Introduction to Distributed and Embedded Multi-agent Systems

Carlos Eduardo Pantoja[1]
Nilson Mori Lazarin[1,2]

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

Introduction

Agent

- Is a computational system capable of perceiving and acting in an environment by its deliberation based on its convictions and motivations.

WOOLDRIDGE, Michael. Intelligent Agents. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Cambridge, MA, USA: MIT Press, 1999. p. 27–77
MICHEL, Fabien; FERBER, Jacques; DROGOUL, Alexis. Multi-Agent Systems and Simulation: a Survey From the Agents Community's Perspective. 2009
BRATMAN, Michael. Intention, plans, and practical reason. 1987.

Introduction

Multi-Agent Systems (MAS)

- A MAS is a group of loosely coupled autonomous agents working in the same environment.

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

BRANDÃO, FABIAN CESAR; LIMA, MARIA ALICE TRINTA; PANTOJA, CARLOS EDUARDO; ZAHN, JEAN; VITERBO, JOSÉ. Engineering Approaches for Programming Agent-Based IoT Objects Using the Resource Management Architecture. SENSORS, v. 21, p. 8110. Disponível em: https://doi.org/10.3390/s21238110.

Introduction

Multi-Agent Systems (MAS)

- A MAS is a group of loosely coupled autonomous agents working in the same environment.

Embedded MAS

- It is a system based on agents that provides autonomy, proactivity, and social ability to a physical device.

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

BRANDÃO, FABIAN CESAR; LIMA, MARIA ALICE TRINTA; PANTOJA, CARLOS EDUARDO; ZAHN, JEAN; VITERBO, JOSÉ. Engineering Approaches for Programming Agent-Based IoT Objects Using the Resource Management Architecture. SENSORS, v. 21, p. 8110. Disponível em: https://doi.org/10.3390/s21238110.

Introduction

Multi-Agent Systems (MAS)

- A MAS is a group of loosely coupled autonomous agents working in the same environment.

Embedded MAS

- It is a system based on agents that provides autonomy, proactivity, and social ability to a physical device.

There is no remote control or external processing.

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

BRANDÃO, FABIAN CESAR; LIMA, MARIA ALICE TRINTA; PANTOJA, CARLOS EDUARDO; ZAHN, JEAN; VITERBO, JOSÉ. Engineering Approaches for Programming Agent-Based IoT Objects Using the Resource Management Architecture. SENSORS, v. 21, p. 8110. Disponível em: https://doi.org/10.3390/s21238110.

Embodied or embedded agents: Conventional approaches

T. LEPPÄNEM et al., Mobile Agents for Integration of Internet of Things and Wireless Sensor Networks. 2013 IEEE International Conference on Systems, Man, and Cybernetics, Manchester, UK, 2013, pp. 14-2.
C. SAVAGLIO, G. FORTINO and M. ZHOU, Towards interoperable, cognitive and autonomic IoT systems: An agent-based approach. 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, VA.
M. E. PÉREZ HERNÁNDEZ and S. REIFF-MARGANIEC, Towards a Software Framework for the Autonomous Internet of Things. 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), Vienna, Austria, 2016, pp. 220-227
HERINGER, V. H.; BARROS, R. S.; PANTOJA, C. E.; MACHADO, L.; LAZARIN, N. M. An Agent-oriented Ground Vehicle's Automation using Jason Framework. In : International Conference on Agents and Artificial Intelligence, 2014, ESEO. Proceedings of the 6th International Conference on Agents and Artificial Intelligence. p. 261-266.

Embodied or embedded agents: Conventional approaches

Centralised Solution

Embodied or embedded agents: Conventional approaches

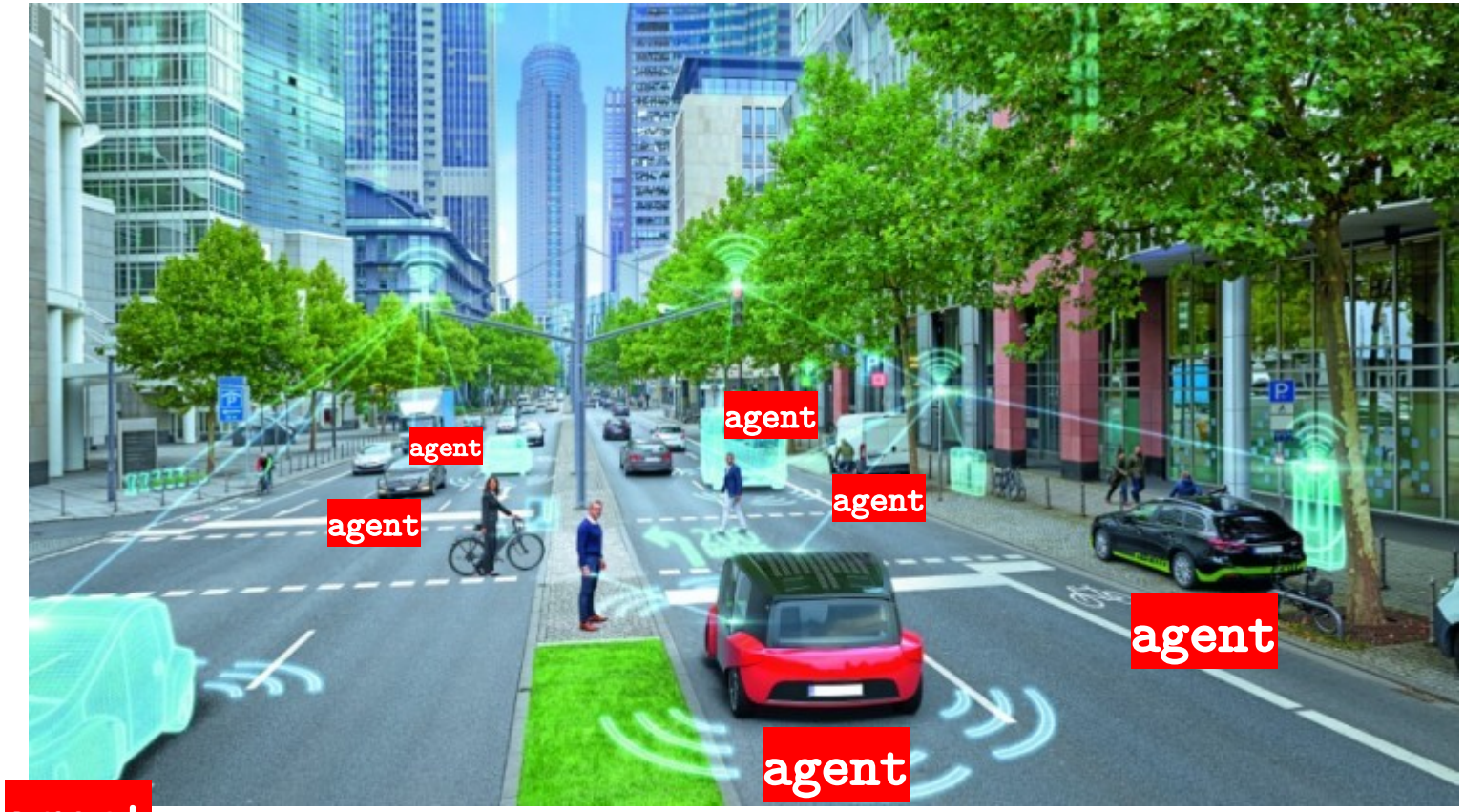
Centralised Solution

- one agent p. device

Embodied or embedded agents: Conventional approaches

Centralised Solution

- one agent p. device
  - performance issues
    - only one agent into the device;
    - a lot of sensors.



Embodied or embedded agents: Conventional approaches

Centralised Solution

- one agent p. device
  - performance issues
    - only one agent into the device;
    - a lot of sensors.
- reactive artifact

Embodied or embedded agents: Conventional approaches

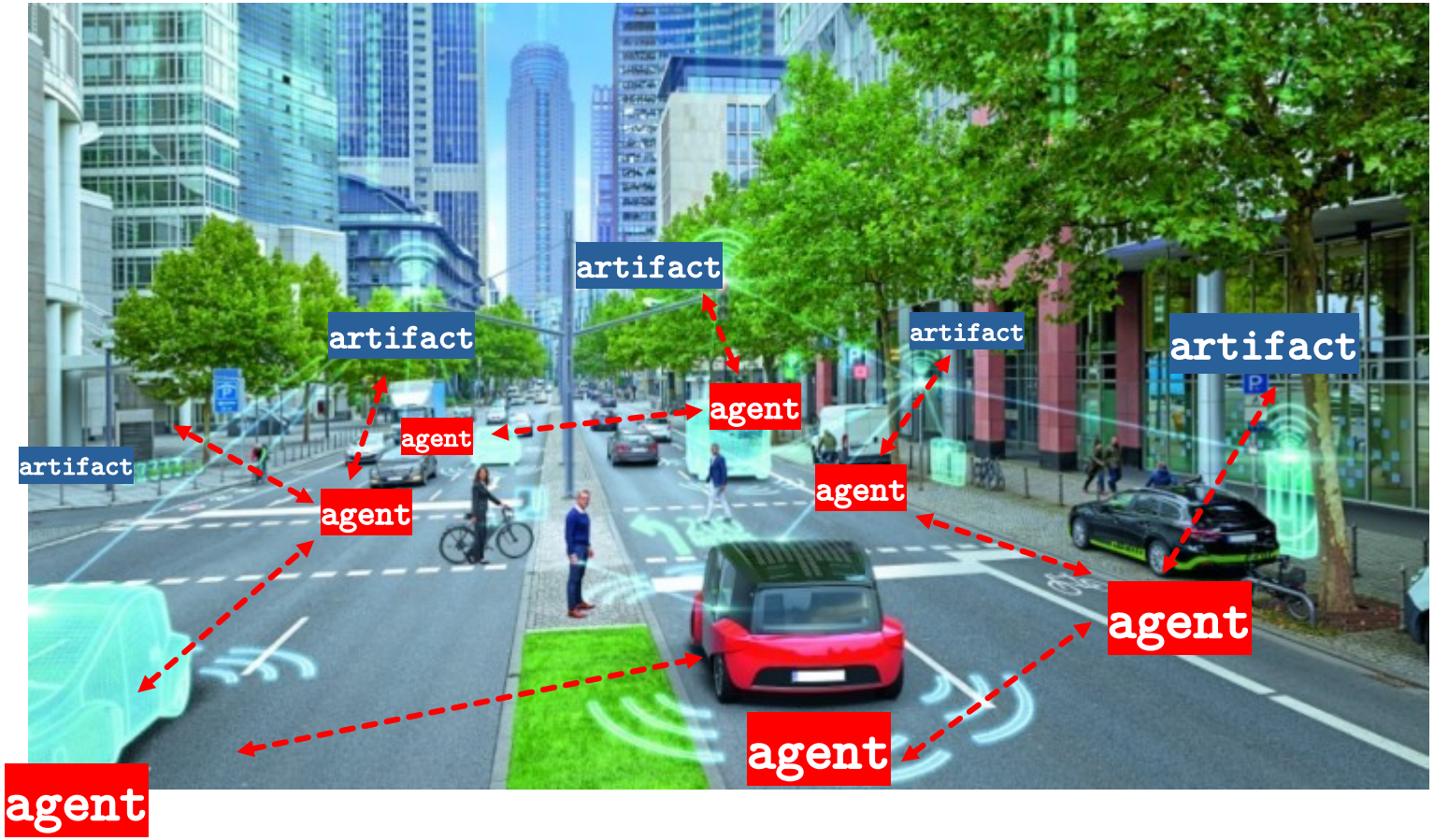
Centralised Solution

- one agent p. device
  - performance issues
    - only one agent into the device;
    - a lot of sensors.
- reactive artifact
  - are data oriented
  - don't have decision-making

Embodied or embedded agents: Conventional approaches

Centralised Solution

- one agent p. device
  - performance issues
    - only one agent into the device;
    - a lot of sensors.
- reactive artifact
  - are data oriented
  - don't have decision-making
- depends on a server



