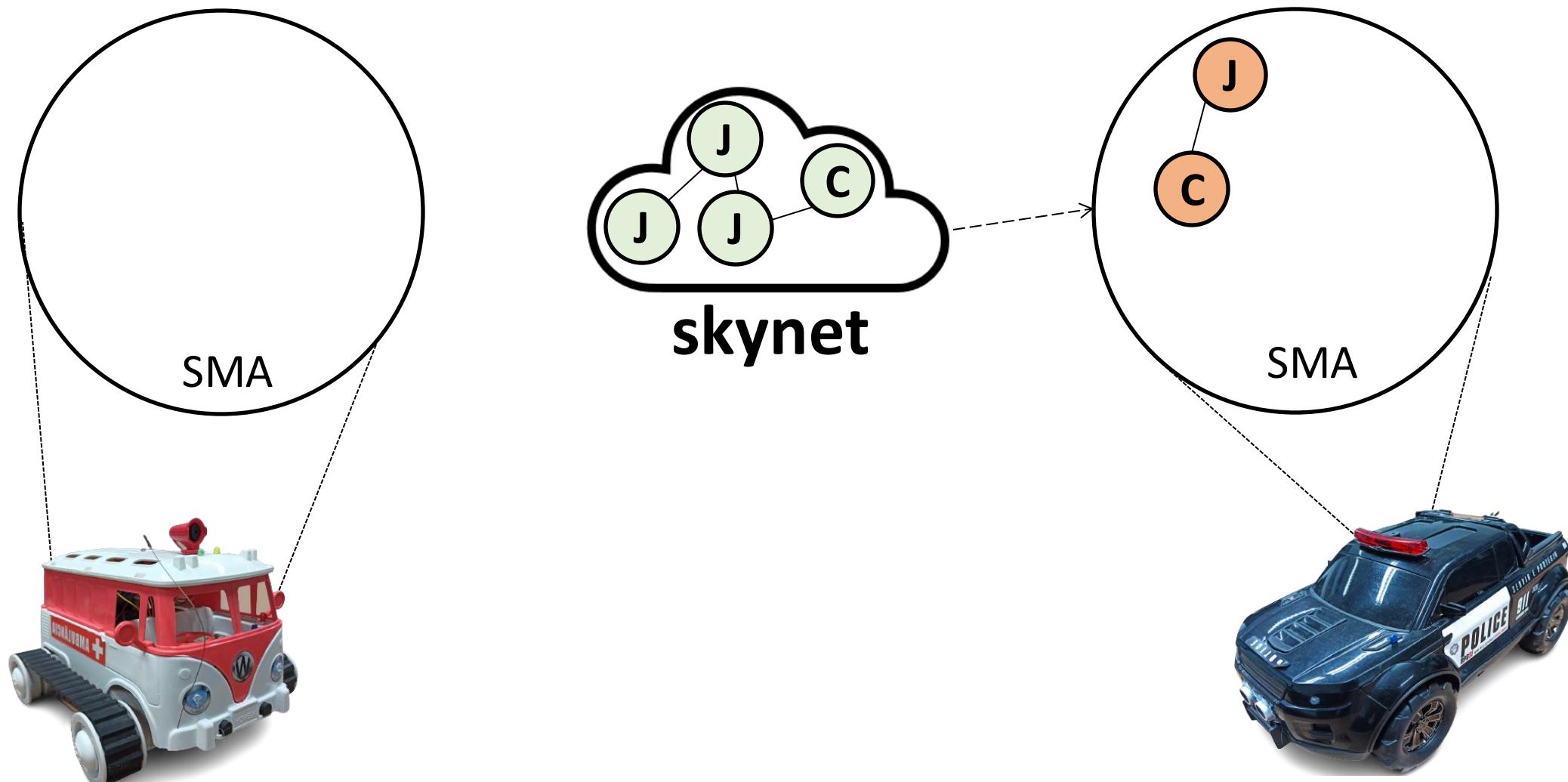
Bio-Inspired: Mutualismo



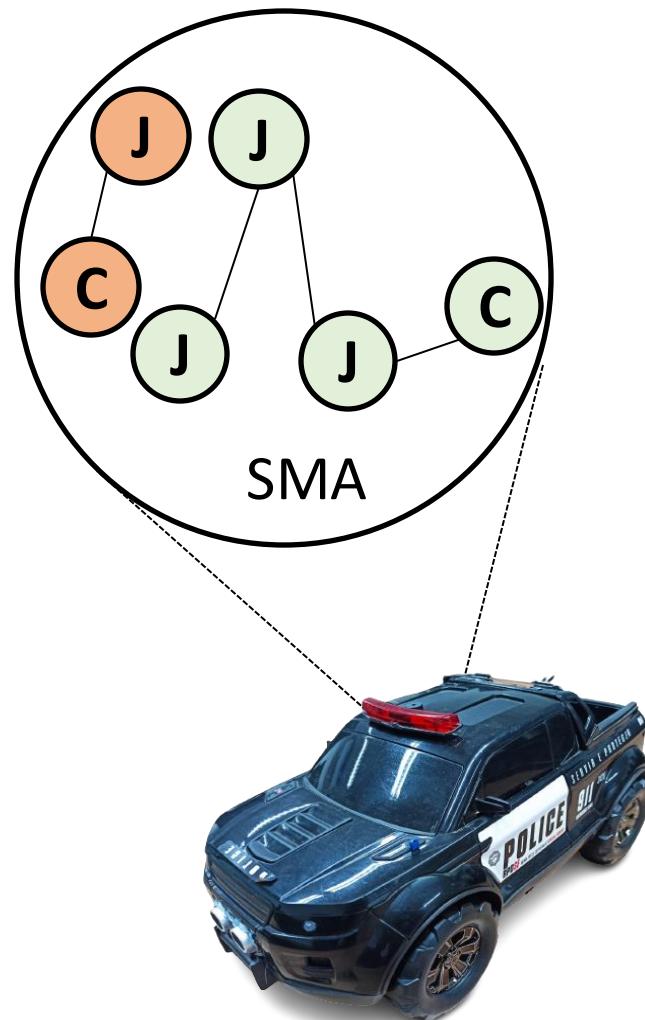
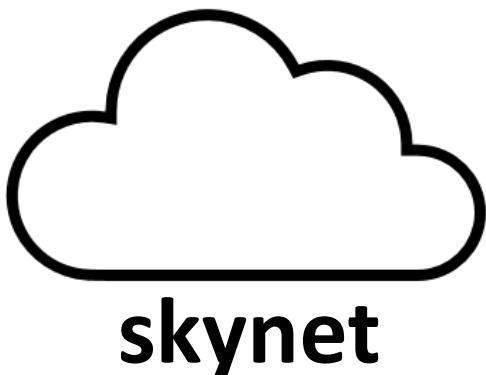
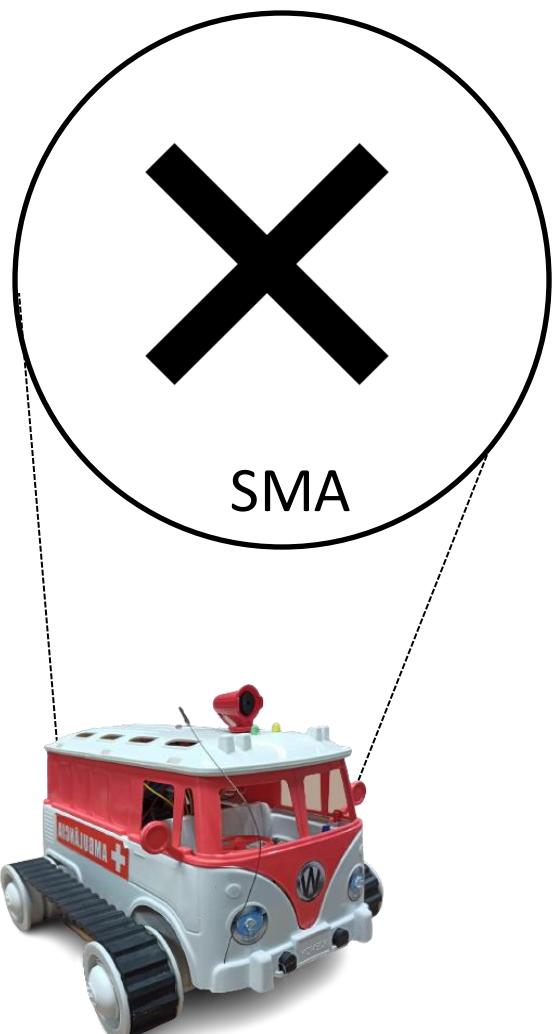
Bio-Inspired: Inquilinismo



