

Introdução a Sistemas Multiagentes Embarcados

Carlos Eduardo Pantoja^{1,2}
Nilson Mori Lazarin^{1,2}
Vinicius Souza de Jesus²
Fabian César Brandão²

1. Centro Federal de Educação Tecnológica (CEFET/RJ) - 2. Universidade Federal Fluminense (UFF), Brasil



Introdução

- Agentes

Introdução

- Agentes
- Sistemas Multiagentes (SMA)

Introdução

- Agentes
- Sistemas Multiagentes (SMA)
- Um SMA Embarcado é um sistema cognitivo baseado em:

Introdução

- Agentes
- Sistemas Multiagentes (SMA)
- Um SMA Embarcado é um sistema cognitivo baseado em:
 - hardware, responsável pelas percepção e atuação no ambiente real,

Introdução

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

Introdução

- Agentes
- Sistemas Multiagentes (SMA)
- Um 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.

Introdução

- Agentes
- Sistemas Multiagentes (SMA)
- Um 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.
- Utiliza-se uma arquitetura de quatro camadas:

Introdução

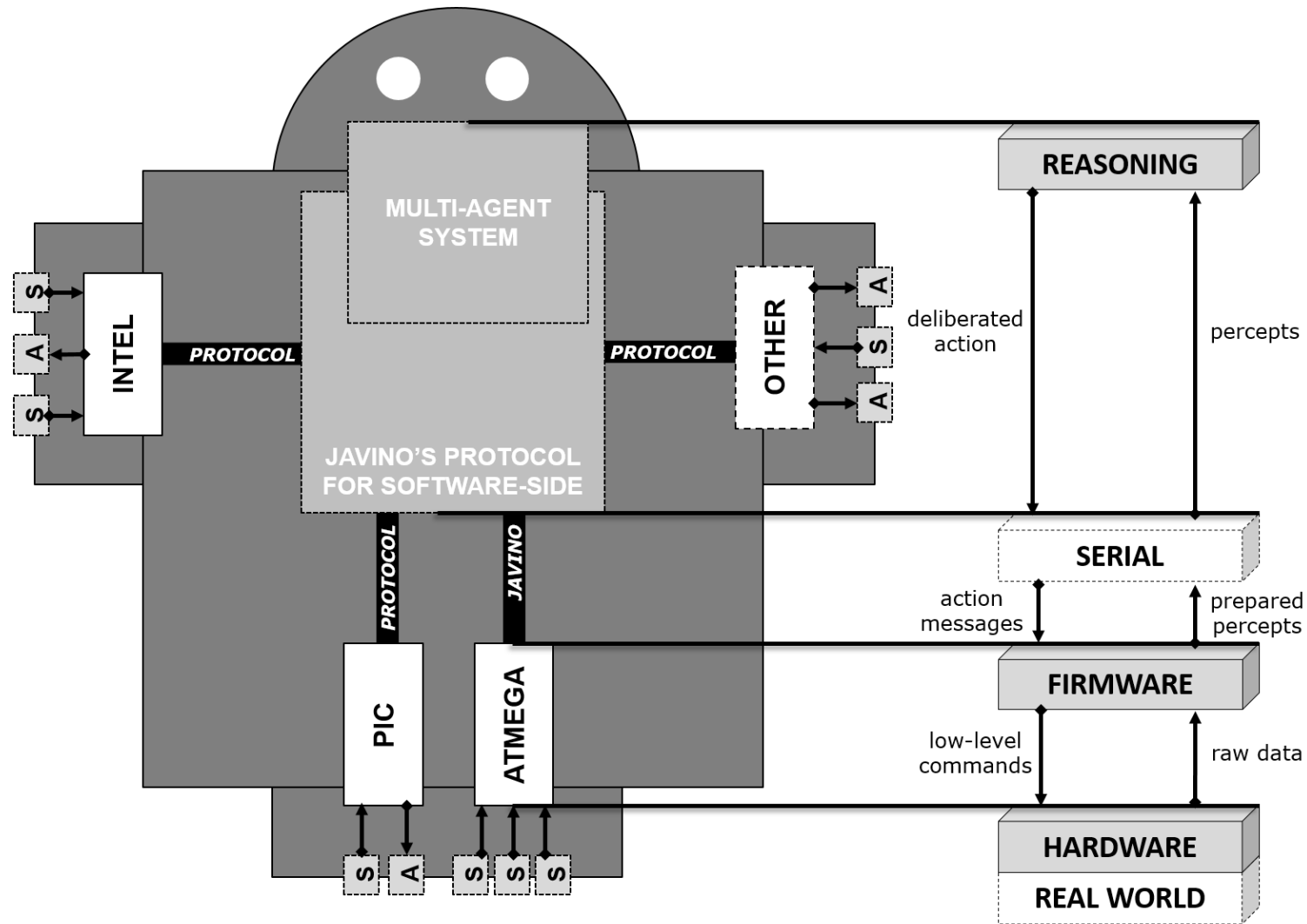
- Agentes
- Sistemas Multiagentes (SMA)
- Um SMA Embarcado é um sistema cognitivo baseado em:
 - hardware, responsável pelas percepção e atuação no ambiente real,
 - e software, responsável pelo raciocínio e controle do dispositivo.
- Não há controle remoto ou processamento externo.
- Utiliza-se uma arquitetura de quatro camadas:
 - **hardware:** conjunto de *recursos* que representam o ambiente do agente no mundo físico;

- Agentes
- Sistemas Multiagentes (SMA)
- Um 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.
- Utiliza-se uma arquitetura de quatro camadas:
 - **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;

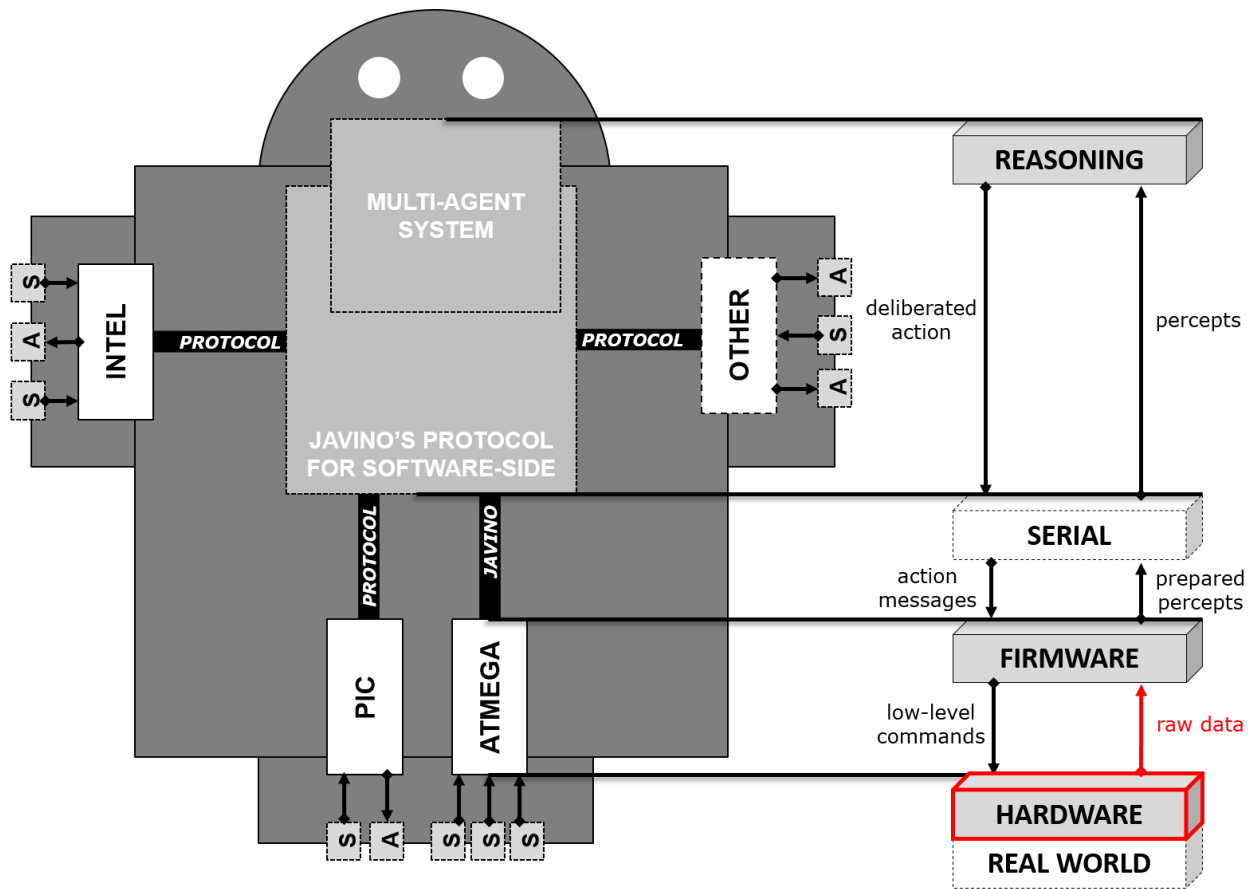
- Agentes
- Sistemas Multiagentes (SMA)
- Um SMA Embarcado é um sistema cognitivo baseado em:
 - hardware, responsável pelas percepção e atuação no ambiente real,
 - e software, responsável pelo raciocínio e controle do dispositivo.
- Não há controle remoto ou processamento externo.
- Utiliza-se uma arquitetura de quatro camadas:
 - **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;

- Agentes
- Sistemas Multiagentes (SMA)
- Um 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.
- Utiliza-se uma arquitetura de quatro camadas:
 - **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.

Arquitetura Física e Lógica

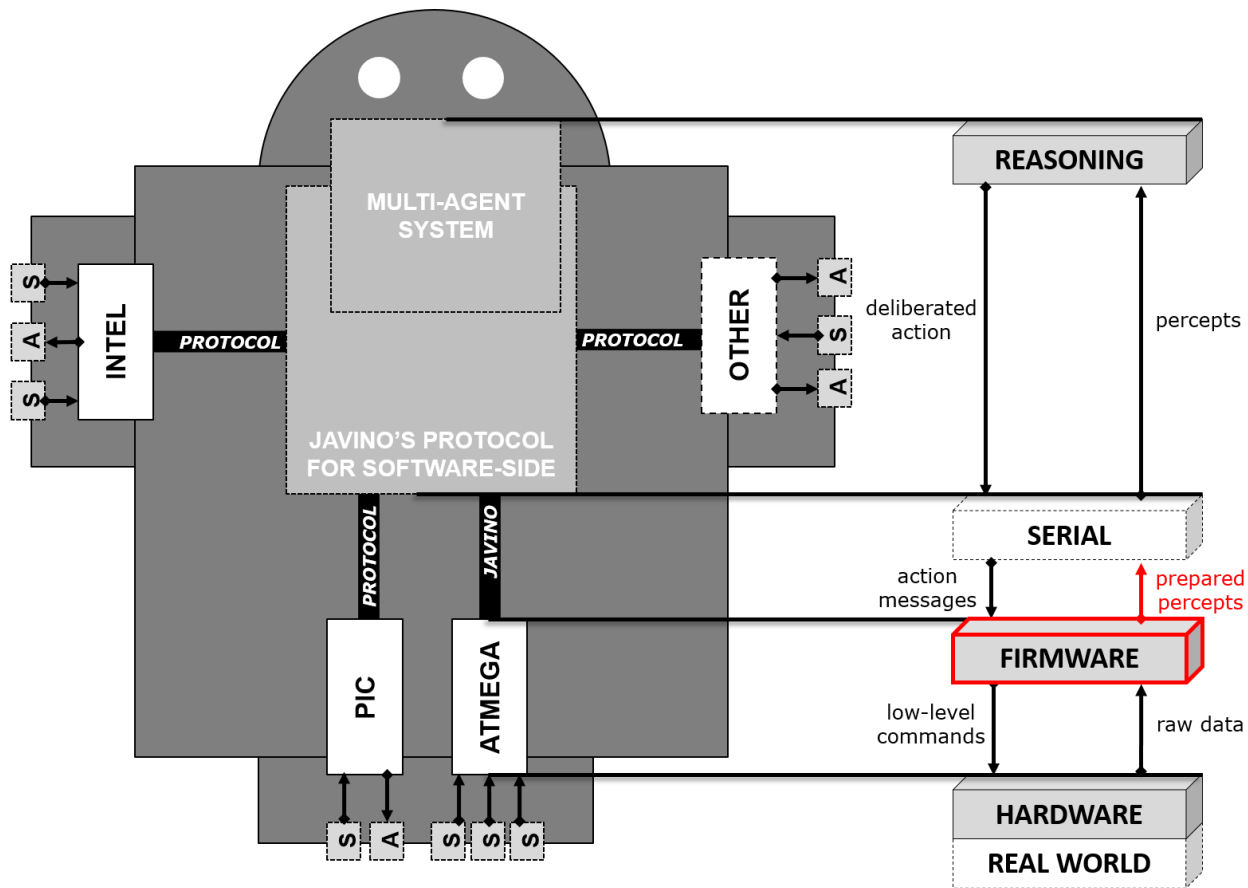


Arquitetura Física e Lógica: Fluxo da Mensagem



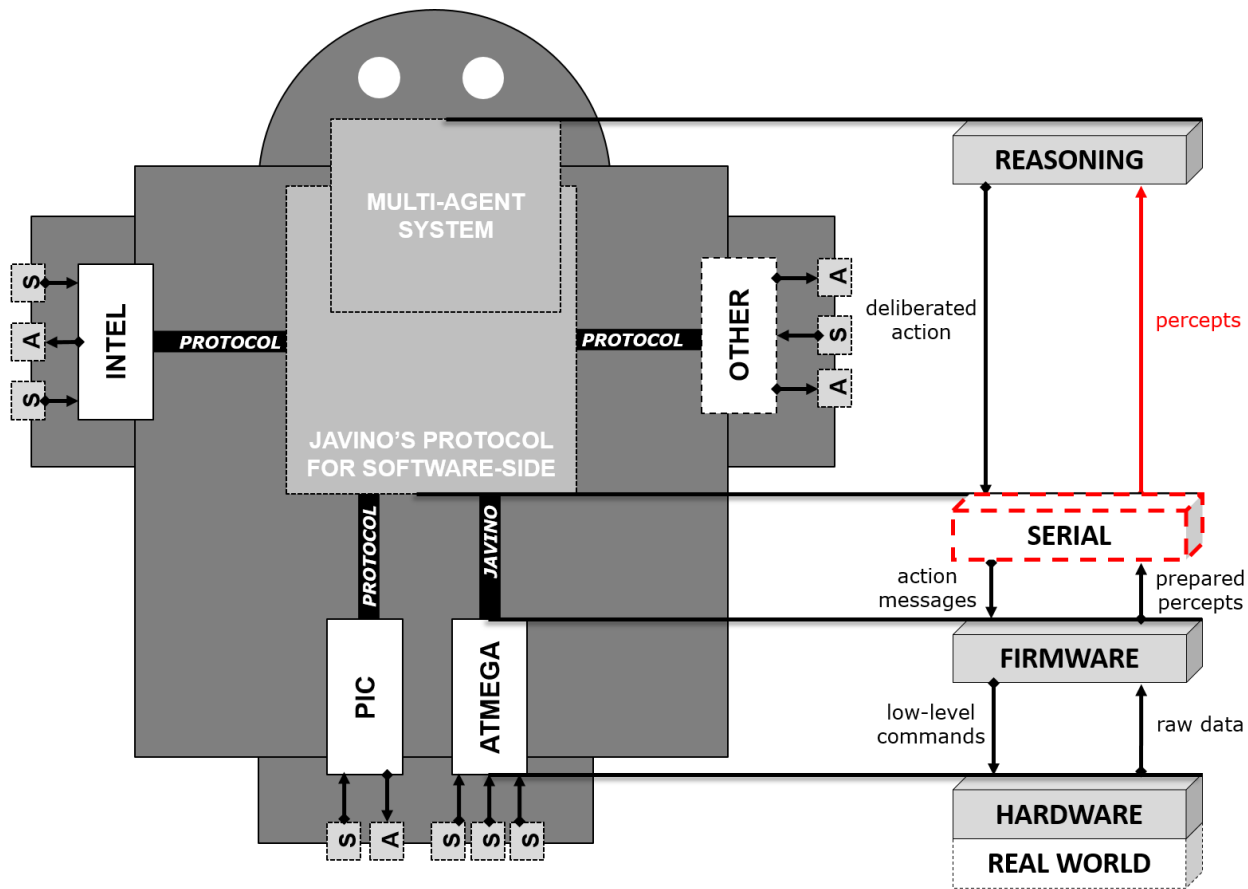
Sensores capturam dados brutos do mundo real e os enviam para um dos microcontroladores escolhido para o projeto.

Arquitetura Física e Lógica: Fluxo da Mensagem



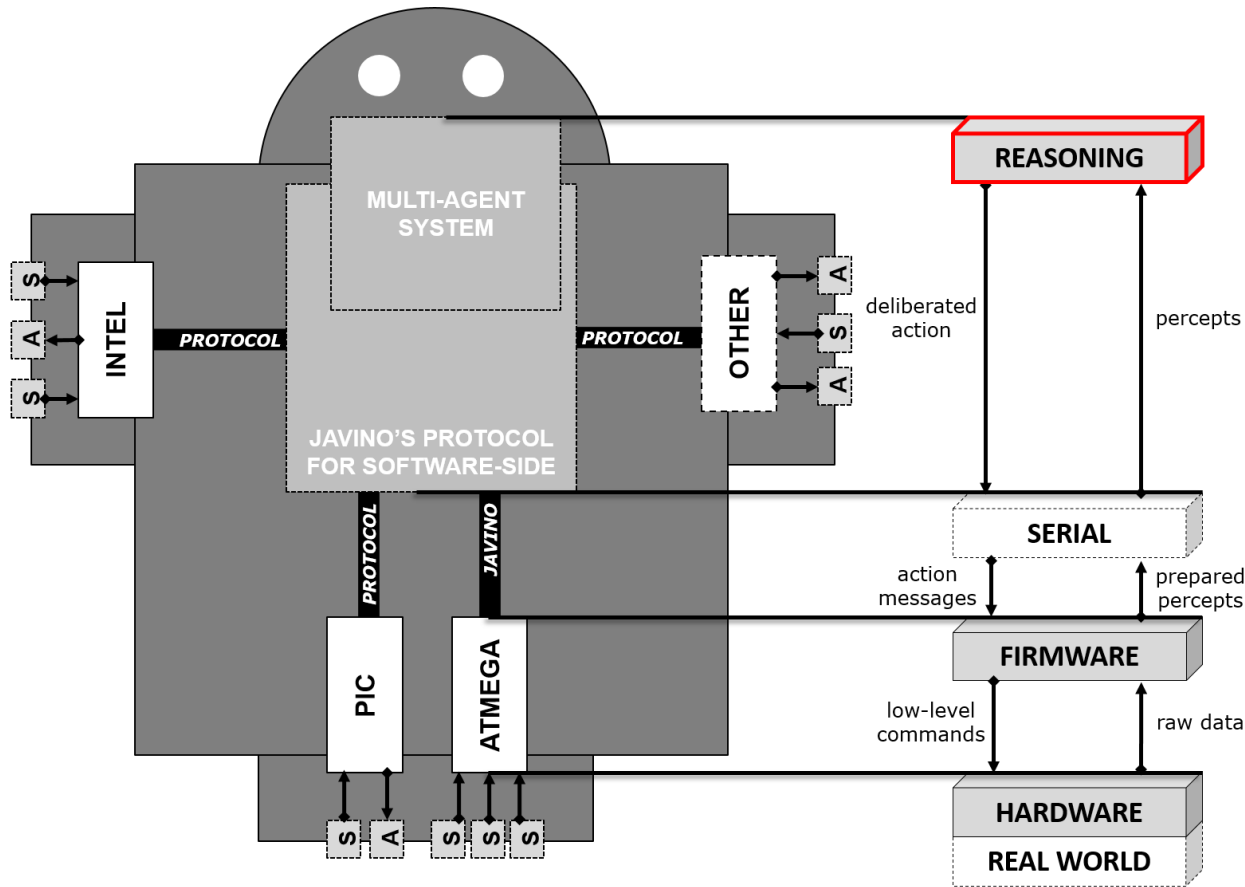
Na programação do microcontrolador, os dados brutos são transformados em percepções baseado na linguagem de programação orientada a agentes escolhida.

Arquitetura Física e Lógica: Fluxo da Mensagem



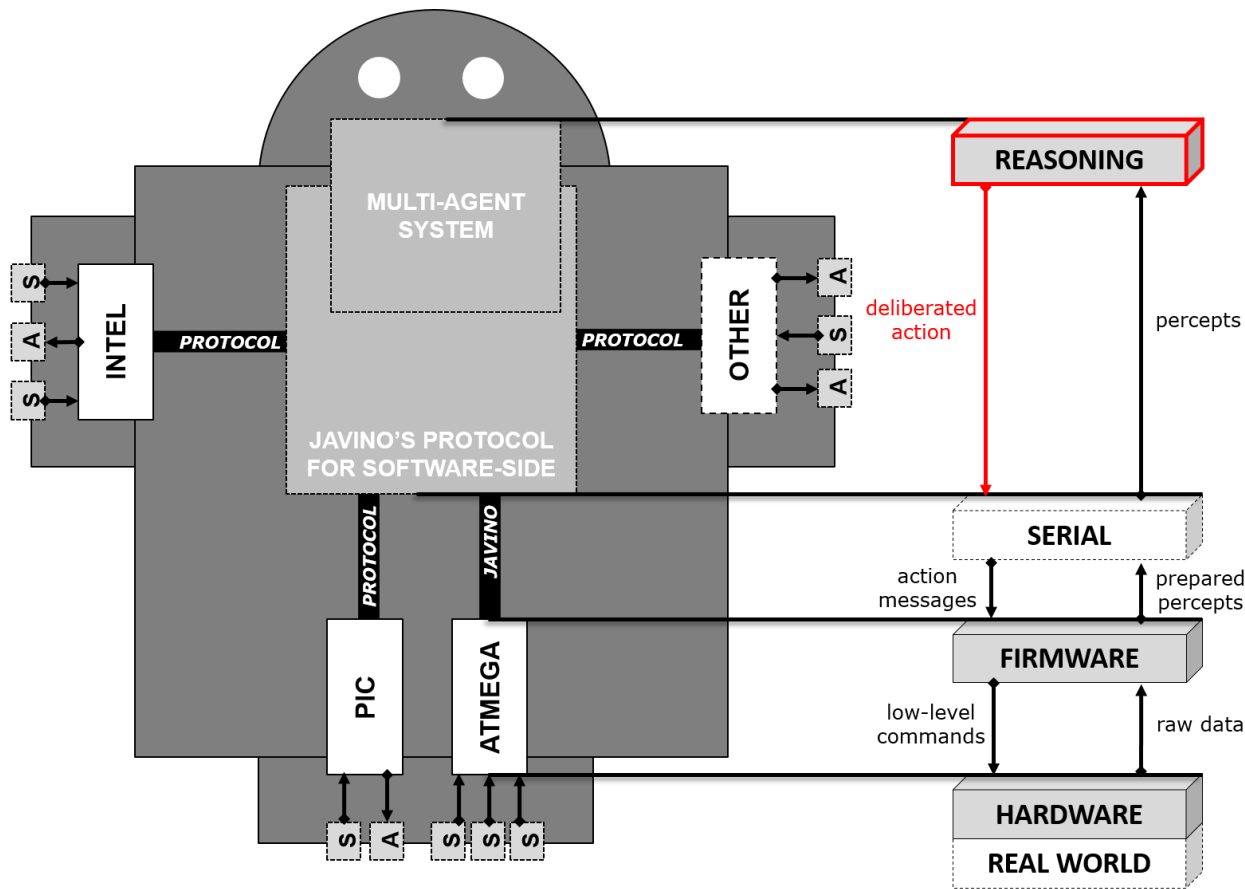
É responsável por enviar as percepções para a camada de raciocínio usando a comunicação serial.

Arquitetura Física e Lógica: Fluxo da Mensagem



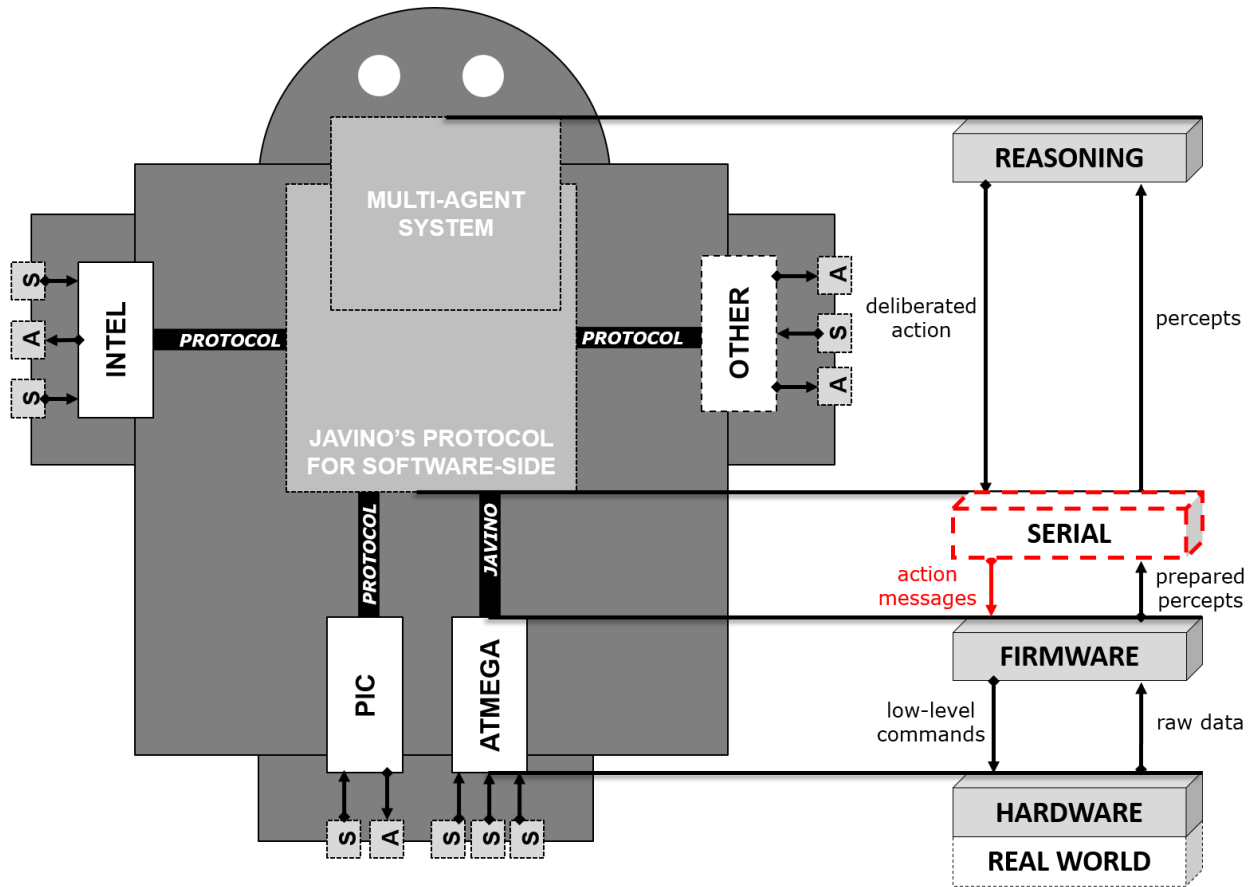
O agente é capaz de raciocinar com as percepções que vem diretamente do mundo real. Além disso, o SMA pode ser embarcado em placas computadorizadas como a Raspberry Pi ou computadores com interface USB.

Arquitetura Física e Lógica: Fluxo da Mensagem



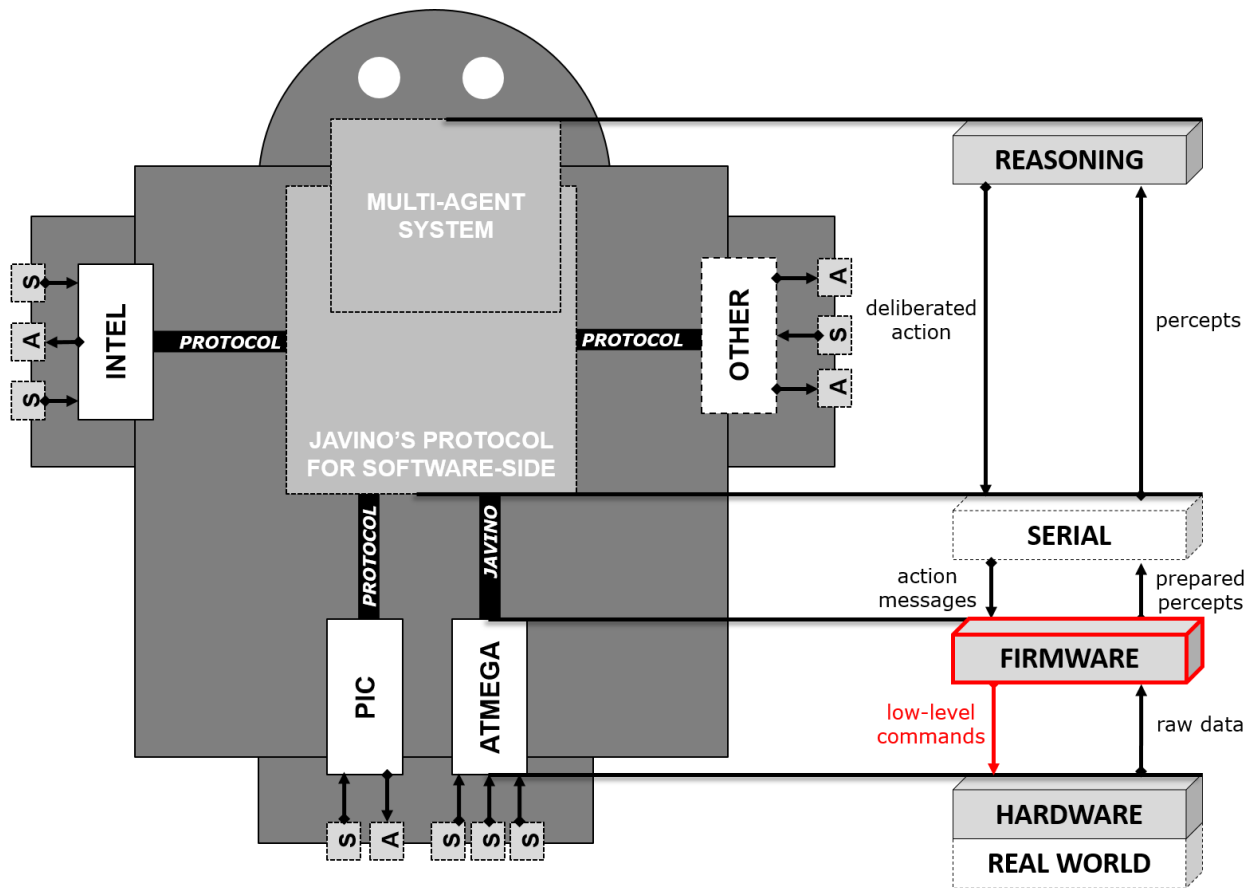
Então, o agente delibera e se alguma ação precisar ser executada. Neste caso, uma mensagem é enviada a camada serial.