Our approach

Distributed Solution

Our approach

Distributed Solution

- one MAS p. device

Our approach

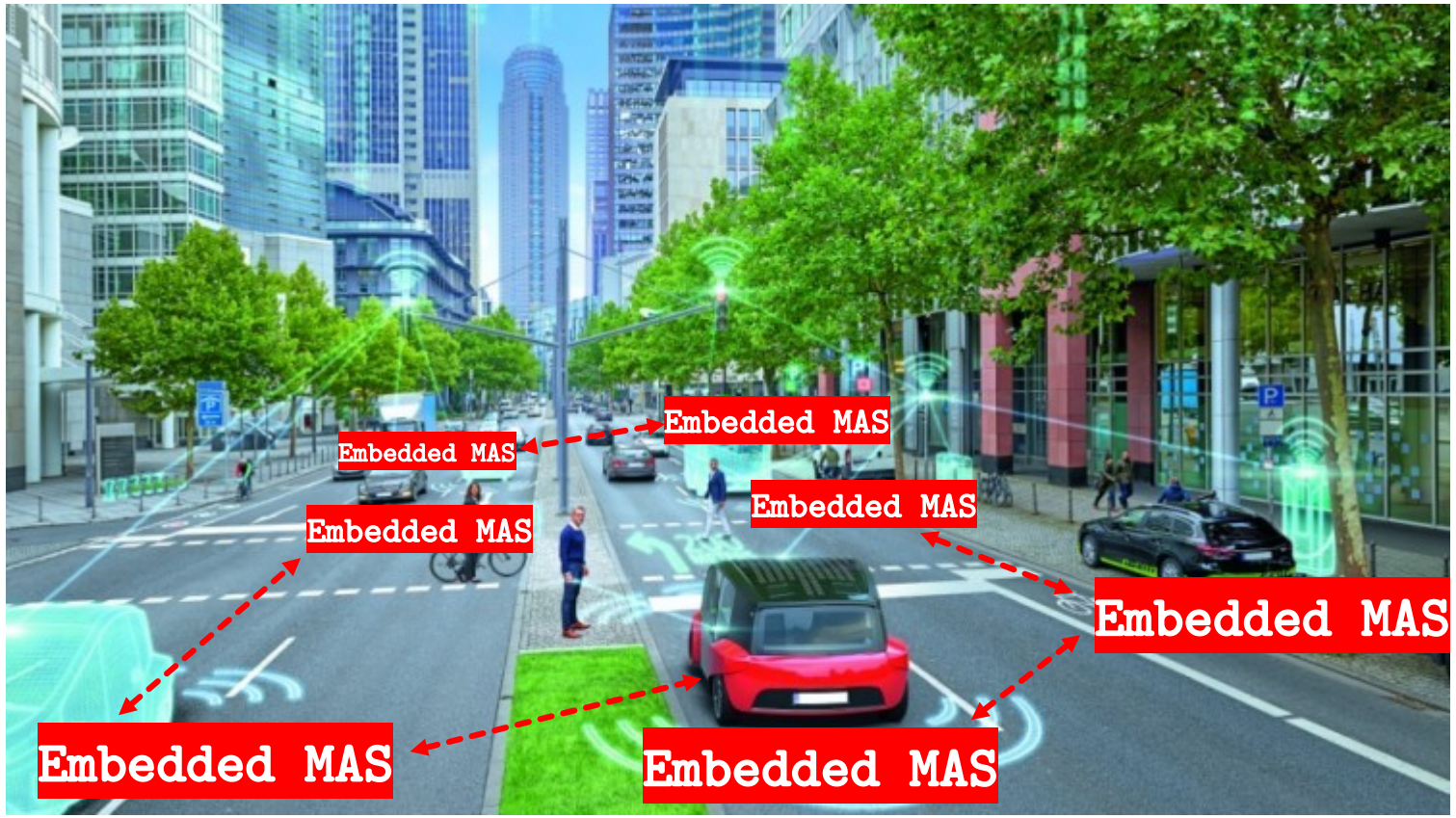
Distributed Solution

- one MAS p. device
- truly autonomy

Our approach

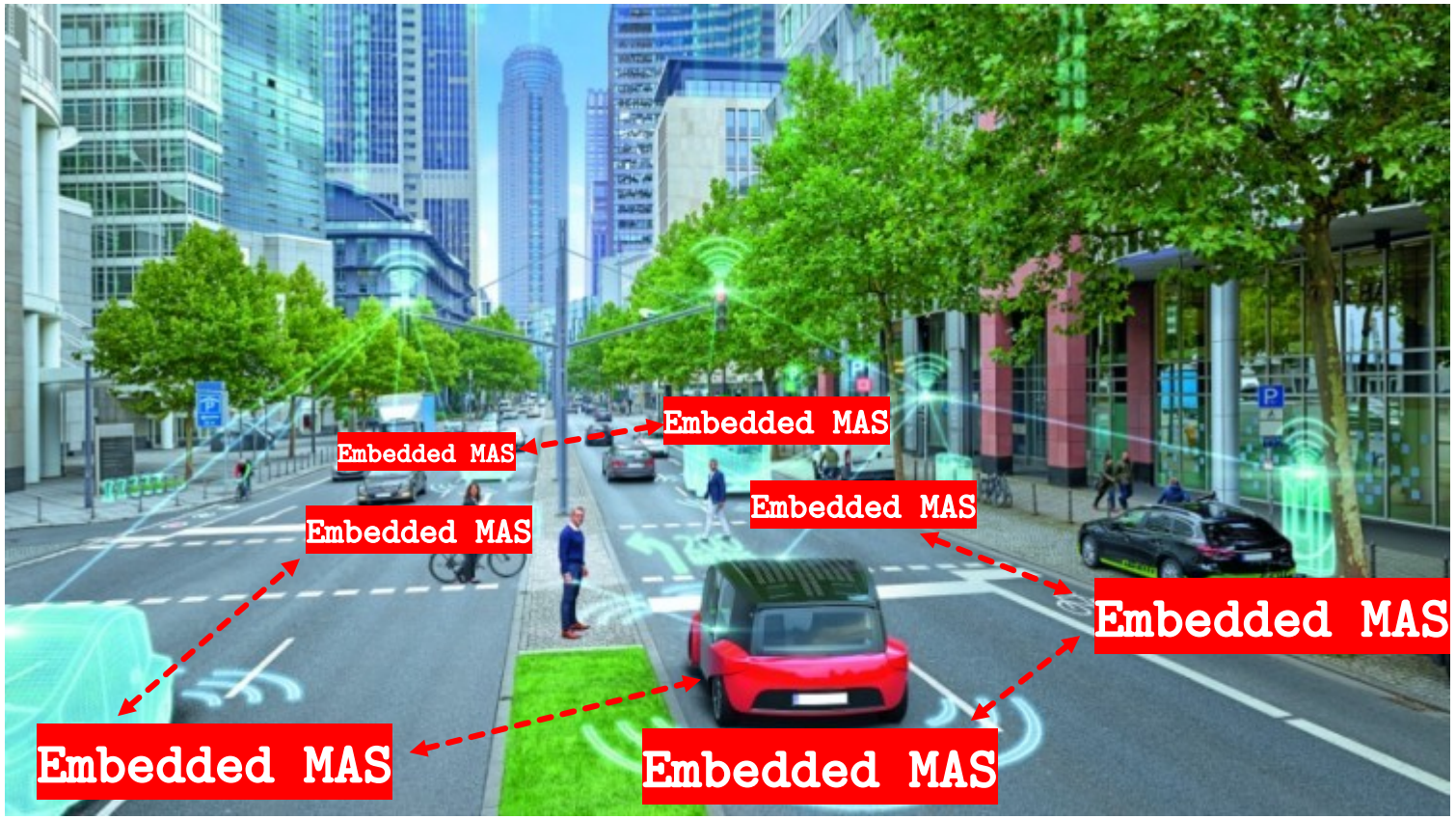
Distributed Solution

- one MAS p. device
- truly autonomy
  - communicability dependency only for external communication