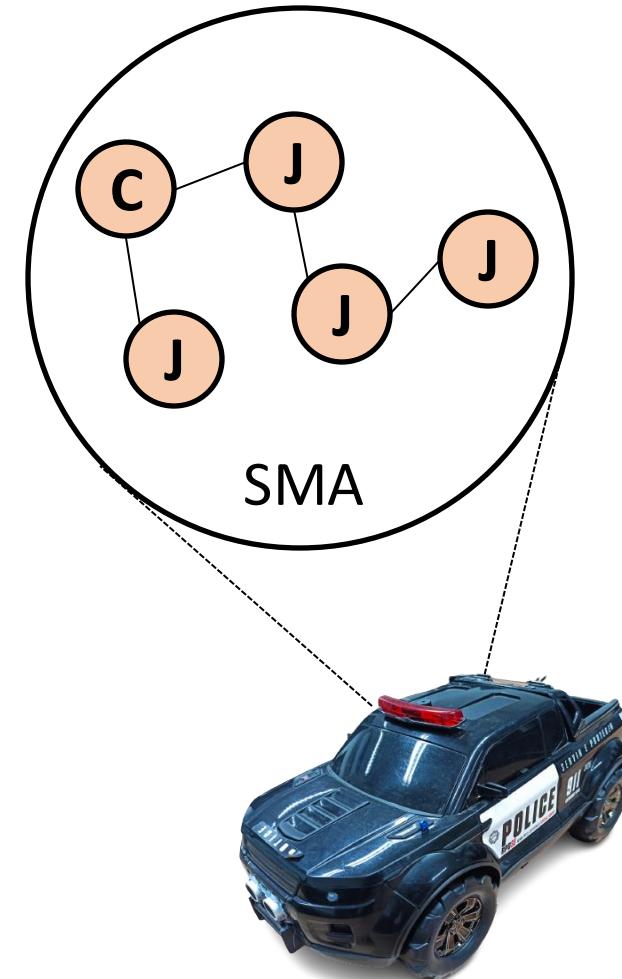
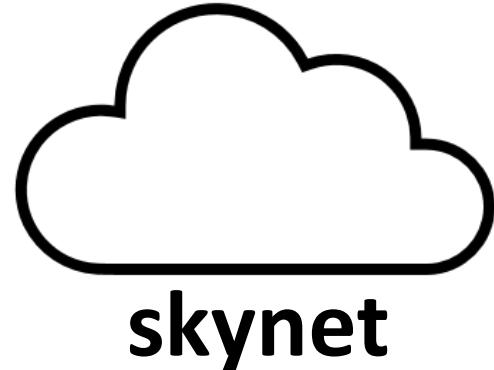
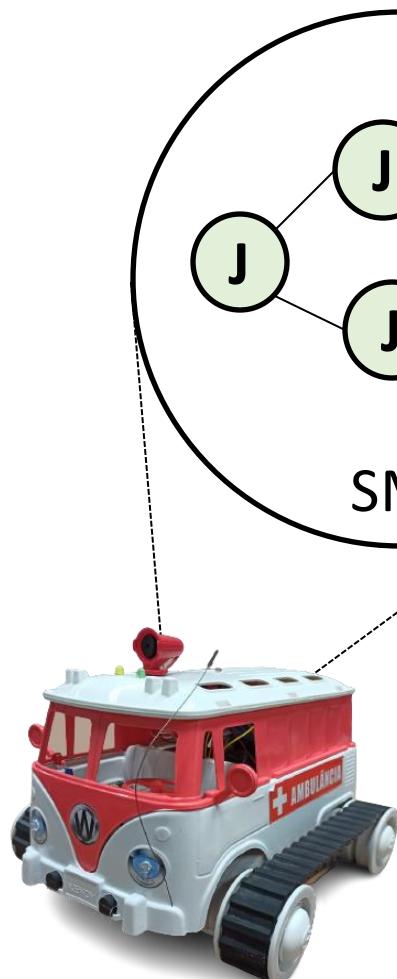
Bio-Inspired: Inquilinismo