Arquitetura Física e Lógica: Fluxo da Mensagem



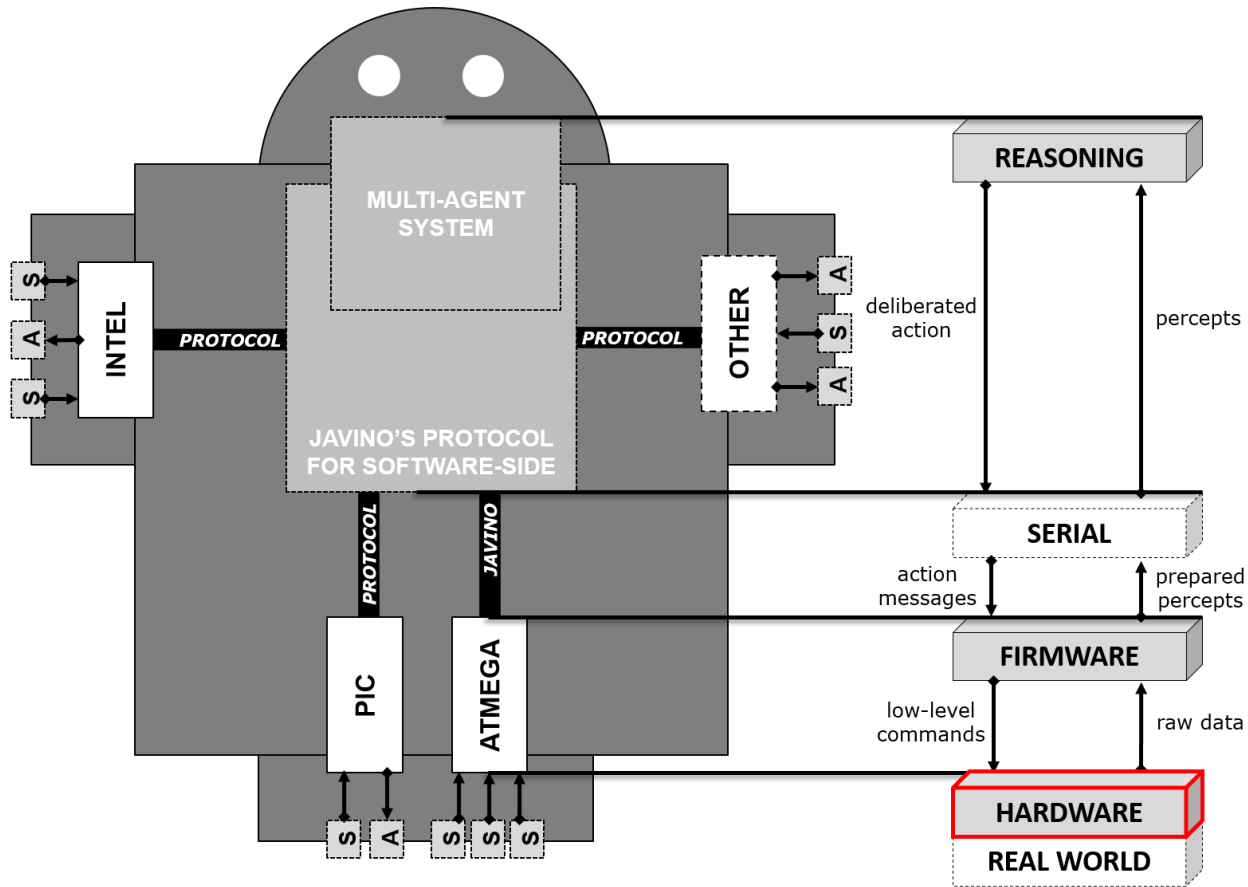
Redireciona as mensagens de ações para o microcontrolador que está conectado na porta USB identificado na mensagem.

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



O efetuator é ativado.

Desafios do Desenvolvimento de SMA Embarcados

- Equipe multidisciplinar

Desafios do Desenvolvimento de SMA Embarcados

- Equipe multidisciplinar
 - domínio de diferentes tecnologias (hardware, linguagens, frameworks, etc.)

Desafios do Desenvolvimento de SMA Embarcados

- Equipe multidisciplinar
 - domínio de diferentes tecnologias (hardware, linguagens, frameworks, etc.)
- Integração e configuração das diversas tecnologias

Desafios do Desenvolvimento de SMA Embarcados

- Equipe multidisciplinar
 - domínio de diferentes tecnologias (hardware, linguagens, frameworks, etc.)
- Integração e configuração das diversas tecnologias
- Heterogeneidade de componentes

Desafios do Desenvolvimento de SMA Embarcados

- Equipe multidisciplinar
 - domínio de diferentes tecnologias (hardware, linguagens, frameworks, etc.)
- Integração e configuração das diversas tecnologias
- Heterogeneidade de componentes
- Múltiplos ambientes de desenvolvimento:

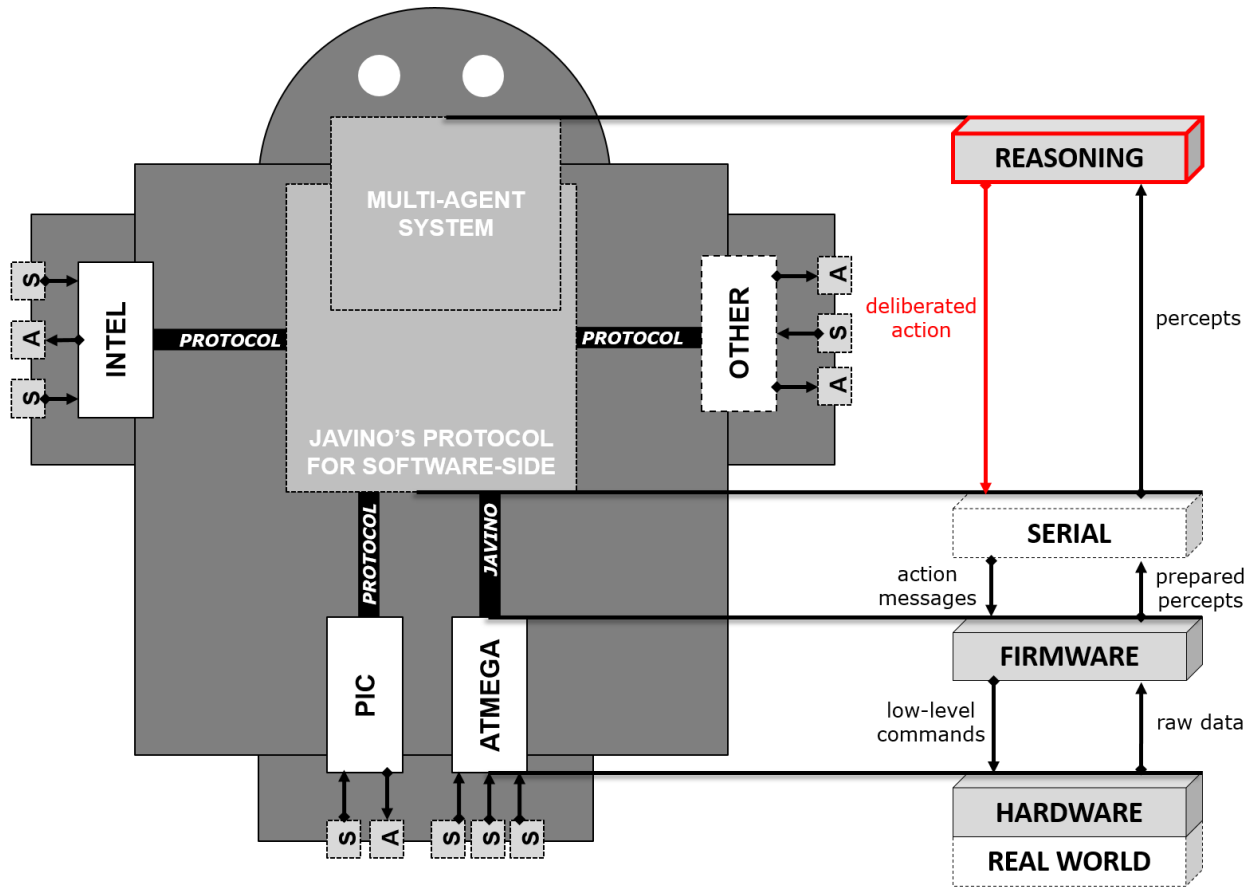
Desafios do Desenvolvimento de SMA Embarcados

- Equipe multidisciplinar
 - domínio de diferentes tecnologias (hardware, linguagens, frameworks, etc.)
- Integração e configuração das diversas tecnologias
- Heterogeneidade de componentes
- Múltiplos ambientes de desenvolvimento:
 - Arduino IDE

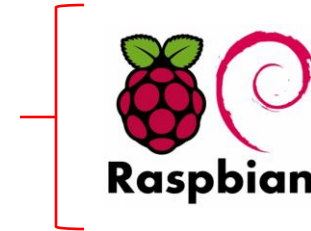
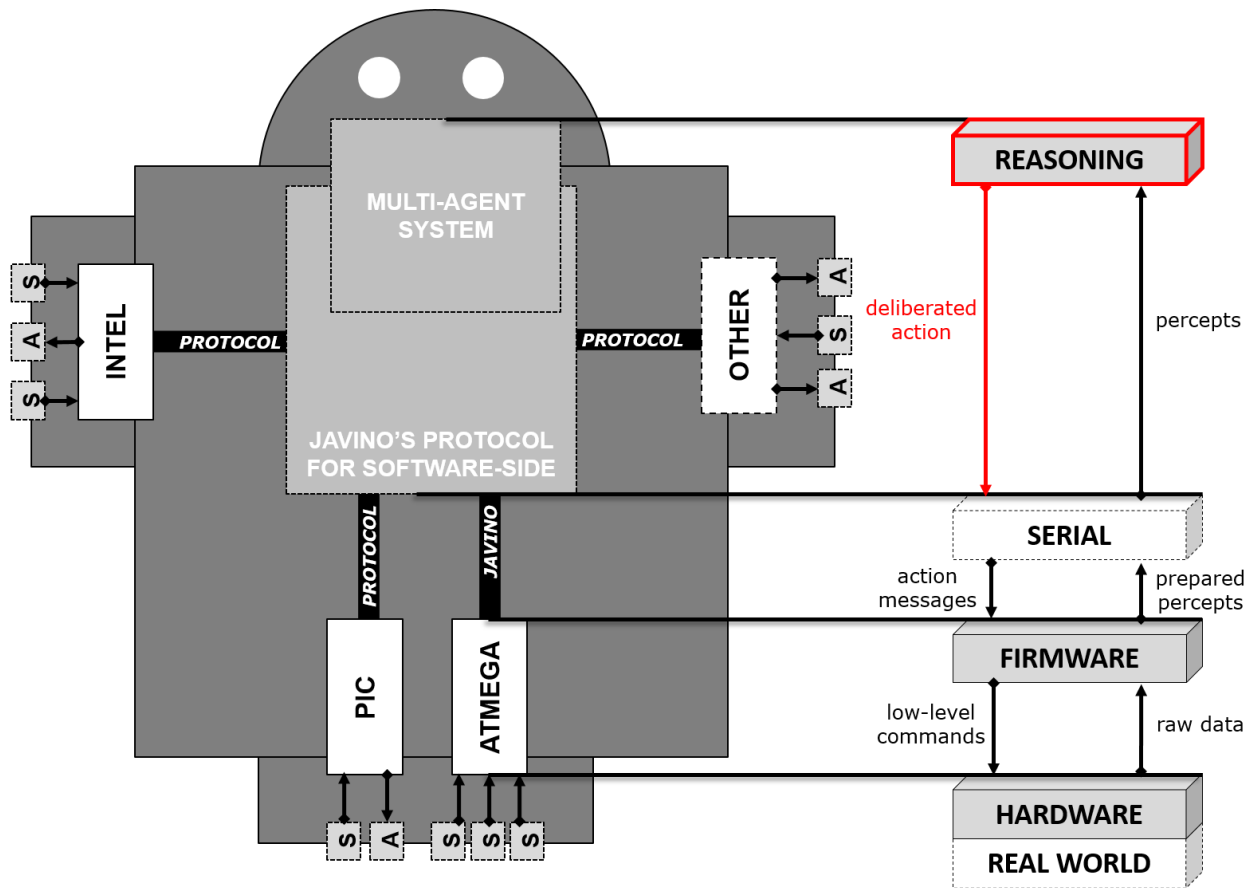
Desafios do Desenvolvimento de SMA Embarcados

- Equipe multidisciplinar
 - domínio de diferentes tecnologias (hardware, linguagens, frameworks, etc.)
- Integração e configuração das diversas tecnologias
- Heterogeneidade de componentes
- Múltiplos ambientes de desenvolvimento:
 - Arduino IDE
 - JEdit/Eclipse/etc.

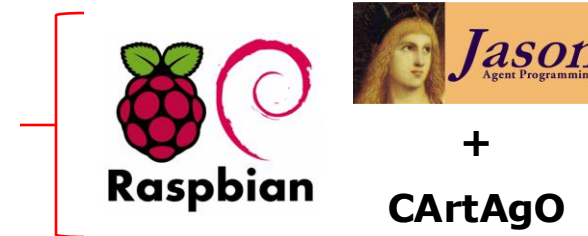
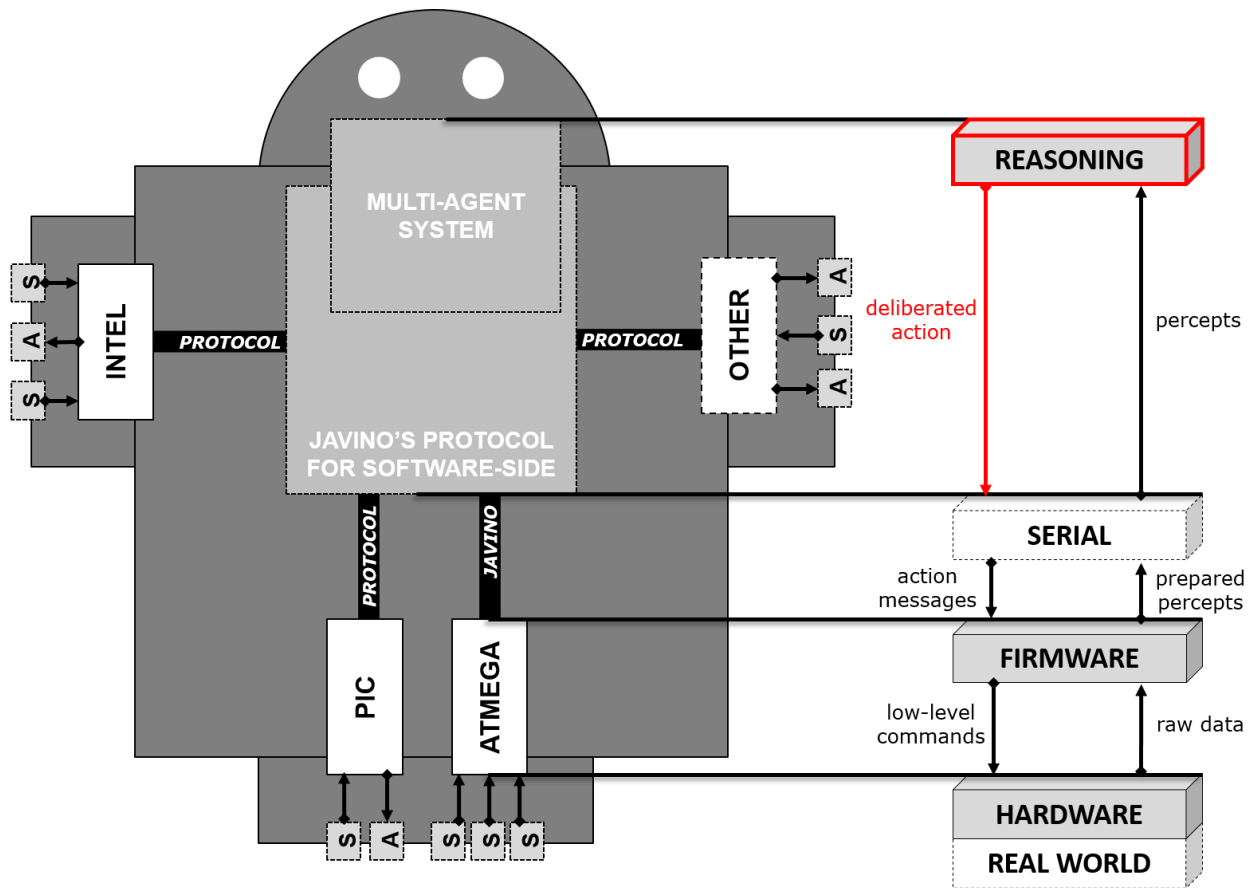
Tecnologias de Desenvolvimento



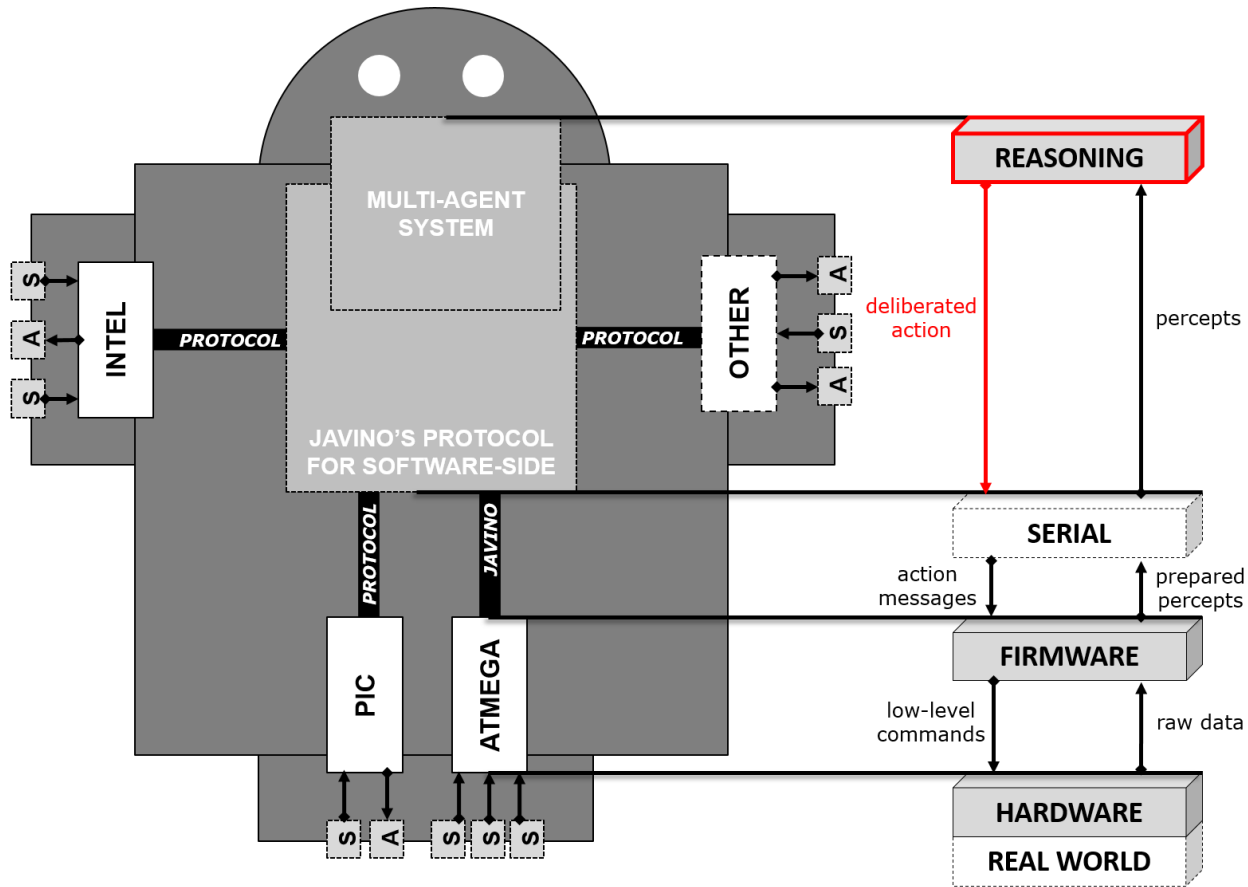
Tecnologias de Desenvolvimento



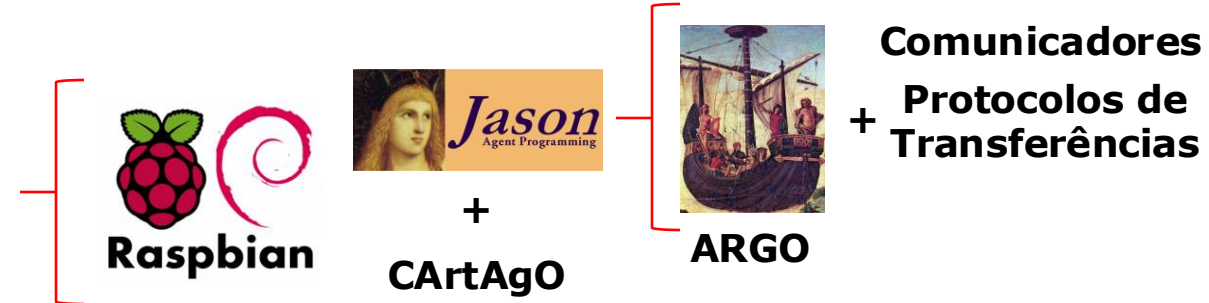
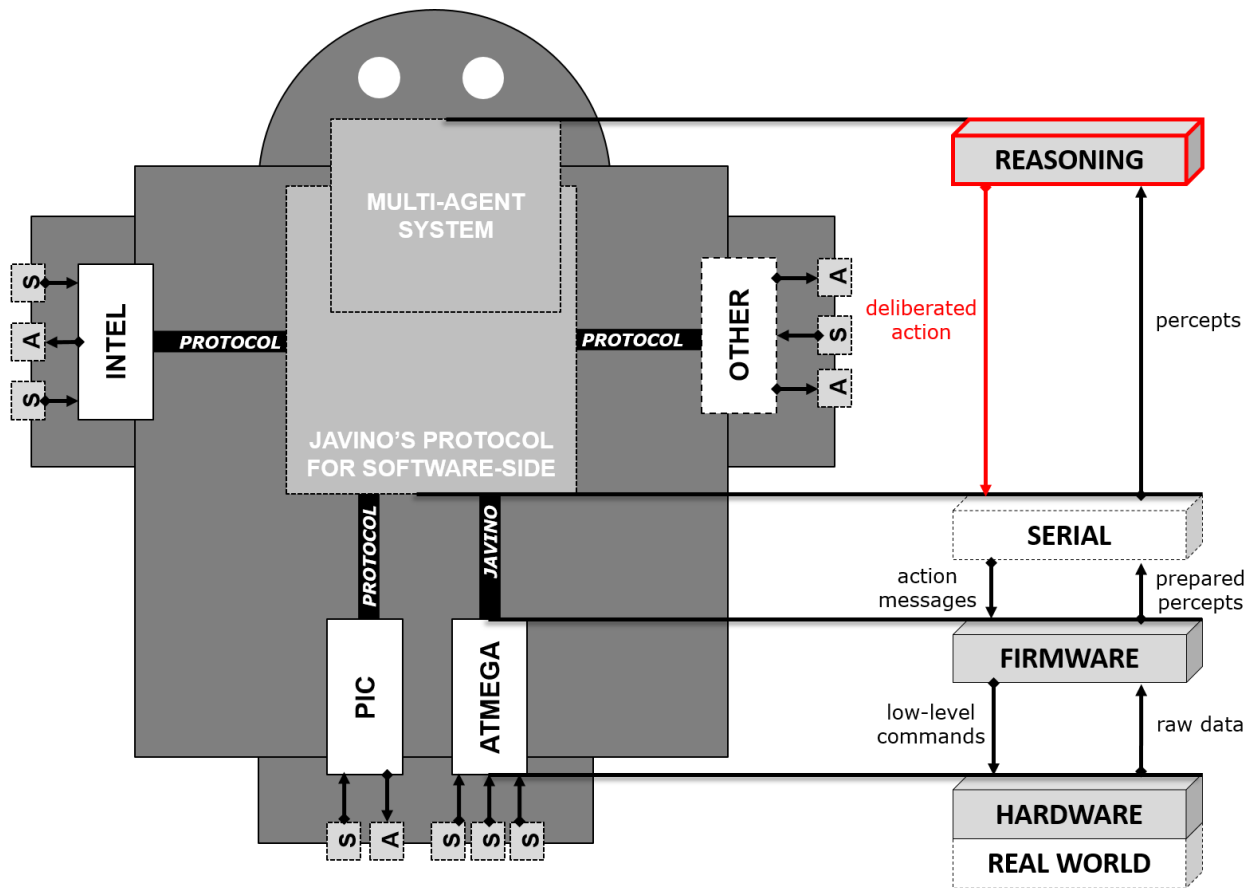
Tecnologias de Desenvolvimento