Our approach

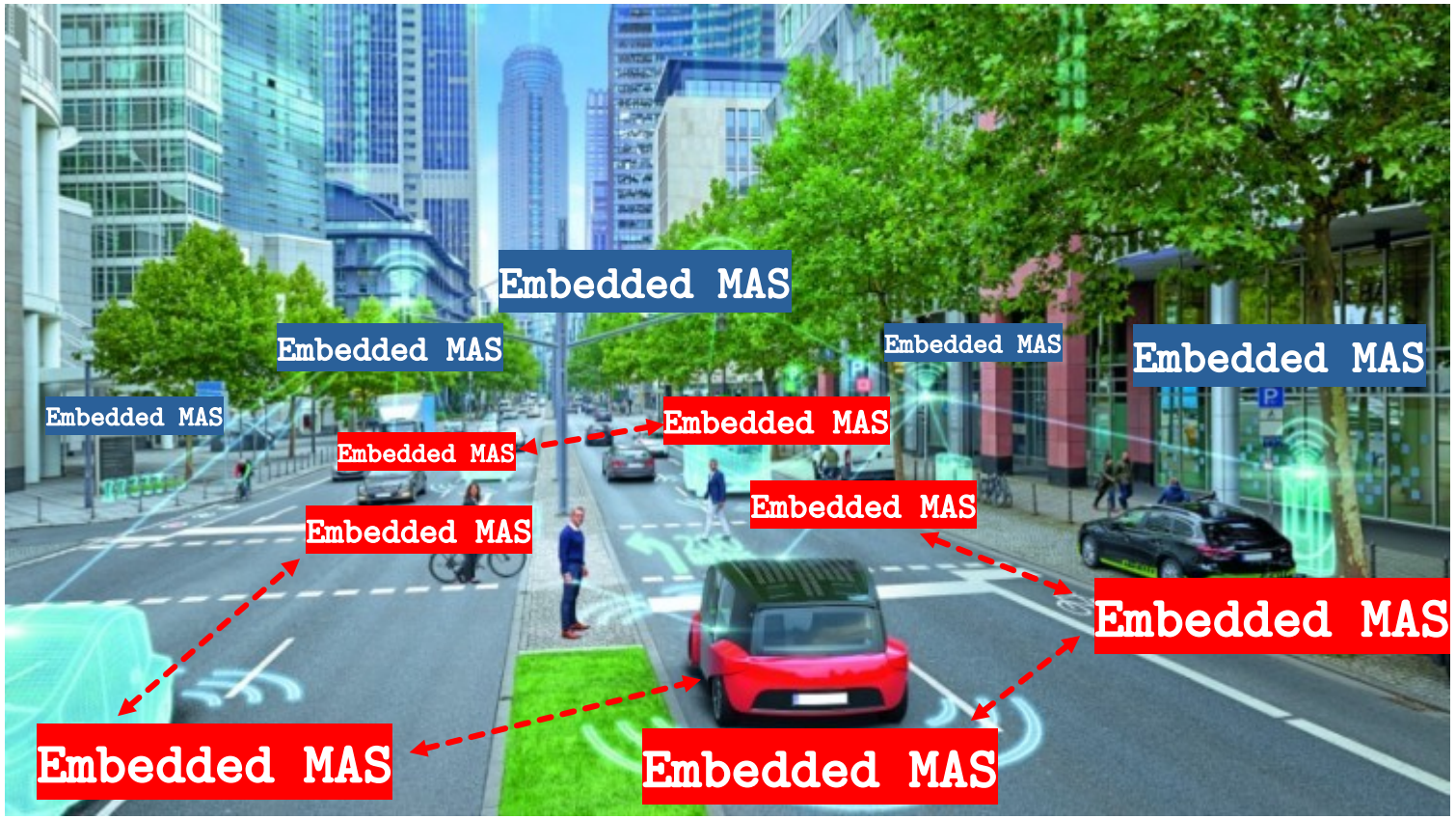
Edge Intelligence



Our approach

Edge Intelligence

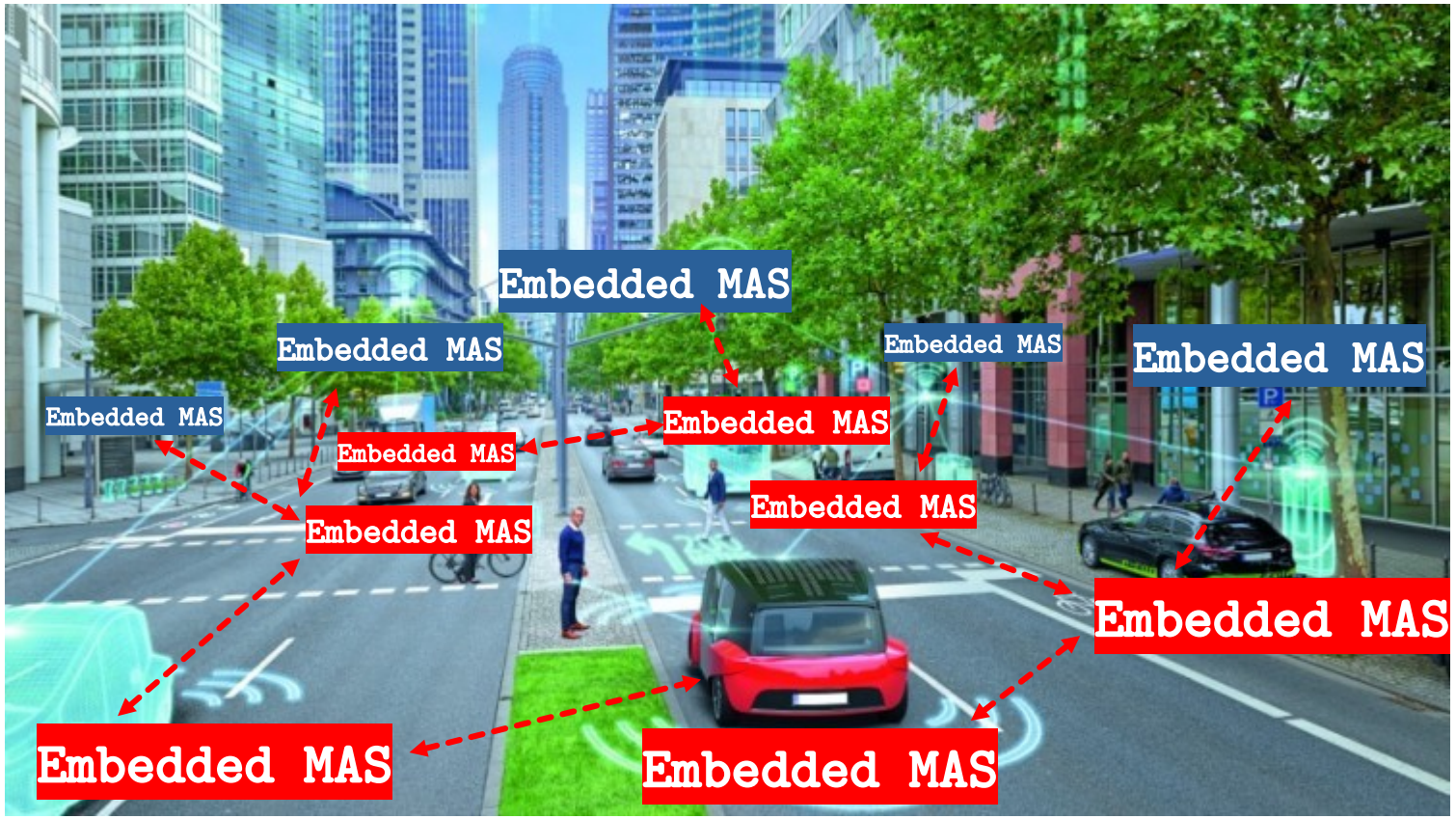
- one MAS p. artifact



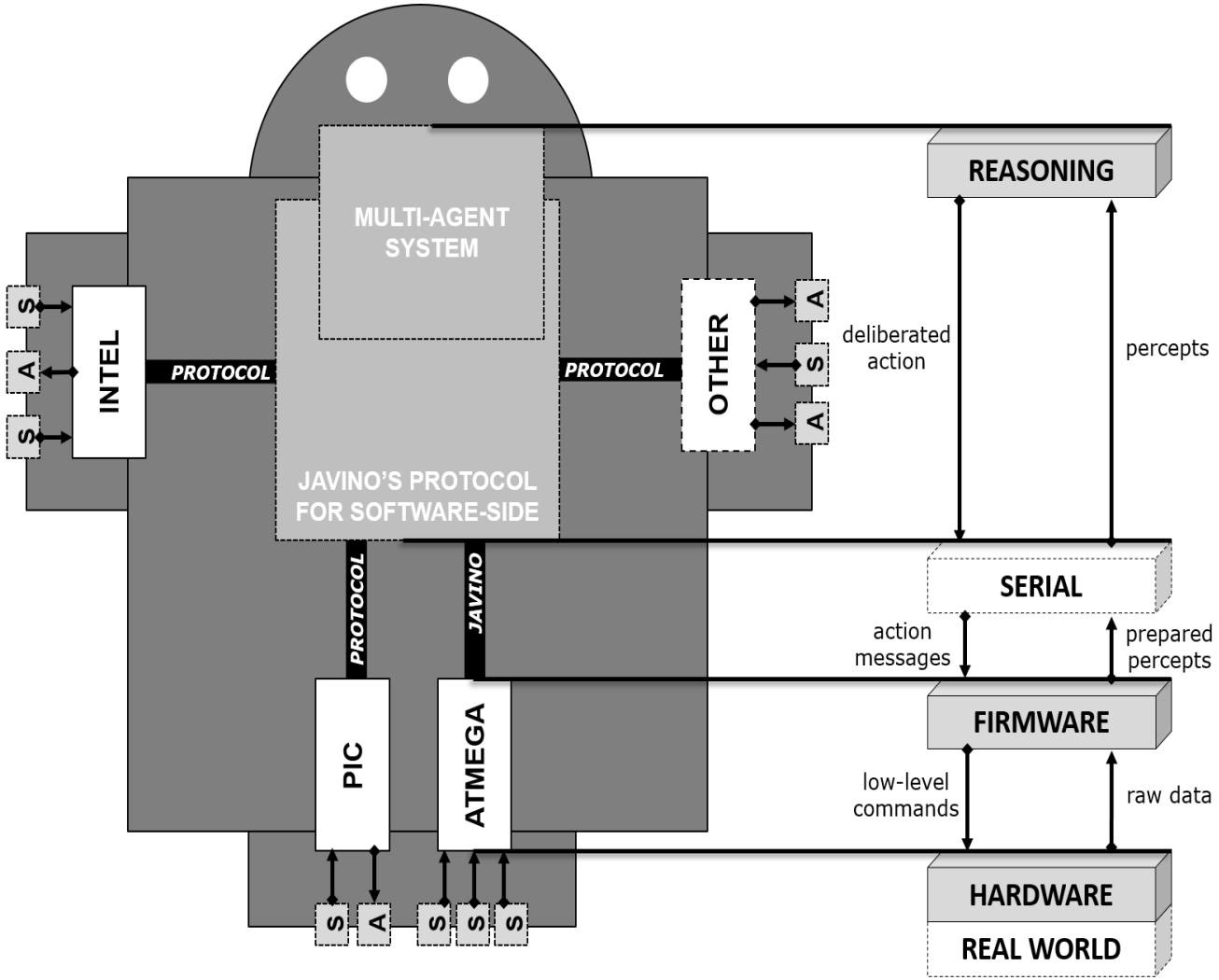
Our approach

Edge Intelligence

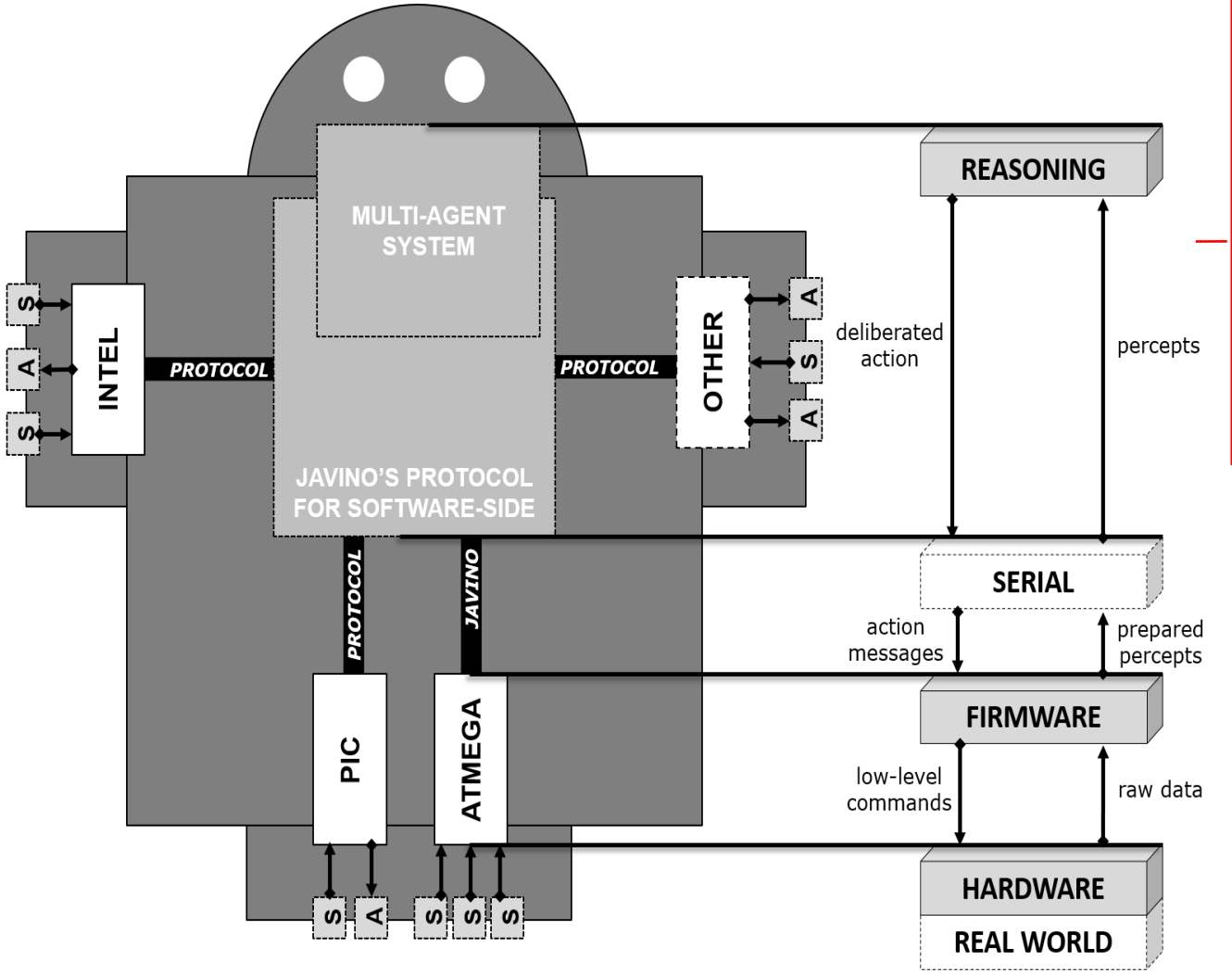
- one MAS p. artifact
  - pro-active artifact
  - decision-making in the edge



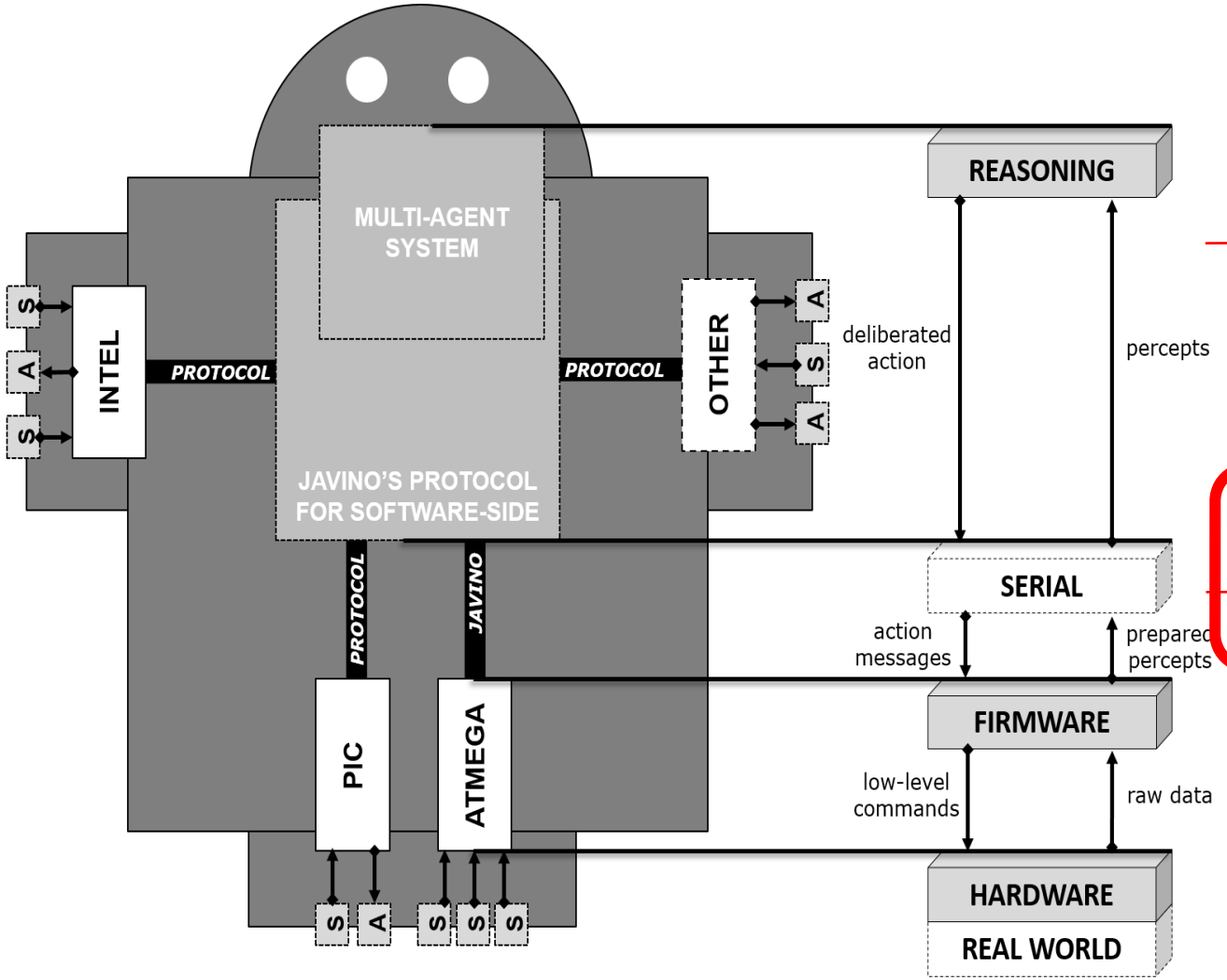
Toolkit to Facilitate Embedded MAS Development



Toolkit to Facilitate Embedded MAS Development



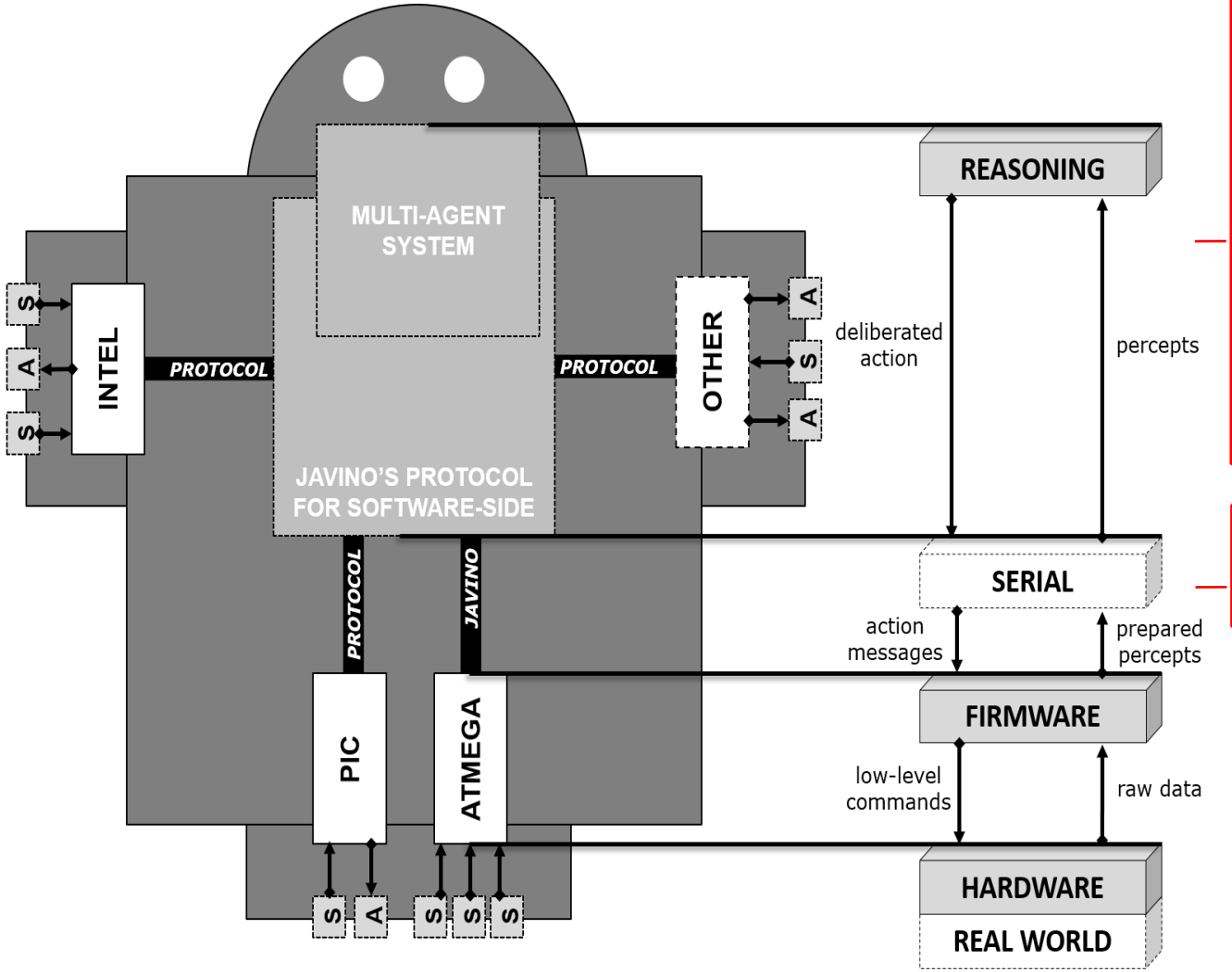
Javino



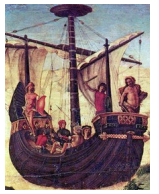
JavINO

LAZARIN, Nilson Mori; PANTOJA, Carlos Eduardo. **A robotic-agent platform for embedding software agents using raspberry pi and arduino boards**. In: 9TH SOFTWARE AGENTS, ENVIRONMENTS AND APPLICATIONS SCHOOL, 2015. Proceedings WESAAC 2015 [...]. Niteroi: UFF, 2015. p. 13–20. Disponível em: http://www2.ic.uff.br/~wesaac2015/Proceedings-WESAAC-2015.pdf.