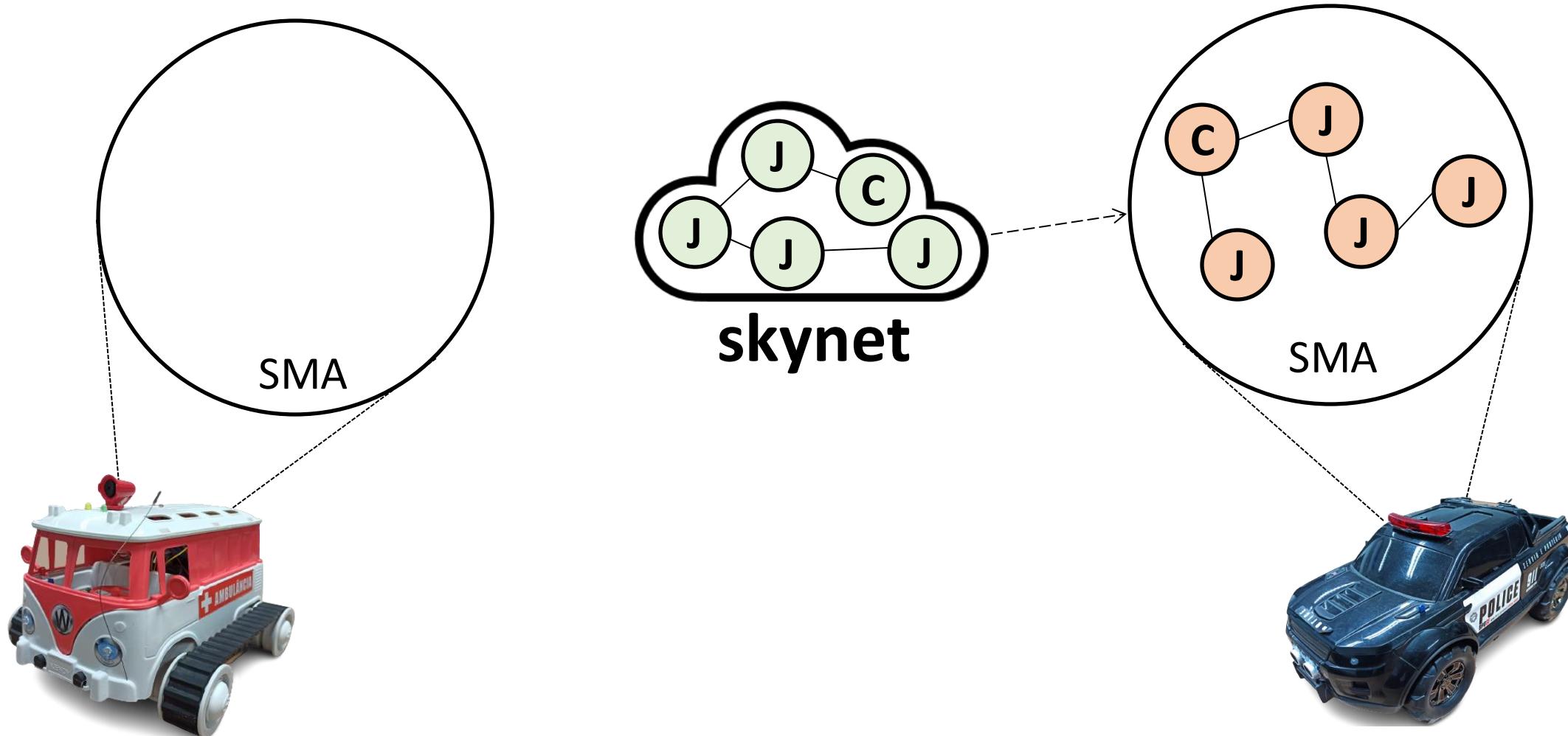
Bio-Inspired: Inquilinismo



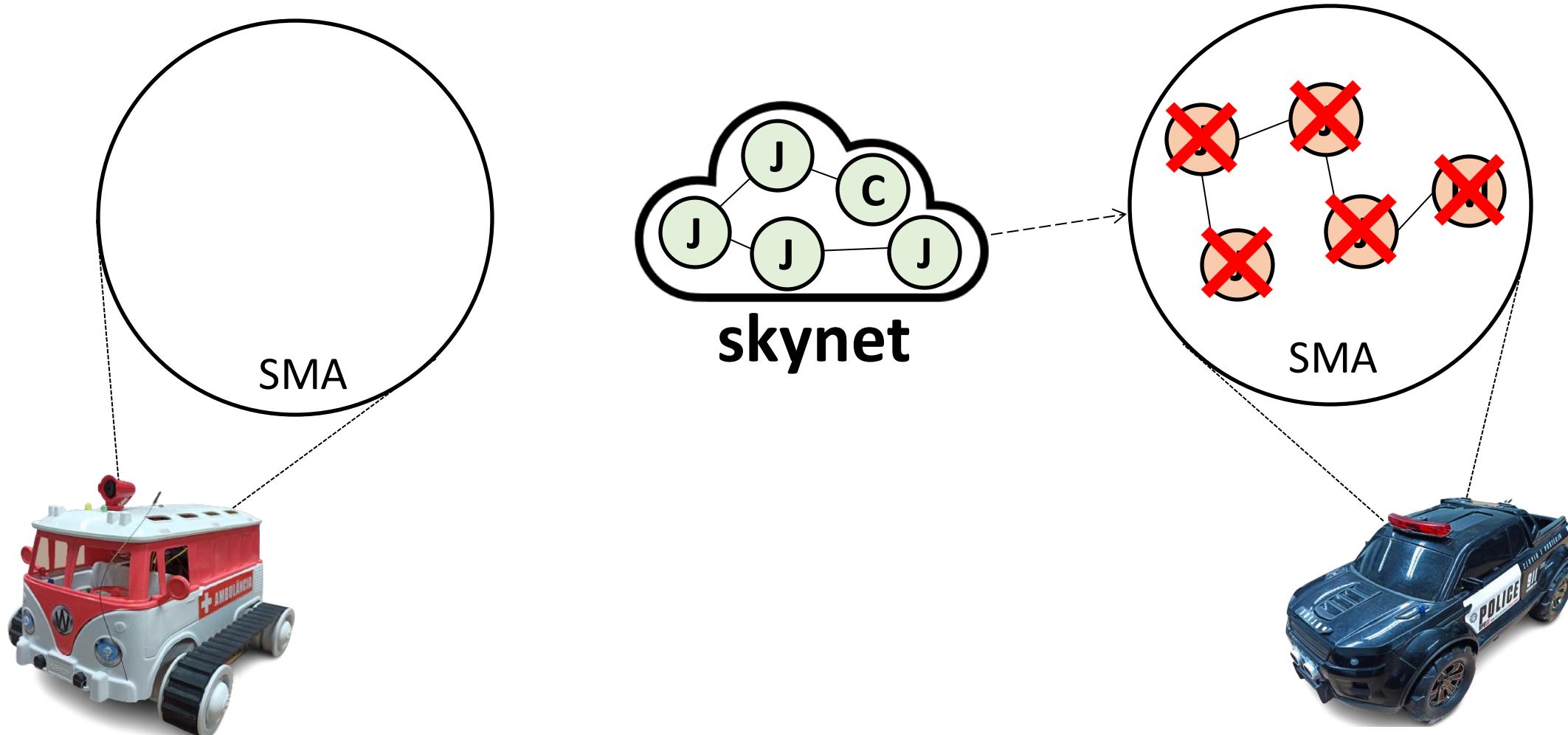
Bio-Inspired: Predatismo



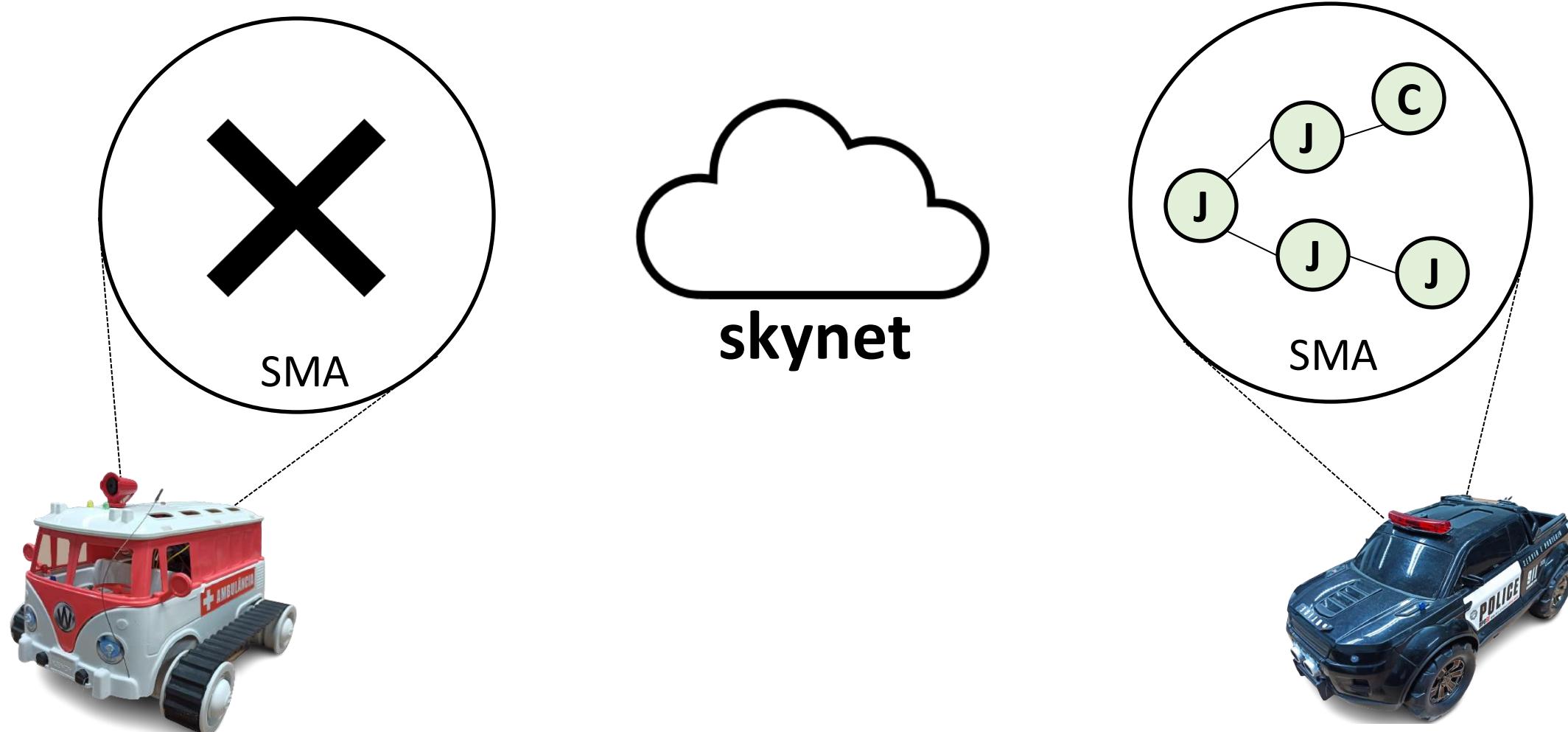
Bio-Inspired: Predatismo



Bio-Inspired: Predatismo



Bio-Inspired: Predatismo



Communicator: Ações Internas

- **Communicator Internal Actions:**

- .moveOut(agentUuid, predation|inquilinism)

move os agentes para um SMA.

- .moveOut(agentUuid, mutualism, agent)

envia um agente específico para outro SMA.