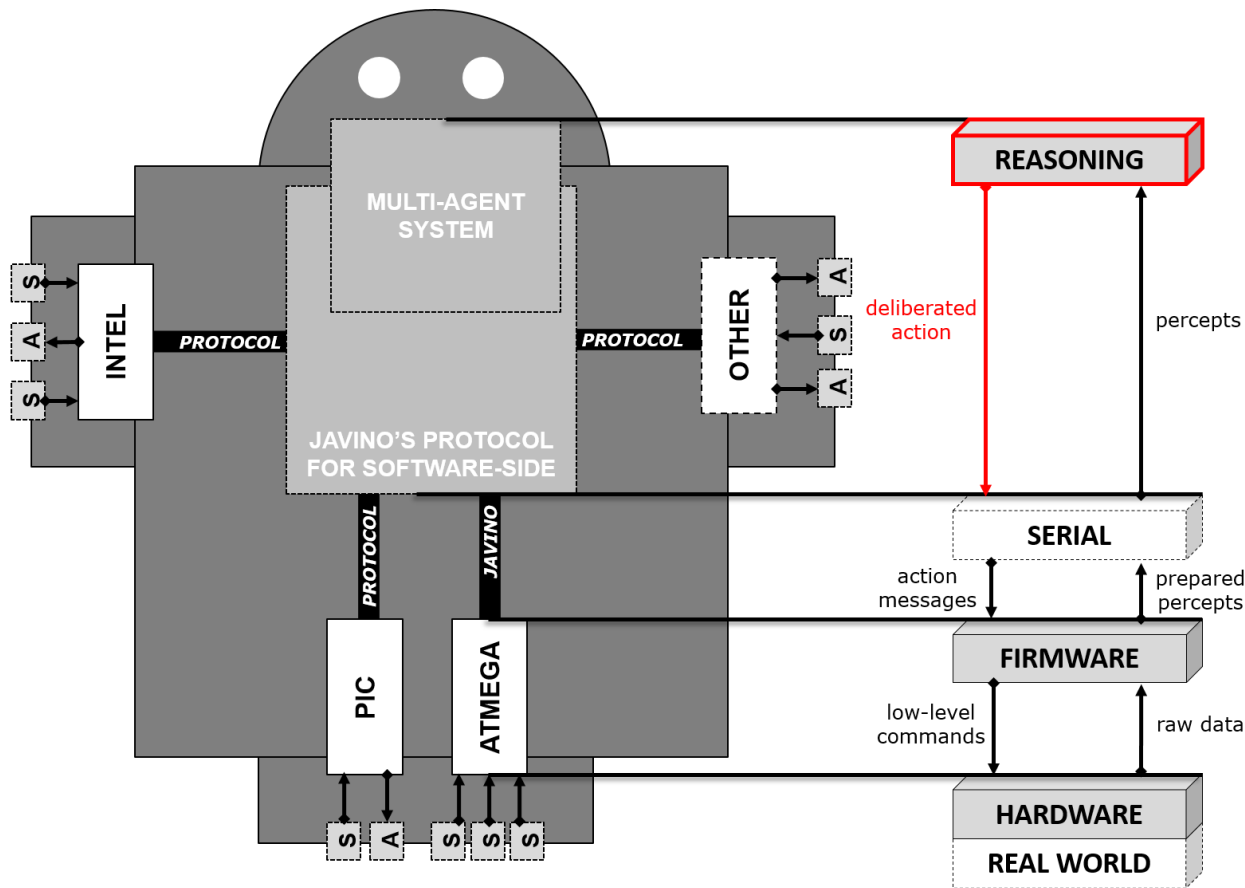
Tecnologias de Desenvolvimento



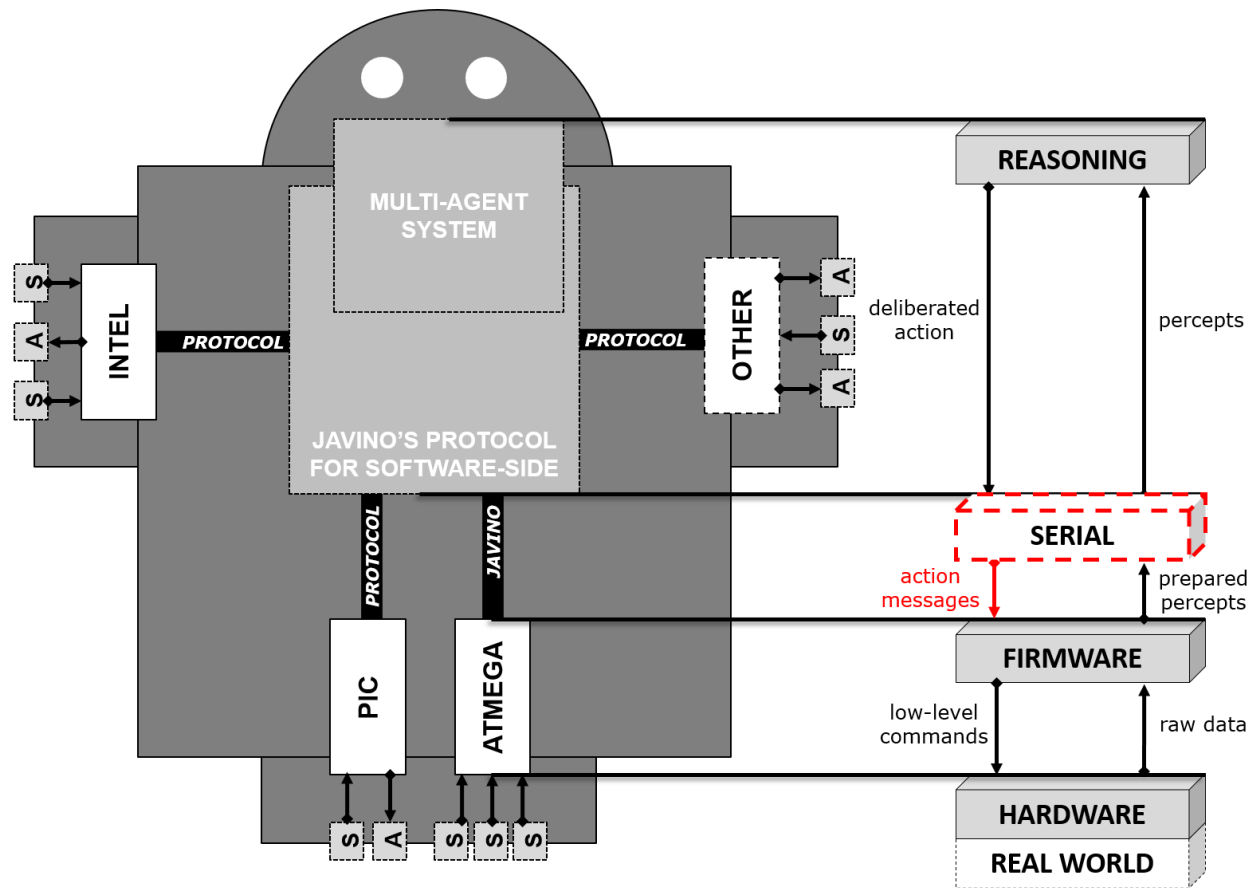
Tecnologias de Desenvolvimento



Tecnologias de Desenvolvimento

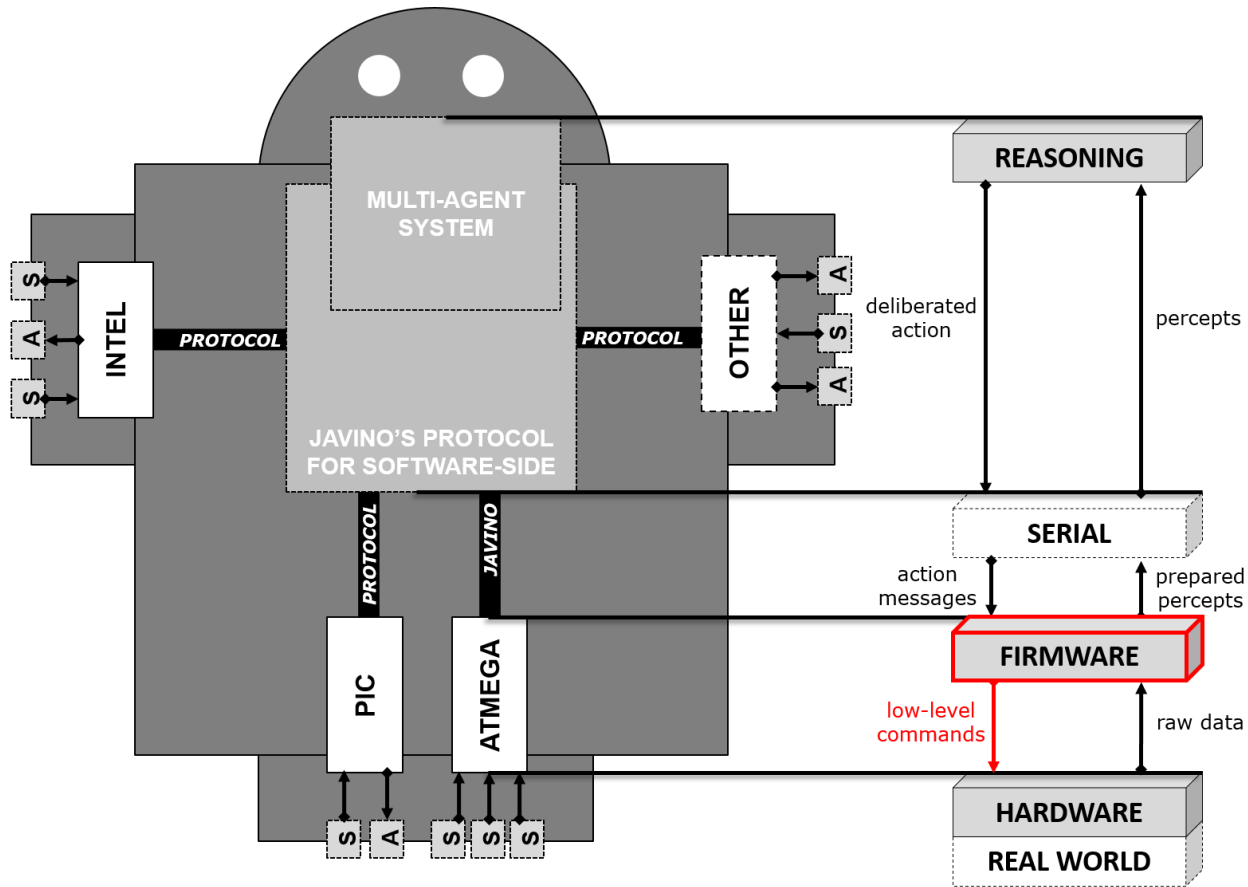


Tecnologias de Desenvolvimento

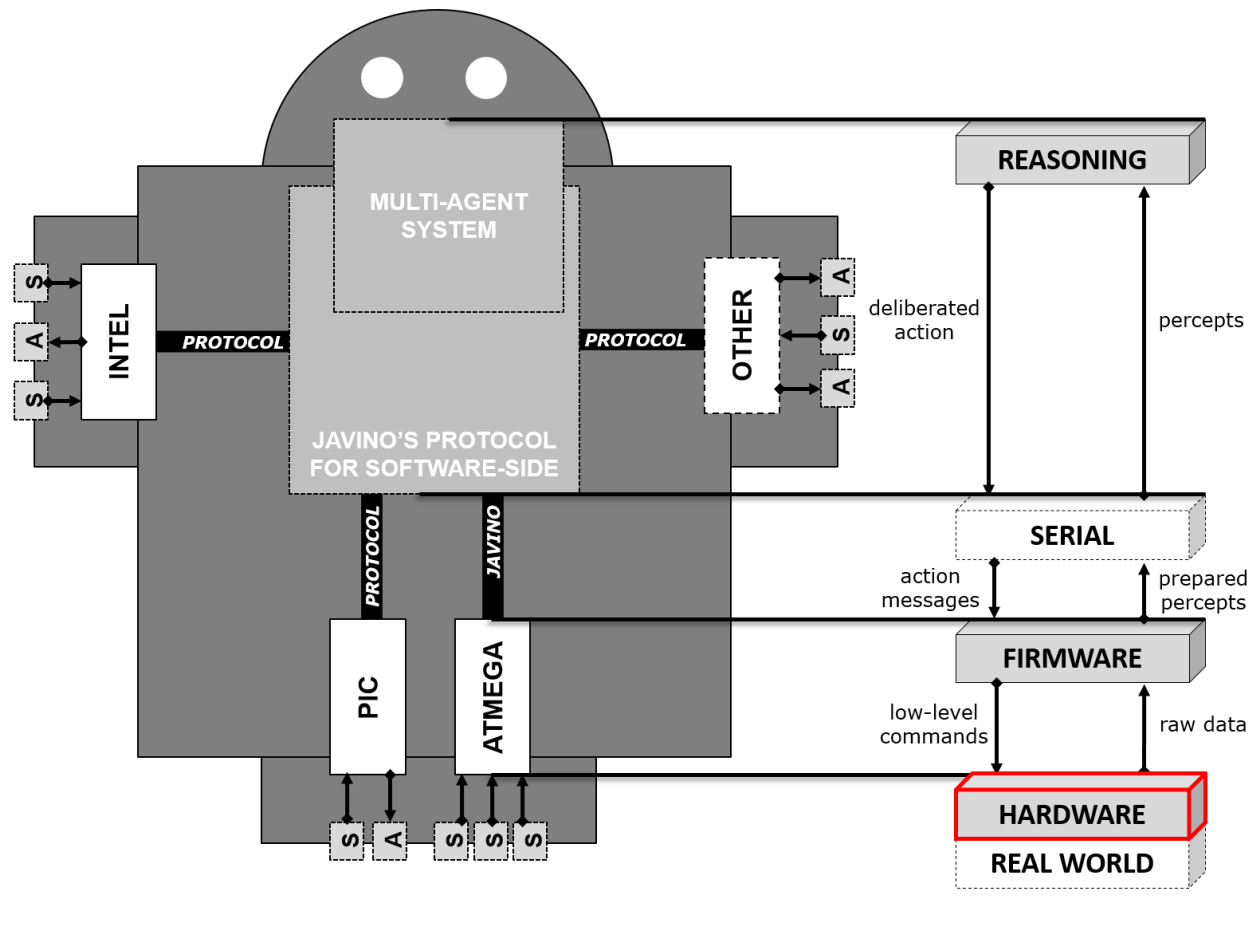


JavINO

Tecnologias de Desenvolvimento



Tecnologias de Desenvolvimento



sensores e atuadores



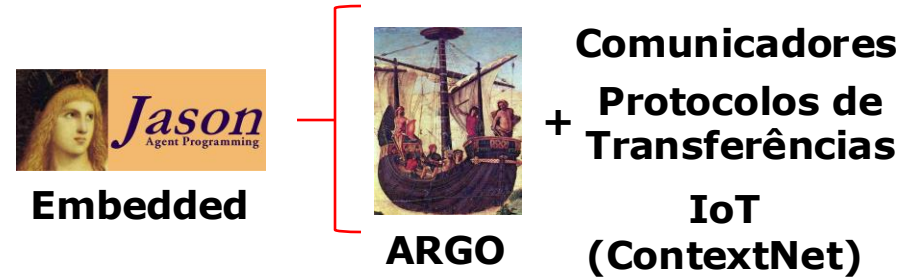
Abordagem Proposta

Abordagem Proposta

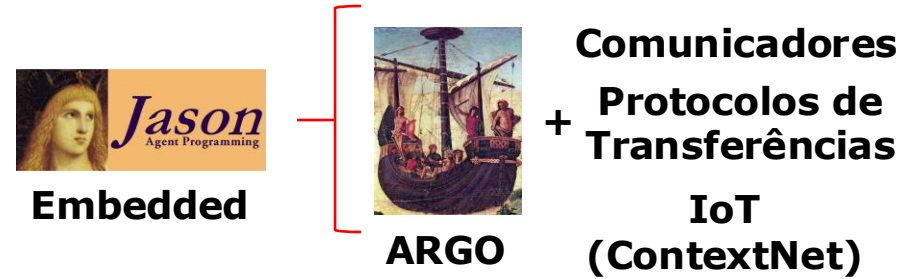


Embedded

Abordagem Proposta

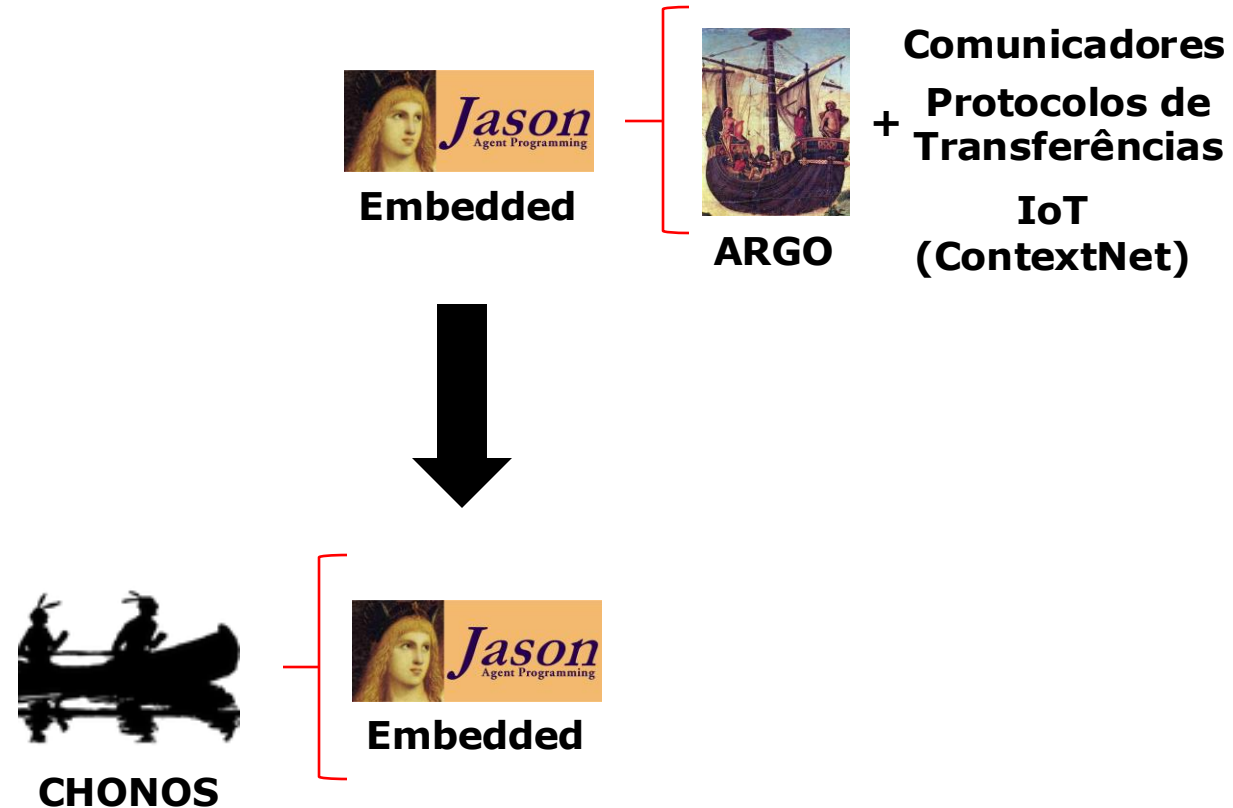


Abordagem Proposta

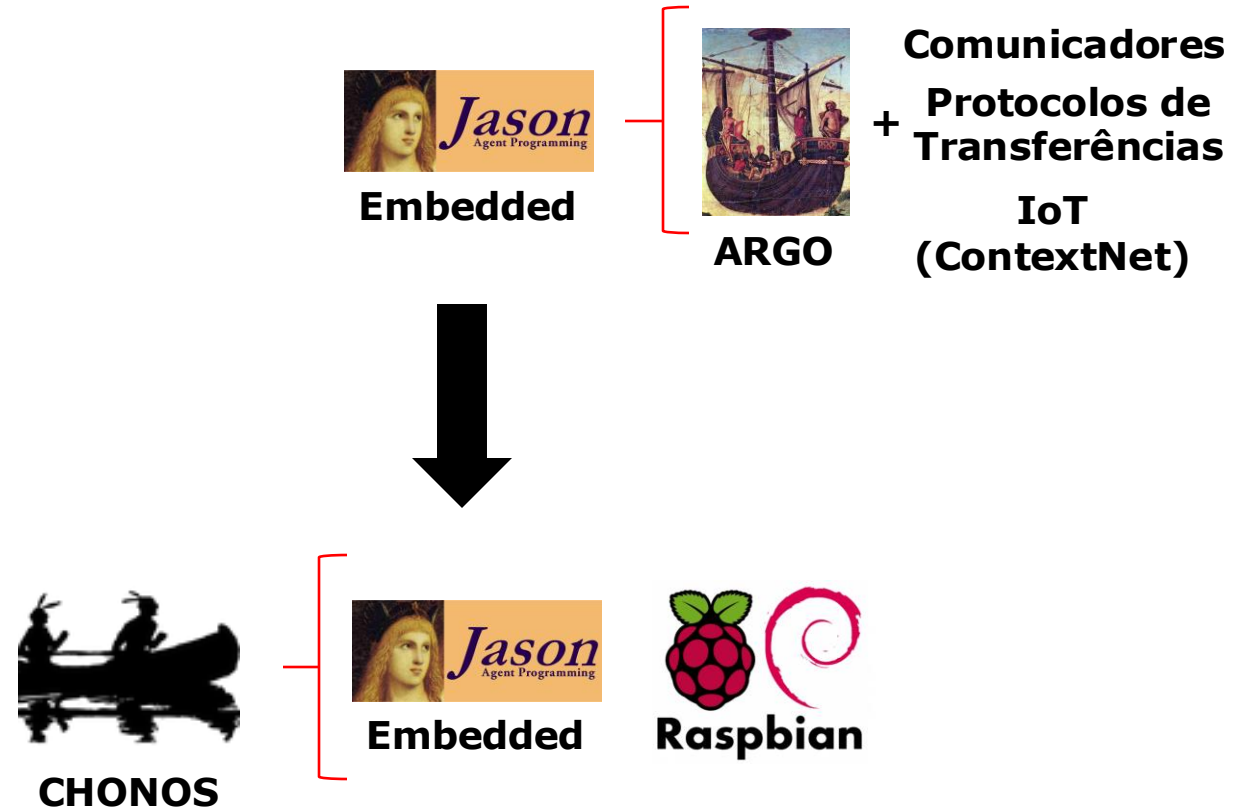


CHONOS

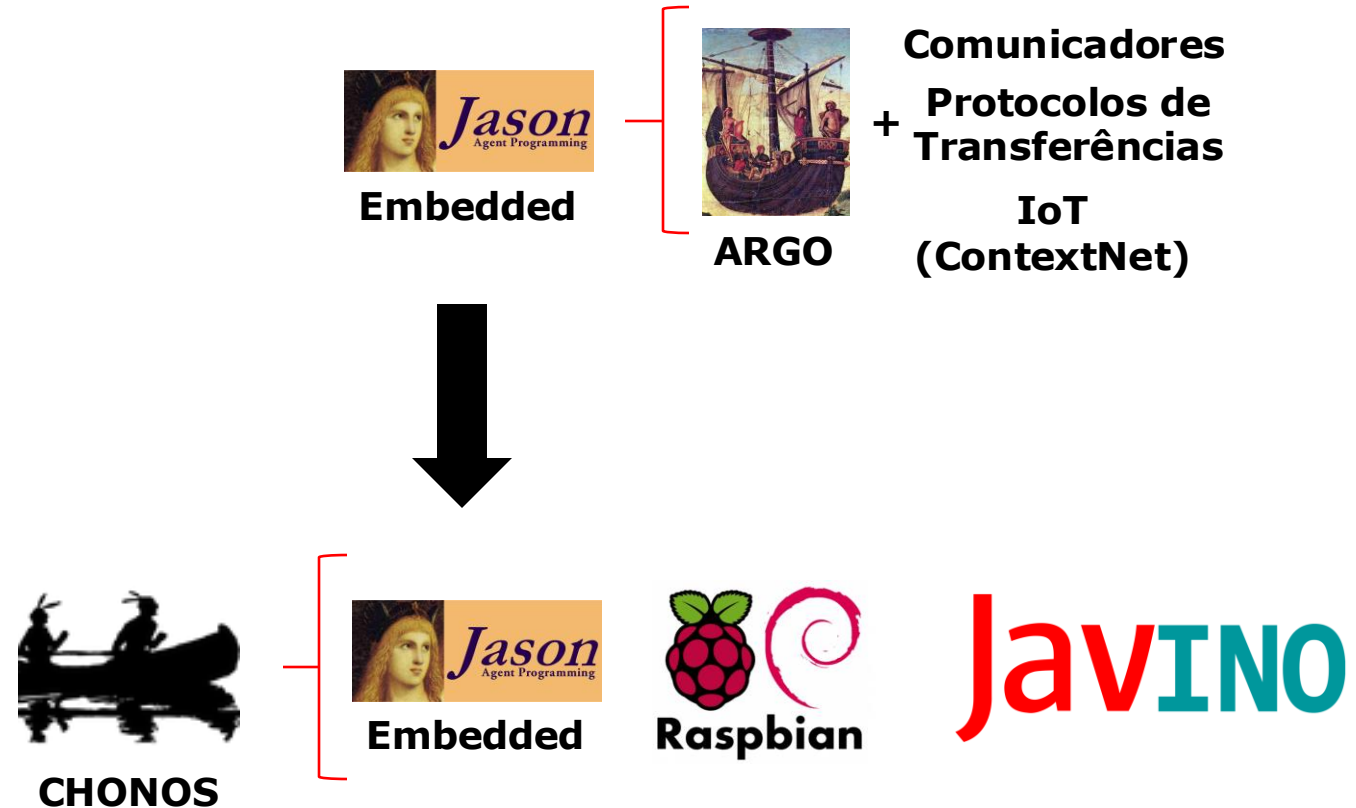
Abordagem Proposta



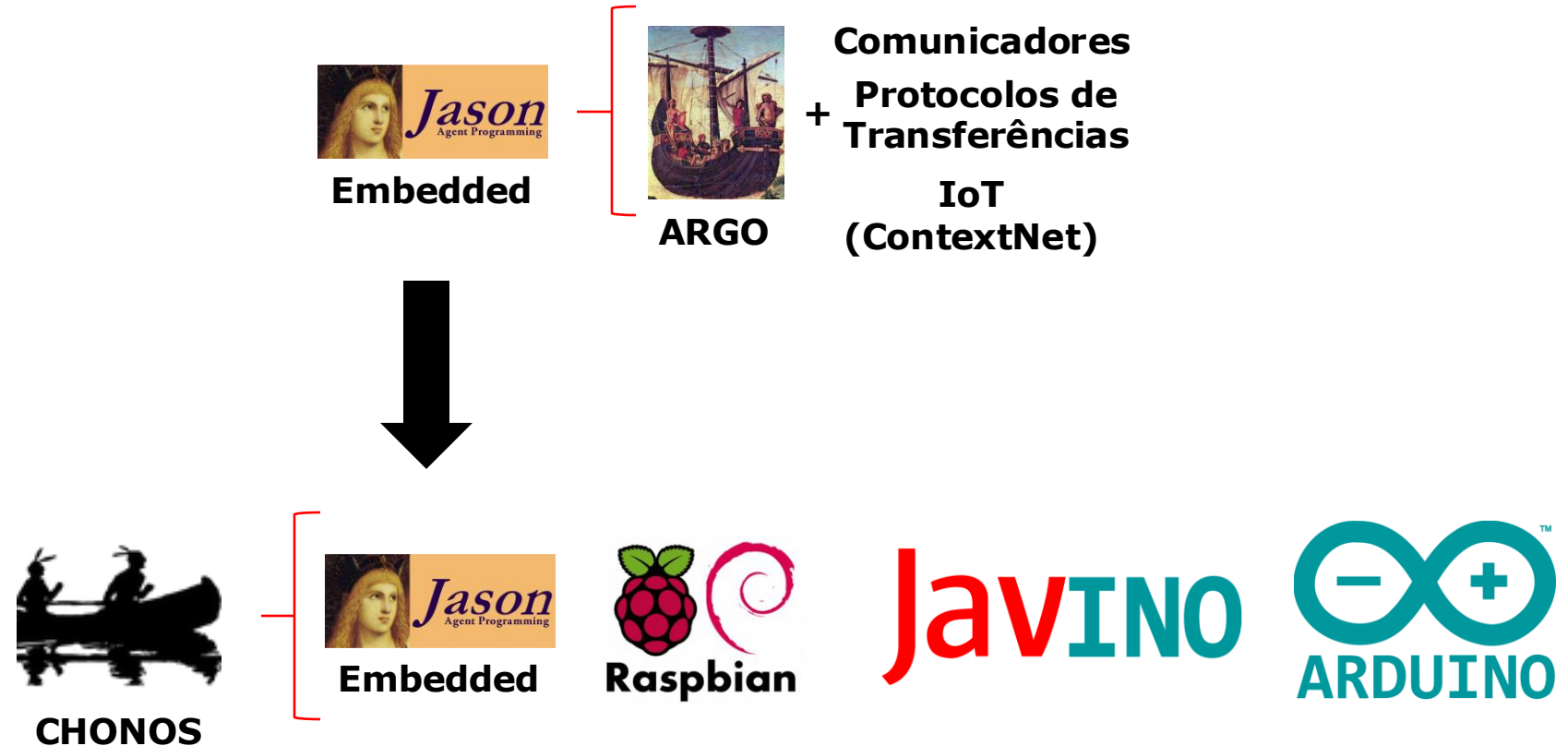
Abordagem Proposta



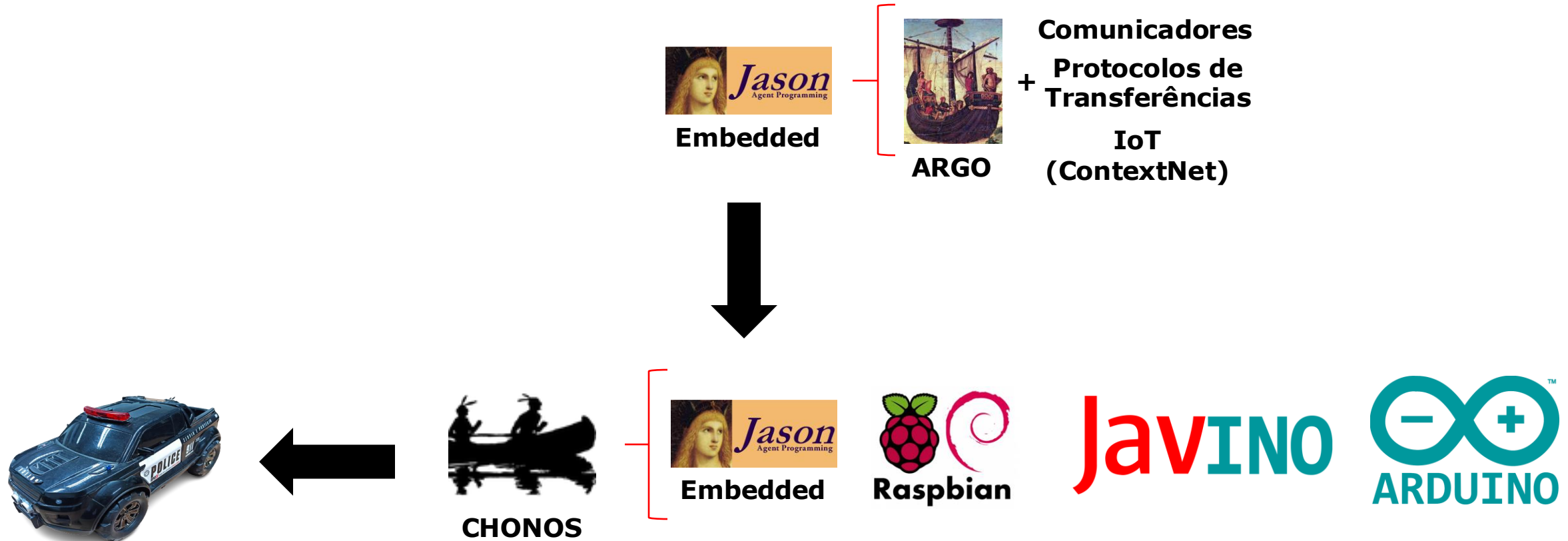
Abordagem Proposta



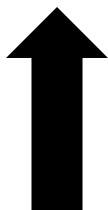
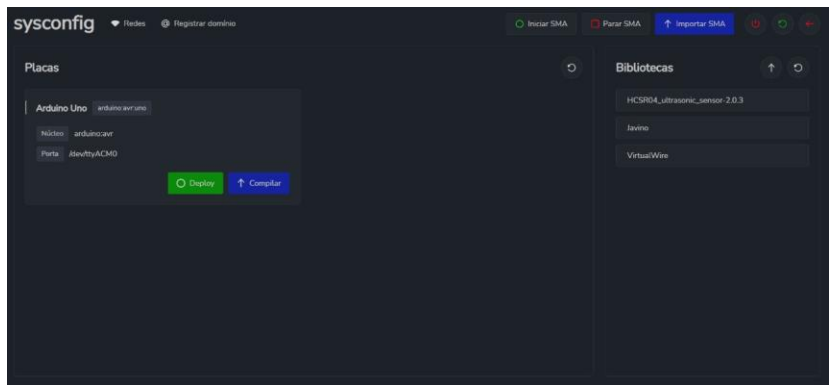
Abordagem Proposta



Abordagem Proposta



Abordagem Proposta



CHONOS



Embedded

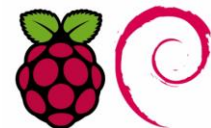


ARGO

**Comunicadores
+
Protocolos de
Transferências
IoT
(ContextNet)**



Embedded



Raspbian

JavINO



ARDUINO

Introdução ao desenvolvimento de SMA Embarcados
utilizando a distro **chonOS** e o spin-off **Jason
Embedded**;

Introdução ao desenvolvimento de SMA Embarcados
utilizando a distro **chonOS** e o spin-off **Jason
Embedded**;

Objetivos secundários:

Introdução ao desenvolvimento de SMA Embarcados utilizando a distro **chonOS** e o spin-off **Jason Embedded**;

Objetivos secundários:

- Instalação e configuração da distro **chonOS**;

Introdução ao desenvolvimento de SMA Embarcados utilizando a distro **chonOS** e o spin-off **Jason Embedded**;

Objetivos secundários:

- Instalação e configuração da distro **chonOS**;
- Criação de agentes **Argo**;

Introdução ao desenvolvimento de SMA Embarcados utilizando a distro **chonOS** e o spin-off **Jason Embedded**;

Objetivos secundários:

- Instalação e configuração da distro **chonOS**;
- Criação de agentes **Argo**;
- Comunicação entre agentes de SMA Embarcados distintos;

Introdução ao desenvolvimento de SMA Embarcados utilizando a distro **chonOS** e o spin-off **Jason Embedded**;

Objetivos secundários:


- Instalação e configuração da distro **chonOS**;
- Criação de agentes **Argo**;
- Comunicação entre agentes de SMA Embarcados distintos;
- Movimentação de agentes;

ATO 1: O CHONOS

Download da Imagem (<http://chonos.sf.net>)



[Open Source Software](#) [Business Software](#) [Resources](#)

[Home](#) / [Browse](#) / ChonOS



ChonOS

A specifical-purpose GNU/Linux distribution for Embedded MAS
Brought to you by: [kadupantoja](#), [nilsonmori](#)

 [Download](#)  [Get Updates](#) [Share This](#)

[Summary](#) [Files](#) [Reviews](#) [Support](#) [Git ▼](#)

ChonOS (Cognitive Hardware on Network - Operational System) is a specifical-purpose GNU/Linux distribution that seeks to facilitate the development of an Embedded MultiAgent System (MAS).

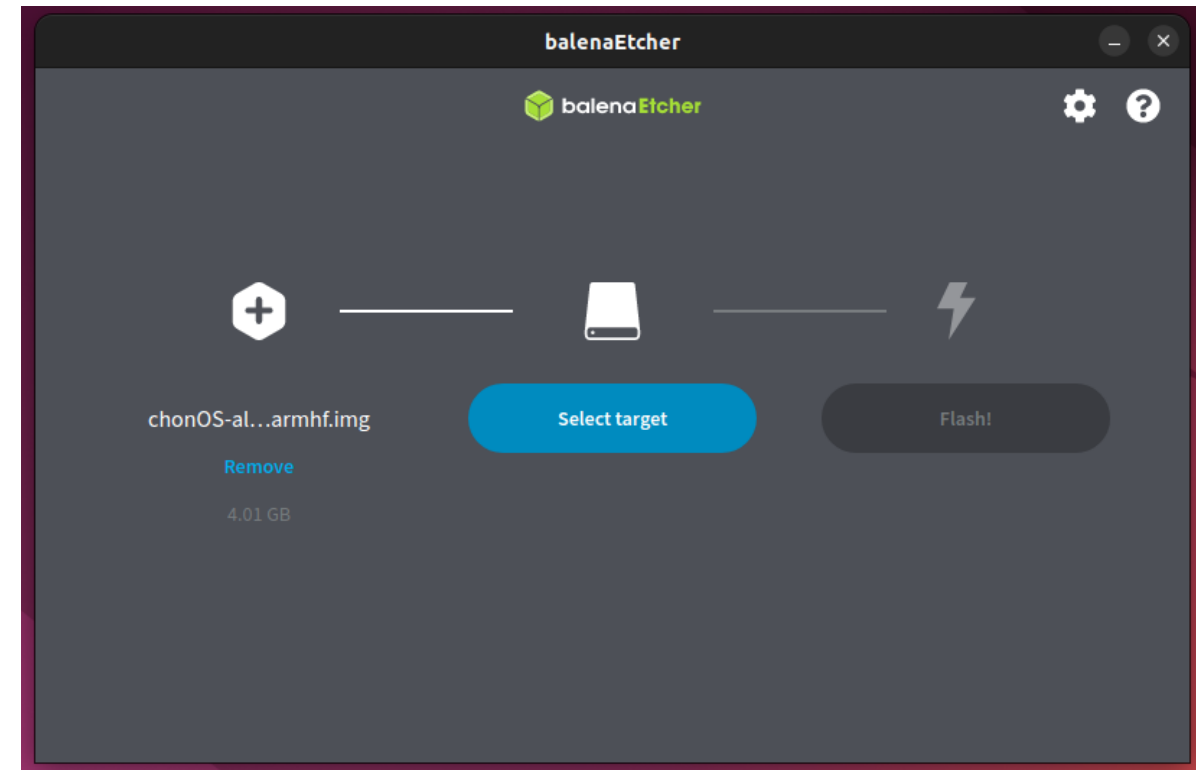
ChonOS enables, without the need to turn off the device or stop the MAS: the deployment of reasoning to the robot; firmware deployment for microcontrollers; the transfer of the MAS from the development environment to the production environment; and the transfer of new agents to the MAS running using the Internet of Things (IoT).