Argo Agents



J



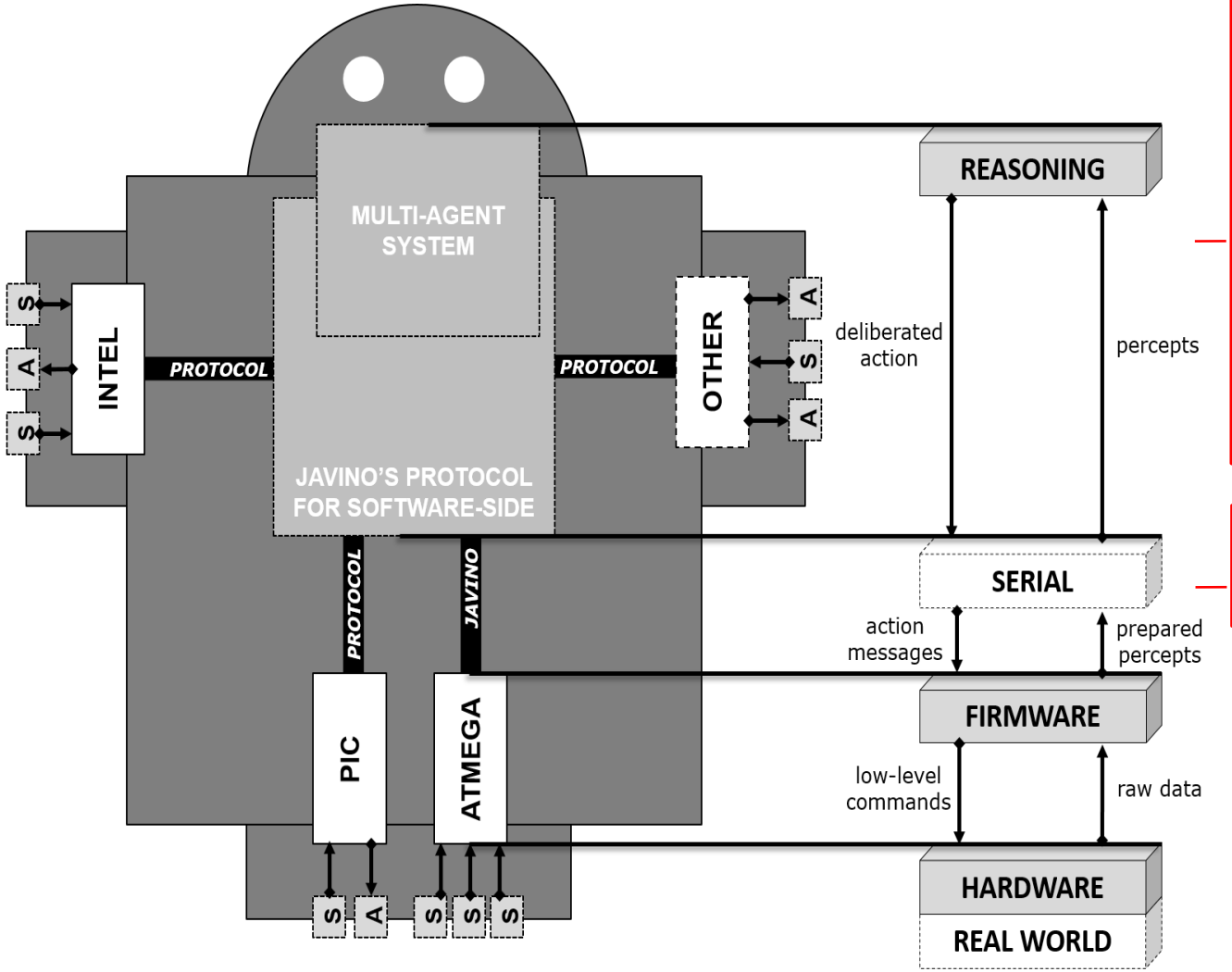
ARGO

A

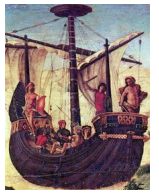
JaVINO

Pantoja, C.E., Stabile, M.F., Lazarin, N.M., Sichman, J.S. (2016). **ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming**. In: Baldoni, M., Müller, J., Nunes, I., Zalila-Wenkstern, R. (eds) Engineering Multi-Agent Systems. EMAS 2016. Lecture Notes in Computer Science(), vol 10093. Springer, Cham. https://doi.org/10.1007/978-3-319-50983-9_8

Communicator Agents



J



ARGO

A

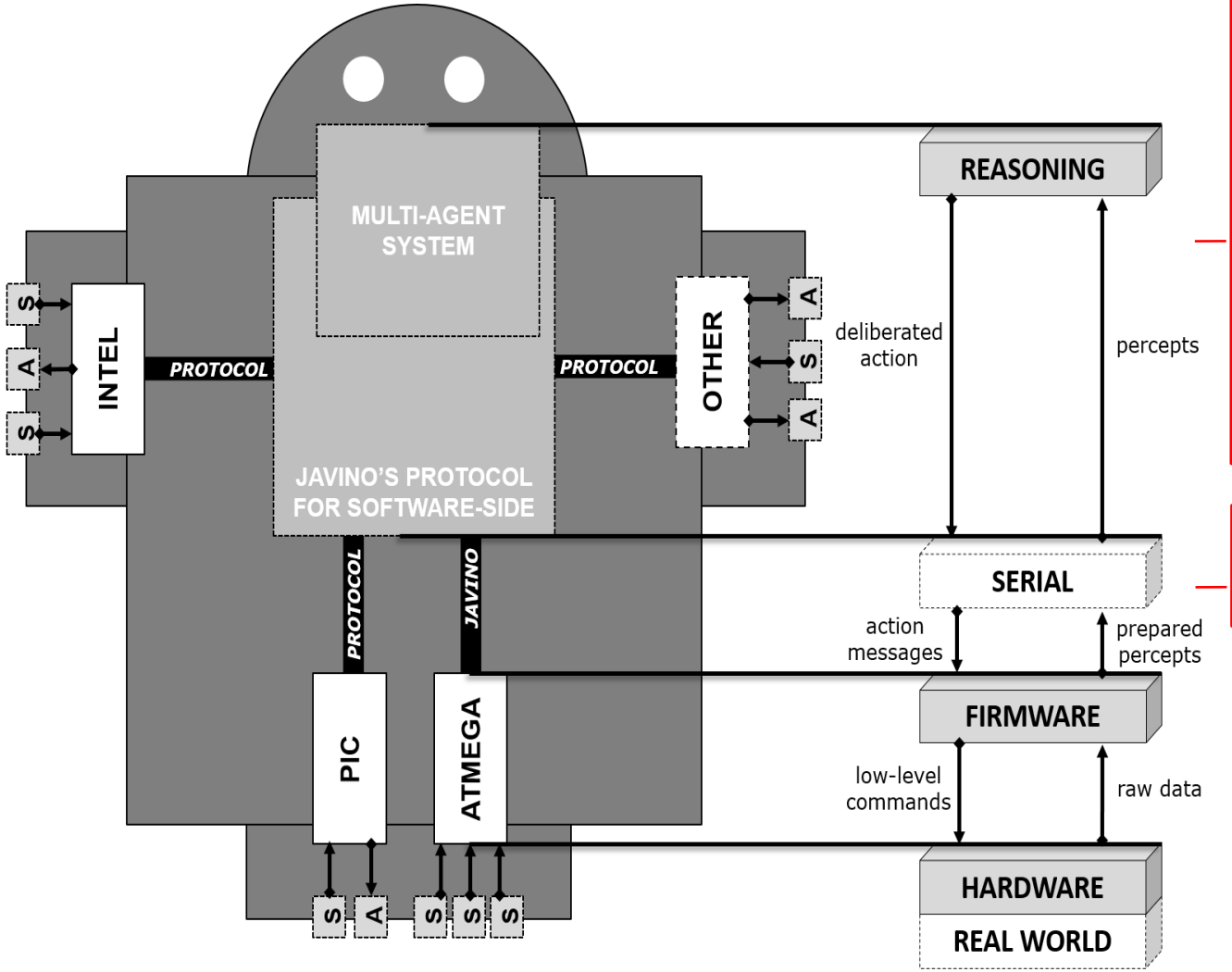
C

Communicators

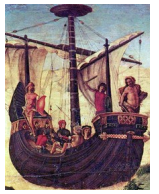
JaVINO

PANTOJA, Carlos Eduardo; SOARES, Heder Dorneles; VITERBO, José; et al. **Exposing IoT Objects in the Internet Using the Resource Management Architecture**. International Journal of Software Engineering and Knowledge Engineering, v. 29, n. 11n12, p. 1703–1725, 2019.

Bio-Inspired Protocols



J



ARGO

A

C

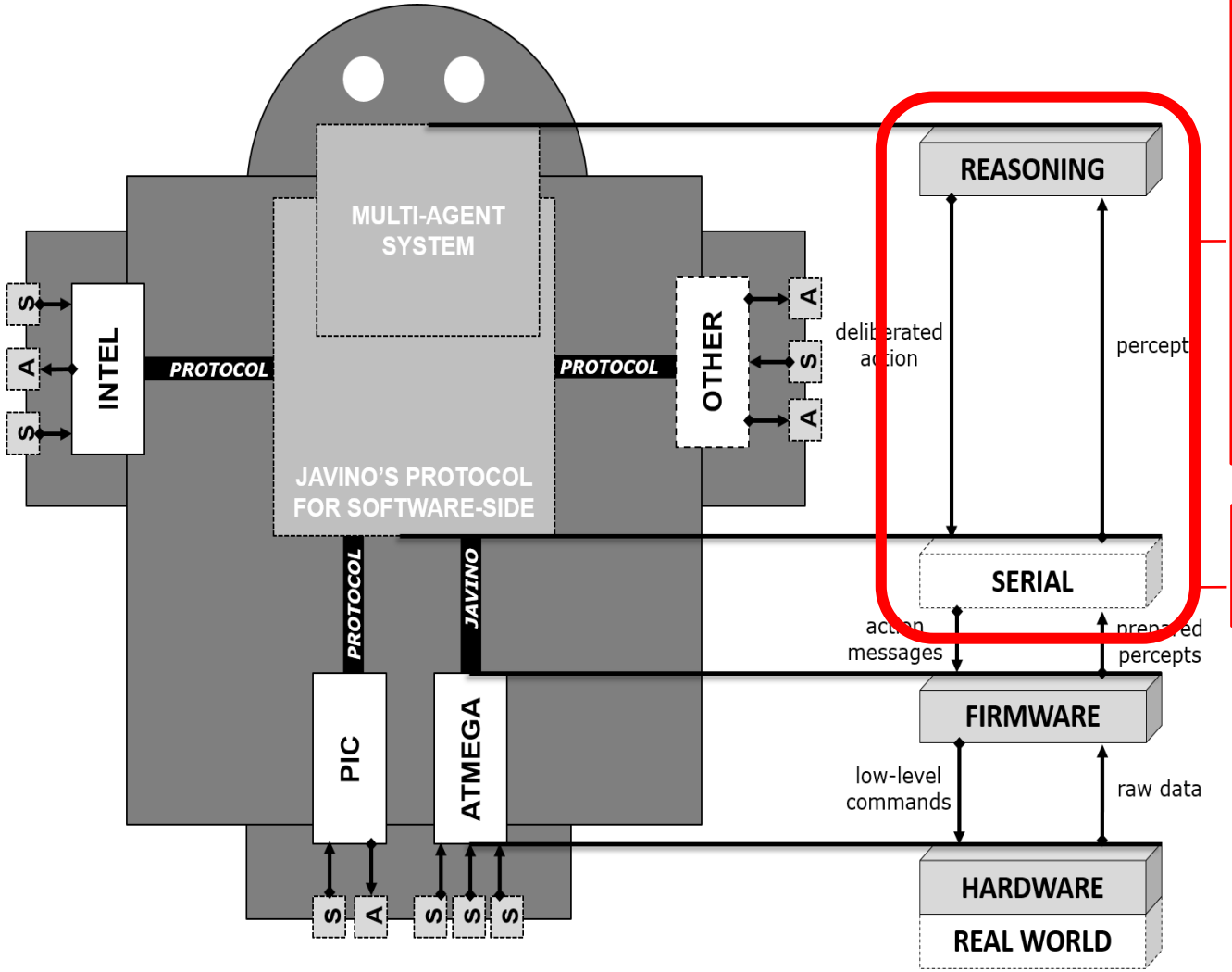
Bio-Inspired Protocols

Communicators

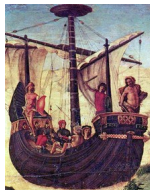
JaVINO

Souza de Jesus V., Pantoja C., Manoel F., Alves G., Viterbo J. and Bezerra E. (2021). **Bio-Inspired Protocols for Embodied Multi-Agent Systems**.In Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART, ISBN 978-989-758-484-8, pages 312-320. DOI: 10.5220/0010257803120320