Exemplo: Bio-Inspired Protocol



Exemplo: Bio-Inspired Protocol



Embedded MAS
enterprise

Exemplo: Bio-Inspired Protocol

agent uhura



Embedded MAS
enterprise

Exemplo: Bio-Inspired Protocol

agent uhura



agent scott



Exemplo: Bio-Inspired Protocol

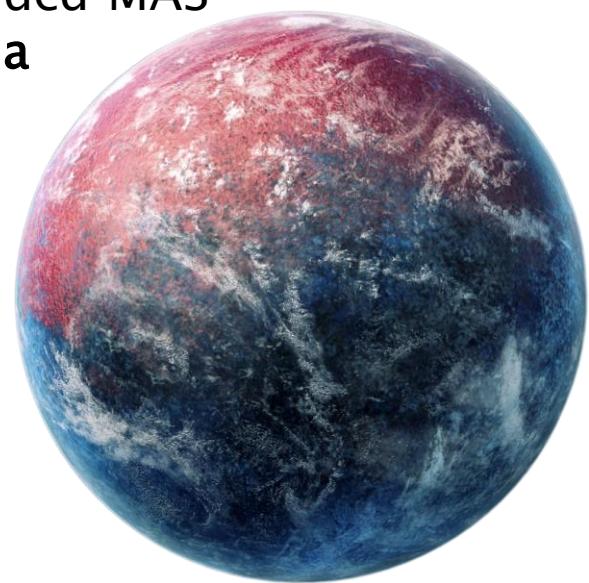
agent uhura



agent scott



Embedded MAS
andoria



Exemplo: Bio-Inspired Protocol

agent uhura



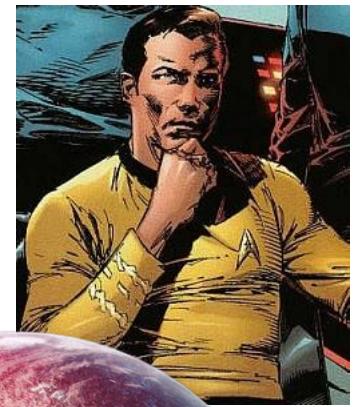
agent scott



Embedded MAS
andoria



agent kirk



Exemplo: Bio-Inspired Protocol

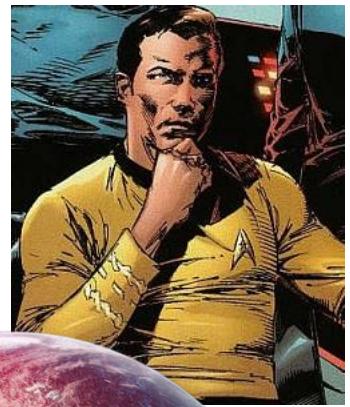
agent uhura



agent scott



agent kirk



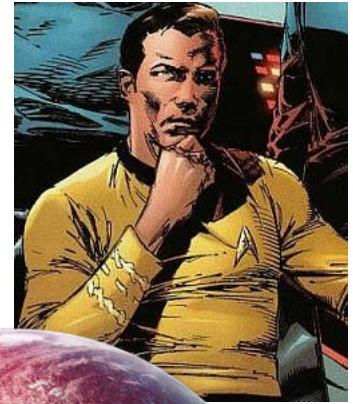
Embedded MAS andoria



agent spok



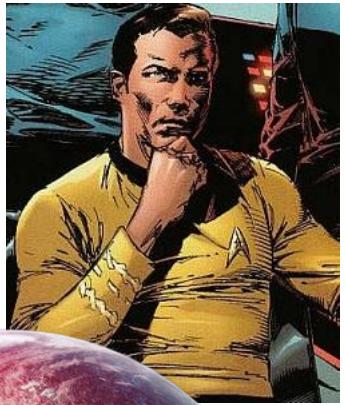
Exemplo: Bio-Inspired Protocol



Exemplo: Bio-Inspired Protocol



Computer,
Commander
Montgomery Scott,
Chief Engineering
Office!

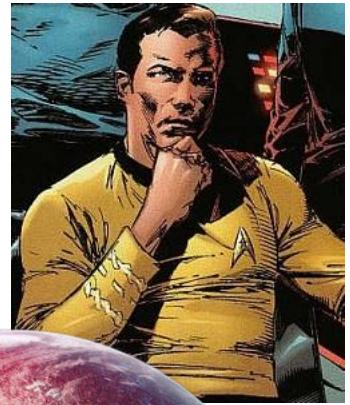


Exemplo: Bio-Inspired Protocol



Computer,
Commander
Montgomery Scott,
Chief Engineering
Office!

Kirk to Scotty...
Beam us up!



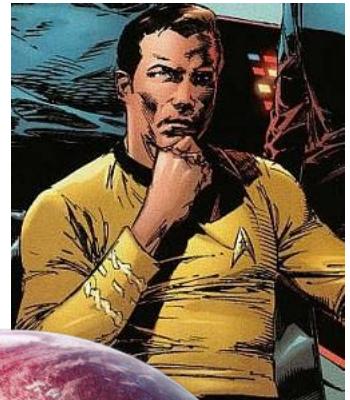
Exemplo: Bio-Inspired Protocol



Computer,
Commander
Montgomery Scott,
Chief Engineering
Office!

Transporter ready for
Kirk and Spok.

Kirk to Scotty...
Beam us up!



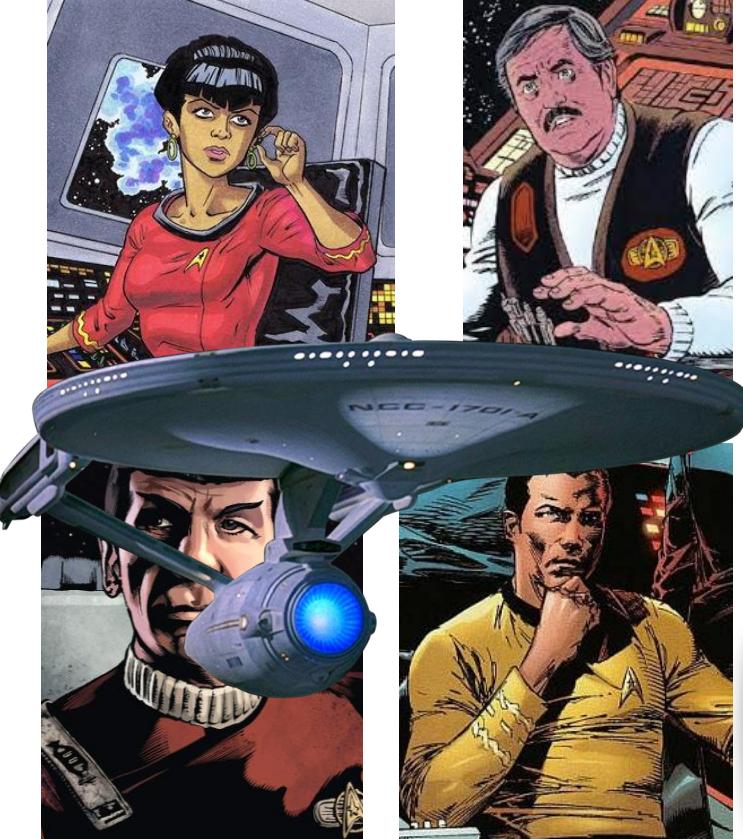
Exemplo: Bio-Inspired Protocol



Computer, Commander
James T. Kirk, Enterprise's
Captain! I'm aboard!



Exemplo: Bio-Inspired Protocol



Computer, Commander
James T. Kirk, Enterprise's
Captain! I'm aboard!



Exemplo: Bio-Inspired Protocol



```
◇ scott.asl
 2 //https://www.uuidgenerator.net/
 3 myUUID("07ba9e4a-d539-4a0e-8c14-4ac336476858").
 4
 5 //skyNetAddress(skynet.chon.group).           //public SkynetServer
 6 skyNetAddress("192.168.0.103").               //local IP of Your SkynetServer
 7 skyNetPort(3273).
 8
 9 /* Initial goals */
10 !start.
11
12 /* Plans */
13 +!start : true <-
14   ?myUUID(ID);
15   ?skyNetAddress(Server);
16   ?skyNetPort(Port);
17   .print("Computer, Commander Montgomery Scott, Chief Engineering Office");
18   .connectCN(Server,Port,ID).
19
20 +beam_us_up_scotty[source(X)] <-
21   .print("Transporter ready for ",X);
22   .sendOut(X,tell,energizing);
23   -beam_us_up_scotty[source(X)].
```

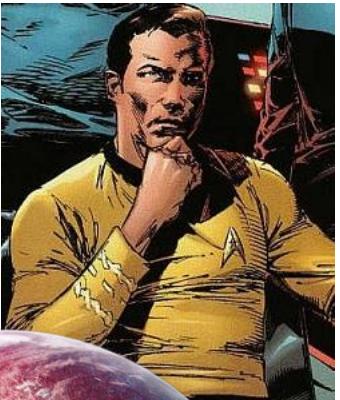
https://sourceforge.net/p/chonos/examples/ci/master/tree/04-openMultiAgent/openMASEExample01/ncc_1701/