ChonOS also features an extended version of Jason specifically for Embedded MAS, which allows communication with hardware and an IoT middleware.

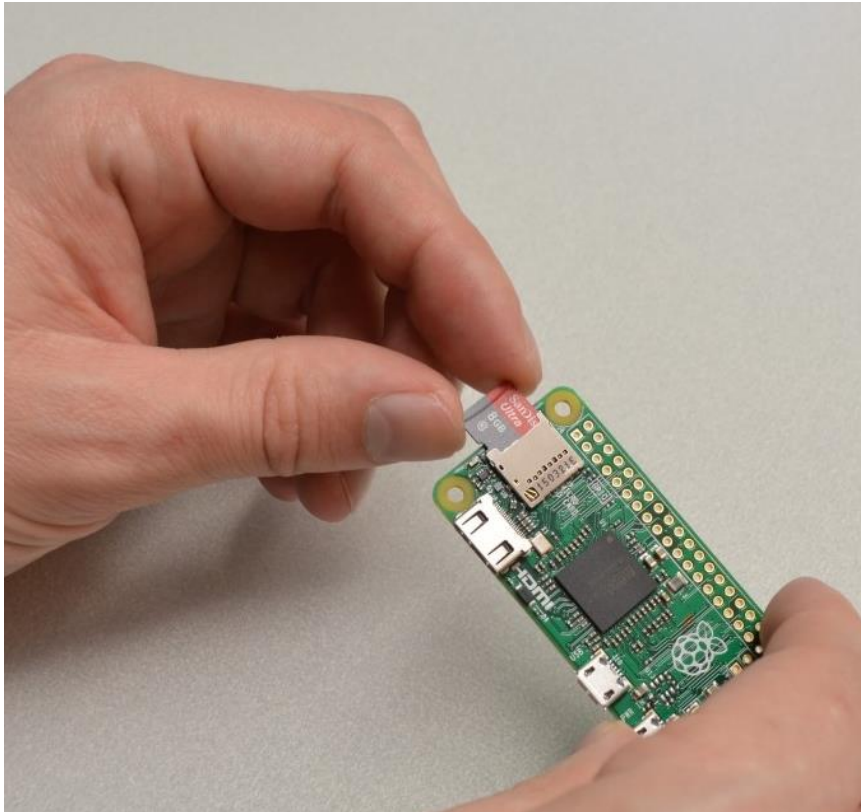
Features

- Wi-Fi Management
- Reasoning Upload
- Mind Inspector
- Firmware Upload
- MAS log monitor

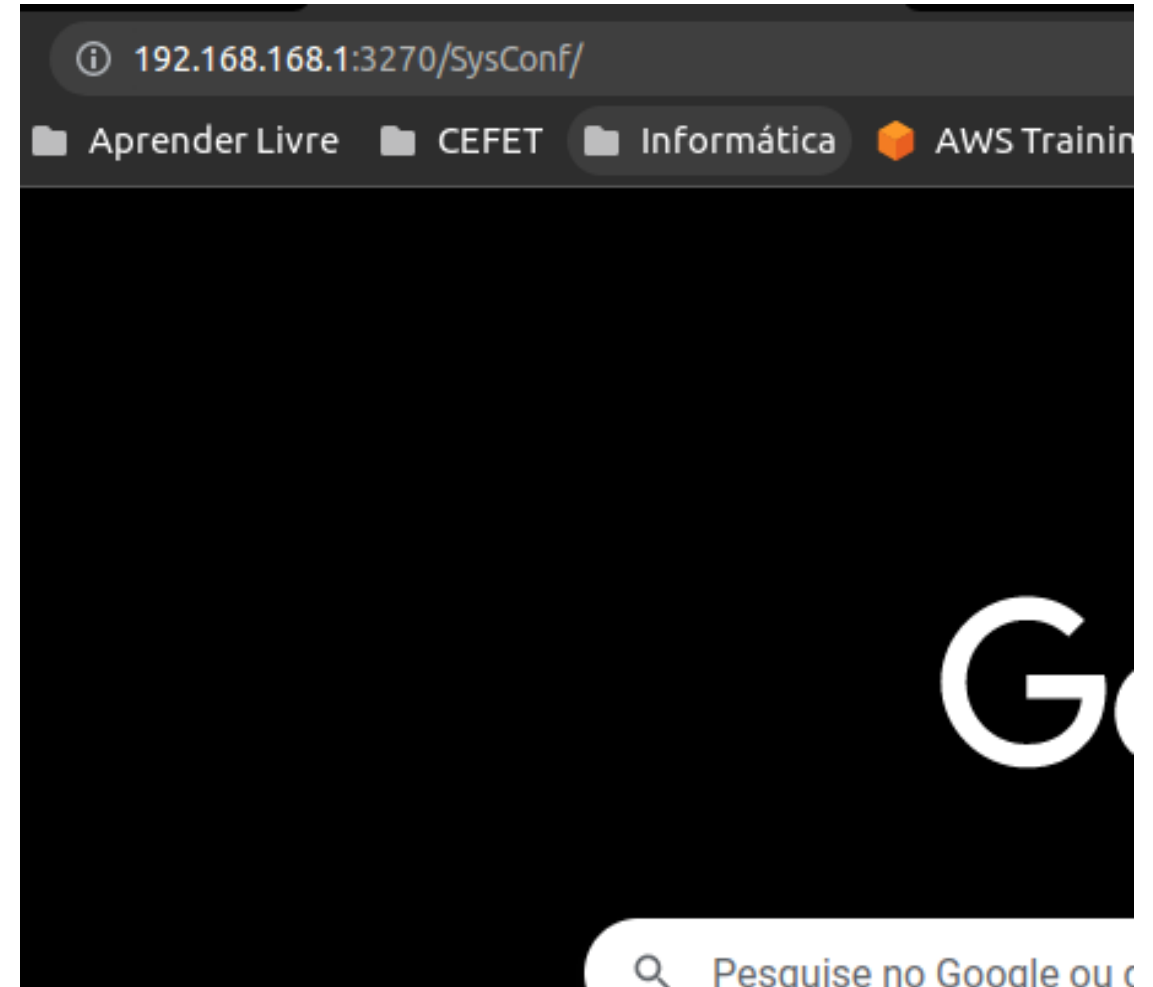
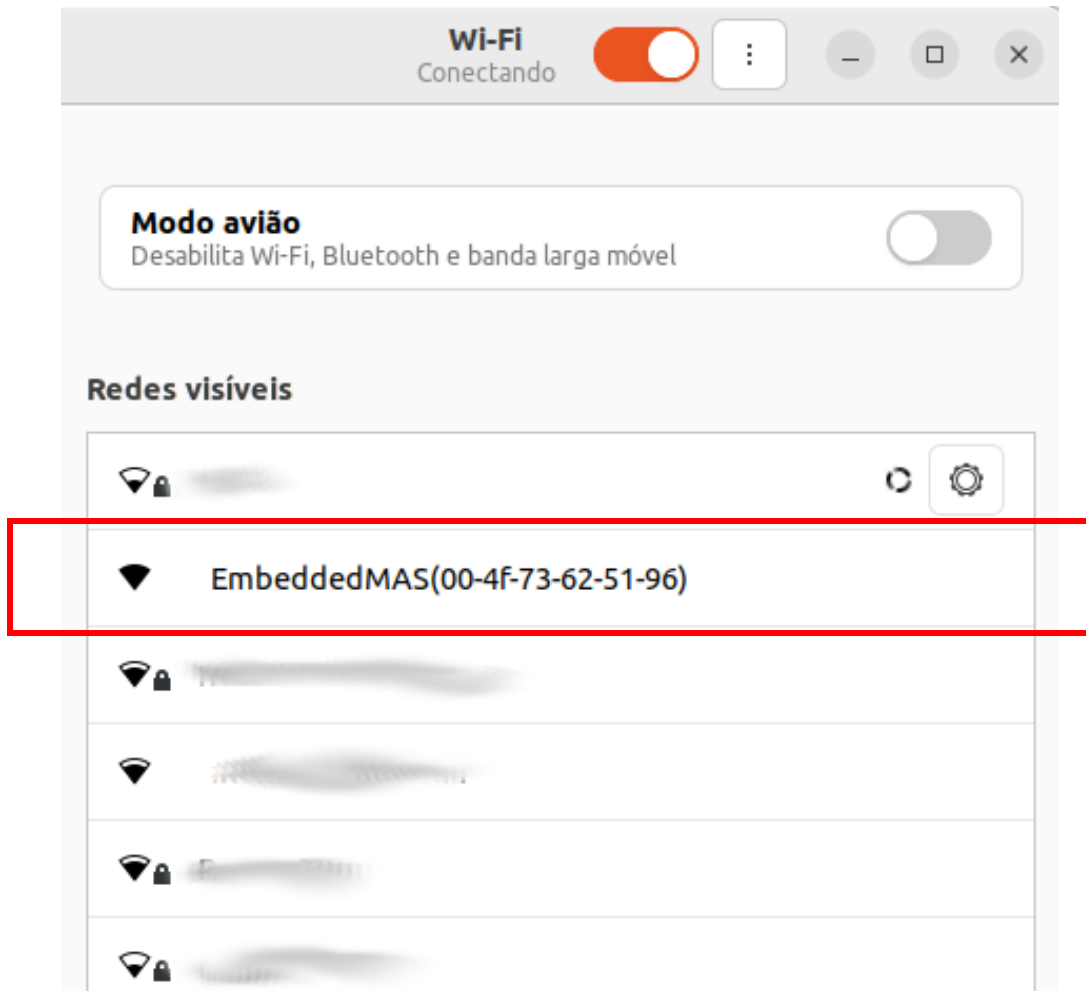
Gravar imagem (<https://balena.io/etcher>)



Ligar Raspberry Pi



Conectando

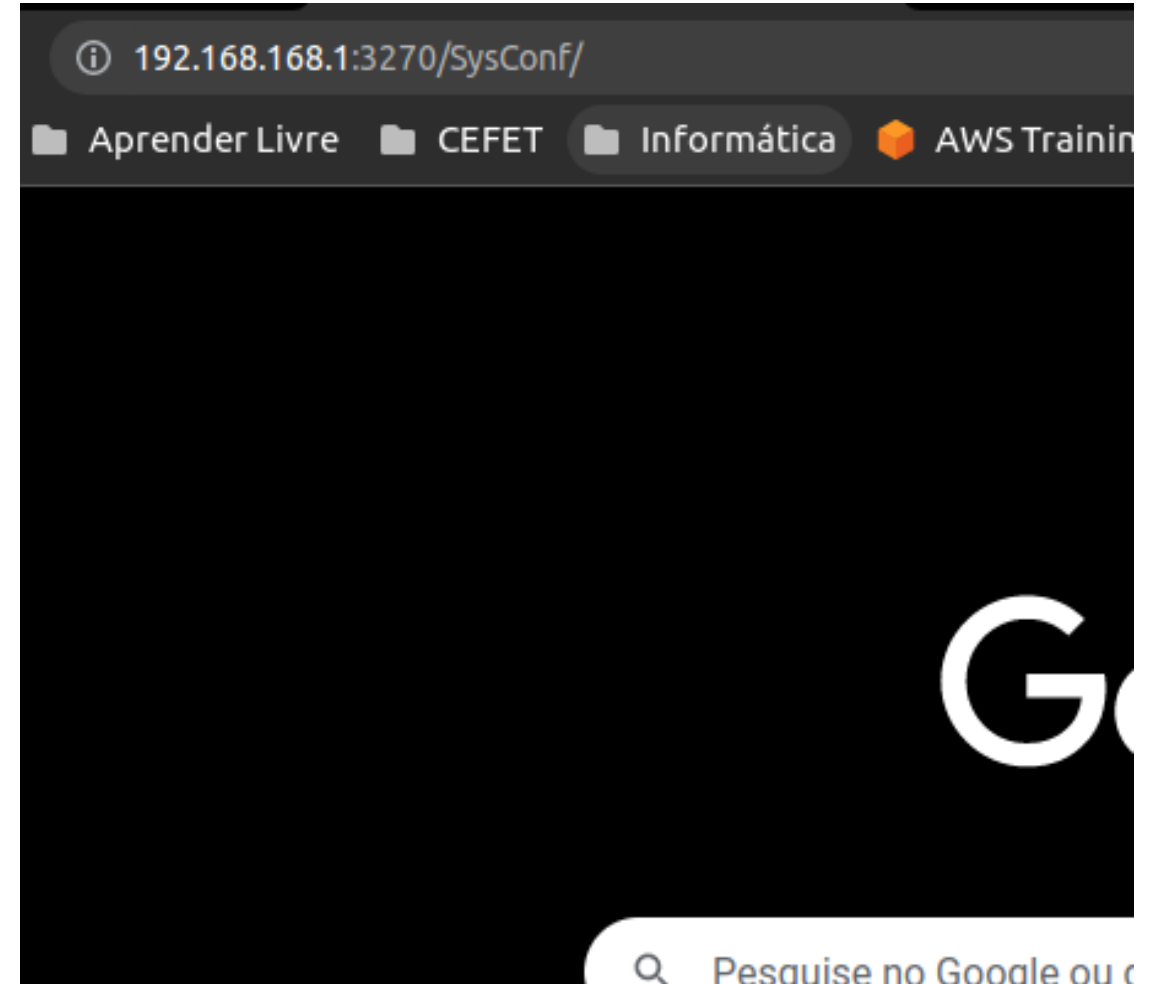


Configurar DDNS e Rede

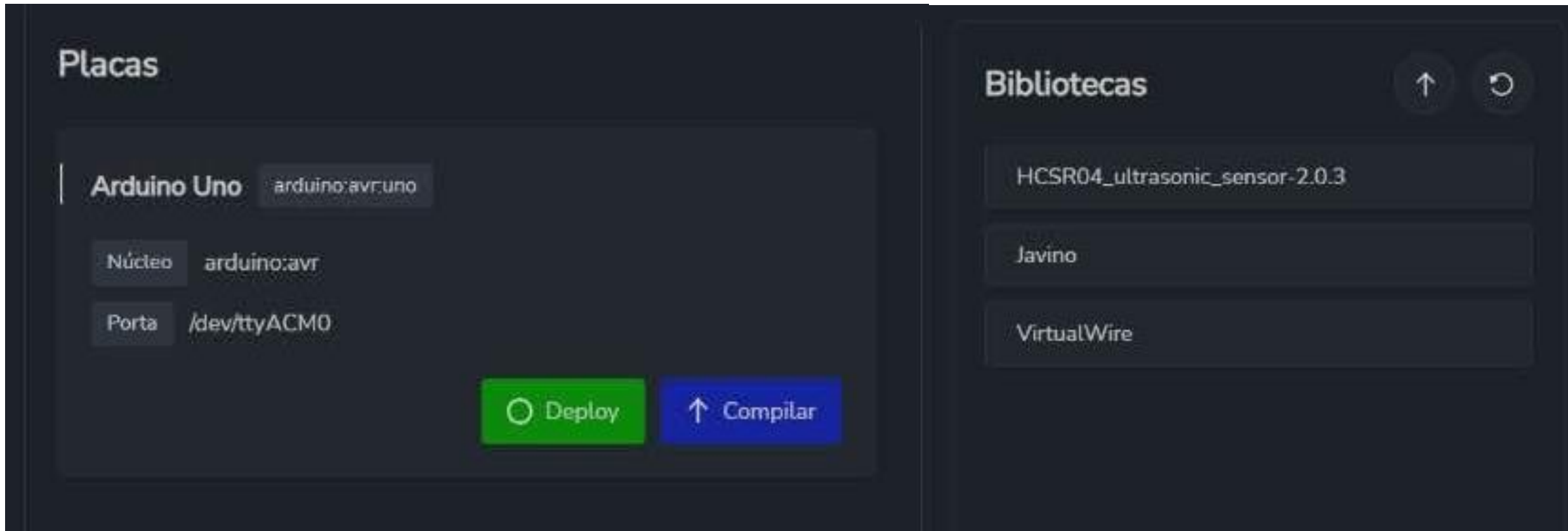
Informe o nome do seu bot

Através desse nome, você será capaz de acessar o sysconfig via: agente-embarcado.bot.chon.group:3270sysconfig

Salvar nome



Upload de Firmware



Upload de Reasoning



TAREFA CHONOS

ATO 2:

ESCOLHA DO SEU BOT

Escolha dos Bots



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



Escolha dos Bots



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



Escolha dos Bots



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



Escolha dos Bots



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



Escolha dos Bots



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




← → ↻ 🏠 🔒 sourceforge.net/p/chonos/examples/ci/master/tree/ 🔍 🔗 ☆ 📄 ⚙️ 🗑️ 👤

SOURCEFORGE Help Create Join Login

Open Source Software Business Software Resources 🔍 Search for software or solutions

Home / Browse / ChonOS / Examples



ChonOS Examples

A specifical-purpose GNU/Linux distribution for Embedded MAS
Brought to you by: [kadupantoja](#), [nilsonmori](#)

Summary Files Reviews Support **Git ▾**

Browse Commits

Fork

Merge Requests (0)

Branches

master

Tree [\[14c7be\]](#) **master** /

Download Snapshot History

HTTPS git:// HTTPS access `git clone https://git.code.sf.net/p/chonos/examples chonos-examples`

File	Date	Author	Commit
basic	2 days ago	Prof Nilson Mori	[14c7be] remotelab
papers	2 days ago	Prof Nilson Mori	[14c7be] remotelab
test	4 days ago	Prof Nilson Mori	[38bd94] test skynet

ATO 3: O ARGO

- **Beliefs**

- 1. Percepções do Ambiente (Percepts)**

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

- 2. Notas Mentais (Mental Notes)**

Informações adicionadas na base de crenças pelo próprio agente resultado de coisas que aconteceram no passado, promessas. Esse tipo de informação geralmente é adicionada pela execução de um plano. constante do ambiente.

- 3. Comunicação**

Informações obtidas pelo agente através da comunicação com outros agentes.

- **Goals**

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

- **Tipos**

1. **Achievement Goals (!)**

É um objetivo para atingir determinado estado desejado pelo agente.

2. **Test Goals (?)**

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

- **Plans**

Triggering_event : **context** <- **body**.

```
+!order(Product,Qtd)[source(Ag)] : true <-  
  ?last_order_id(N);  
  OrderId = N + 1;  
  -+last_order_id(OrderId);  
  deliver(Product,Qtd);  
  .send(Ag, tell, delivered(Product,Qtd,OrderId)).
```

- **Plans**

- 1. Addition**

São ativados quando um plano é transformado de um desejo para uma intenção na mente do agente.

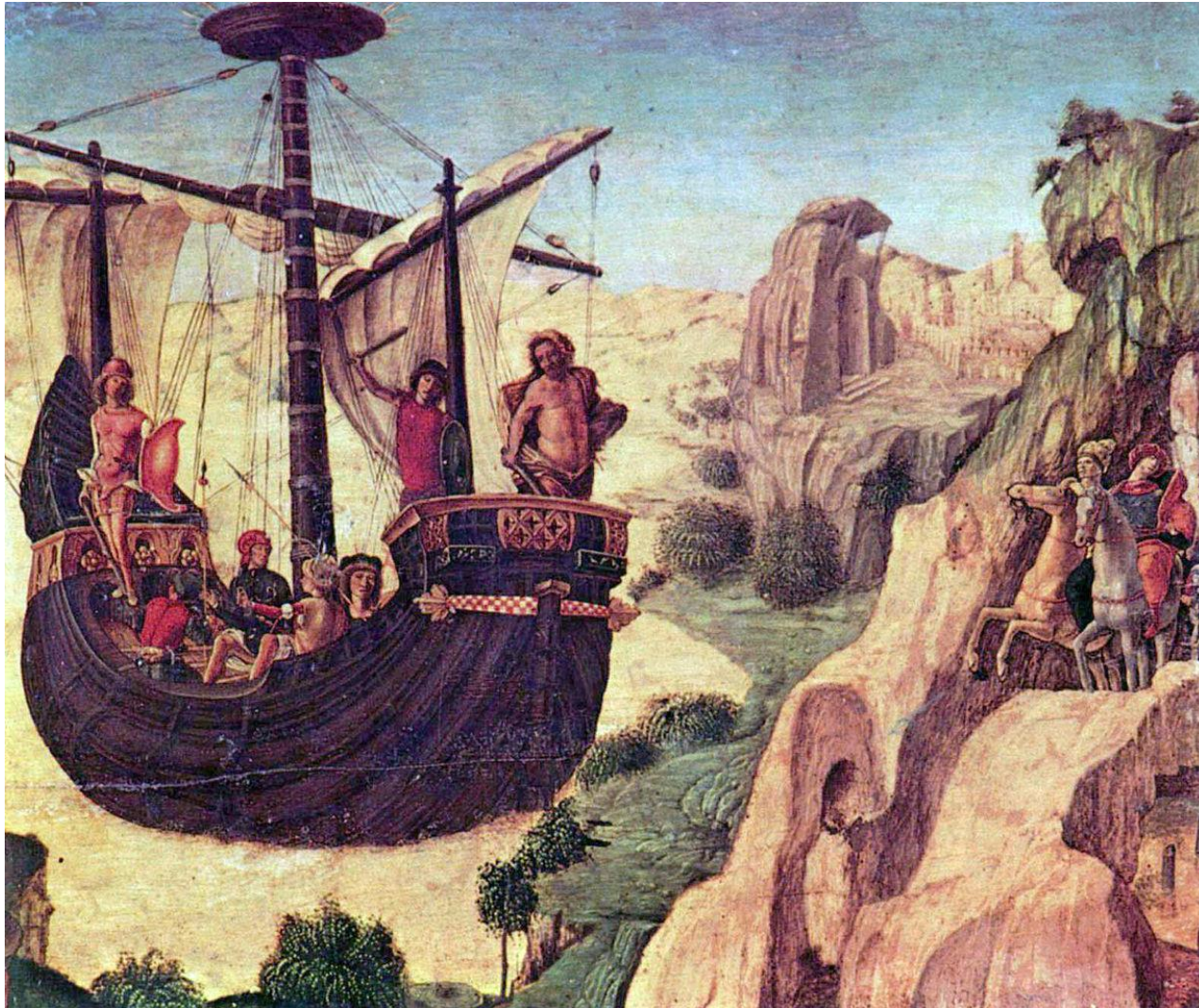
- 2. Test Goal**

São objetivos que recuperam informações da base de crenças.

- 3. Belief**

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

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

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.

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.

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.

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.

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.

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.

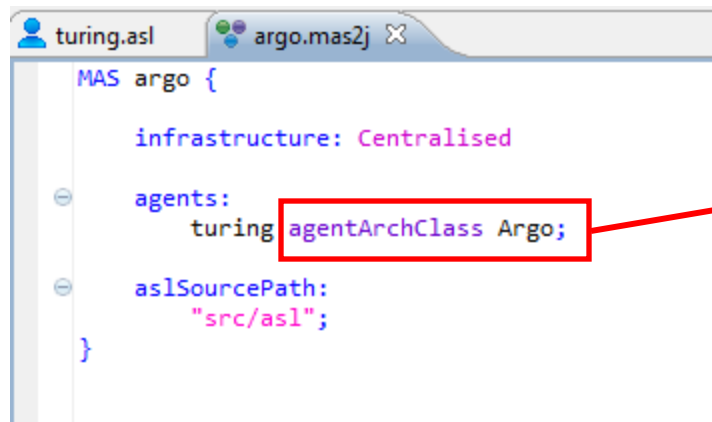
O **ARGO** permite:

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

Argo for Jason: Definição de um Agente

- Criando um agente ARGO e definindo sua arquitetura

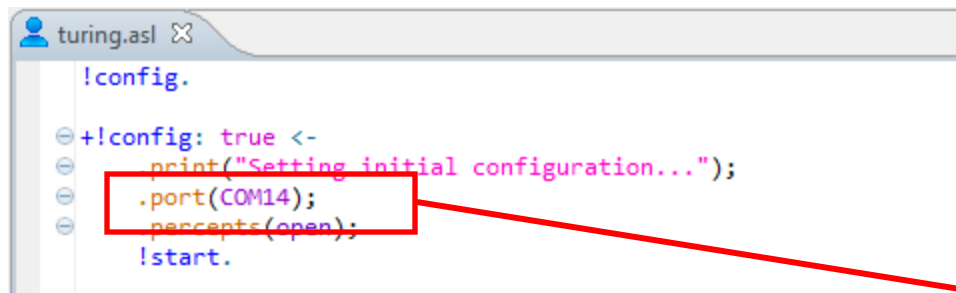


```
MAS argo {  
  infrastructure: Centralised  
  agents:  
    turing agentArchClass Argo;  
  aslSourcePath:  
    "src/asl";  
}
```

É necessário
especificar a classe
da arquitetura
customizada para
cada agente ARGO.

Argo for Jason: Acessando uma Porta

- Criando e definindo a arquitetura de um agente ARGO



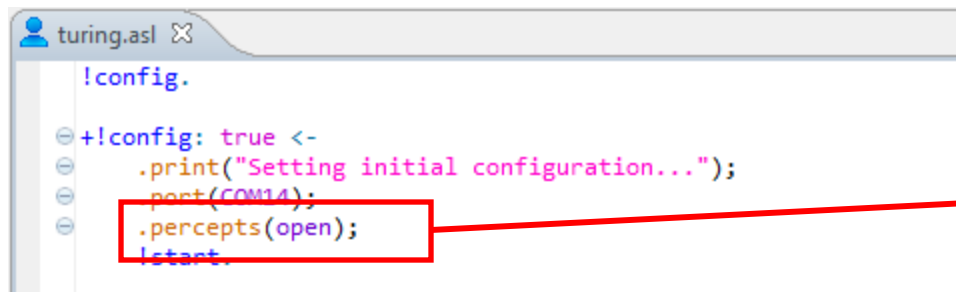
```
turing.asl X
!config.

+!config: true <-
  print("Setting initial configuration...");
  .port(COM14);
  percepts(open);
!start.
```

É preciso escolher qual dispositivo o agente irá controlar definindo a porta serial onde o mesmo estiver conectado.

Argo for Jason: Fluxo de Percepções

- Criando e definindo a arquitetura de um agente ARGO

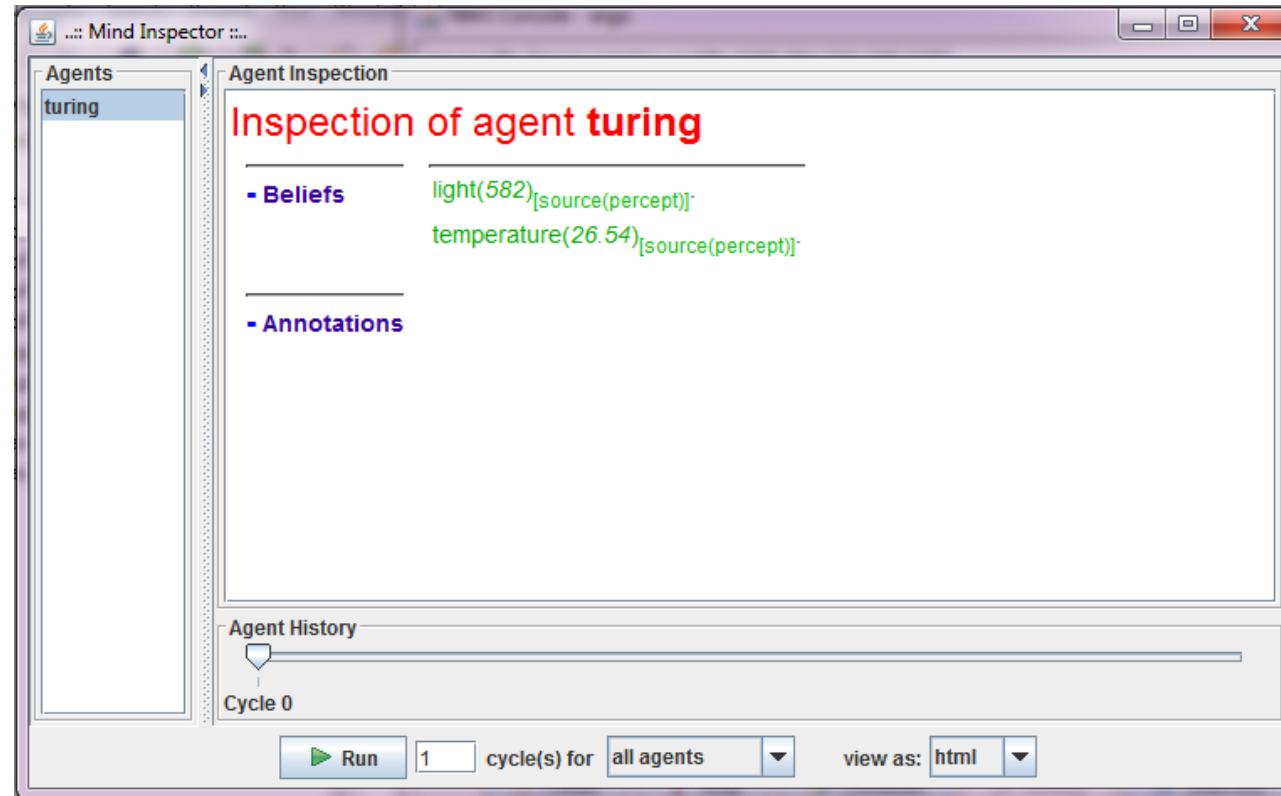


```
turing.asl X
!config.
+!config: true <-
    .print("Setting initial configuration...");
    .port(COM14);
    .percepts(open);
end.
```

Por padrão, as percepções provenientes dos sensores estão inicialmente bloqueadas. Para isso é necessário desbloqueá-las .