Swapping Physical Resources at Runtime



J



ARGO

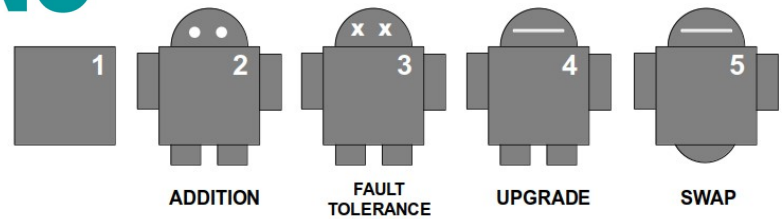
A

C

Bio-Inspired Protocols

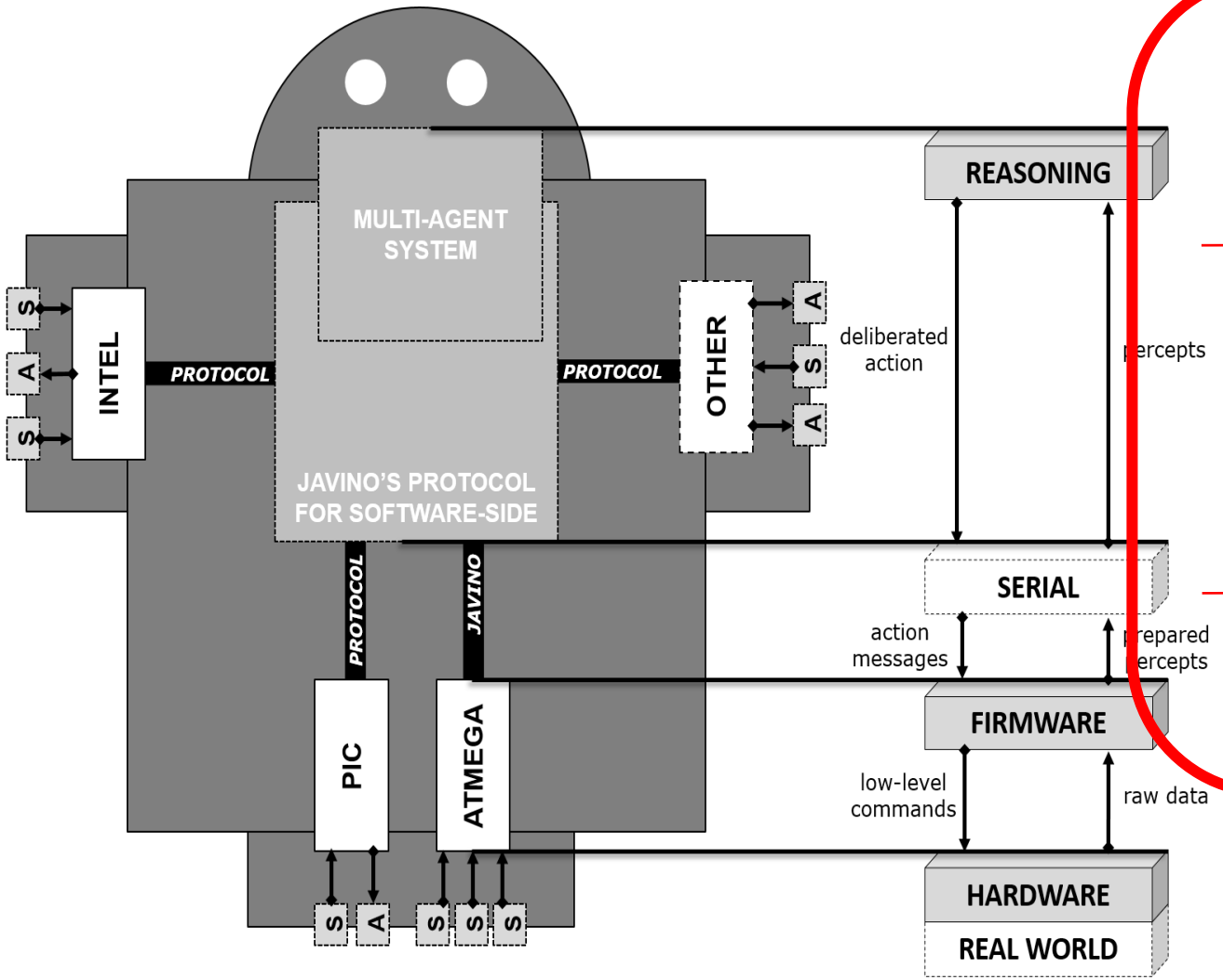
Communicators

JavINO



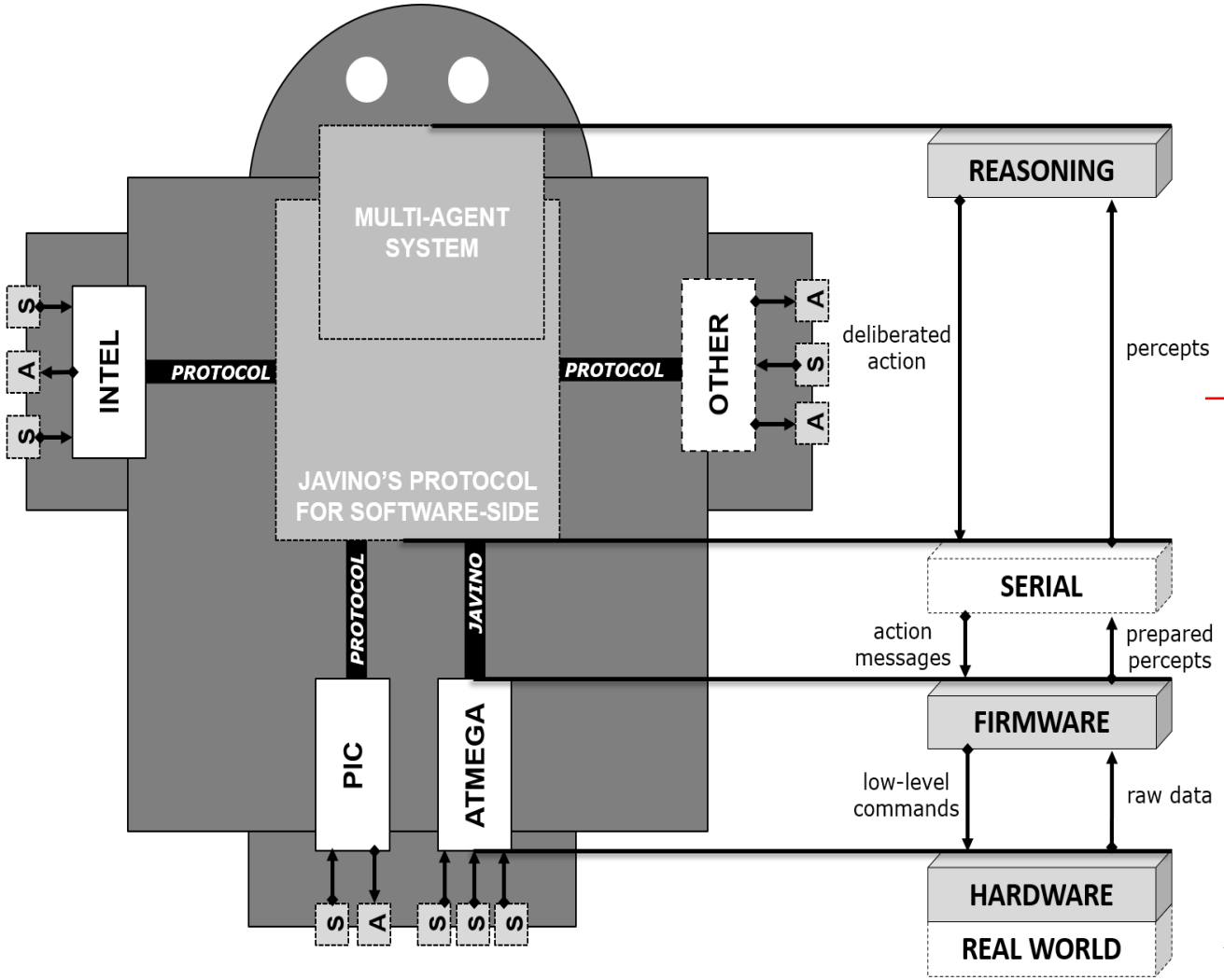
Lazarin, N.; Pantoja, C. and Viterbo, J. (2023). **Swapping Physical Resources at Runtime in Embedded MultiAgent Systems**. In Proceedings of the 15th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART; ISBN 978-989-758-623-1; ISSN 2184-433X, SciTePress, pages 93-104. DOI: 10.5220/0011750700003393

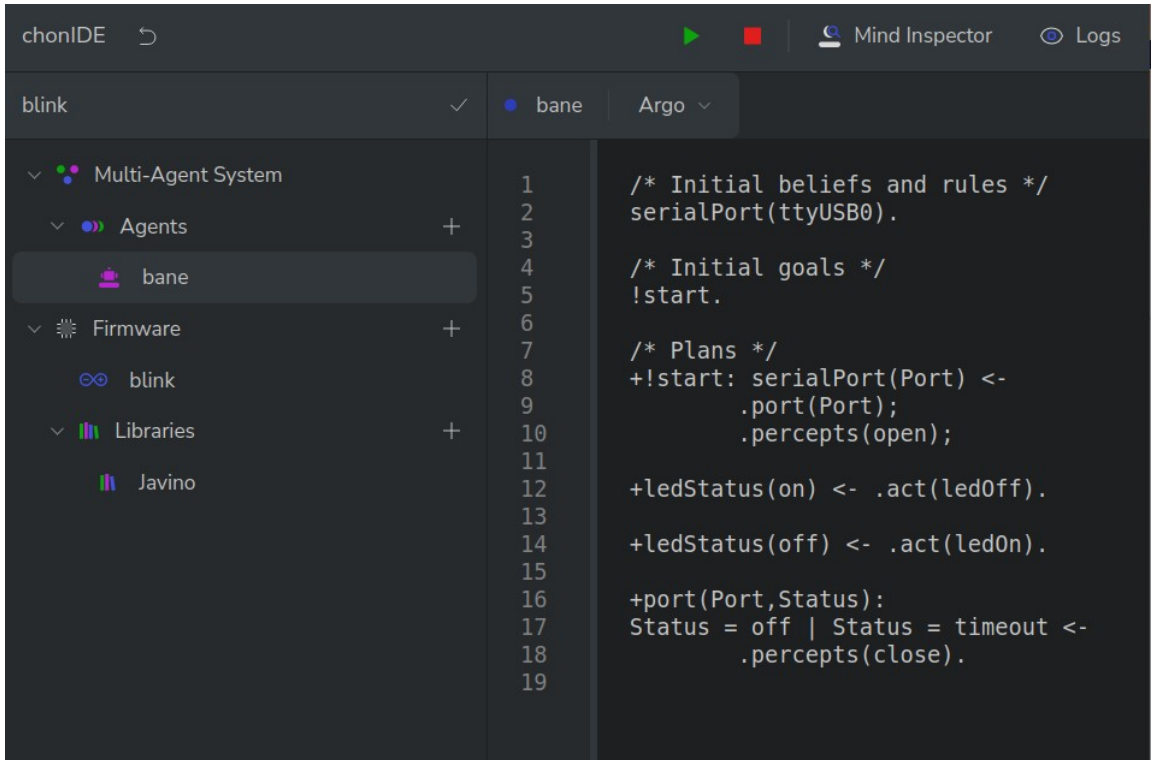
Jason Embedded and Packages



PANTOJA, CARLOS EDUARDO; JESUS, VINICIUS SOUZA; LAZARIN, NILSON MORI; VITERBO, JOSÉ. ***A Spin-off Version of Jason for IoT and Embedded Multiagent Systems***. In : Proceedings of the 12nd Brazilian Conference on Intelligent Systems, BRACIS 2023, Belo Horizonte, Brazil, 2023

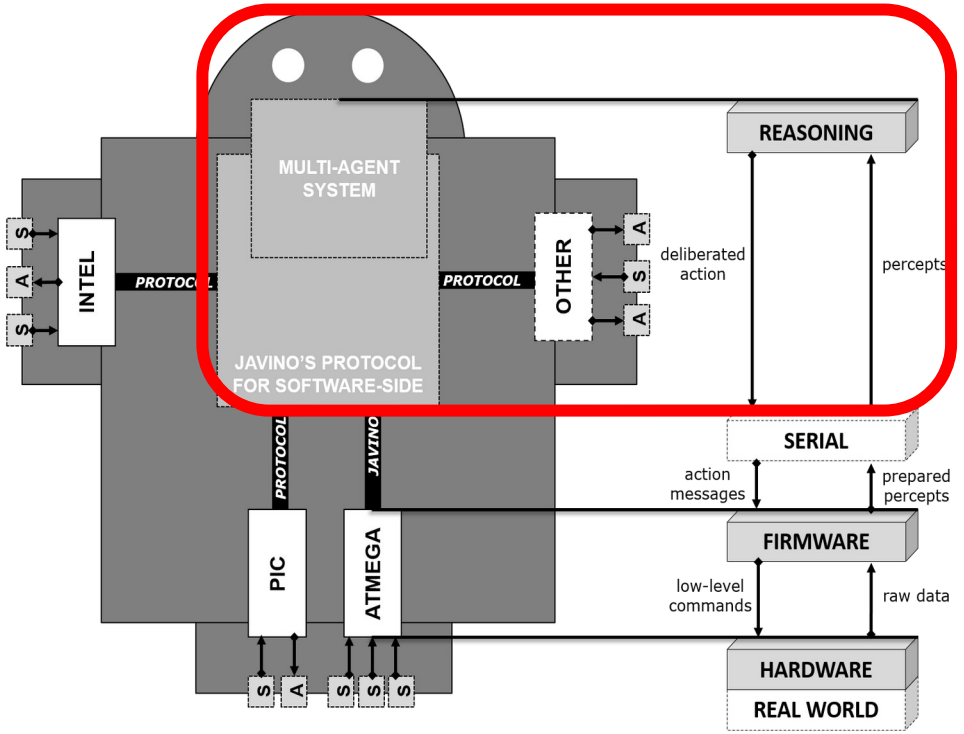
ChonIDE



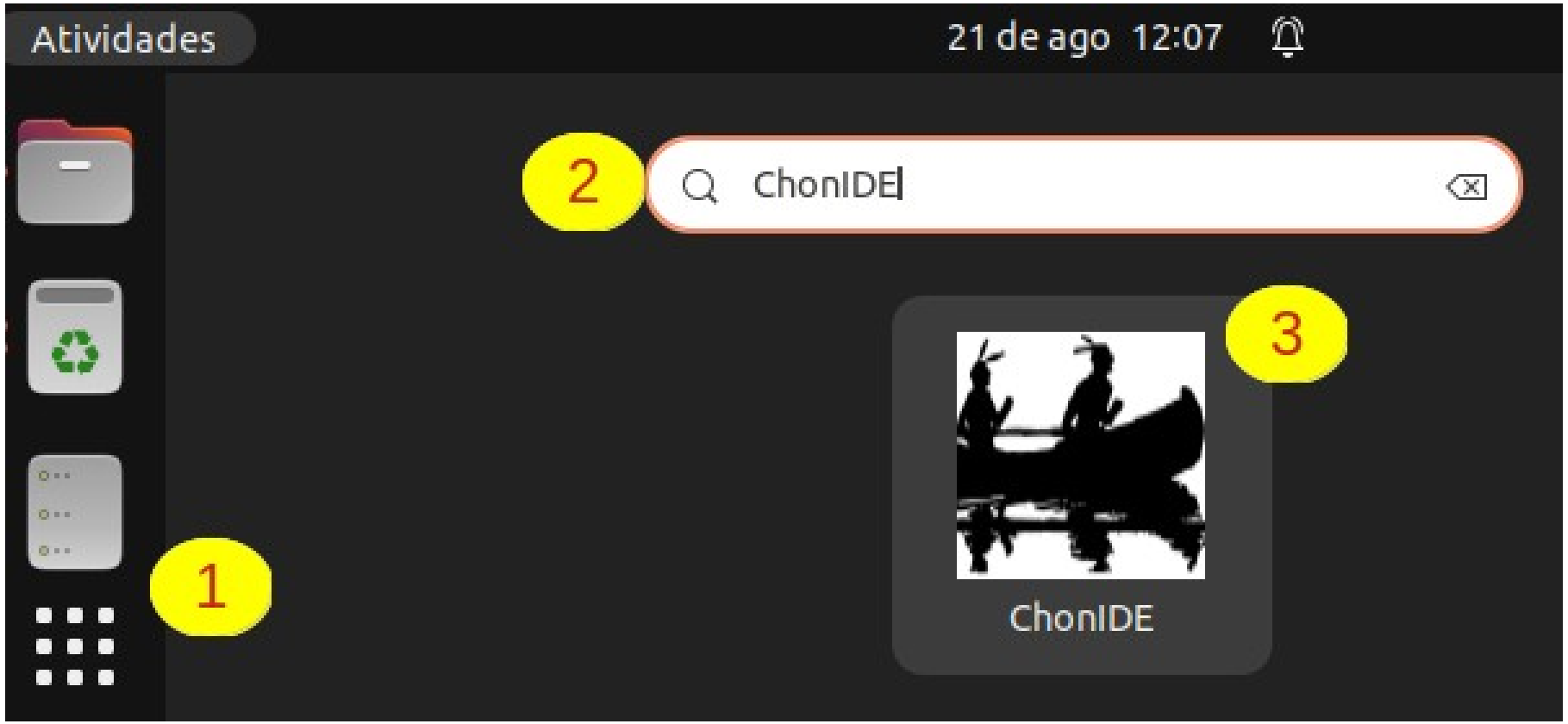


Souza de Jesus, V., Mori Lazarin, N., Pantoja, C.E., Vaz Alves, G., Ramos Alves de Lima, G., Viterbo, J. (2023). **An IDE to Support the Development of Embedded Multi-Agent Systems**. In: Mathieu, P., Dignum, F., Novais, P., De la Prieta, F. (eds) Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection. PAAMS 2023. Lecture Notes in Computer Science(), vol 13955. Springer, Cham. https://doi.org/10.1007/978-3-031-37616-0_29