Exemplo: Bio-Inspired Protocol

```
> kirk.asl ◇ scott.asl
1 /* Initial beliefs and rules */
2 abroad(yes). scott("07ba9e4a-d539-4a0e-8c14-4ac336476858").
3 skyNetAddress("192.168.0.103"). skyNetPort(3273).
4 myUUID("41ff1712-b2f0-416d-8232-fef834651e77").
5 /* Initial goals */
6 !conf.
7 /* Plans */
8 +!conf : abroad(yes) & (myUUID(ID)
9   & (skyNetAddress(Server) & skyNetPort(Port))) <-
10   .connectCN(Server,Port,ID); !start.
11
12 +!conf : abroad(no) <-
13   .print("Computer, Commander James T. Kirk, Enterprise's Captain").
14
15 +!start : (abroad(yes) & energizing[source(X)]) <-
16   .print("Transporter ready!"); -energizing[source(X)]; +ready; !start.
17
18 +!start : (abroad(yes) & (ready & scott(UUID))) <-
19   -ready; -abroad(yes); +abroad(no); .moveOut(UUID,inquiline).
20
21 +!start : (abroad(yes) & scott(UUID)) <-
22   .print("Kirk to Scotty... ");
23   .sendOut(UUID,tell,beam_us_up_scotty); .wait(2500); !start.
24
25 +!start : abroad(no) <-
26   .print("Computer, Commander James T. Kirk, Enterprise's Captain").
27
28 -!start <- !conf.
```

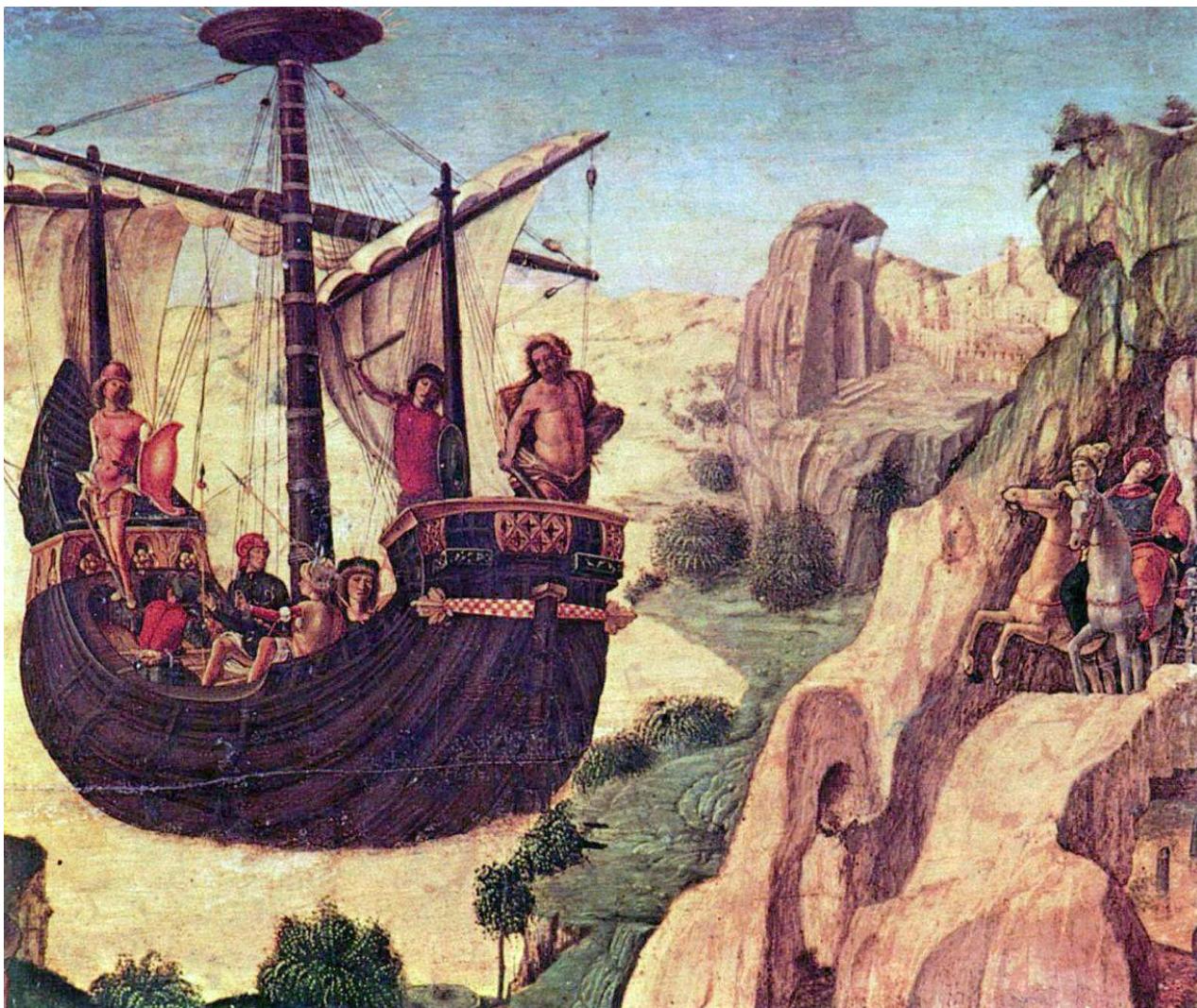
<https://sourceforge.net/p/chonos/examples/ci/master/tree/04-openMultiAgent/openMASEExample01/andoria/>

```
kirk.asl ◇ scott.asl ◇ spock.asl
1 /* Initial beliefs and rules */
2 abroad(yes)[source(kirk)].
3
4 /* Initial goals */
5 !start.
6
7 /* Plans */
8 +!start : abroad(no) <-
9   .wait(2000);
10  .print("Computer, Lieutenant Commander Spock, First Officer").
11
12 +!start : abroad(yes) <-
13   .wait(2000);
14   -abroad(yes);
15   .send(kirk,askOne,abroad(Status),Reply);
16   +Reply;
17   !start.
```



Embedded MultiAgent System

Argo for Jason



Argo foi o barco que Jasão (**Jason**) e os Argonautas navegaram na busca pelo **velocino de ouro** na mitologia grega.