Argo for Jason: A Mente do Agente

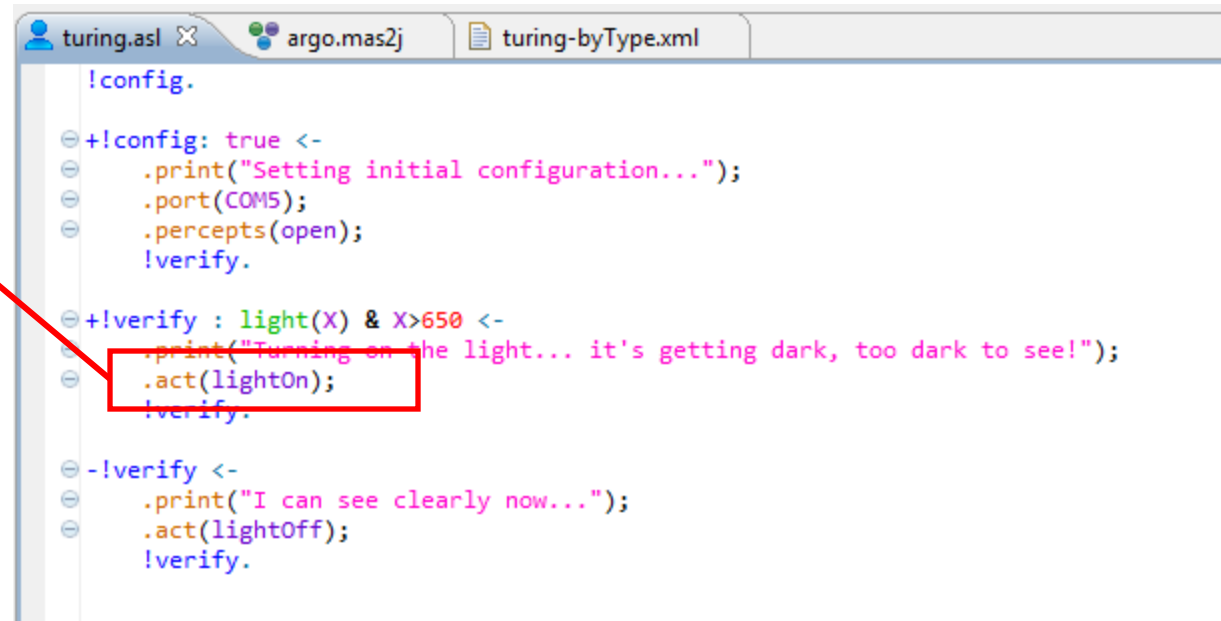
- Verificando as percepções adquiridas



Argo for Jason: Ativando Atuadores

- **Ativando um efetuator**

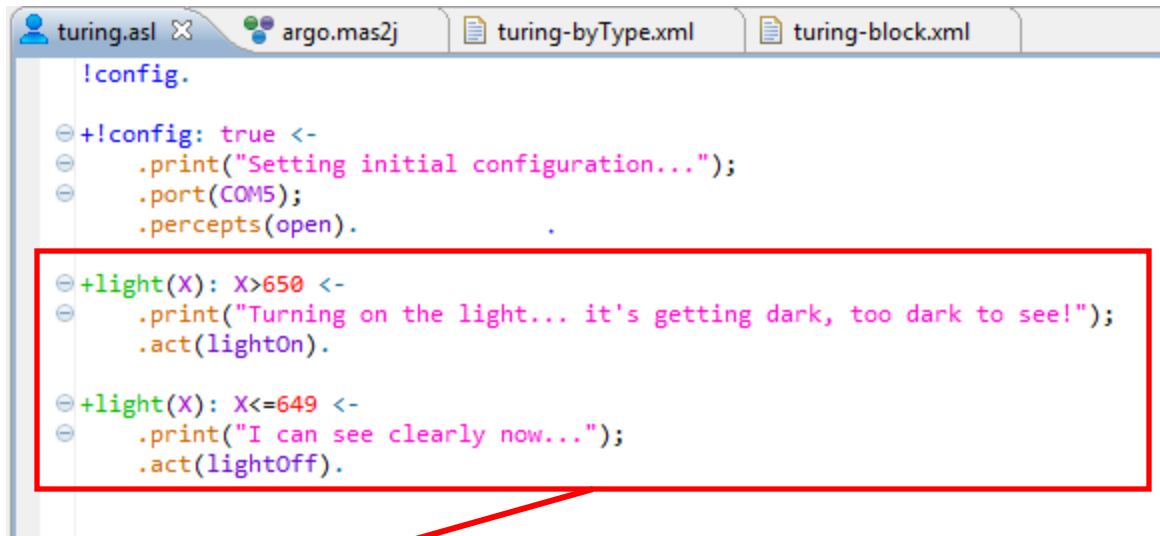
Ativando um
efetuator
utilizando a ação
interna *act*.



```
!config.  
+!config: true <-  
  .print("Setting initial configuration...");  
  .port(COM5);  
  .percepts(open);  
  !verify.  
+!verify : light(X) & X>650 <-  
  .print("Turning on the light... it's getting dark, too dark to see!");  
  .act(lightOn);  
  !verify.  
-!verify <-  
  .print("I can see clearly now...");  
  .act(lightOff);  
  !verify.
```

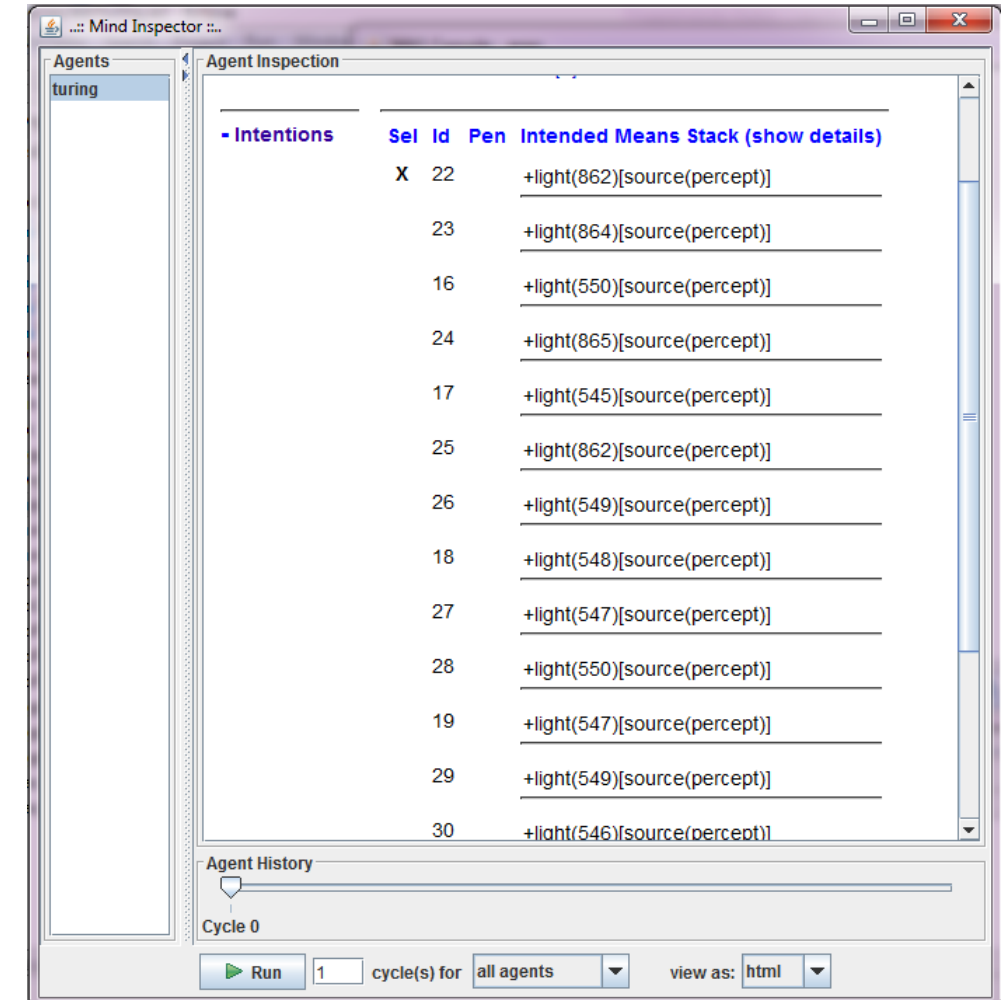
Argo for Jason: Cuidados

- **Ativando um efetuator**



```
!config.  
  
+!config: true <-  
  .print("Setting initial configuration...");  
  .port(COM5);  
  .percepts(open).  
  
+light(X): X>650 <-  
  .print("Turning on the light... it's getting dark, too dark to see!");  
  .act(lightOn).  
  
+light(X): X<=649 <-  
  .print("I can see clearly now...");  
  .act(lightOff).
```

Gera um alto número de eventos na pilha de intenções a ser tratado.



Exemplo: Argo



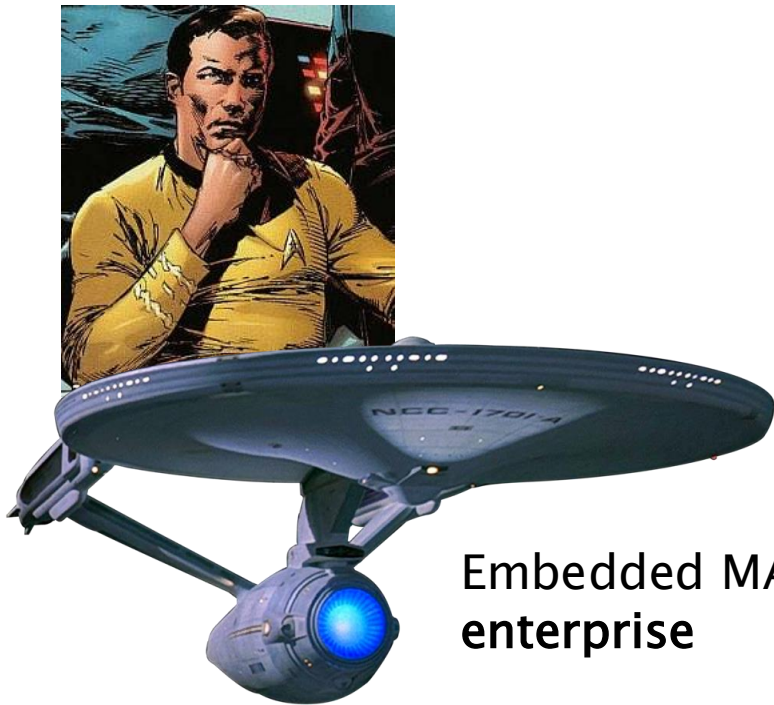
Exemplo: Argo



Embedded MAS
enterprise

Exemplo: Argo

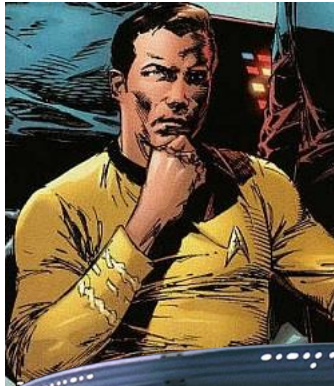
agent kirk



Embedded MAS
enterprise

Exemplo: Argo

agent kirk



agent sulu



Embedded MAS
enterprise

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);`

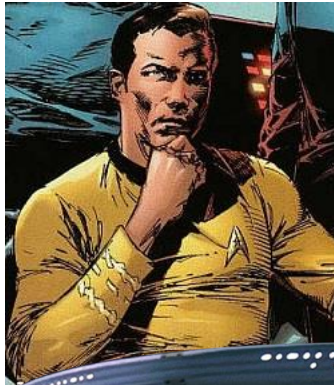


`.port(ttyACM0);`



Exemplo: Argo

`.send(sulu,
achieve, fire);`



`.port(ttyACM0);`

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



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.

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. ”

percepções disponíveis

- luminosity(468)
- buzzerStatus(on)
- buzzerStatus(off)

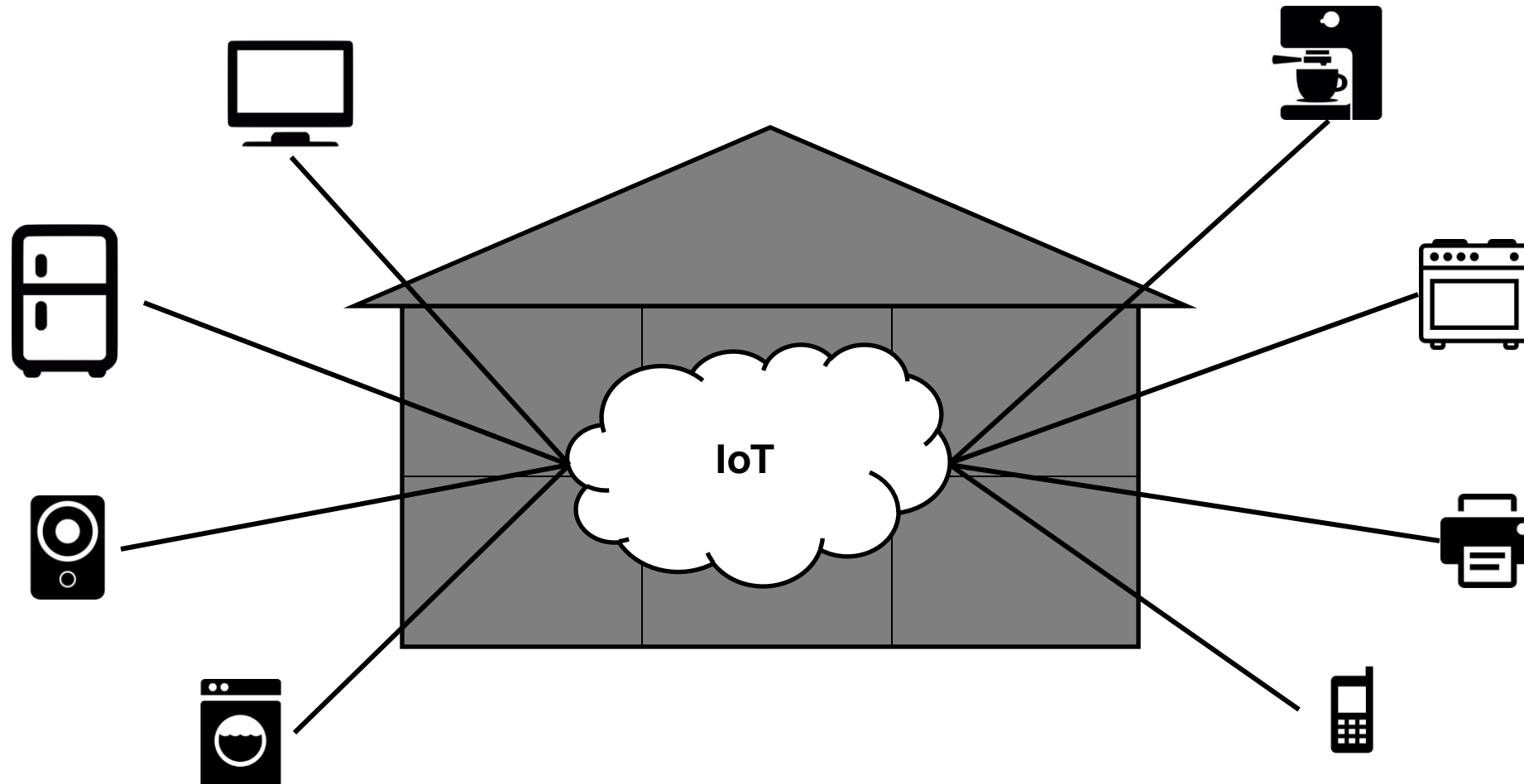
ações disponíveis

- buzzerOn
- buzzerOff

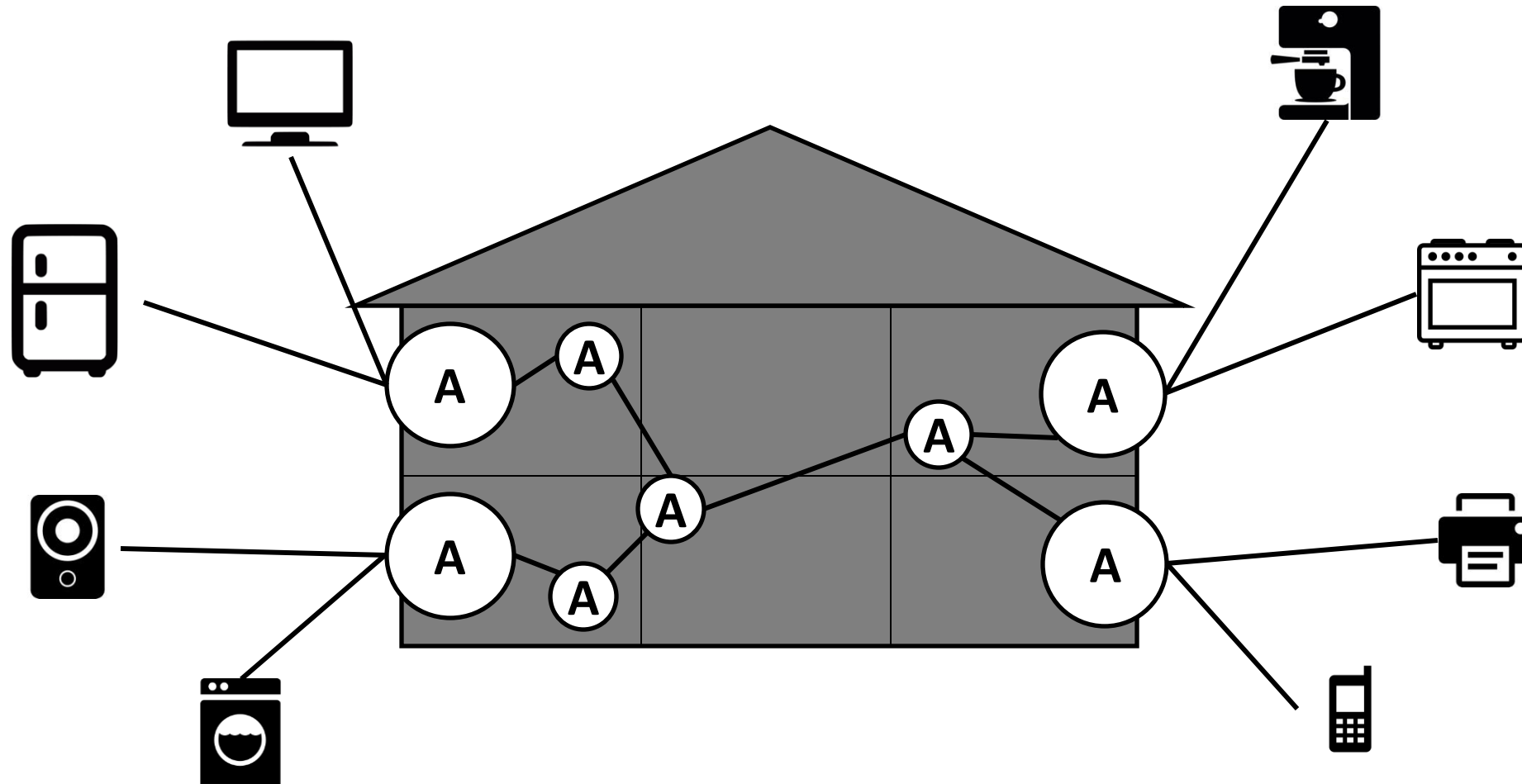
ATO 4:

OS COMUNICADORES

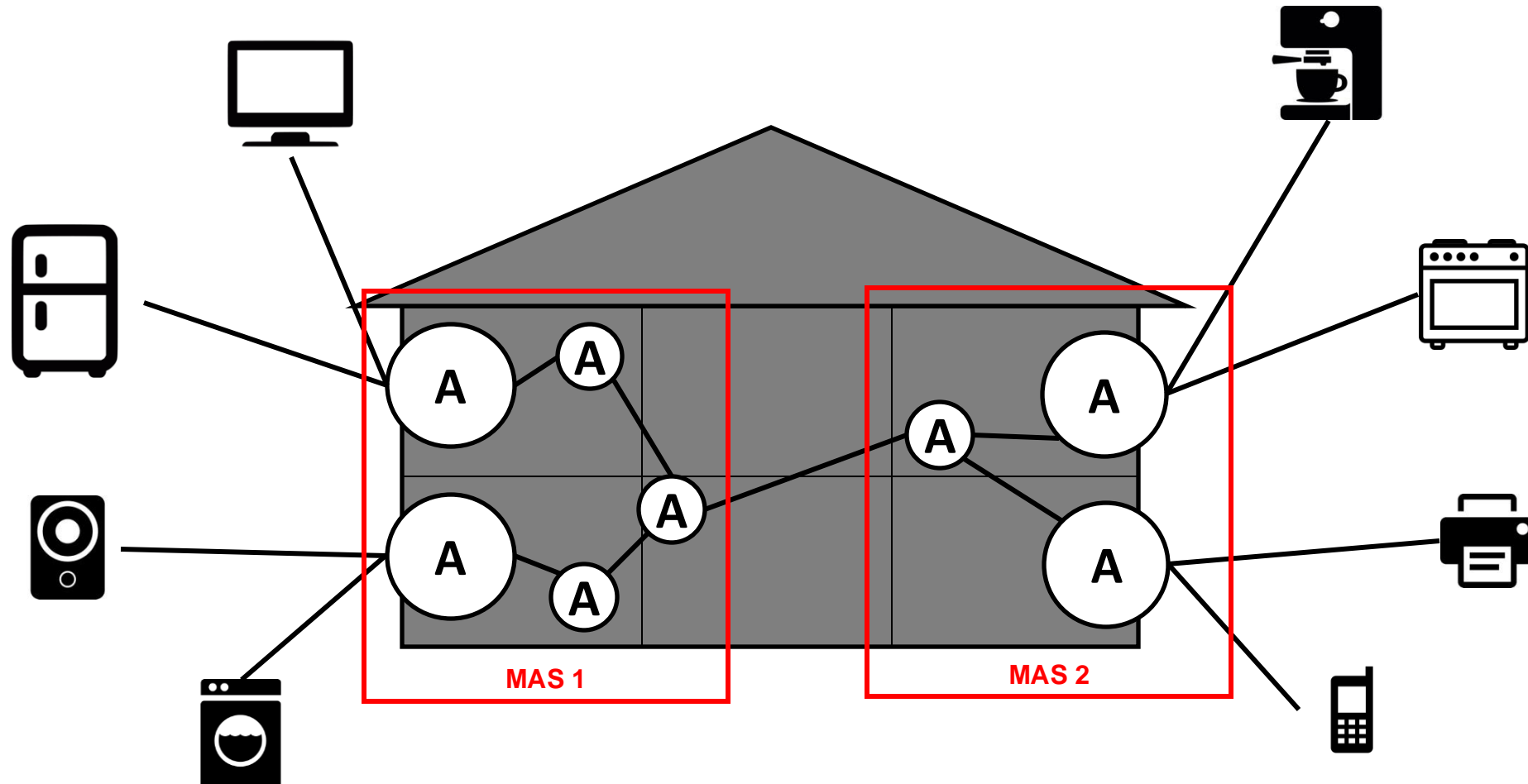
Agentes Comunicadores: IoT



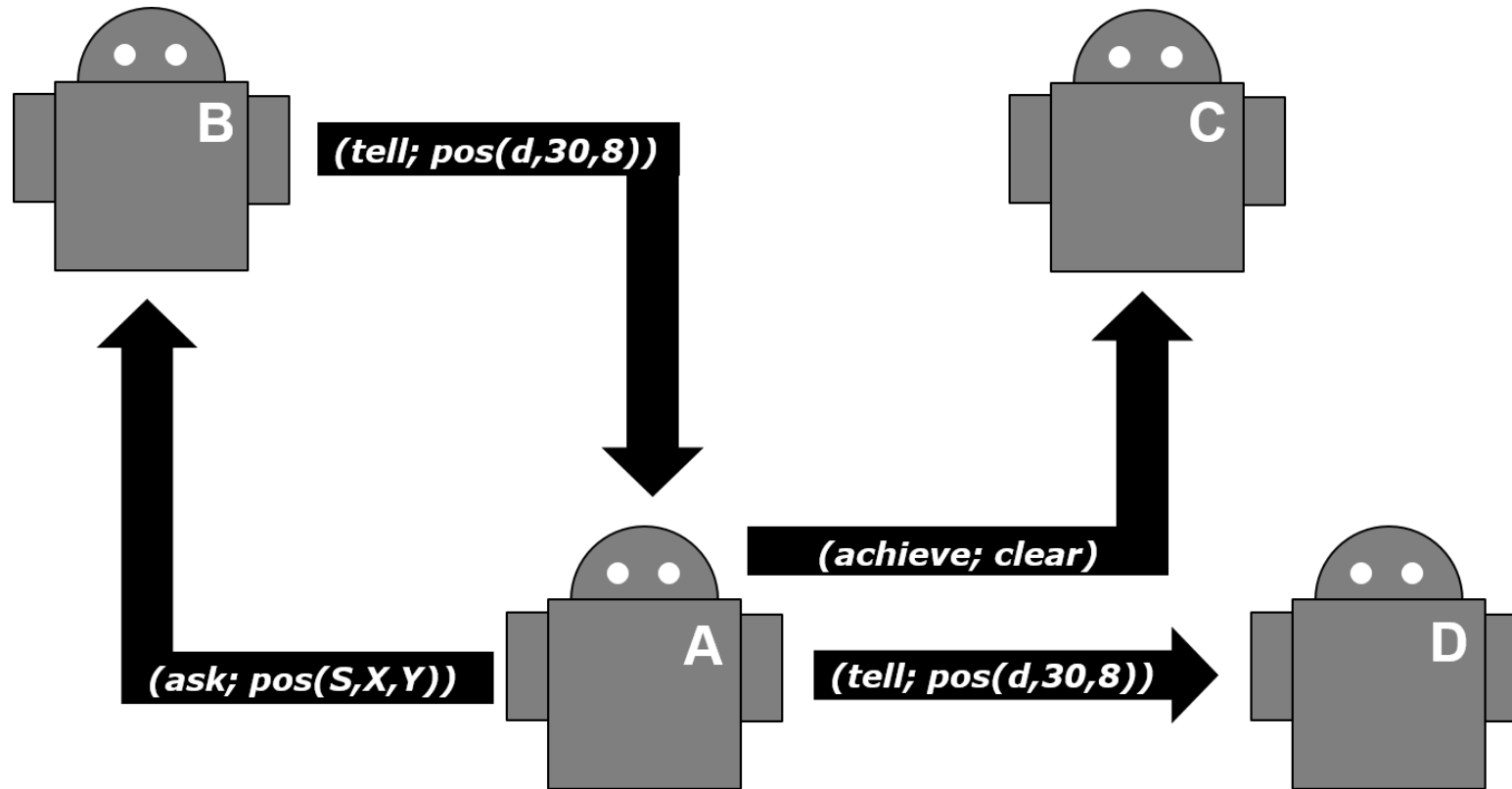
Agentes Comunicadores: IoT e SMA



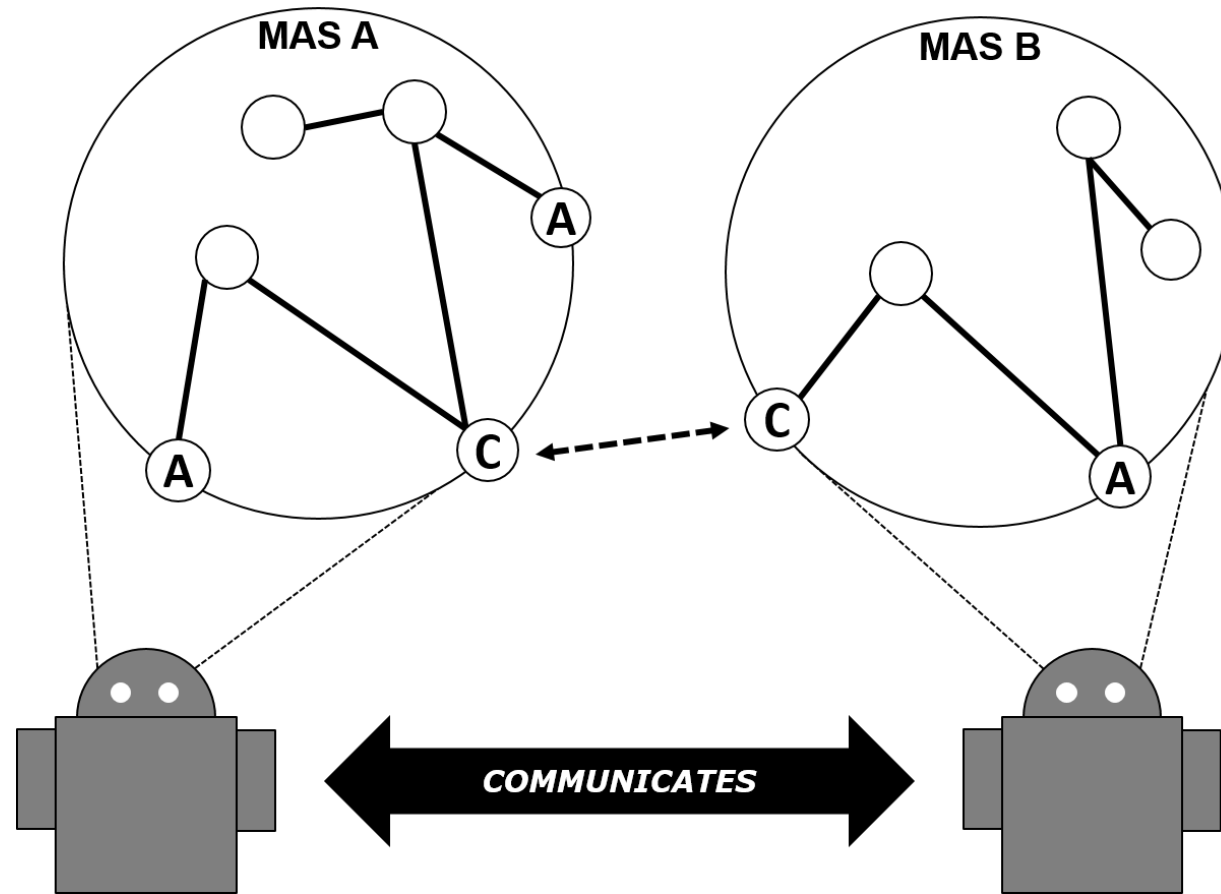
Agentes Comunicadores: IoT e SMA



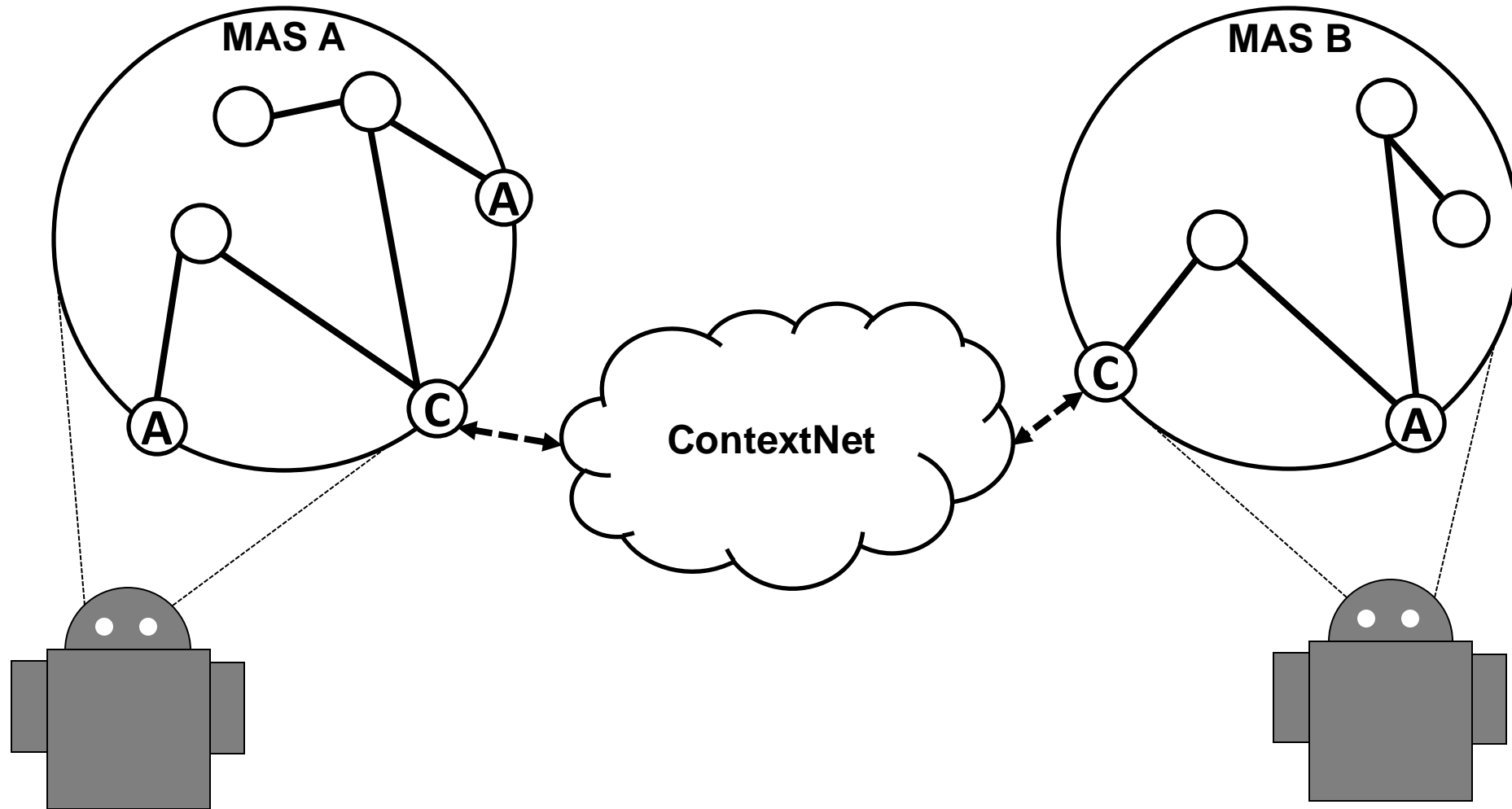
Agentes Comunicadores



Agentes Comunicadores



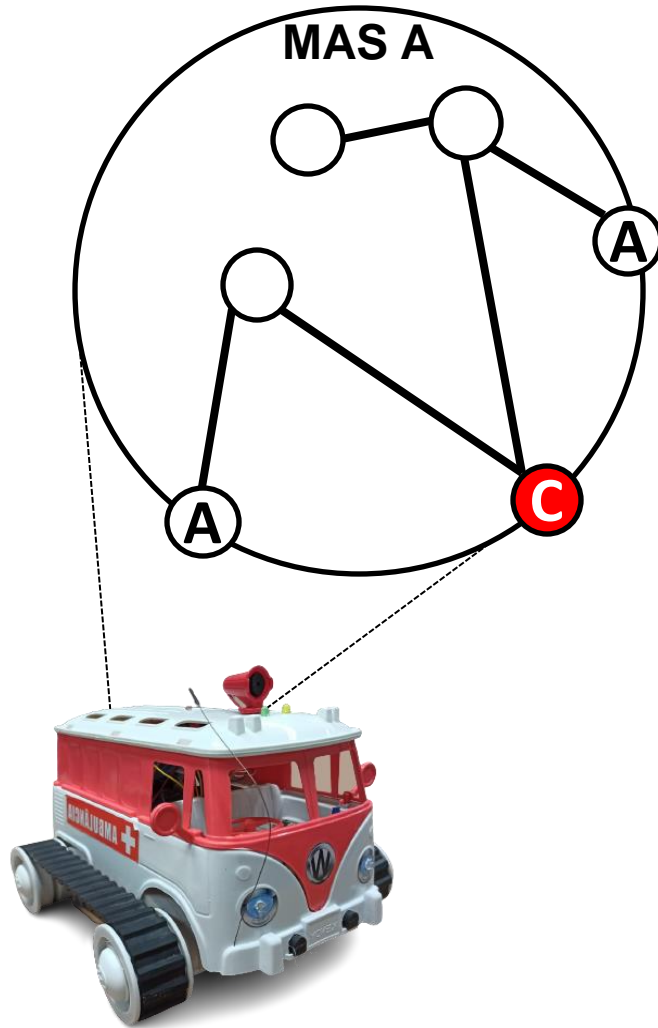
Agentes Comunicadores: Middleware IoT



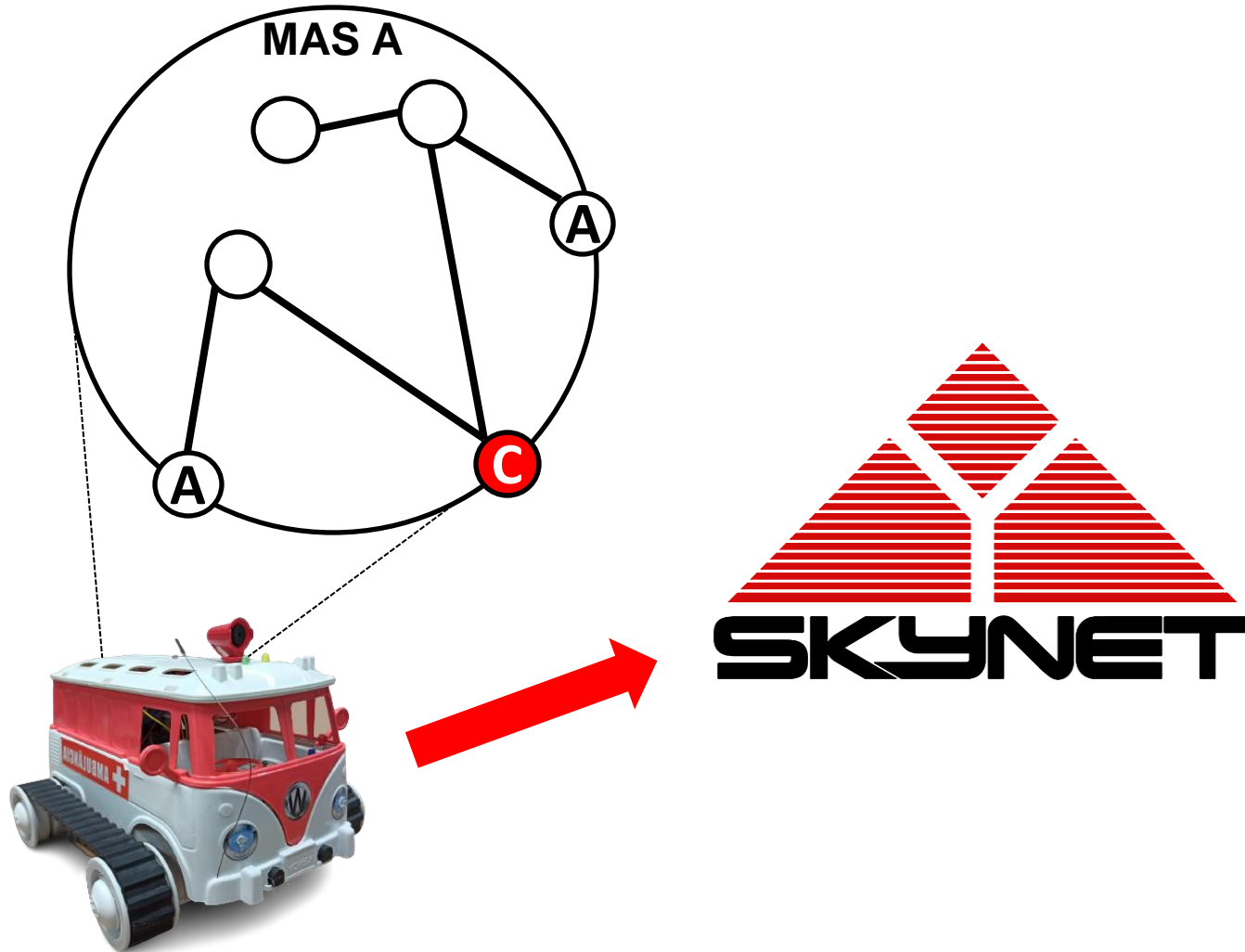
Agentes Comunicadores: Middleware IoT

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**.

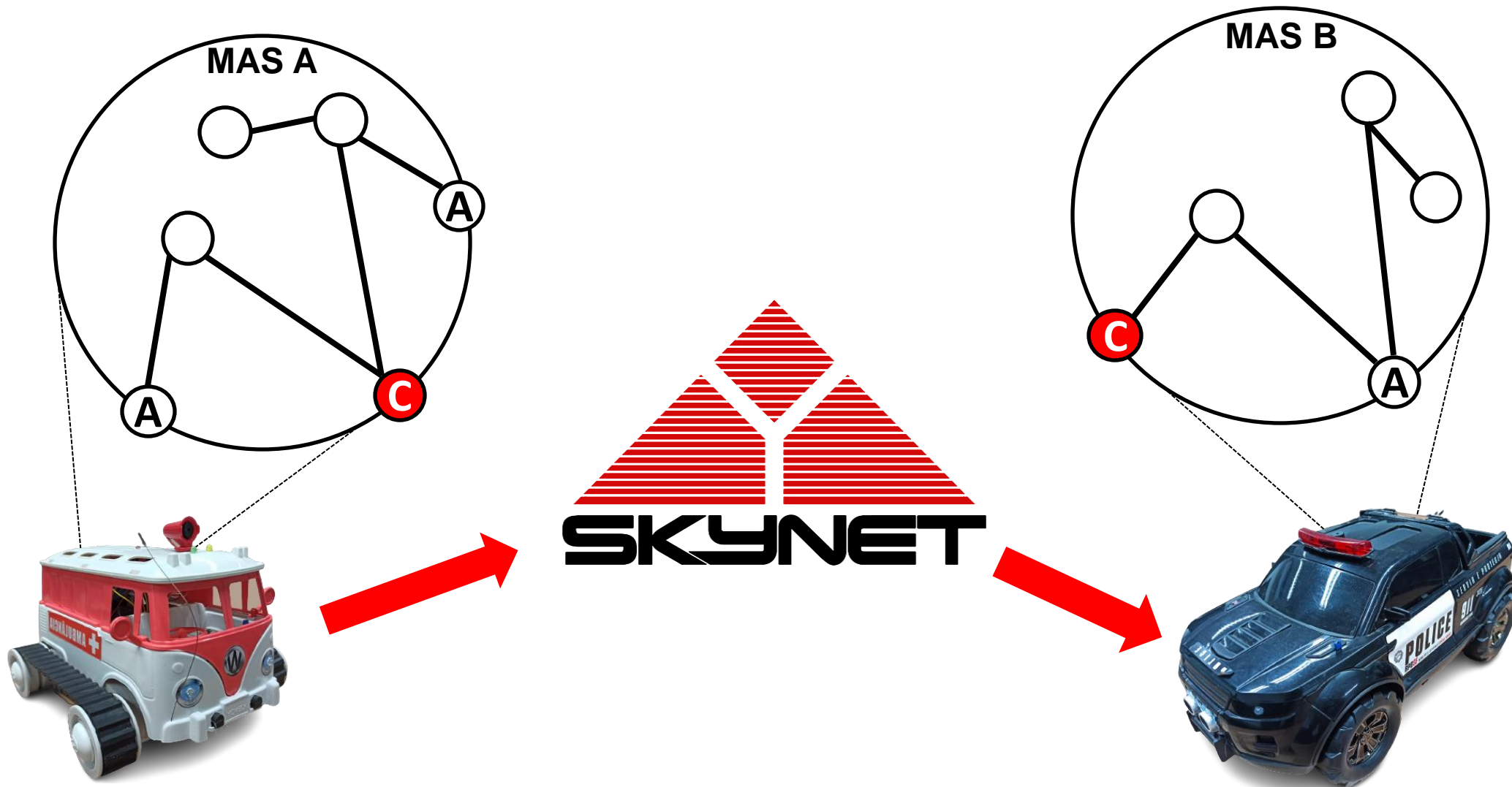
Agentes Comunicadores: Middleware IoT



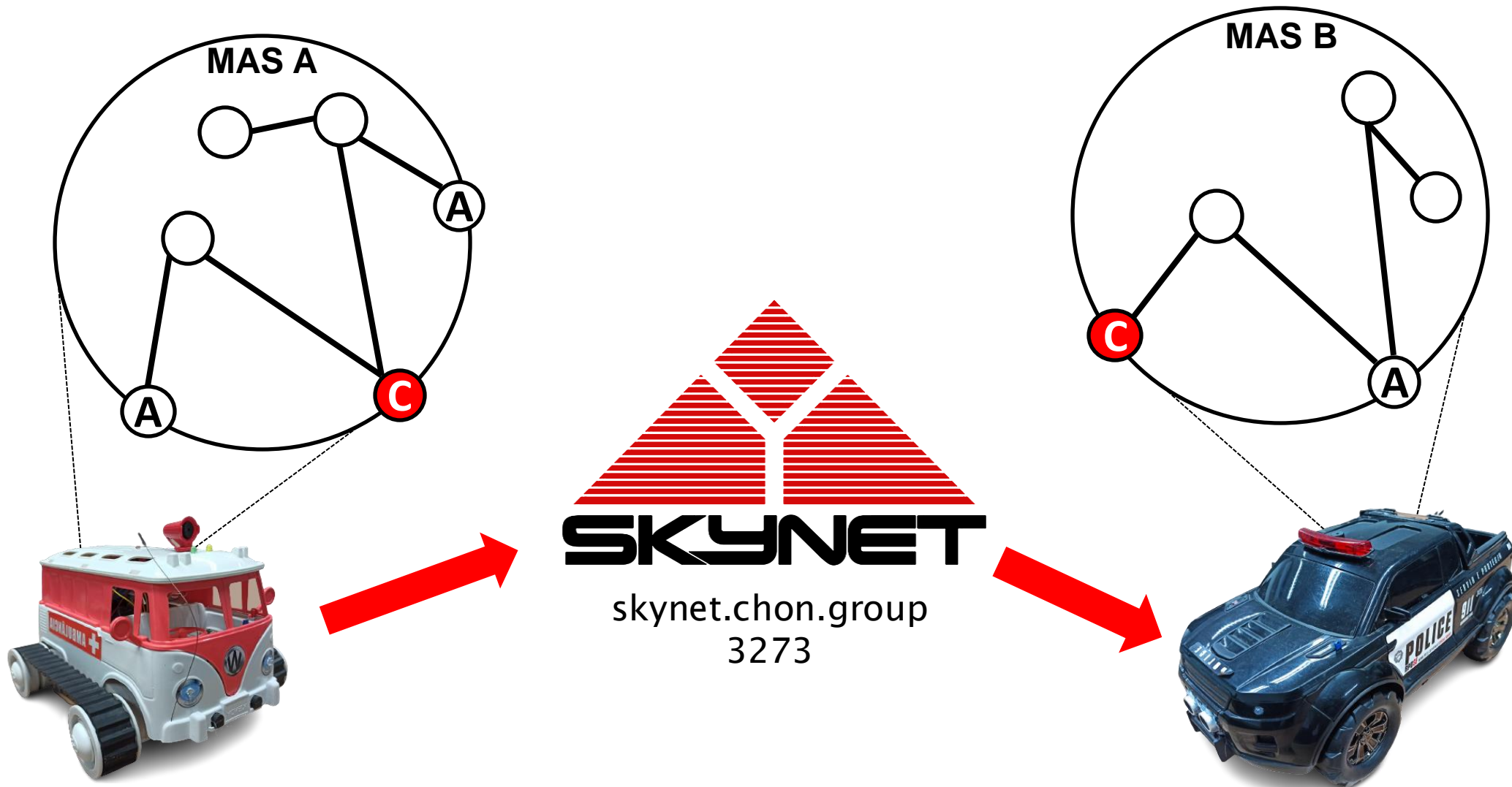
Agentes Comunicadores: Middleware IoT



Agentes Comunicadores: Middleware IoT



Agentes Comunicadores: Middleware IoT



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(ip, porta, agentUuid): 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

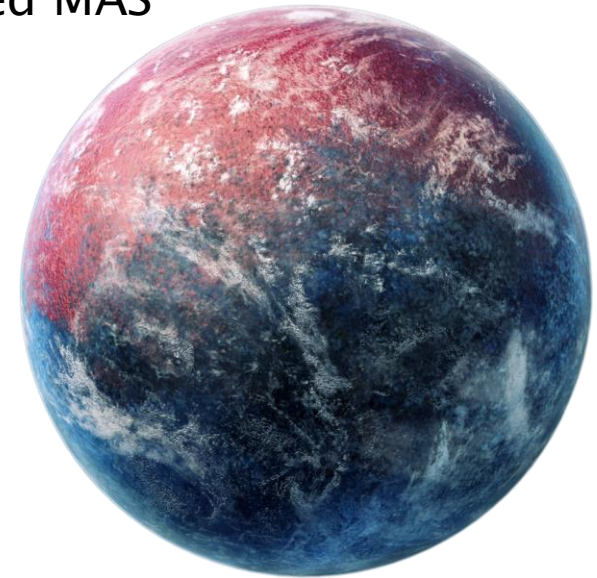


agent uhura



Embedded MAS
enterprise

Embedded MAS
bajor



Exemplo: Comunicação entre SMA Embarcados

agent scott

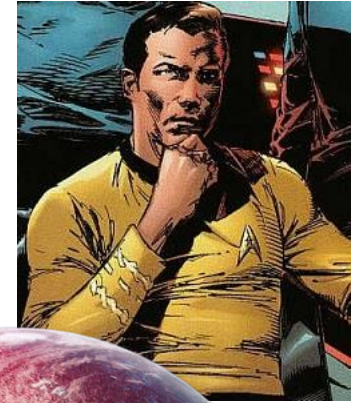


agent uhura



Embedded MAS
enterprise

agent kirk



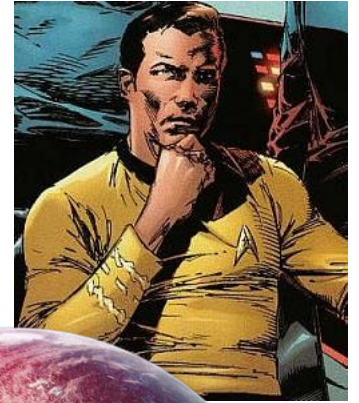
Embedded MAS
bajor



Exemplo: Comunicação entre SMA Embarcados



This is Lieutenant Nyota Uhura, Communications officer.

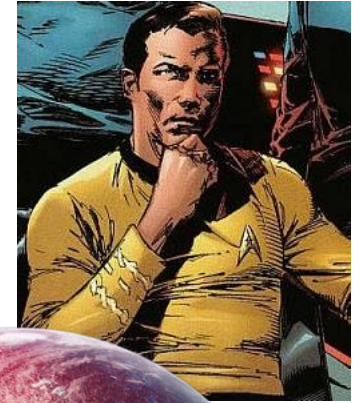


Exemplo: Comunicação entre SMA Embarcados



This is Lieutenant Nyota Uhura, Communications officer.

Kirk to Enterprise...
Damage report!



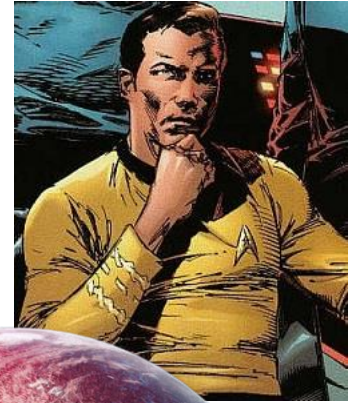
Exemplo: Comunicação entre SMA Embarcados



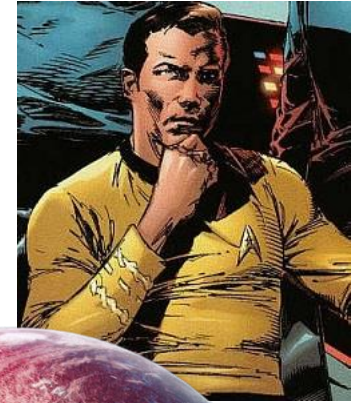
This is Lieutenant Nyota Uhura, Communications officer.

Kirk to Enterprise...
Damage report!

Commander Kirk,
Deck 2 compromised!

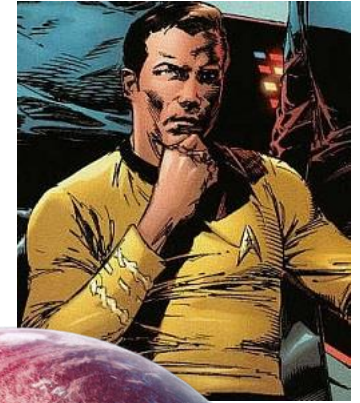


Exemplo: Comunicação entre SMA Embarcados



Exemplo: Comunicação entre SMA Embarcados

```
.connectCN("skynet.cho  
n.group",3273,"07b11c  
f3-92cb-47b4-b321-  
fb5051497150").
```

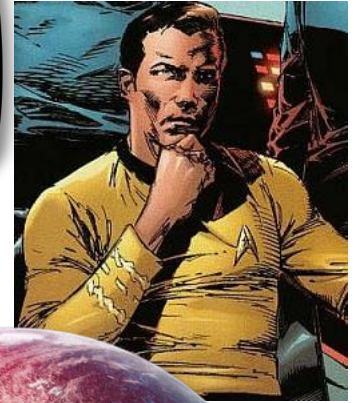


Exemplo: Comunicação entre SMA Embarcados



`.connectCN("skynet.cho
n.group",3273,"07b11c
f3-92cb-47b4-b321-
fb5051497150").`

`.sendOut(uhura,
achieve,
damageReport)`



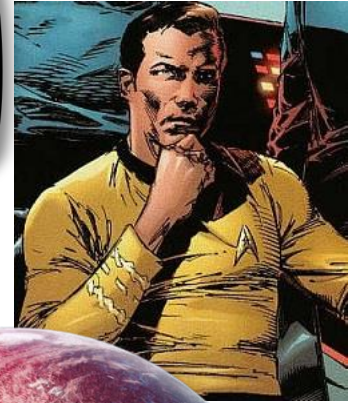
Exemplo: Comunicação entre SMA Embarcados



`.connectCN("skynet.cho
n.group",3273,"07b11c
f3-92cb-47b4-b321-
fb5051497150").`

`.sendOut(uhura,
achieve,
damageReport)`

`.sendOut(kirk, tell,
report(Damages))`



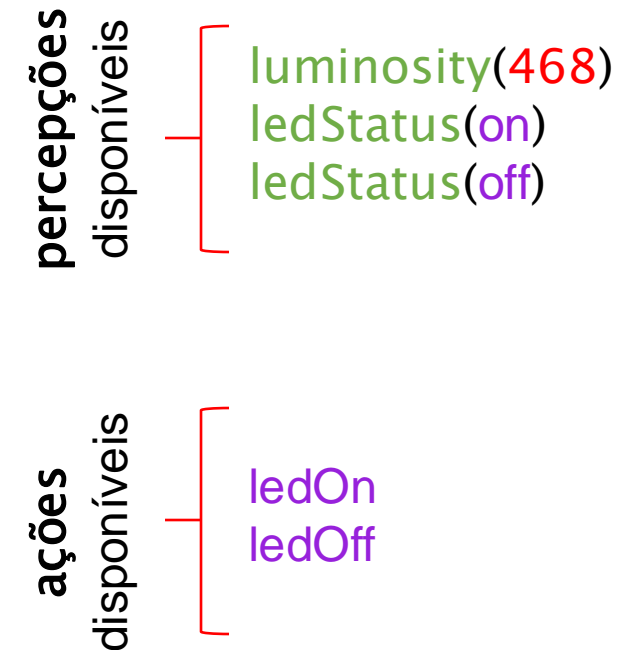
TAREFA SKYNET

“

1. Criar um SMA com um agente Argo e um Comunicador.
2. Quando o agente Argo perceber que está escuro ou claro, ele deve informar ao agente Comunicador.
3. Ao receber a informação, o agente Comunicador da Equipe A informará ao agente Comunicador da Equipe B.
4. Quando o agente Comunicador da Equipe B receber a informação, este deve informar ao agente Argo para acender ou apagar a luz do seu hardware.

Quem assumiu o papel da Equipe A deve complementar a implementação com o comportamento da Equipe B e vice-versa.

”



ATO 5: O PROTOCOLO ULTRON

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.

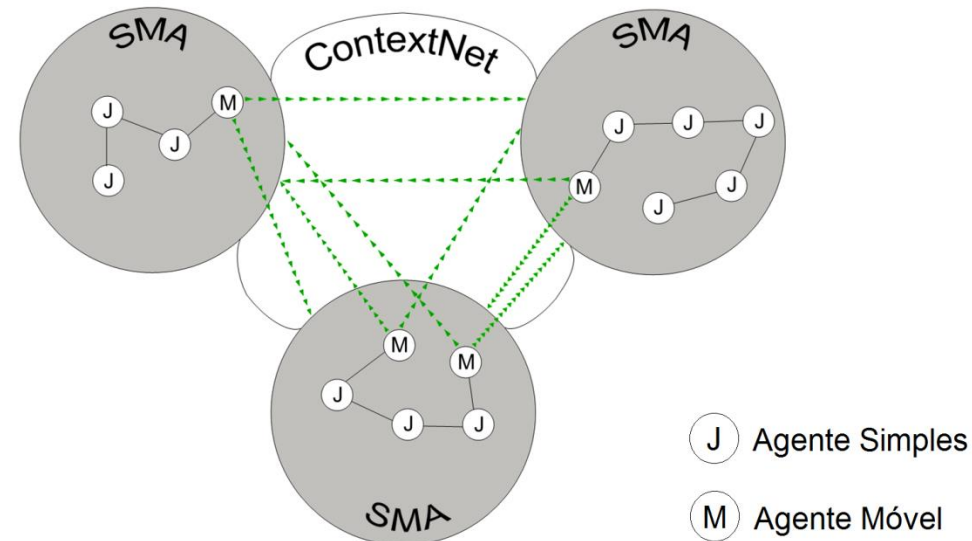


Figura 1. Transferência de agentes móveis

Protocolo de Transferência de Agentes

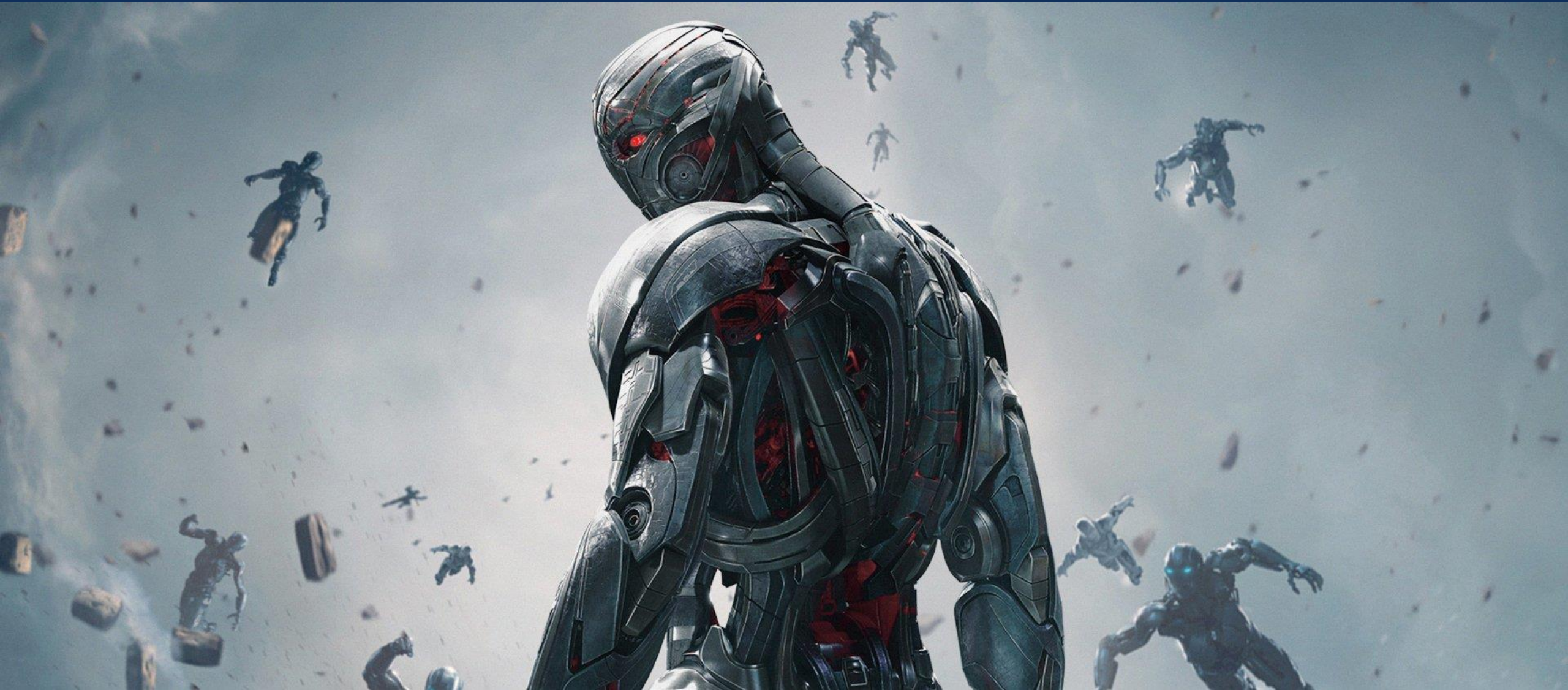
Um **agente cognitivo** está situado em um **SMA** qualquer e ambientado em um **dispositivo**, este agente fica “preso” ao SMA e ao dispositivo, e caso o dispositivo seja danificado o agente cognitivo **não consegue se transferir** para outro SMA.

Protocolo de Transferência de Agentes

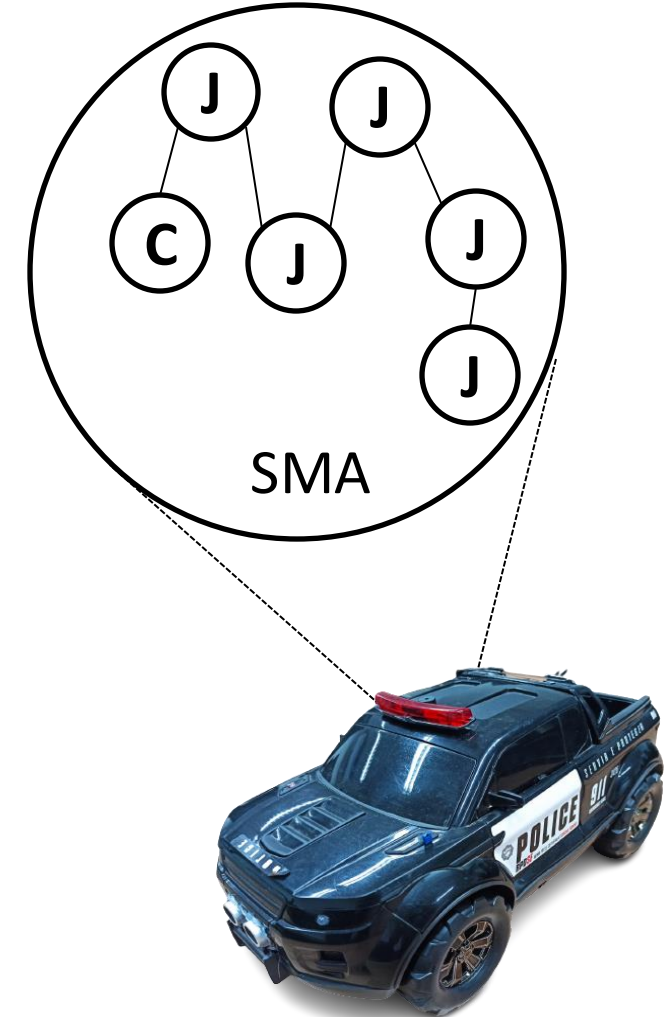
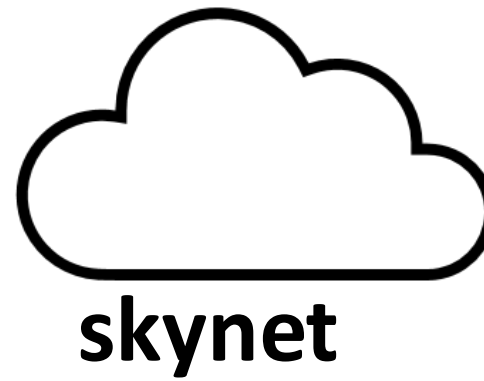
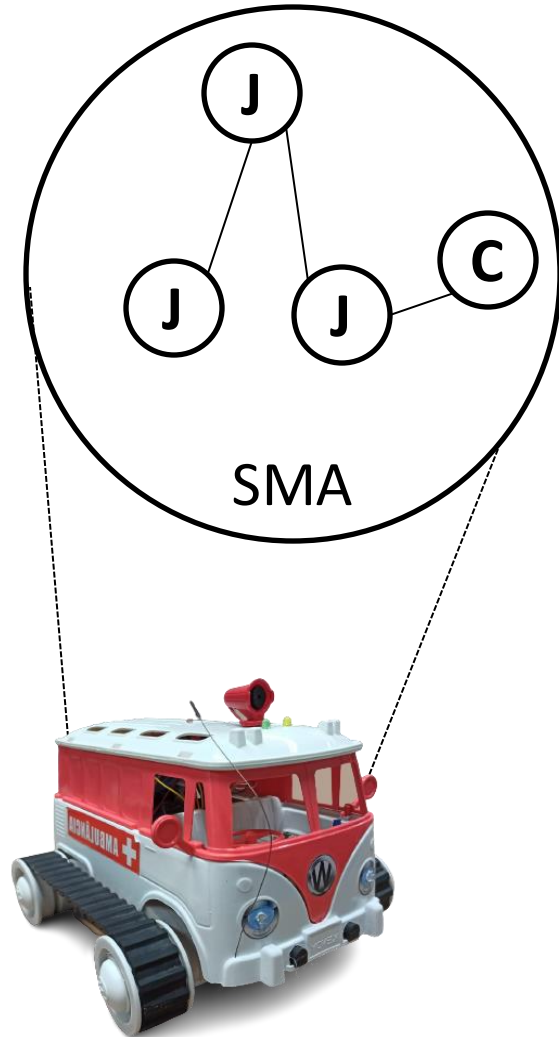
O protocolo de transferência de agentes prevê três possíveis relações entre o agente móvel com o novo SMA.