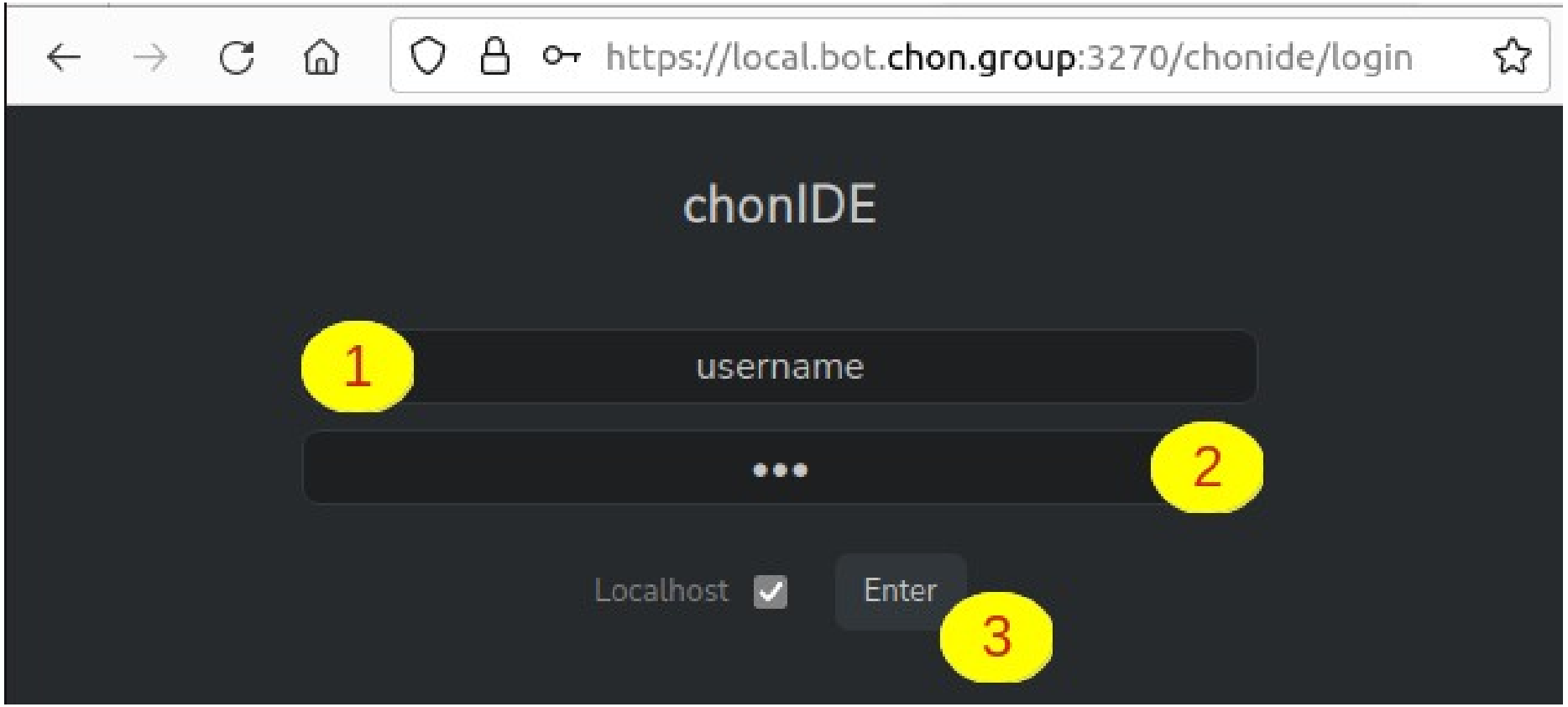
REASONING LAYER USING CHONIDE



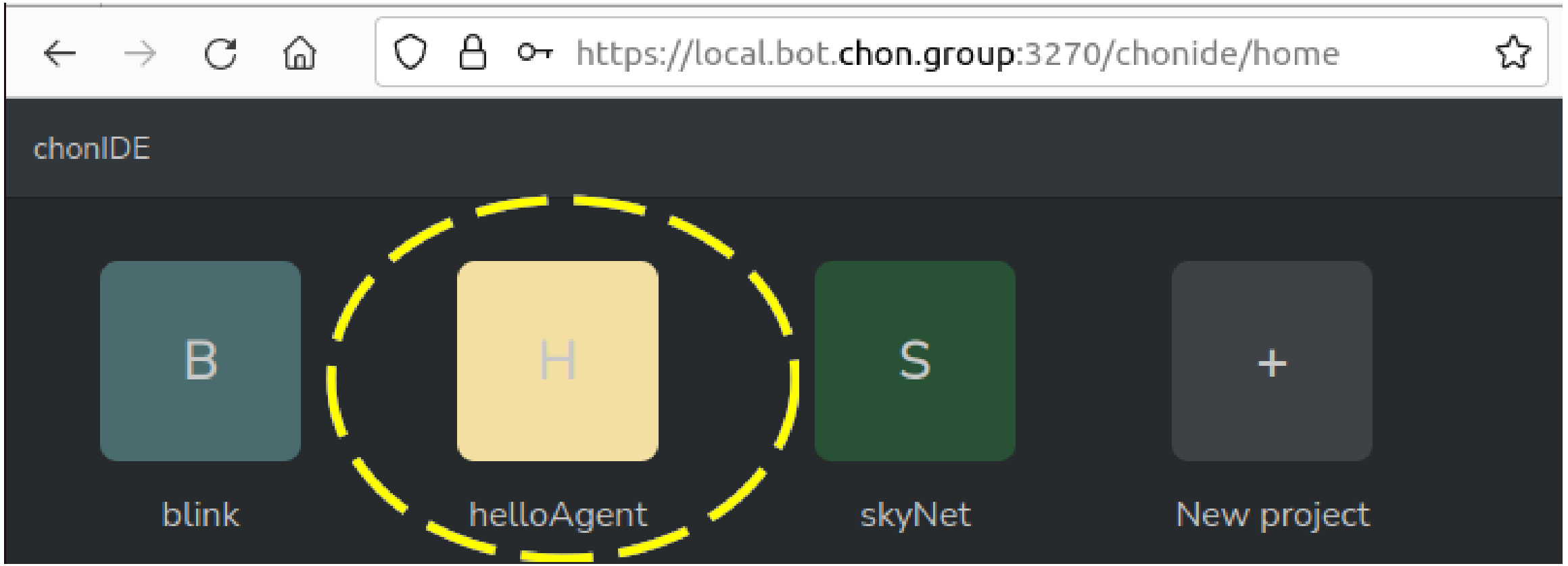
ChonIDE: Hello World



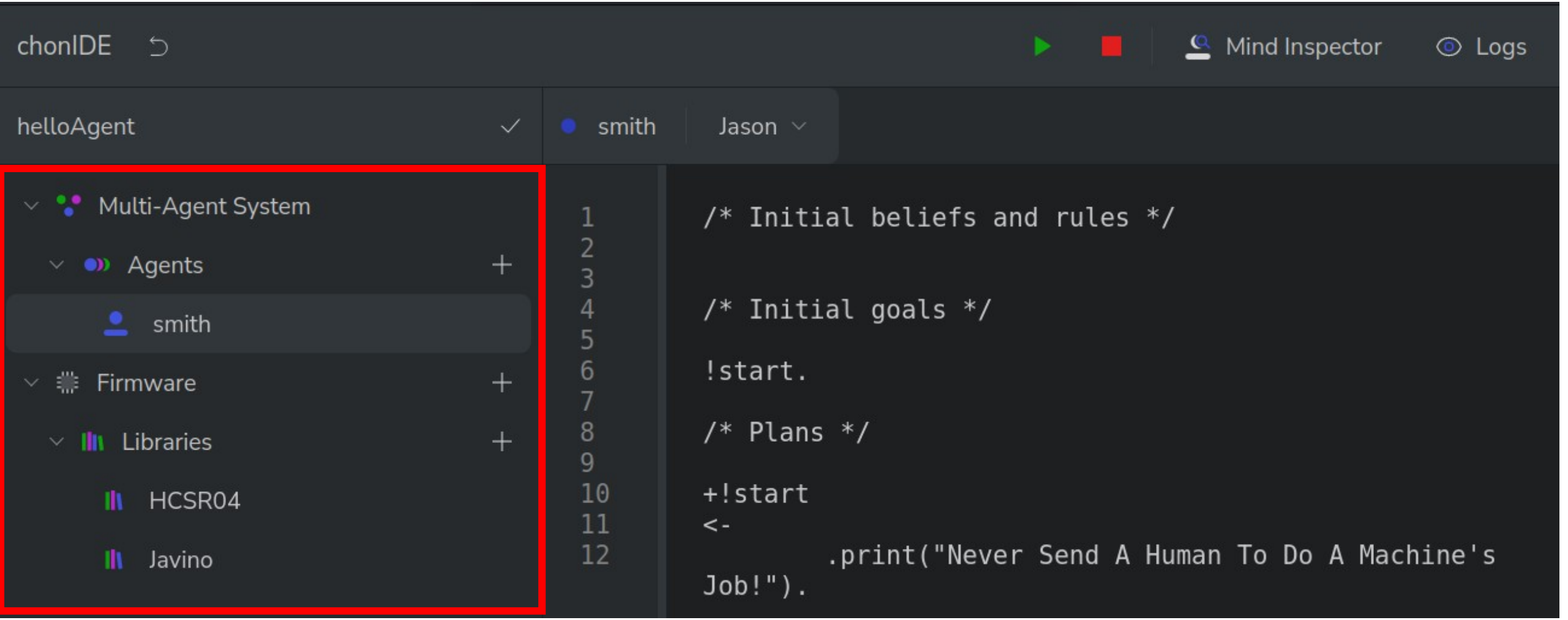
ChonIDE: Hello World



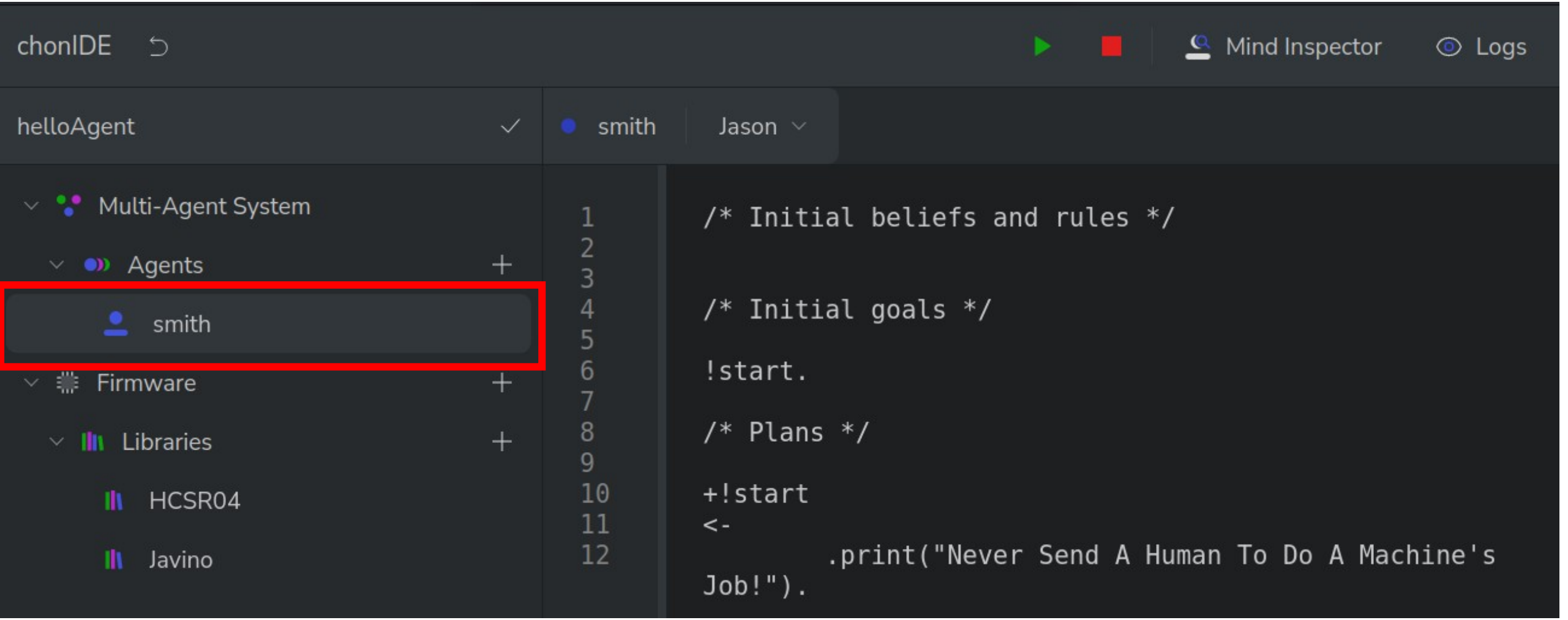
ChonIDE: Hello World



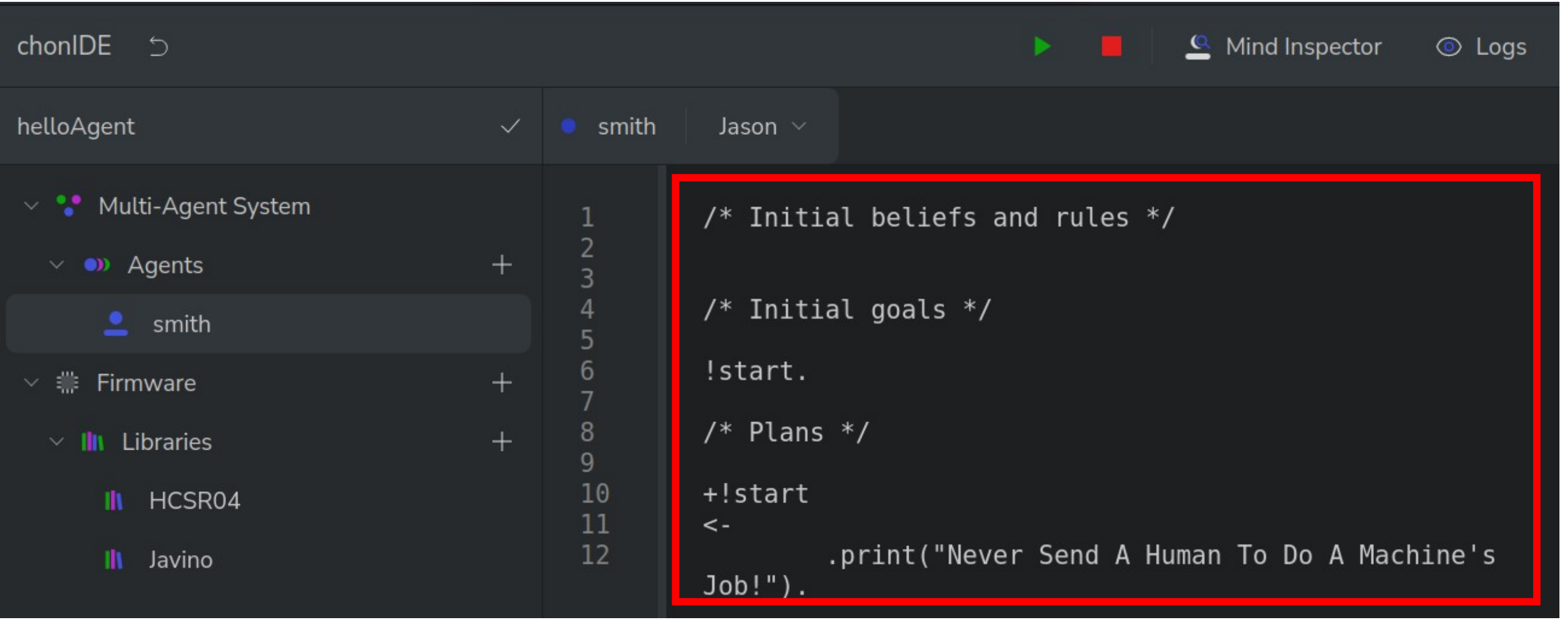
ChonIDE: Hello World



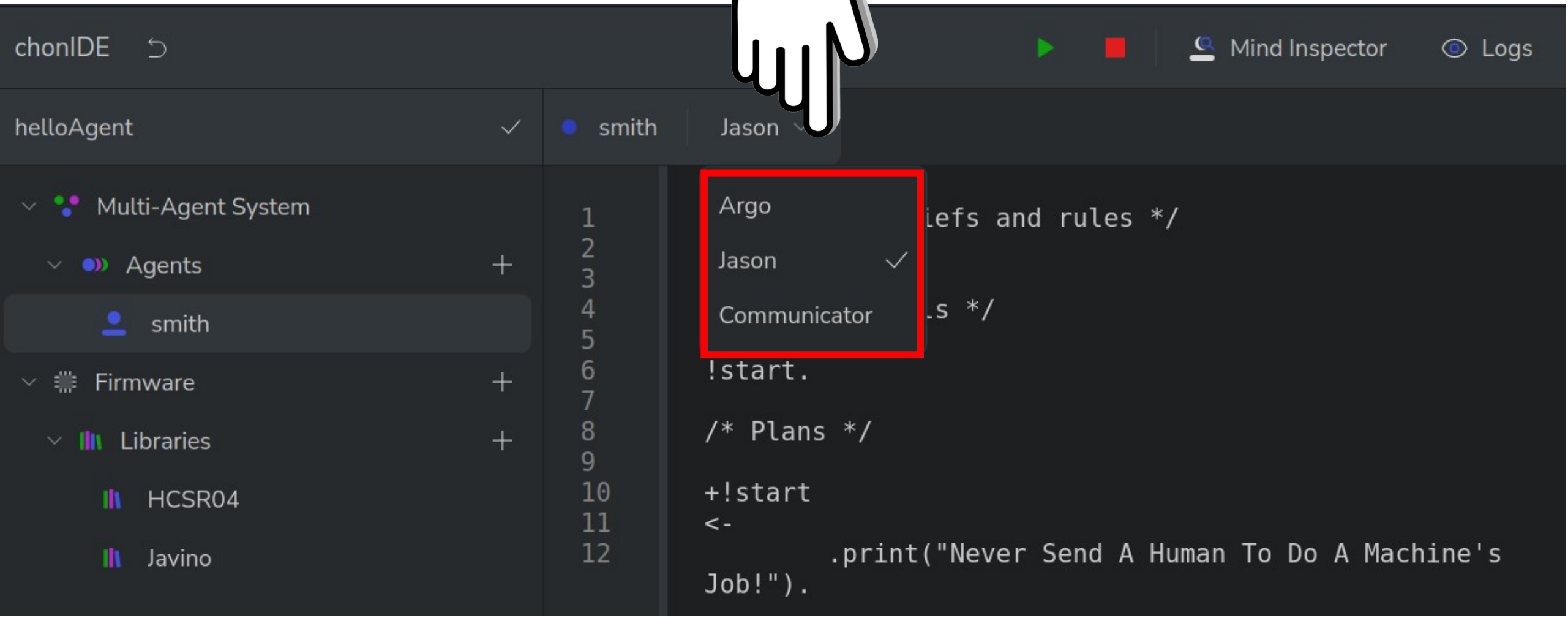
ChonIDE: Hello World



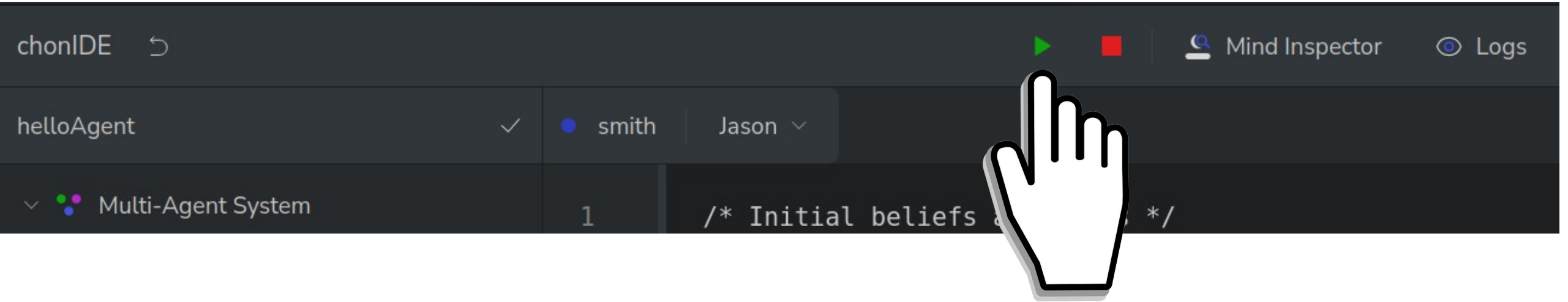
ChonIDE: Hello World



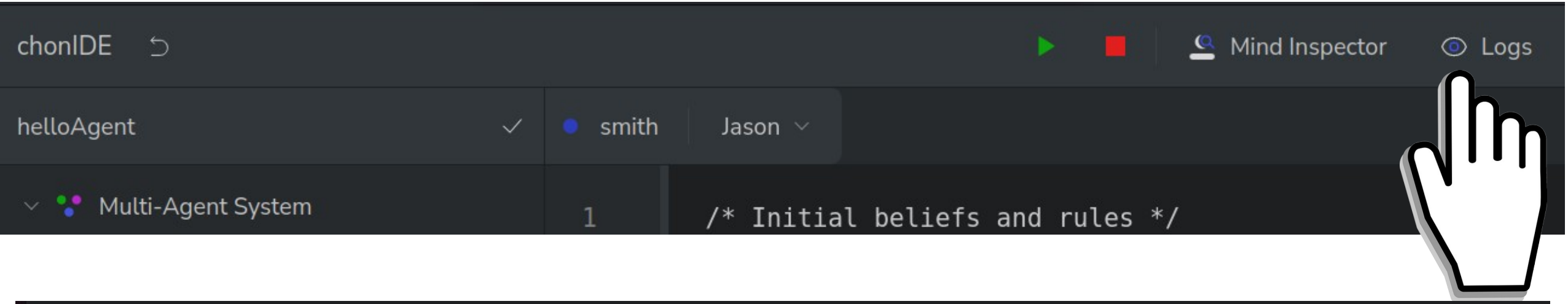
ChonIDE: Hello World



ChonIDE: Hello World

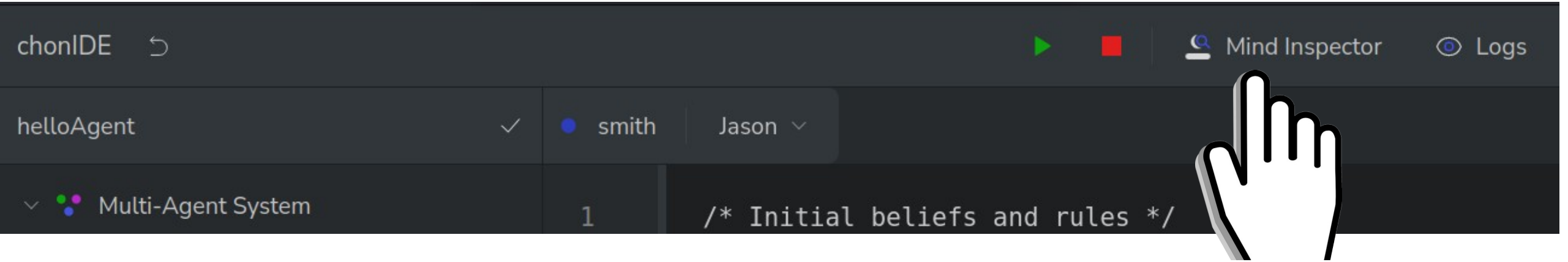


ChonIDE: Hello World




```
[ChonOS EmbeddedMAS] Starting the Multi-Agent System.