The Argo
by Lorenzo Costa

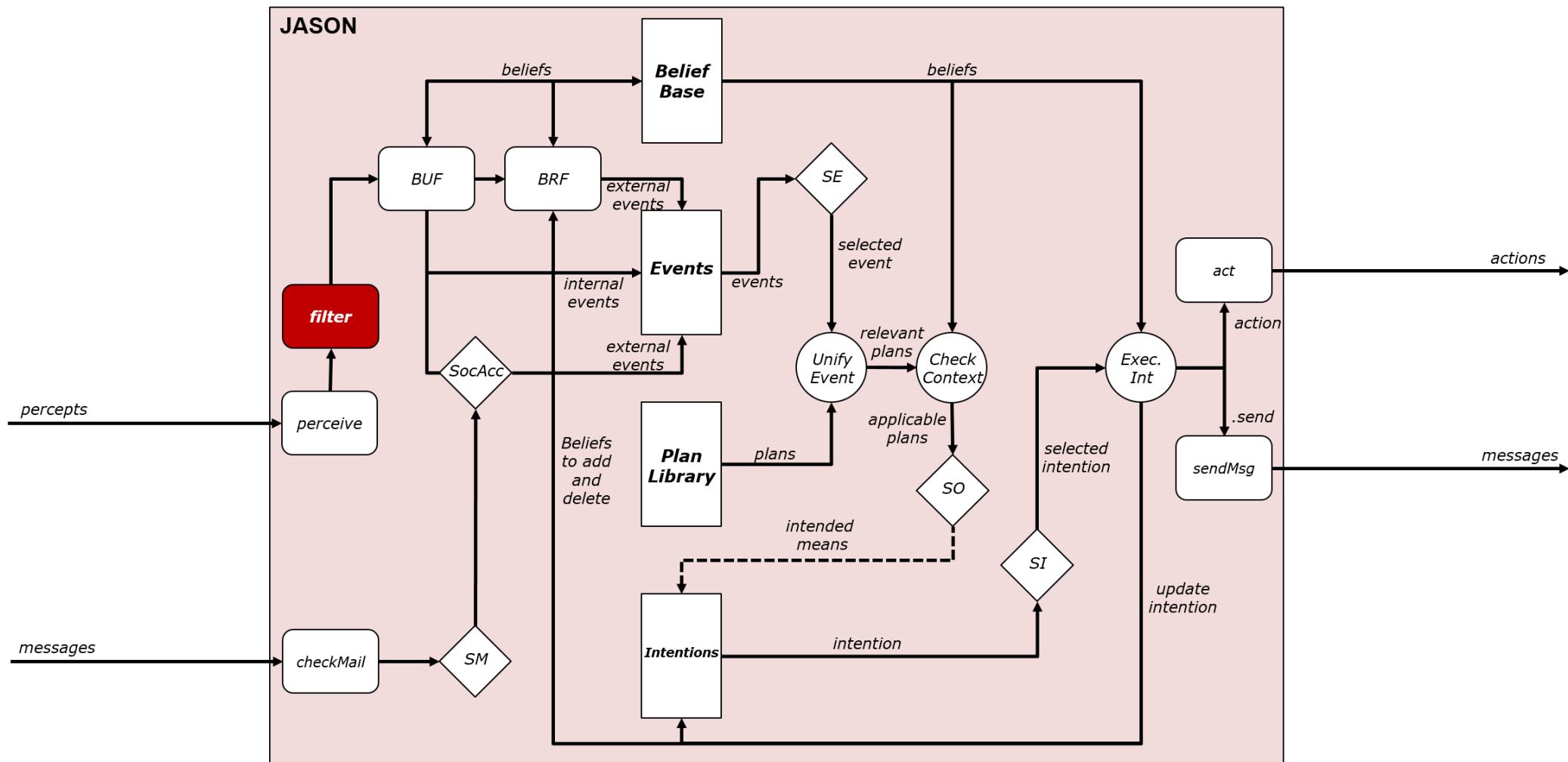
Argo for Jason

O **ARGO** é uma arquitetura customizada que emprega o **middleware Javino** [Lazarin e Pantoja, 2015], que provê uma **ponte** entre o agente inteligente e os sensores e atuadores do robô.

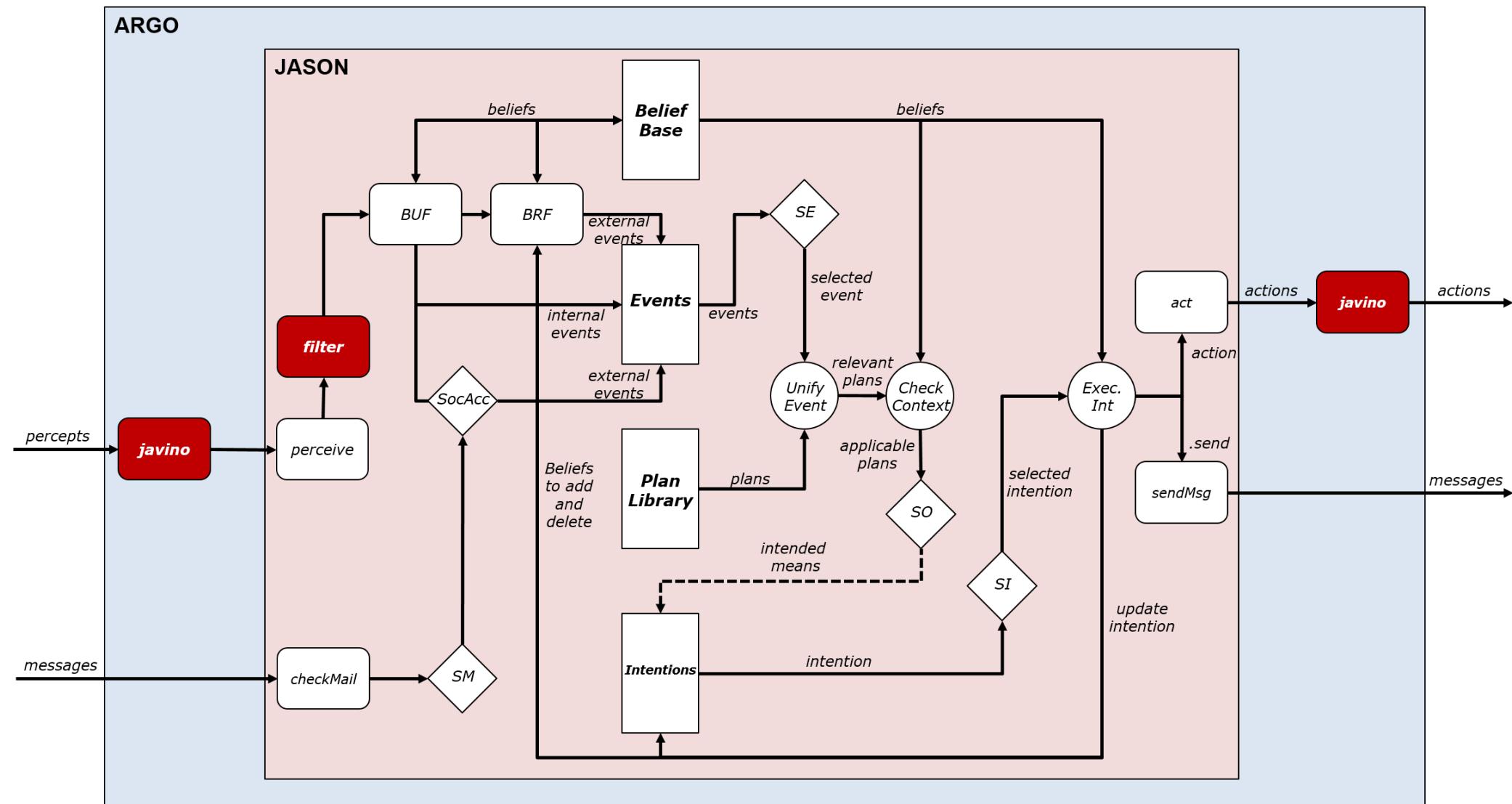
Além disso, o **ARGO** possui um mecanismo de **filtragem de percepções** [Stabile Jr e Sichman, 2015] em tempo de execução.

O **ARGO** tem como objetivo ser uma arquitetura prática para a **programação de agentes robóticos embarcados** usando agentes BDI em **Jason** e placas microcontroladas.

Argo for Jason



Argo for Jason



O **ARGO** permite:

- 1. Controlar diretamente os atuadores em tempo de execução;**
2. Receber percepções dos sensores automaticamente dentro de um período de tempo pré-definido;
3. Mudar os filtros de percepção em tempo de execução;
4. Alterar quais os dispositivos que estão sendo acessados em tempo de execução;
5. Se comunicar com outros agentes em Jason;
6. Decidir quando perceber ou não o mundo real em tempo de execução.

Argo for Jason

O **ARGO** permite:

1. Controlar diretamente os atuadores em tempo de execução;
2. **Receber percepções dos sensores automaticamente dentro de um período de tempo pré-definido;**
3. Mudar os filtros de percepção em tempo de execução;
4. Alterar quais os dispositivos que estão sendo acessados em tempo de execução;
5. Se comunicar com outros agentes em Jason;
6. Decidir quando perceber ou não o mundo real em tempo de execução.

Argo for Jason

O **ARGO** permite:

1. Controlar diretamente os atuadores em tempo de execução;
2. Receber percepções dos sensores automaticamente dentro de um período de tempo pré-definido;
- 3. Mudar os filtros de percepção em tempo de execução;**
4. Alterar quais os dispositivos que estão sendo acessados em tempo de execução;
5. Se comunicar com outros agentes em Jason;
6. Decidir quando perceber ou não o mundo real em tempo de execução.

Argo for Jason

O **ARGO** permite:

1. Controlar diretamente os atuadores em tempo de execução;
2. Receber percepções dos sensores automaticamente dentro de um período de tempo pré-definido;
3. Mudar os filtros de percepção em tempo de execução;
- 4. Alterar quais os dispositivos que estão sendo acessados em tempo de execução;**
5. Se comunicar com outros agentes em Jason;
6. Decidir quando perceber ou não o mundo real em tempo de execução.

Argo for Jason

O **ARGO** permite:

1. Controlar diretamente os atuadores em tempo de execução;
2. Receber percepções dos sensores automaticamente dentro de um período de tempo pré-definido;
3. Mudar os filtros de percepção em tempo de execução;
4. Alterar quais os dispositivos que estão sendo acessados em tempo de execução;
- 5. Se comunicar com outros agentes em Jason;**
6. Decidir quando perceber ou não o mundo real em tempo de execução.