- Mutualismo
- Inquilinismo
- Predatismo

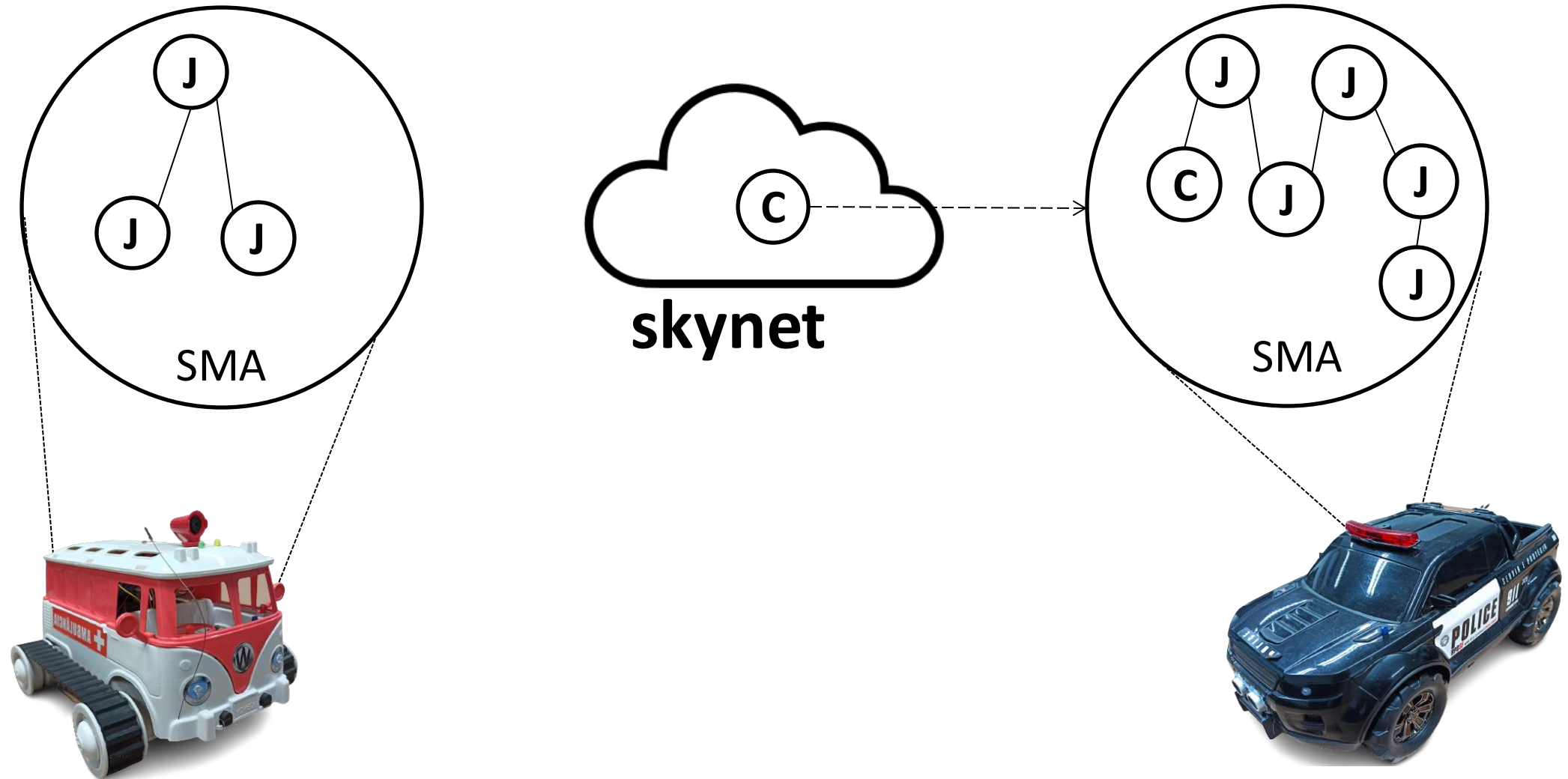
Bio-Inspired: Protocolo Ultron



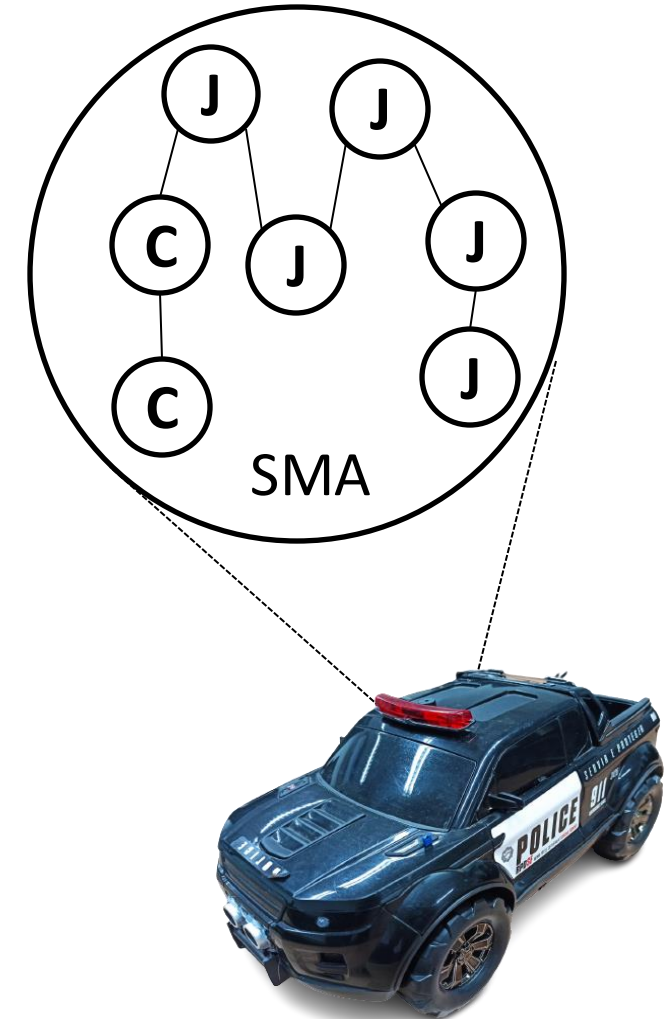
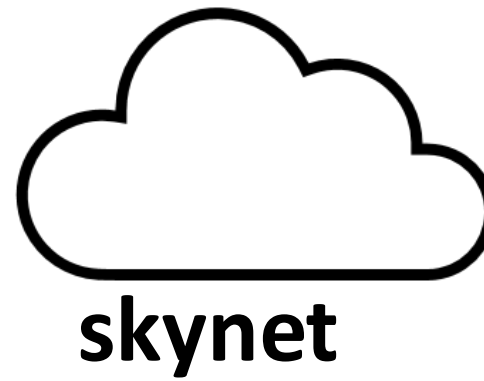
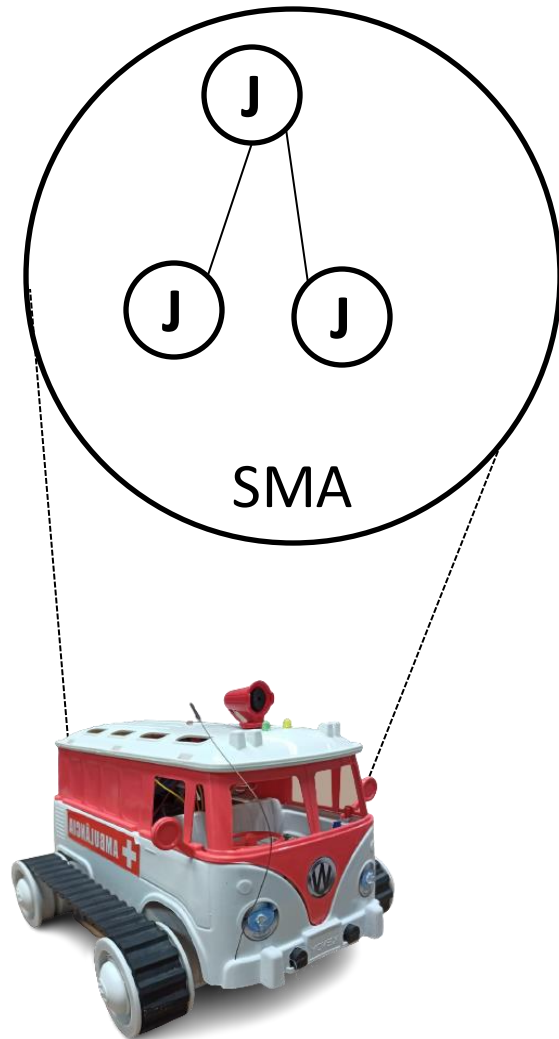
Bio-Inspired: Mutualismo



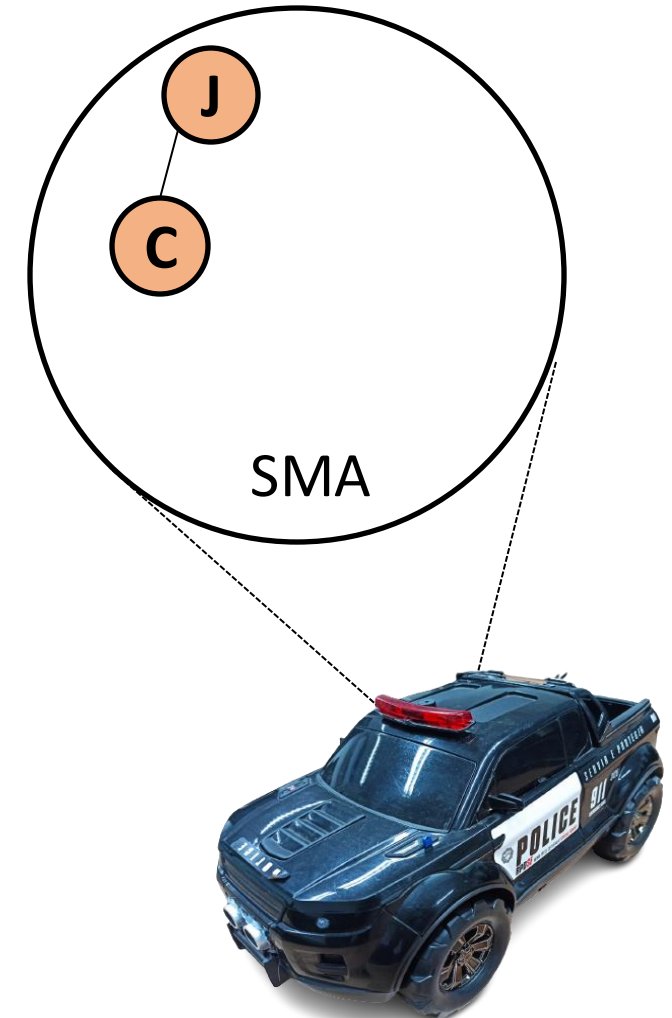
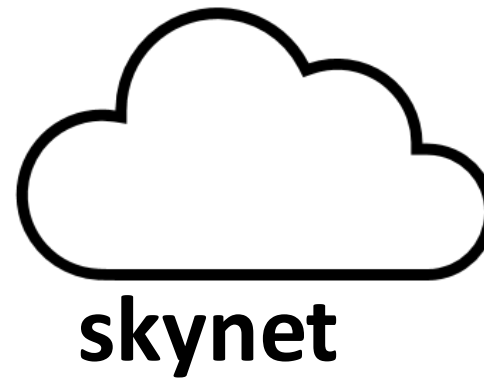
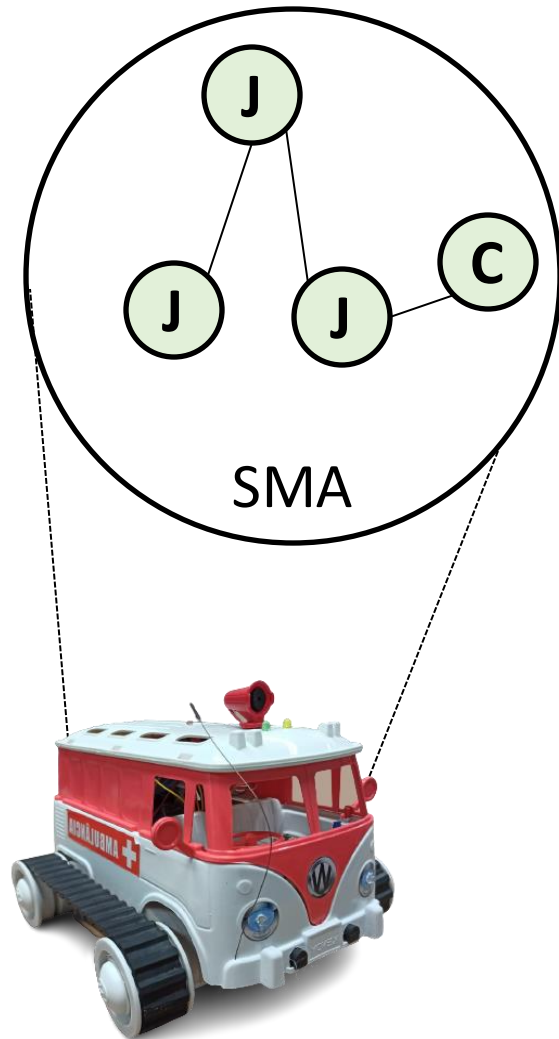
Bio-Inspired: Mutualismo



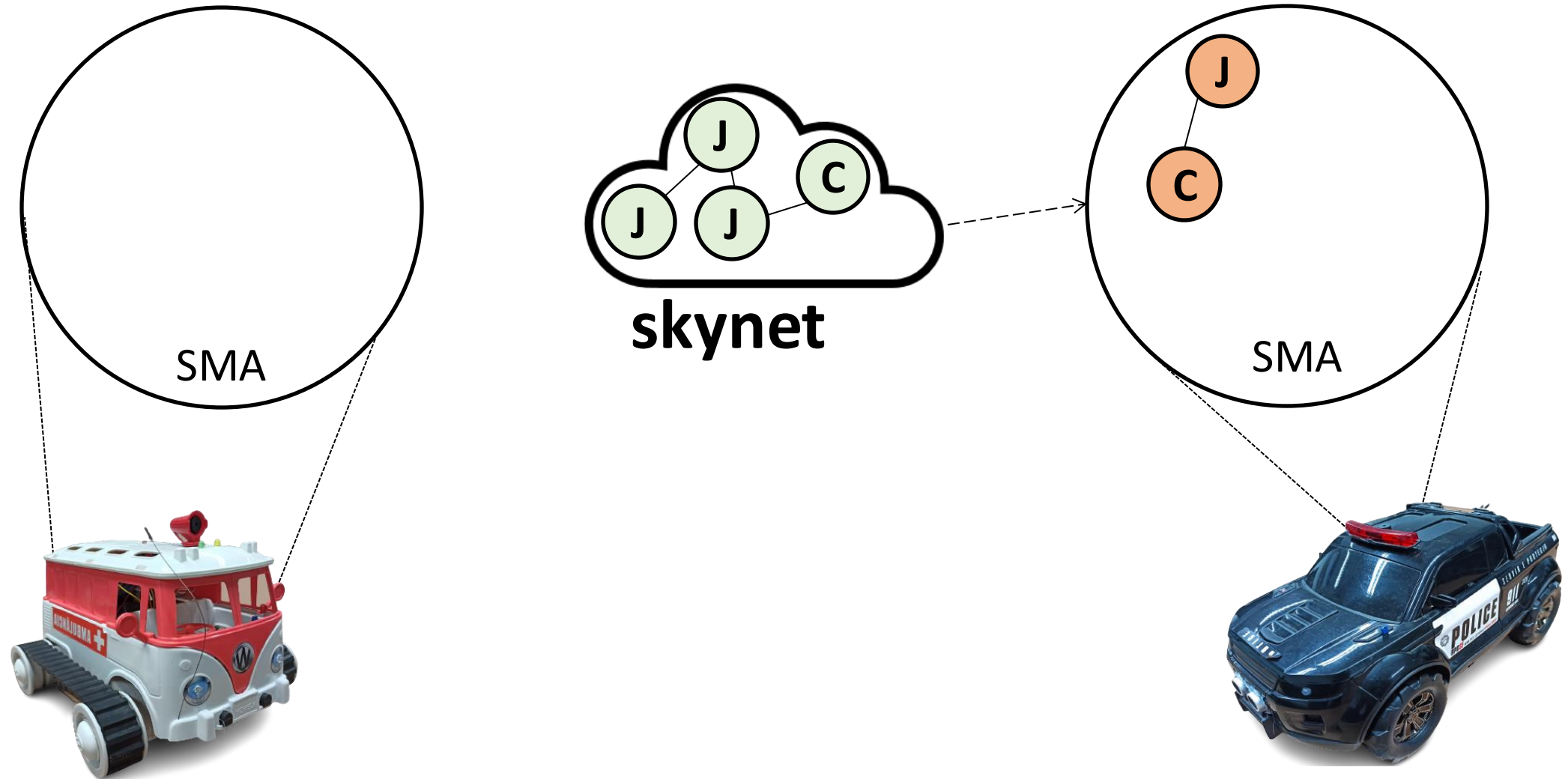
Bio-Inspired: Mutualismo



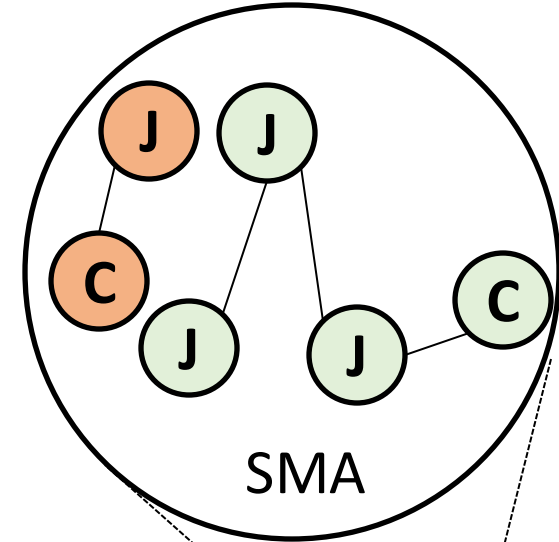
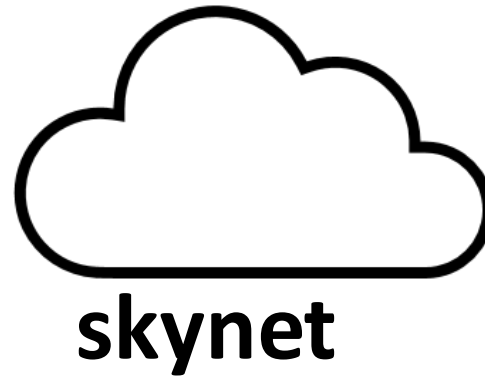
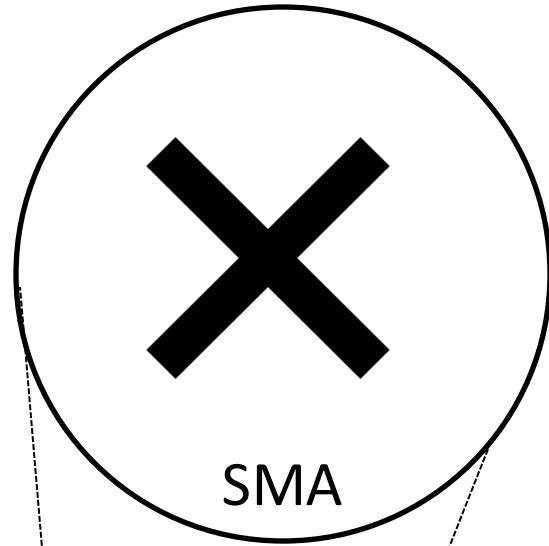
Bio-Inspired: Inquilinismo



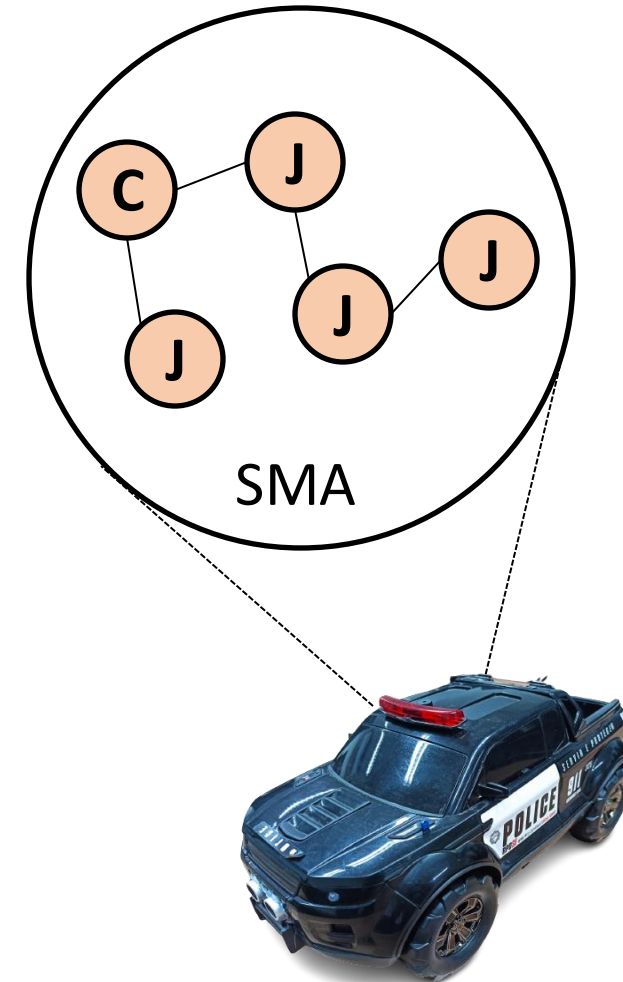
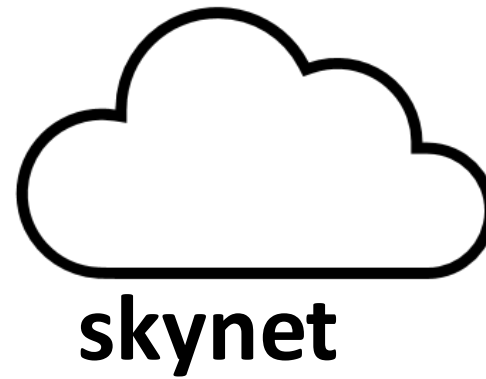
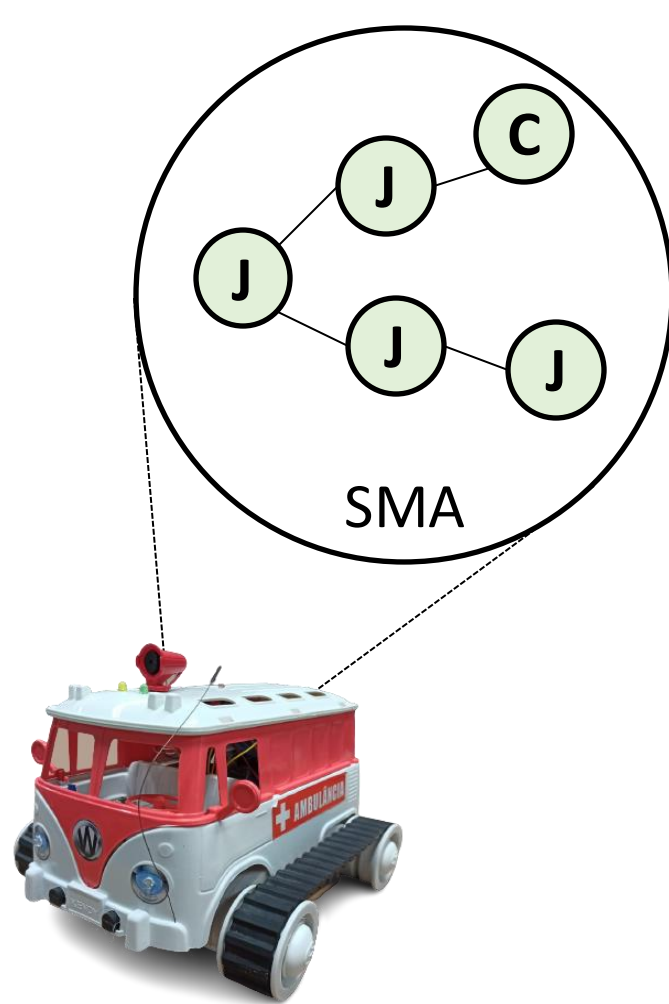
Bio-Inspired: Inquilinismo