NOTE: Picked up JDK_JAVA_OPTIONS:  --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.
--add-opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED
Jason Http Server running on http://192.168.0.111:3272
[smith] Never Send A Human To Do A Machine's Job!
```

ChonIDE: Hello World



Agents

- smith

by Jason

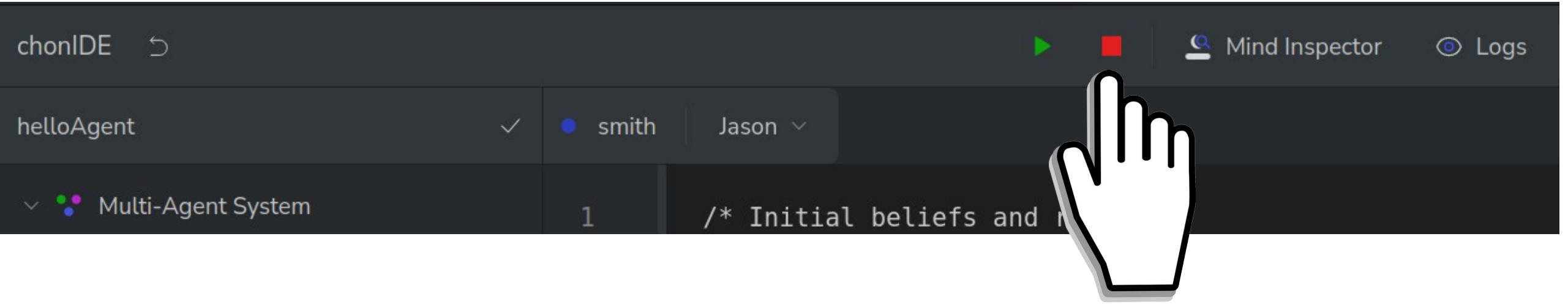
Inspection of agent **smith**

- **Beliefs**

belief(another)[source(self)].
oneBelief[source(self)].

- **Annotations**

ChonIDE: Hello World




```
[ChonOS EmbeddedMAS] Stopping the Multi-agent System Gently.
[ChonOS EmbeddedMAS] Multi-Agent System Successfully Terminated!
```