Argo for Jason

O **ARGO** permite:

1. Controlar diretamente os atuadores em tempo de execução;
2. Receber percepções dos sensores automaticamente dentro de um período de tempo pré-definido;
3. Mudar os filtros de percepção em tempo de execução;
4. Alterar quais os dispositivos que estão sendo acessados em tempo de execução;
5. Se comunicar com outros agentes em Jason;
- 6. Decidir quando perceber ou não o mundo real em tempo de execução.**

- **ARGO Internal Actions:**

- `.limit(x)`: define um intervalo de tempo para perceber o ambiente
- `.port(y)`: define qual porta serial deve ser utilizada pelo agente
- `.percepts(open|block)`: decide quando perceber ou não o mundo real
- `.act(w)`: envia ao microcontrolador uma ação para ser executada por um efetuador
- `.change_filter(filterName)`: define um filtro de percepção para restringir percepções em tempo real

Exemplo: Argo



Exemplo: Argo



Embedded MAS
enterprise

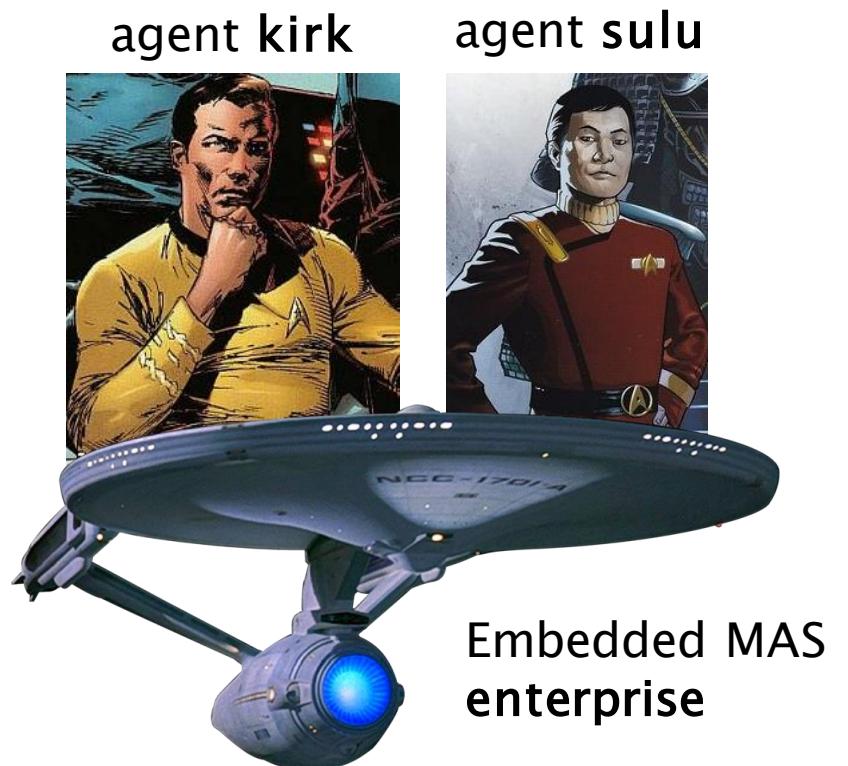
Exemplo: Argo

agent kirk

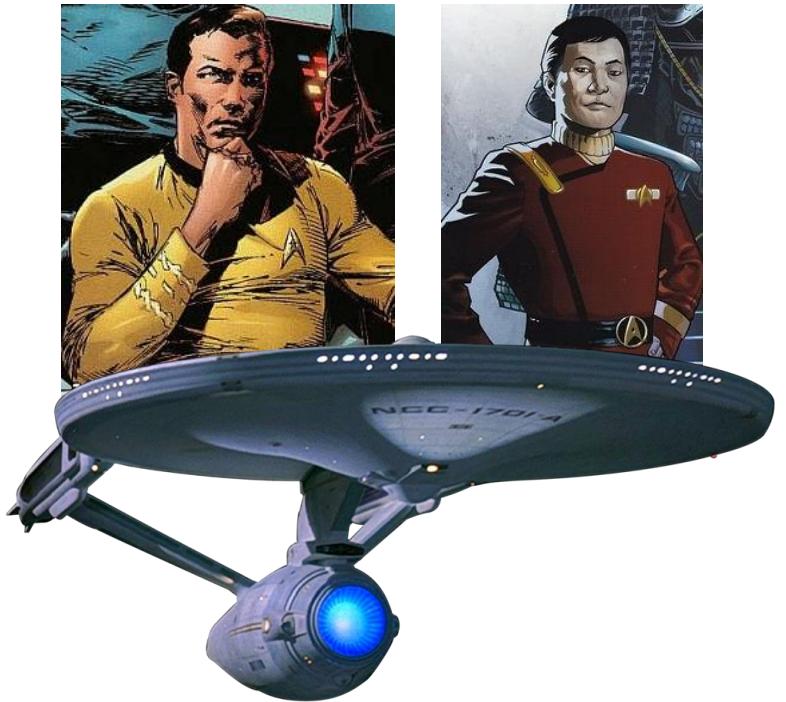


Embedded MAS
enterprise

Exemplo: Argo

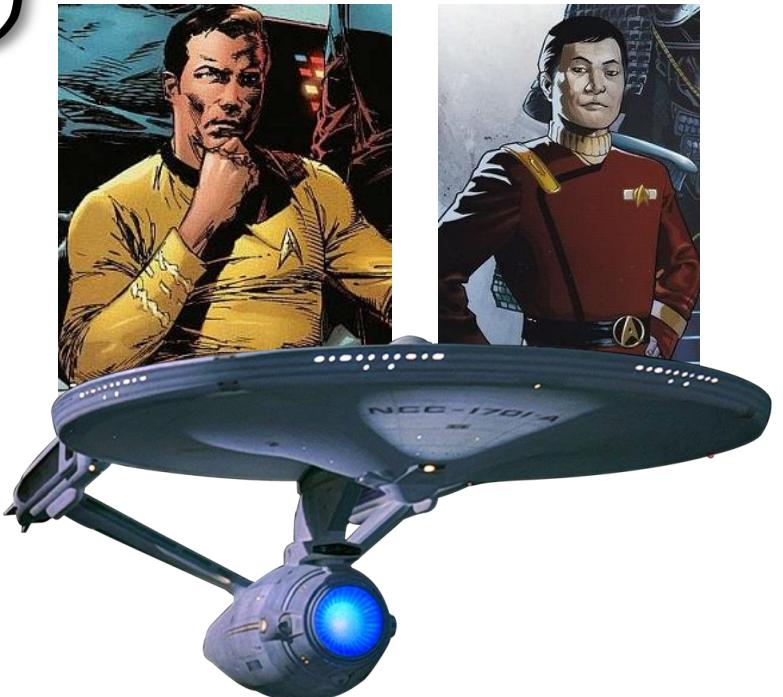


Exemplo: Argo



Exemplo: Argo

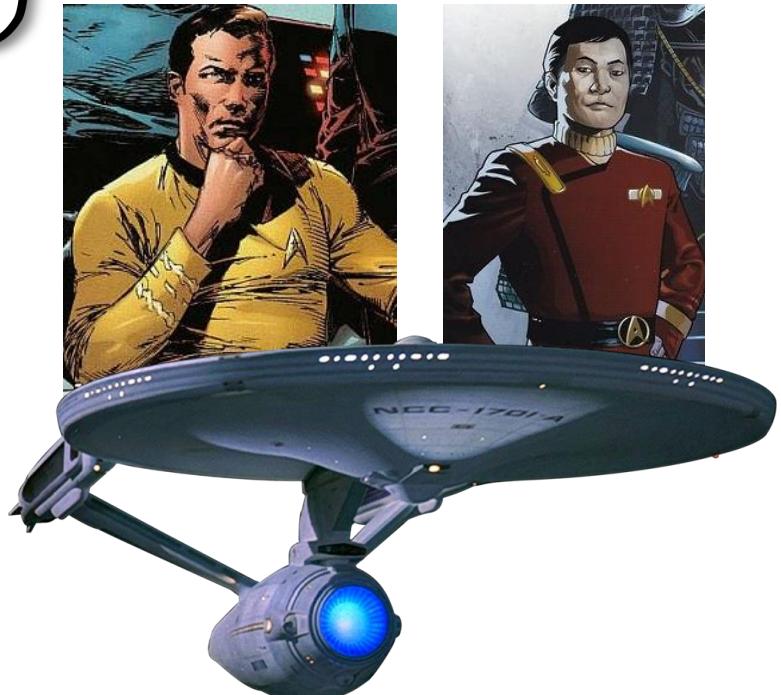
Sulu...
Fire Photon
Torpedo!



Exemplo: Argo

Sulu...
Fire Photon
Torpedo!

Target Acquired!