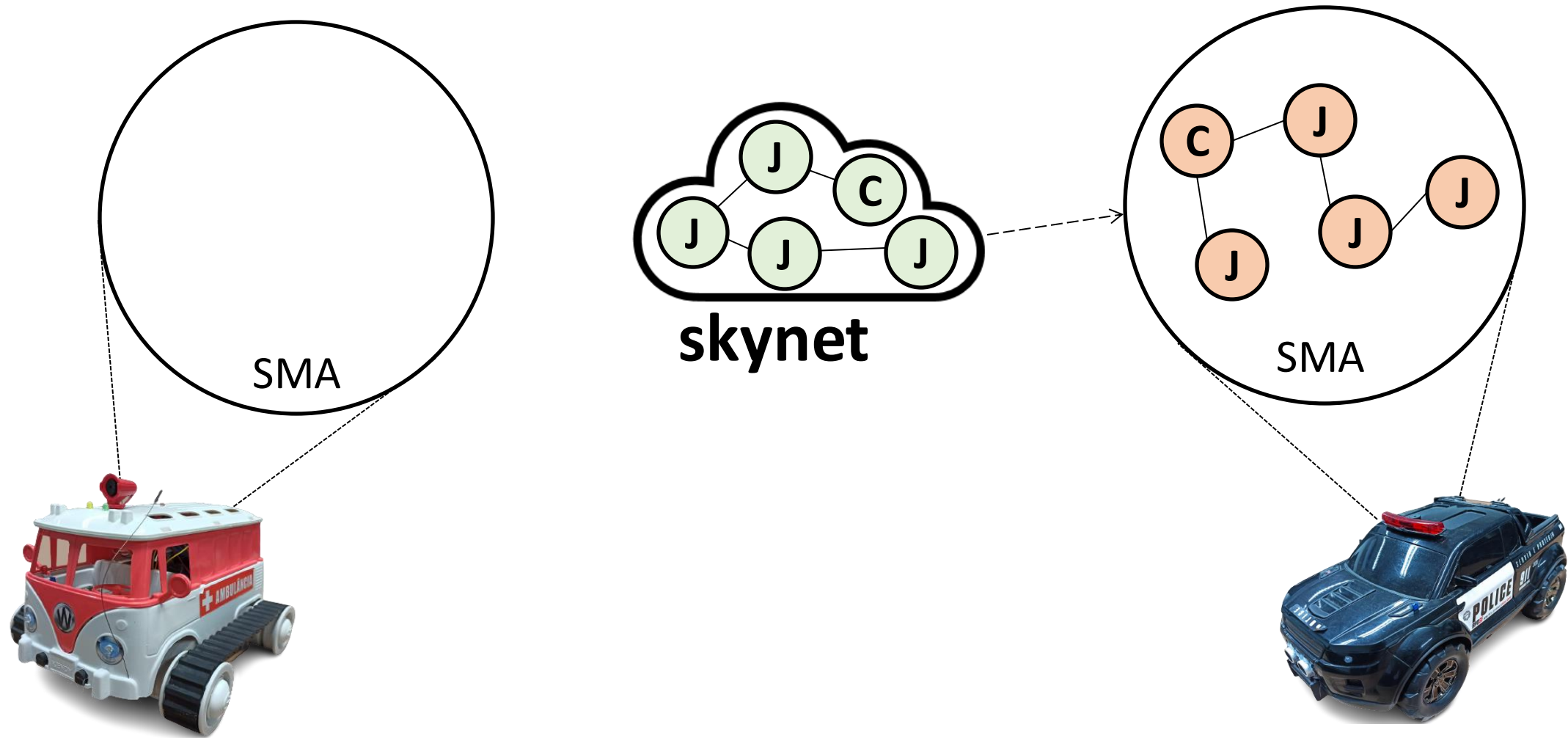
Bio-Inspired: Inquilinismo



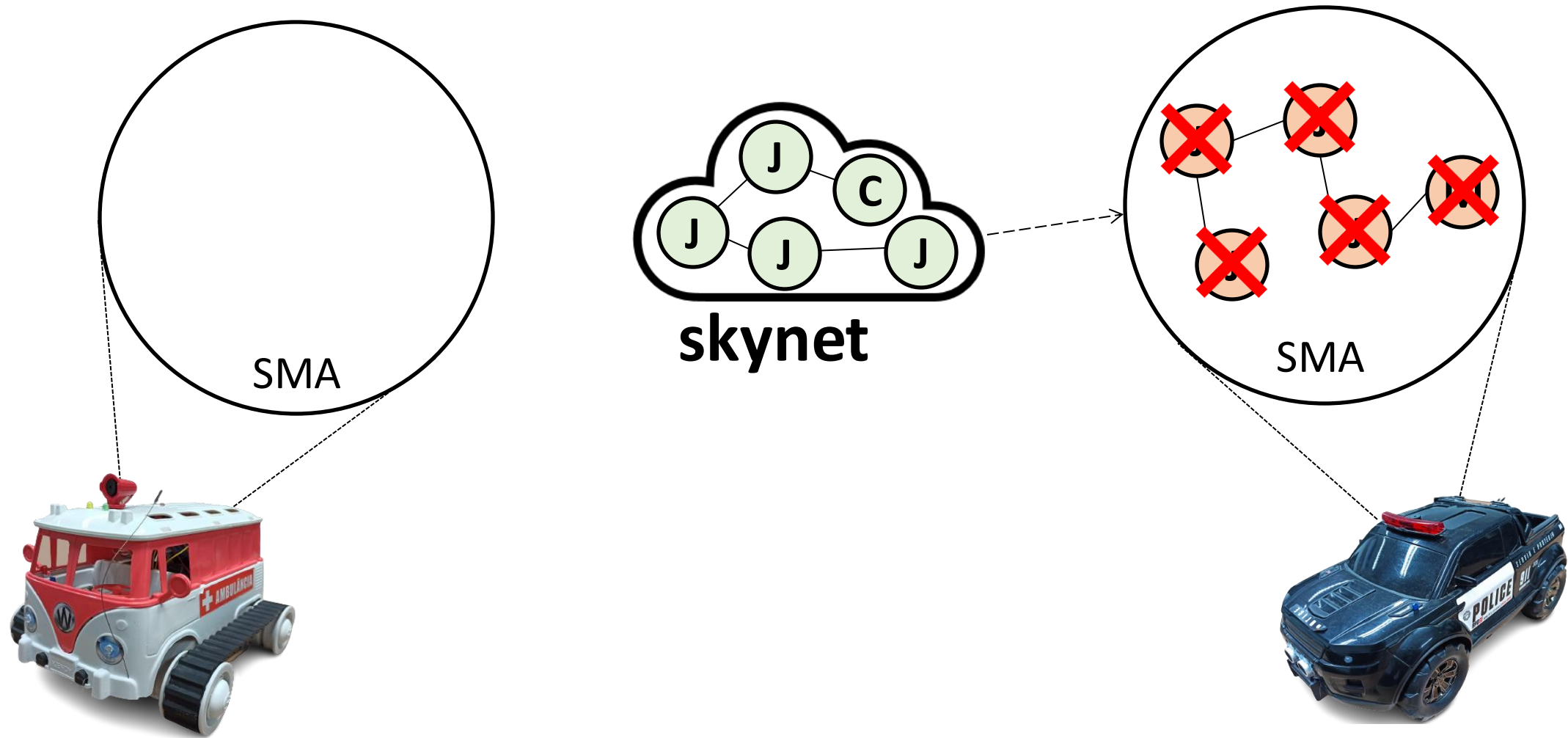
Bio-Inspired: Predatismo



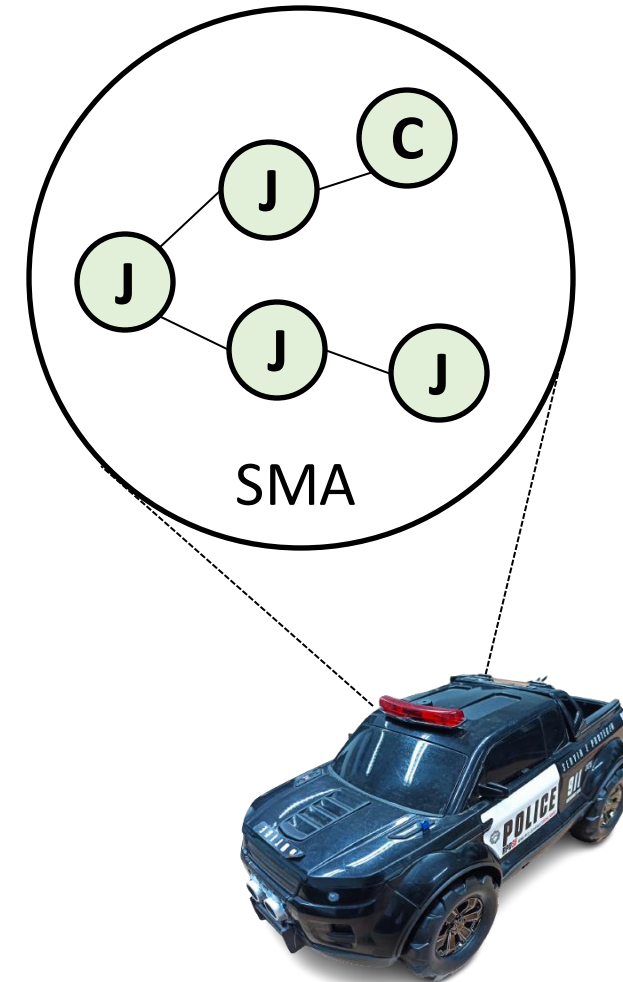
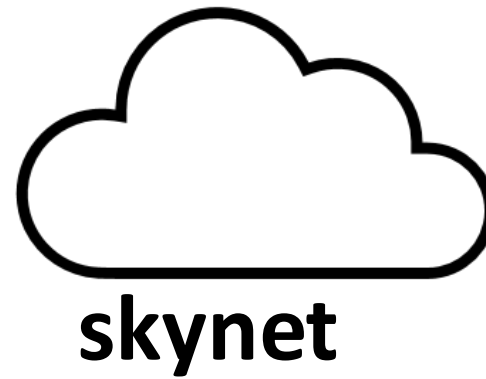
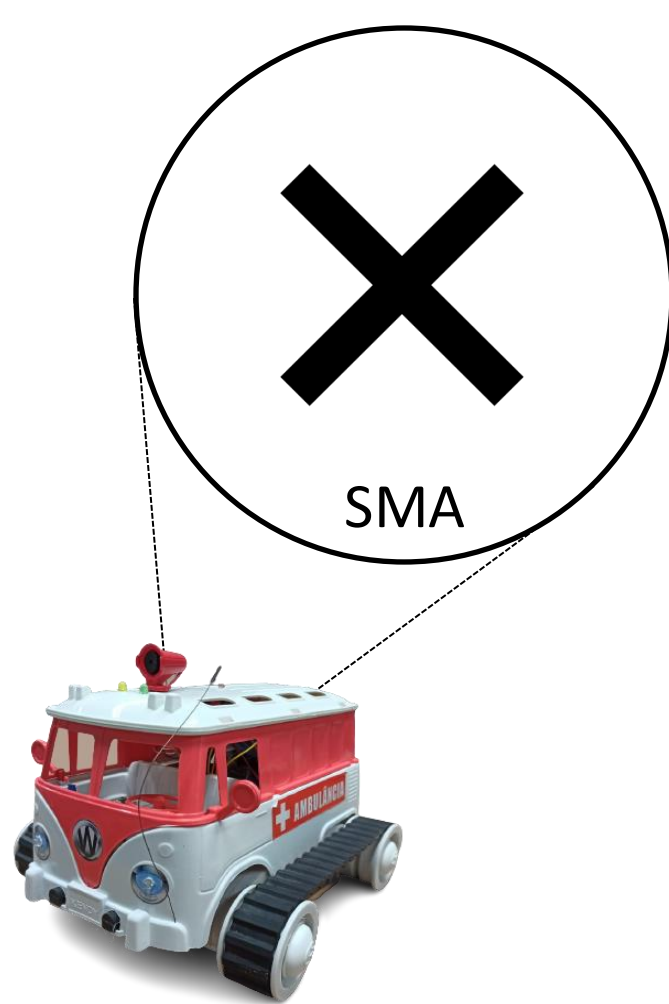
Bio-Inspired: Predatismo



Bio-Inspired: Predatismo



Bio-Inspired: Predatismo



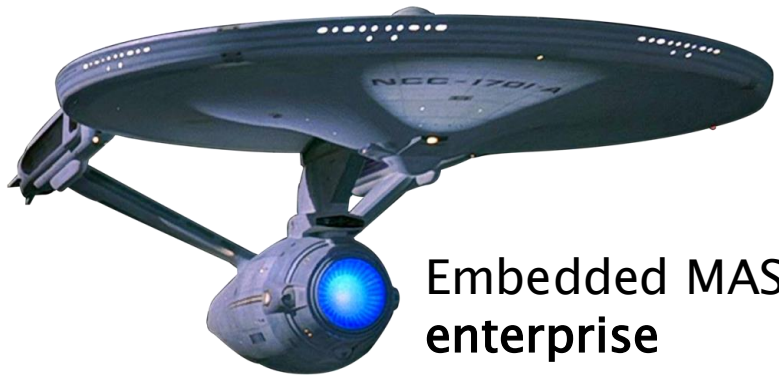
Communicator: Ações Internas

- **Communicator** Internal Actions:
 - .moveOut(agentUuid, predation|inquilinism): **envia uma mensagem de um agente comunicador para outro comunicador de um SMA distinto.**
 - .moveOut(agentUuid, mutualism, agent): **envia uma agente específico para outro comunicador de um SMA distinto.**

Exemplo: Bio-Inspired Protocol



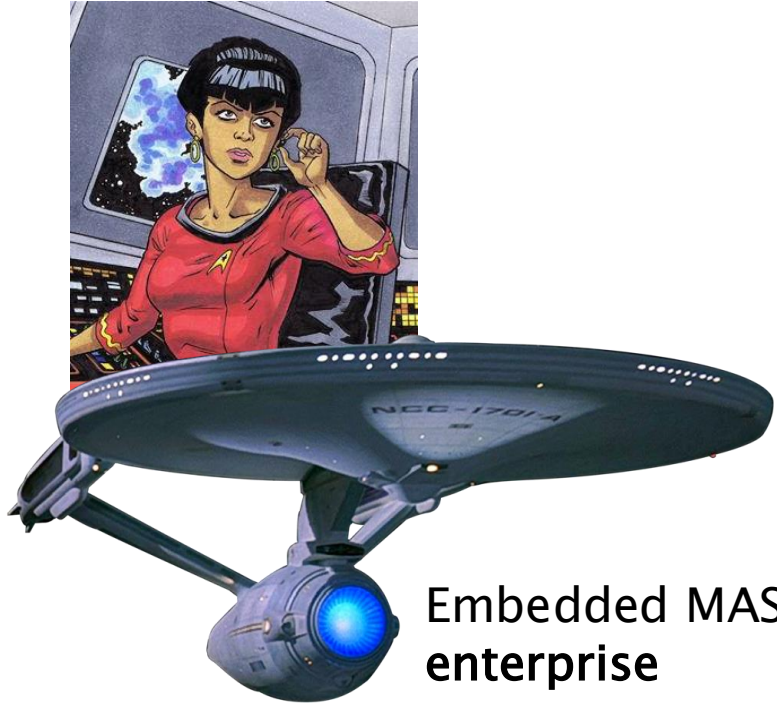
Exemplo: Bio-Inspired Protocol



Embedded MAS
enterprise

Exemplo: Bio-Inspired Protocol

agent uhura



Embedded MAS
enterprise

Exemplo: Bio-Inspired Protocol

agent uhura



agent scott



Embedded MAS
enterprise

Exemplo: Bio-Inspired Protocol

agent uhura

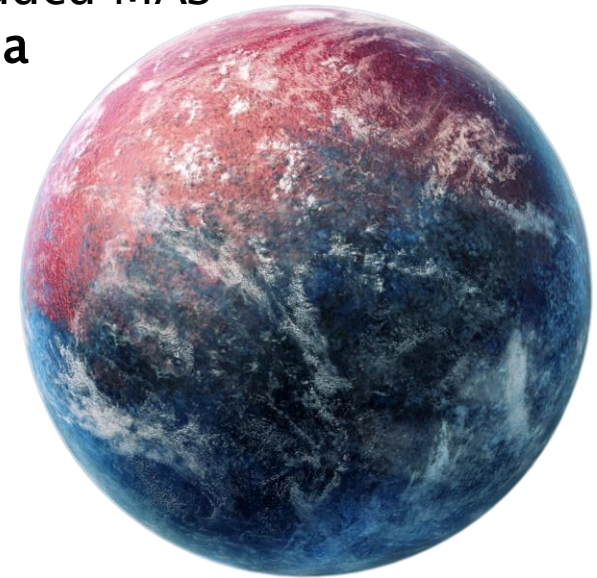


agent scott



Embedded MAS
enterprise

Embedded MAS
andoria



Exemplo: Bio-Inspired Protocol

agent uhura

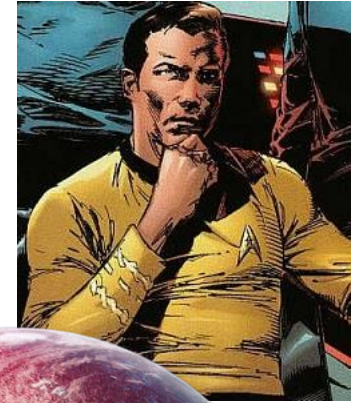


agent scott



Embedded MAS
enterprise

agent kirk



Embedded MAS
andoria



Exemplo: Bio-Inspired Protocol

agent uhura

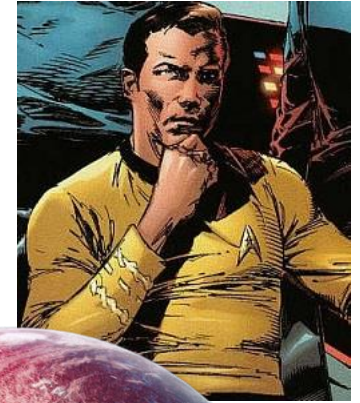


agent scott



Embedded MAS
enterprise

agent kirk



Embedded MAS
andoria



agent spok

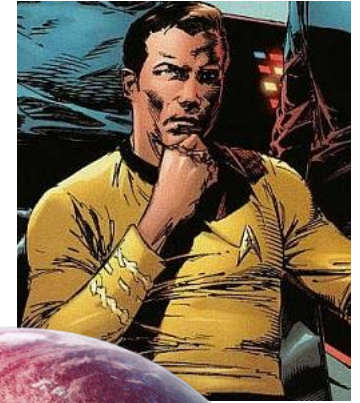


Exemplo: Bio-Inspired Protocol



Exemplo: Bio-Inspired Protocol

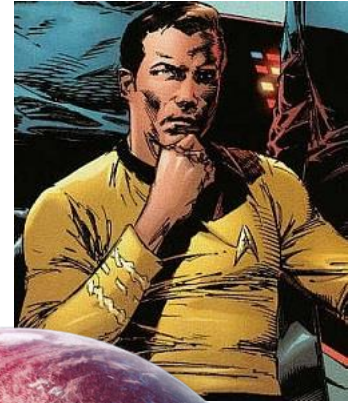
Computer,
Commander
Montgomery Scott,
Chief Engineering
Office!



Exemplo: Bio-Inspired Protocol

Computer,
Commander
Montgomery Scott,
Chief Engineering
Office!

Kirk to Scotty...
Beam us up!



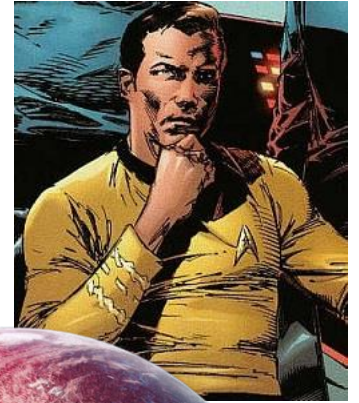
Exemplo: Bio-Inspired Protocol



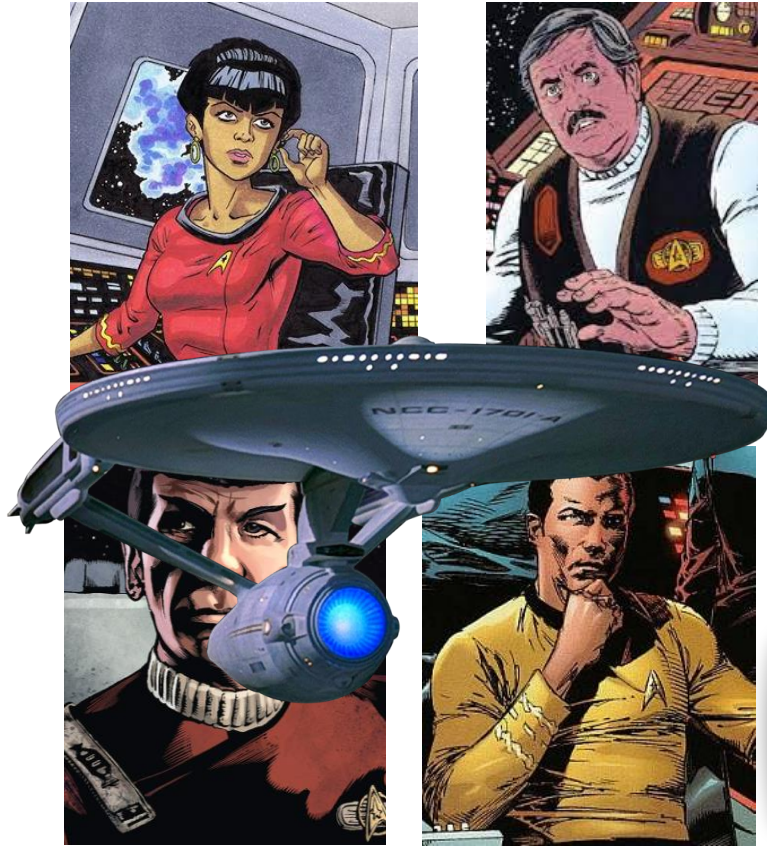
Computer,
Commander
Montgomery Scott,
Chief Engineering
Office!

Transporter ready for
Kirk and Spok.

Kirk to Scotty...
Beam us up!



Exemplo: Bio-Inspired Protocol

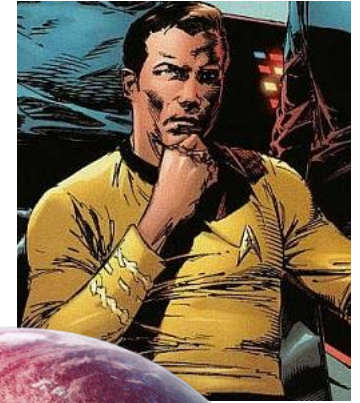


Computer, Commander
James T. Kirk, Enterprise's
Captain! I'm aboard!



Exemplo: Bio-Inspired Protocol

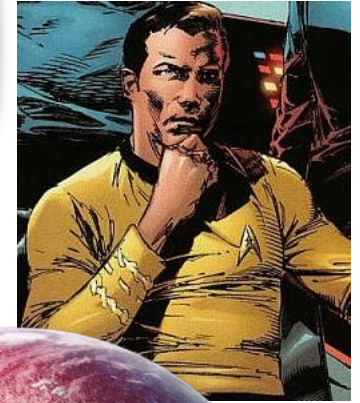
```
.connectCN("skynet.cho  
n.group",3273,"07ba9e  
4a-d539-4a0e-8c14-  
4ac336476858").
```



Exemplo: Bio-Inspired Protocol

```
.connectCN("skynet.cho  
n.group",3273,"07ba9e  
4a-d539-4a0e-8c14-  
4ac336476858").
```

```
.sendOut(scott,  
tell,  
beam_us_up)
```



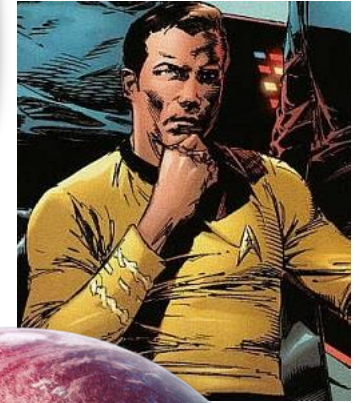
Exemplo: Bio-Inspired Protocol



```
.connectCN("skynet.cho  
n.group",3273,"07ba9e  
4a-d539-4a0e-8c14-  
4ac336476858").
```

```
.sendOut(scott,  
tell,  
beam_us_up)
```

```
.sendOut(kirk,tell,  
energizing);
```



Exemplo: Bio-Inspired Protocol

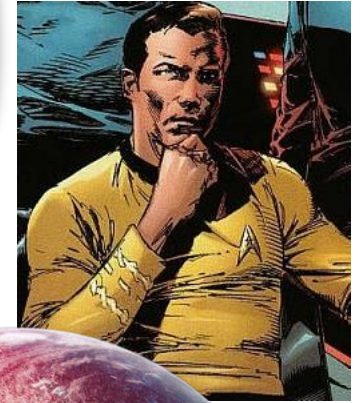


```
.connectCN("skynet.cho  
n.group",3273,"07ba9e  
4a-d539-4a0e-8c14-  
4ac336476858").
```

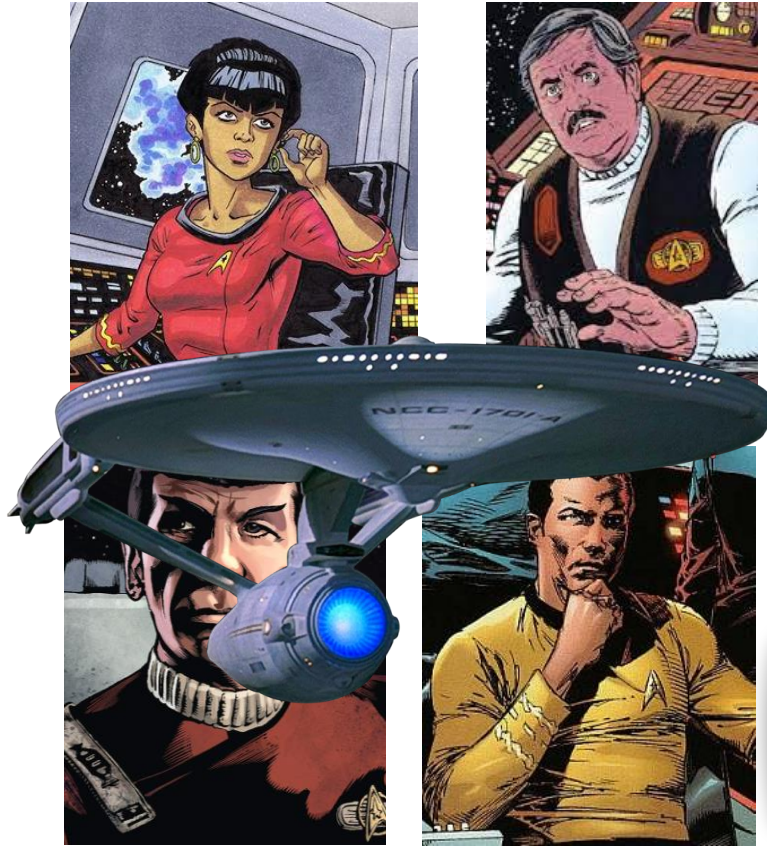
```
.sendOut(scott,  
tell,  
beam_us_up)
```

```
.moveOut(scott,  
inquilinism).
```

```
.sendOut(kirk,tell,  
energizing);
```



Exemplo: Bio-Inspired Protocol



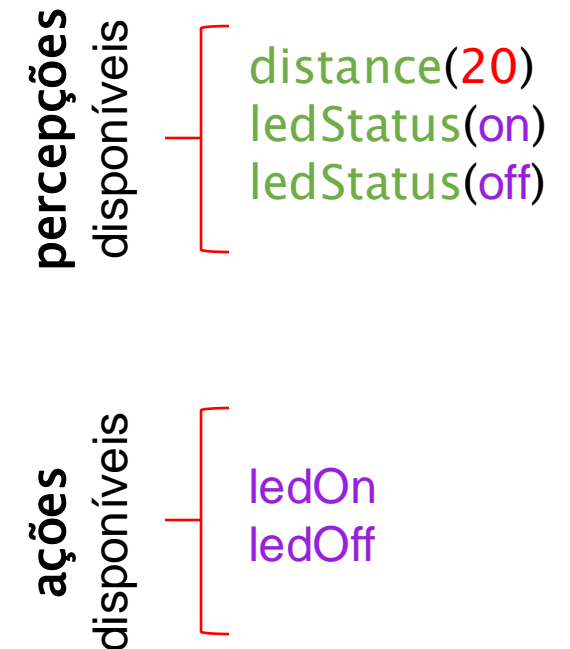
TAREFA ULTRON

“

1. Criar um SMA com um agente Argo, um Comunicador e um Tradicional.
2. Quando o agente Argo da Equipe A perceber que a distância entre ele e um obstáculo for menor que 10cm, ele deve informar ao agente Comunicador.
3. Ao receber a informação, o agente Comunicador da Equipe A deverá ativar o protocolo de Inquilinismo transportando todo o agente Tradicional para a Equipe B.

Quem assumiu o papel da Equipe A deve complementar a implementação com o comportamento da Equipe B e vice-versa.

”



ATO 6: O DESAFIO FINAL

TAREFA ENTERPRISE

Conclusão

- Uma distribuição para facilitar o desenvolvimento de SMA Embarcados

Conclusão

- Uma distribuição para facilitar o desenvolvimento de SMA Embarcados
- Integração de diversas tecnologias em uma versão spin-off

Conclusão

- Uma distribuição para facilitar o desenvolvimento de SMA Embarcados
- Integração de diversas tecnologias em uma versão spin-off
- Uma interface de acesso para configuração do bot.

Conclusão

- Uma distribuição para facilitar o desenvolvimento de SMA Embarcados
- Integração de diversas tecnologias em uma versão spin-off
- Uma interface de acesso para configuração do bot.
- Trabalhos Futuros:

Conclusão

- Uma distribuição para facilitar o desenvolvimento de SMA Embarcados
- Integração de diversas tecnologias em uma versão spin-off
- Uma interface de acesso para configuração do bot.
- Trabalhos Futuros:
 - **JaCa Embedded:** atualizar para a versão mais atual do JaCaMo e adicionar os artefatos de interfaceamento de hardware;

Conclusão

- Uma distribuição para facilitar o desenvolvimento de SMA Embarcados
- Integração de diversas tecnologias em uma versão spin-off
- Uma interface de acesso para configuração do bot.
- Trabalhos Futuros:
 - **JaCa Embedded:** atualizar para a versão mais atual do JaCaMo e adicionar os artefatos de interfaceamento de hardware;
 - **IDE Lite:** uma IDE embarcada e leve para facilitar o desenvolvimento do firmware e do SMA;

Conclusão

- Uma distribuição para facilitar o desenvolvimento de SMA Embarcados
- Integração de diversas tecnologias em uma versão spin-off
- Uma interface de acesso para configuração do bot.
- Trabalhos Futuros:
 - **JaCa Embedded:** atualizar para a versão mais atual do JaCaMo e adicionar os artefatos de interfaceamento de hardware;
 - **IDE Lite:** uma IDE embarcada e leve para facilitar o desenvolvimento do firmware e do SMA;
 - **IDE:** integrada com uma linguagem de modelagem com função drag-and-drop para design da solução e que permita o acesso a todos os seus bots;

Conclusão

- Uma distribuição para facilitar o desenvolvimento de SMA Embarcados
- Integração de diversas tecnologias em uma versão spin-off
- Uma interface de acesso para configuração do bot.
- Trabalhos Futuros:
 - **JaCa Embedded:** atualizar para a versão mais atual do JaCaMo e adicionar os artefatos de interfaceamento de hardware;
 - **IDE Lite:** uma IDE embarcada e leve para facilitar o desenvolvimento do firmware e do SMA;
 - **IDE:** integrada com uma linguagem de modelagem com função drag-and-drop para design da solução e que permita o acesso a todos os seus bots;
 - **Skynet:** substituir o ContextNet por uma solução usando o XMPP ou SMTP baseado em KQML;

Conclusão

- Uma distribuição para facilitar o desenvolvimento de SMA Embarcados
- Integração de diversas tecnologias em uma versão spin-off
- Uma interface de acesso para configuração do bot.
- Trabalhos Futuros:
 - **JaCa Embedded:** atualizar para a versão mais atual do JaCaMo e adicionar os artefatos de interfaceamento de hardware;
 - **IDE Lite:** uma IDE embarcada e leve para facilitar o desenvolvimento do firmware e do SMA;
 - **IDE:** integrada com uma linguagem de modelagem com função drag-and-drop para design da solução e que permita o acesso a todos os seus bots;
 - **Skynet:** substituir o ContextNet por uma solução usando o XMPP ou SMTP baseado em KQML;
 - **ChonOS beta:** inserir um serviço de gerenciamento de recursos.

Conclusão

- Uma distribuição para facilitar o desenvolvimento de SMA Embarcados
- Integração de diversas tecnologias em uma versão spin-off
- Uma interface de acesso para configuração do bot.
- Trabalhos Futuros:
 - **JaCa Embedded:** atualizar para a versão mais atual do JaCaMo e adicionar os artefatos de interfaceamento de hardware;
 - **IDE Lite:** uma IDE embarcada e leve para facilitar o desenvolvimento do firmware e do SMA;
 - **IDE:** integrada com uma linguagem de modelagem com função drag-and-drop para design da solução e que permita o acesso a todos os seus bots;
 - **Skynet:** substituir o ContextNet por uma solução usando o XMPP ou SMTP baseado em KQML;
 - **ChonOS beta:** inserir um serviço de gerenciamento de recursos.
- Trabalhos Futuros Genéricos:
 - **Segurança**
 - **Desempenho**
 - **Eficiência Energética**

“ FREE TIME ”

percepções
disponíveis

[
distance(20)
luminosity(468)
ledStatus(on)
ledStatus(off)
buzzerStatus(on)
buzzerStatus(off)
motorStatus(stopped)
motorStatus(running)
motorStatus(turningRight)
motorStatus(turningLeft)

ações
disponíveis

[
ledOn
ledOff
buzzerOn
buzzerOff
stop
goAhead
goRight
goLeft

Referências Bibliográficas

- [\[Bordini et al. 2007\]](#) Bordini, R.H., Hubner, J.F., Wooldridge, M. Programming Multi-Agent Systems in AgentSpeak Using Jason. John Wiley & Sons Ltd., 2007.
- [\[Bratman, 1987\]](#) Bratman, M. Intentions, Plans, and Practical Reason. Harvard University Press, 1987.
- [\[Guinelli et al., 2016\]](#) Guinelli, J. V. ; Junger, D. S. ; Pantoja, C. E. . An Analysis of Javino Middleware for Robotic Platforms Using Jason and JADE Frameworks. In: Workshop-Escola de Sistemas de Agentes, Seus Ambientes e Aplicações, Maceió. Anais do X Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações, 2016.
- [\[Huber, 1999\]](#) Huber MJ. Jam: a bdi-theoretic mobile agent architecture. In Proceedings of the third annual conference on Autonomous Agents, AGENTS '99, pags. 236-243, New York, 1999
- [\[Lazarin and Pantoja, 2015\]](#) Lazarin, N.M., Pantoja, C.E. : A robotic-Agent Platform For Embedding Software Agents Using Raspberry Pi and Arduino Boards. In: 9th Software Agents, Environments and Applications School, 2015
- [\[Pantoja et al., 2016\]](#) Pantoja, C. E.; Stabile Jr, M. F. ; Lazarin, N. M. ; Sichman, J. S. ARGO: A Customized Jason Architecture for Programming Embedded Robotic Agents. In: Workshop on Engineering Multi-Agent Systems, 2016, Singapore. Proceedings of the Third International Workshop on Engineering Multi-Agent Systems (EMAS 2016), 2016.

Referências Bibliográficas

- [Rao 1996] Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In: de Velde, W.V., Perram, J.W. (eds.) Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world. Lecture Notes in Artificial Intelligence, vol. 1038, pp. 42-55. Springer-Verlag, Secaucus. USA, 1996.
- [Stabile Jr. and Sichman, 2015] Stabile Jr., M.F., Sichman, J.S. Evaluating Perception Filters In BDI Jason Agents. In: 4th Brazilian Conference On Intelligent Systems, 2015.
- [Winikoff, 2005] Winikoff M. Jack intelligent agents: An industrial strength platform. Em Bordini R, Dastani M, Dix J, Fallah AS, Weiss G, editors. Multi-Agent Programming, volume 15 of Multiagent Systems, Artificial Societies, and Simulated Organizations, pages. 175-193. Springer US, 2005.
- [Wooldridge, 2000] Wooldridge, M. *Reasoning about rational agents. Intelligent robotics and autonomous agents. MIT Press*, 2000.
- [Wooldridge, 2009] Wooldridge M. An Introduction to MultiAgent Systems. John Wiley & Sons, 2009.
- [Zambonelli et al., 2001] Zambonelli F, Jennings NR, Omicini A, Wooldridge M. Agent-Oriented Software Engineering for Internet Applications. In: Omicini A, Zambonelli F, Klusch M, Tolksdorf R, editors. Coordination of Internet Agents. Springer Verlag; 2001. p.326-345, 2001

OBRIGADO!

pantoja@cefet-rj.br
nilson.lazarin@cefet-rj.br
vsjesus@id.uff.br
fabiancpbm@gmail.com