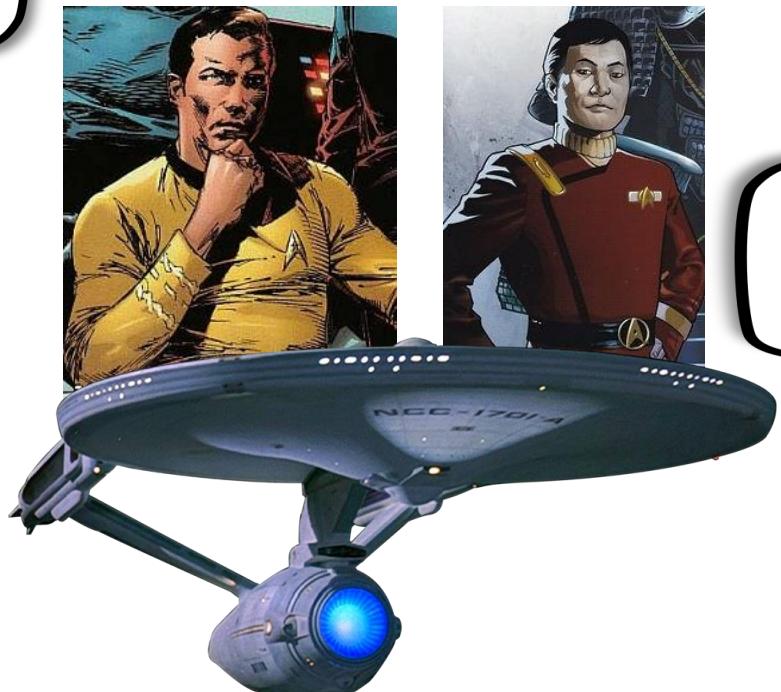


Exemplo: Argo

Sulu...
Fire Photon
Torpedo!

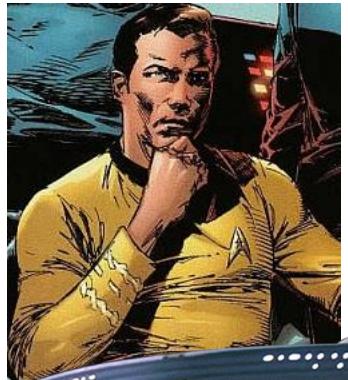
Target Acquired!

Photon torpedo fired.



Exemplo: Argo

.send(sulu,
achieve, fire);



Exemplo: Argo

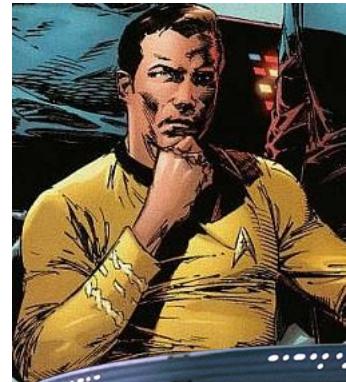
.send(sulu,
achieve, fire);

.port(ttyACM0);



Exemplo: Argo

.send(sulu,
achieve, fire);



.port(ttyACM0);

.act(buzzerOn);
.act(buzzerOff);

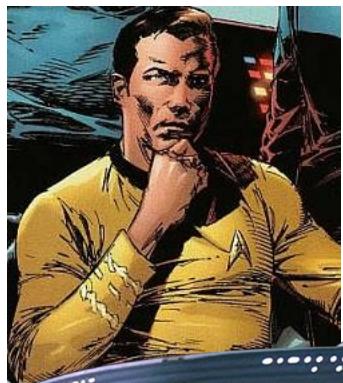


Argo for Jason: Limitações

Algumas características:

- Limite de 127 portas seriais
 - O limite da USB.
- Uma porta de cada vez
 - Sem competição de porta para evitar conflitos.
 - As portas podem ser mudadas em tempo de execução.
- Só agentes ARGO podem controlar dispositivos
 - Agentes em Jason não possuem as funcionalidades do ARGO.
 - Só pode existir uma instância para cada arquivo do agente
 - Se mais de um agente com o mesmo código for instanciado, conflitos acontecem.

Exemplo: Argo .act()



kirk

```
1  /* Initial beliefs and rules */
2
3  /* Initial goals */
4  !start.
5
6  /* Plans */
7
8  +!start <-
9      .print("This is Comm")
10     .wait(200);
11     +ship(klingow).
12
13
14  +ship(Ship): Ship == klingow <-
15      .print("Sulu, fire the photon torpedo.");
16      .send(sulu, achieve, fire).
17
```

sulu

```
1  /* Initial beliefs and rules */
2
3  /* Initial goals */
4  !start.
5
6  /* Plans */
7
8  +!start <-
9      .port(ttyACM0).
10
11  +!fire <-
12      .print("Target Acquired!");
13      .act(buzzerOn);
14      .wait(100);
15      .act(buzzerOff);
16      .print("Photon torpedo fired.").
```

<https://sourceforge.net/p/chonos/examples/ci/master/tree/05-embeddedMultiAgent/argoAgentExample01/>

Exemplo: Argo .percepts()



Agents

- kirk
- sulu

by [Jason](#)

```
sulu

1      /* Initial beliefs and rules */
2
3      /* Initial goals */
4      !start.
5
6      /* Plans */
7
8      +!start <-
9          .port(ttyACM0);
10         .percepts(open).
```

[latest state](#) [2](#) [1](#) [clear history](#)

Inspection of agent sulu

- Beliefs

breakLStatus(off)[source(percept)]·
buzzerStatus(off)[source(percept)]·
distance(63)[source(percept)]·
ledStatus(off)[source(percept)]·
lightStatus(off)[source(percept)]·
lineLeft(1008)[source(percept)]·
lineRight(1006)[source(percept)]·
luminosity(198)[source(percept)]·
motorStatus(stopped)[source(percept)]·

Exemplo: Argo .percepts()



```
argoAgentExample02.mas2j └ kirk.asl └ s
1 /* Initial beliefs and rules */
2
3 /* Initial goals */
4
5 /* Plans */
6
7 +danger[source(X)] <-
8   .print("Sulu, Red Alert!");
9   .send(sulu,achieve,redAlert);
10  -danger[source(X)].
```

```
argoAgentExample02.mas2j └ sulu.asl └ s
1
2
3 /* Initial goals */
4 !start.
5 /* Plans */
6 +!start <-
7   .port(ttyACMO);
8   .percepts(open).
9 +distance(D): D <= 20 & not danger<-
10  +danger;
11  .percepts(block);
12  .print("Kirk, Danger...");
13  .send(kirk,tell,danger).
14 +!redAlert <-
15   .act(lightOn);
16   .act(ledOn);
```

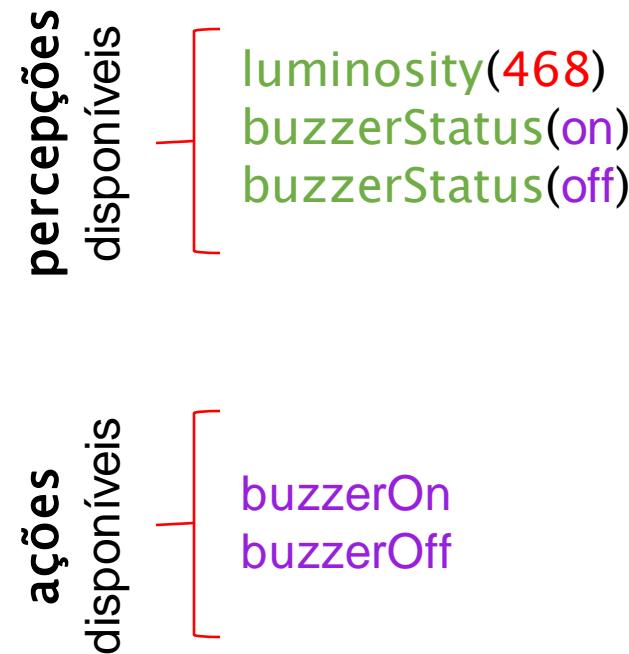
16

```
17   .act(buzzerOn);
18   .print("Enterprise, Red Alert!!!!");
19   .act(buzzerOff);
20   .act(lightOff);
21   .act(ledOff);
22   .percepts(open);
23   -danger.
```

<https://sourceforge.net/p/chonos/examples/ci/master/tree/05-embeddedMultiAgent/argoAgentExample02/>

Tarefa Argo

“ Criar um agente Argo que seja capaz de ativar o Buzzer quando estiver a perceber que está escuro e desativá-lo quando estiver claro. ”



Agradecimentos

OBRIGADO!

pantoja@cefet-rj.br
nilson.lazarin@cefet-rj.br