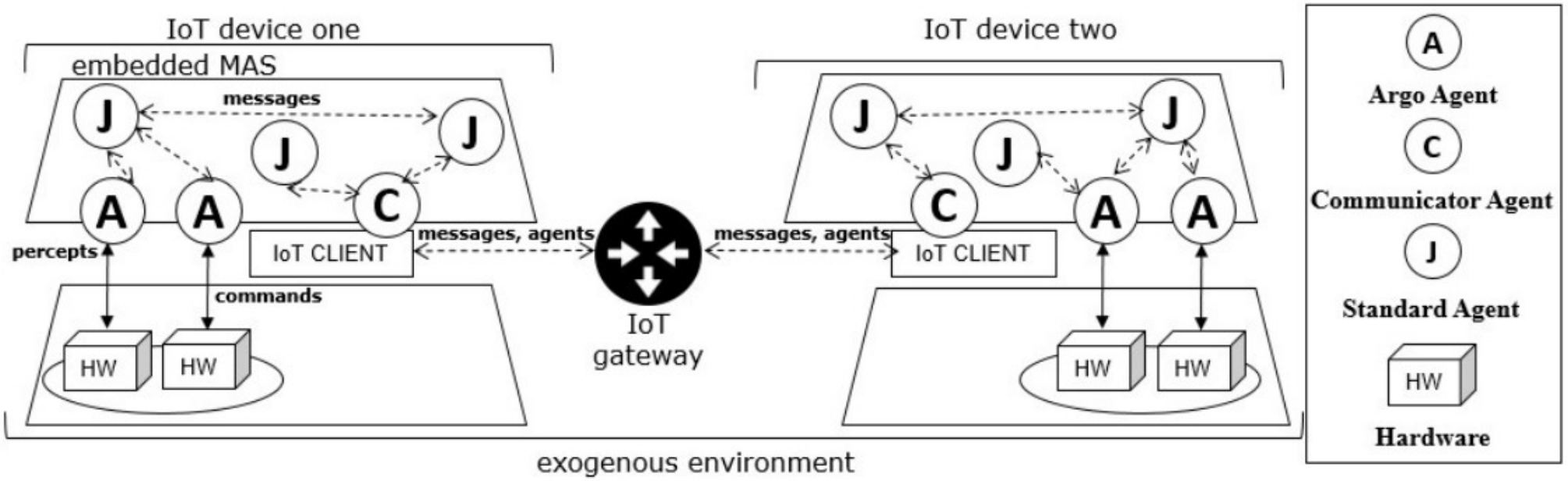
PROGRAMMING EMBEDDED MAS

Jason Embedded

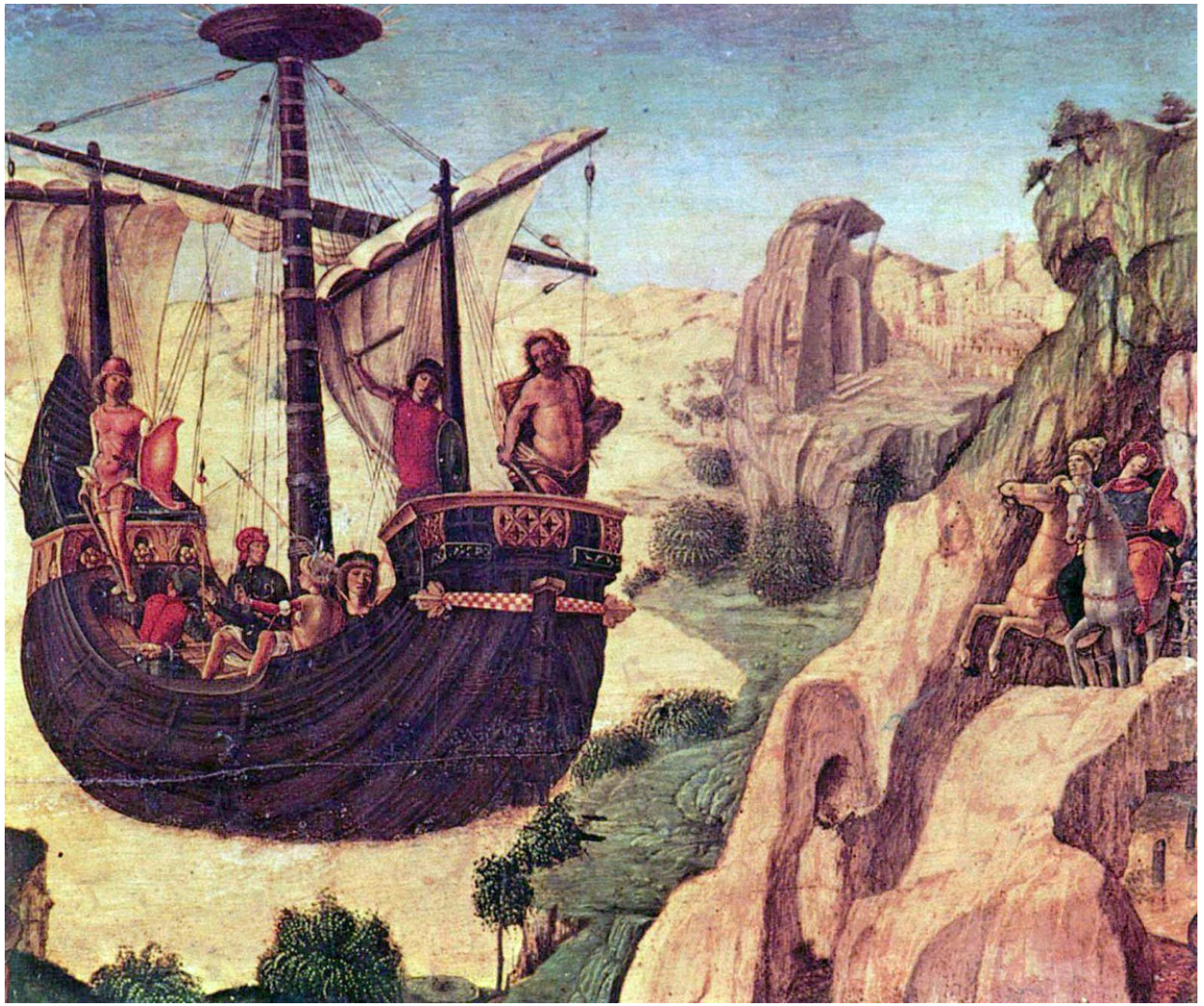
It is a spin-off Version of Jason for IoT and Embedded Multiagent Systems.

Jason Embedded

It is a spin-off Version of Jason for IoT and Embedded Multiagent Systems.



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

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

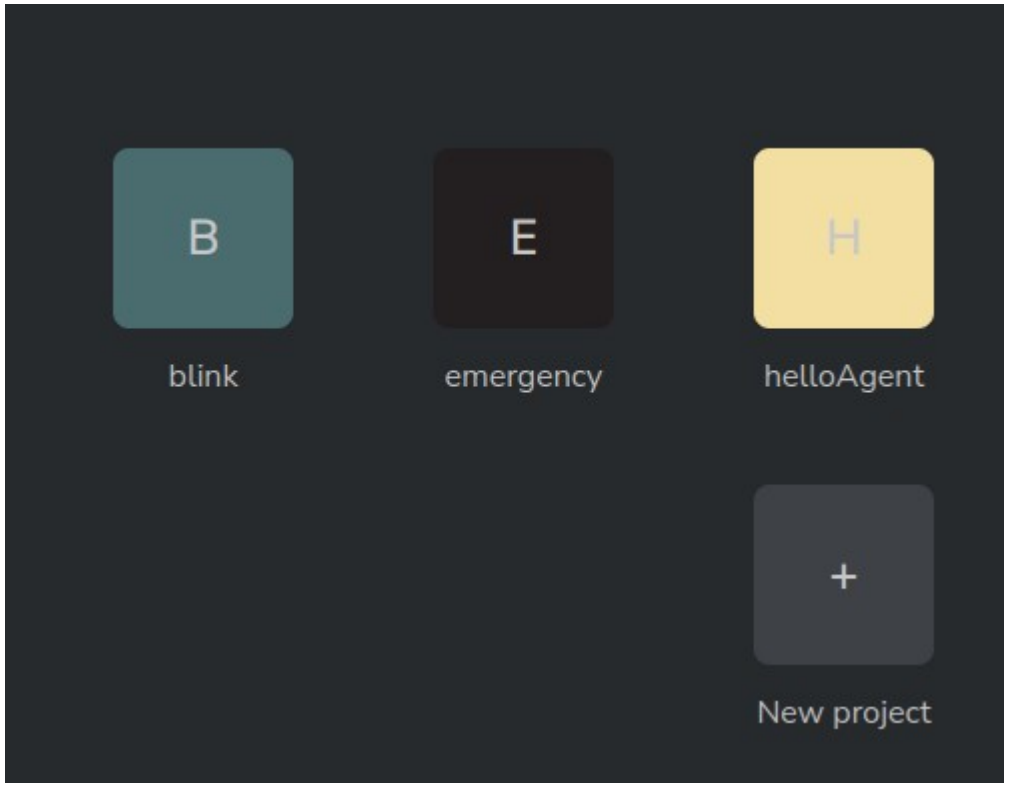
O **ARGO** permite:

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.**

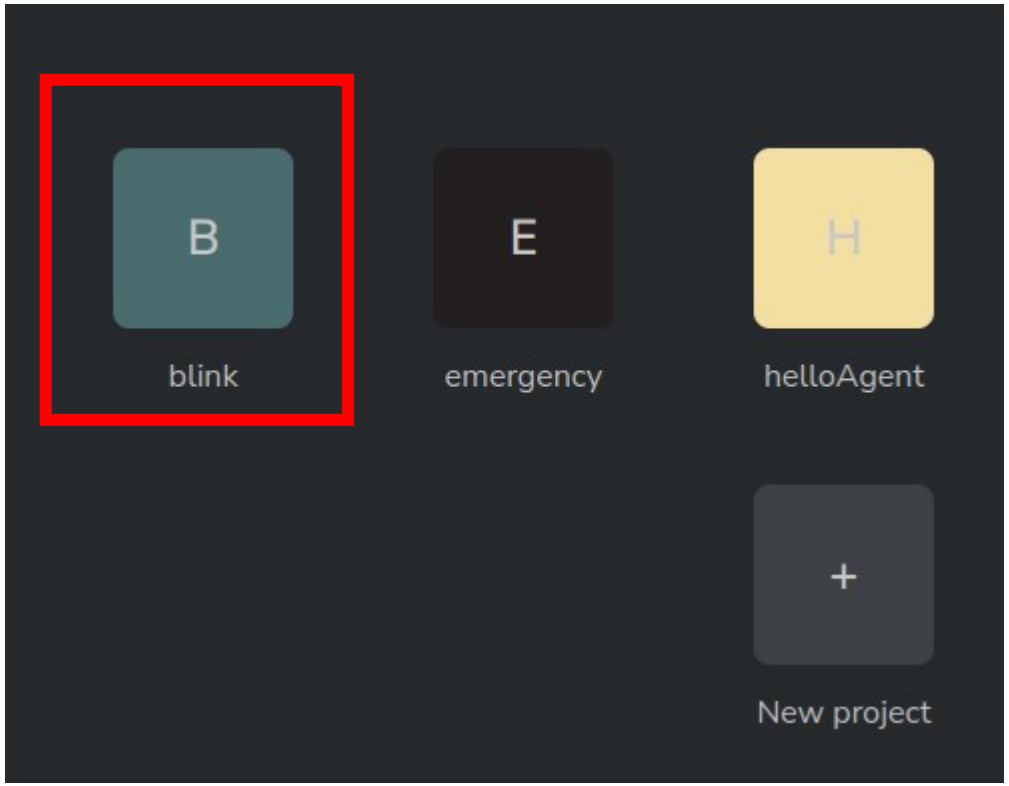
Argo for Jason

- **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|close): 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: blink



Exemplo: blink

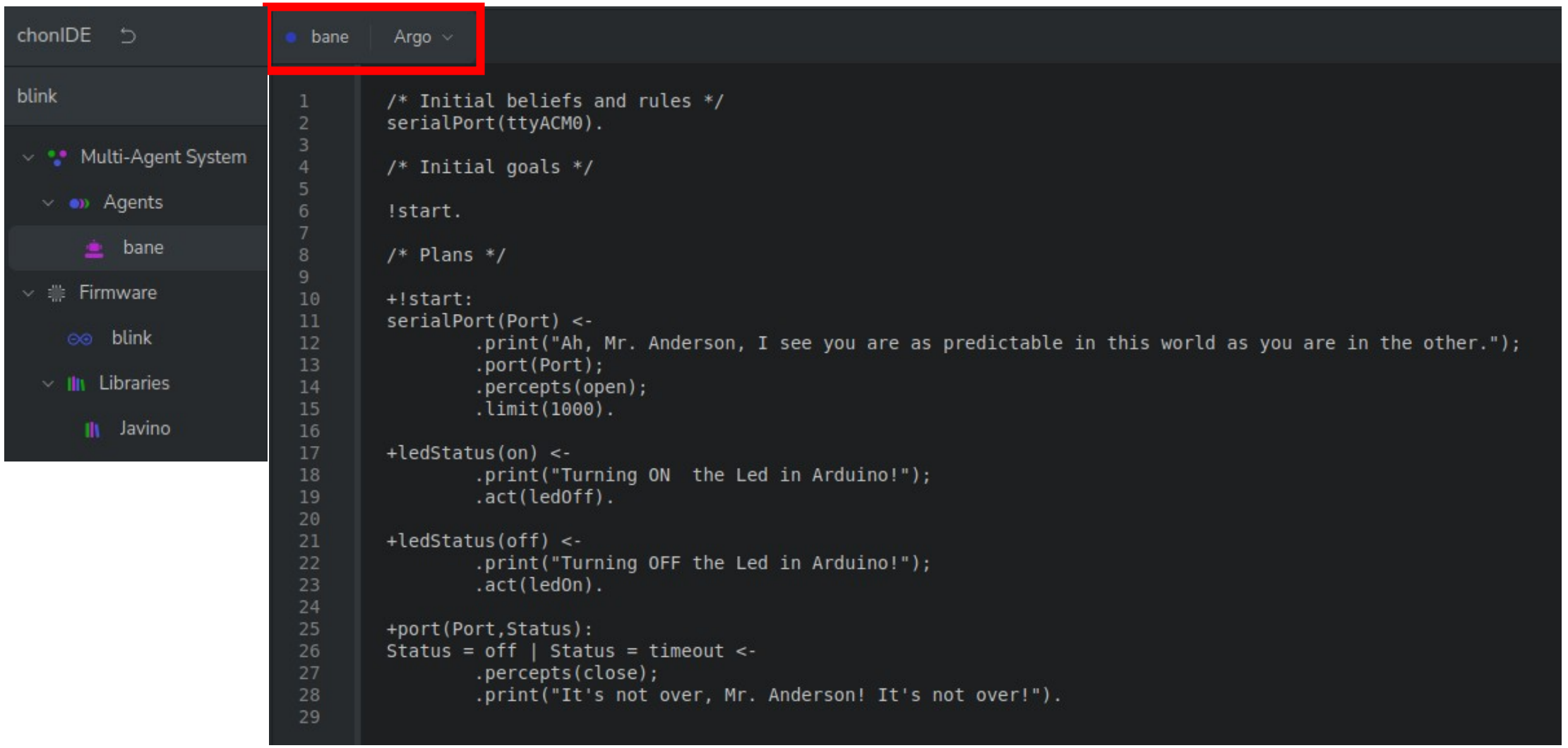


Exemplo: blink

bane | Argo

```
1   /* Initial beliefs and rules */
2   serialPort(ttyACM0).
3
4   /* Initial goals */
5
6   !start.
7
8   /* Plans */
9
10  +!start:
11  serialPort(Port) <-
12          .print("Ah, Mr. Anderson, I see you are as predictable in this world as you are in the other.");
13          .port(Port);
14          .percepts(open);
15          .limit(1000).
16
17  +ledStatus(on) <-
18          .print("Turning ON  the Led in Arduino!");
19          .act(ledOff).
20
21  +ledStatus(off) <-
22          .print("Turning OFF the Led in Arduino!");
23          .act(ledOn).
24
25  +port(Port,Status):
26  Status = off | Status = timeout <-
27          .percepts(close);
28          .print("It's not over, Mr. Anderson! It's not over!").
29
```

Exemplo: blink




```
/* Initial beliefs and rules */
serialPort(ttyACM0).

/* Initial goals */

!start.

/* Plans */

+!start:
serialPort(Port) <-
        .print("Ah, Mr. Anderson, I see you are as predictable in this world as you are in the other.");
        .port(Port);
        .percepts(open);
        .limit(1000).

+ledStatus(on) <-
        .print("Turning ON  the Led in Arduino!");
        .act(ledOff).

+ledStatus(off) <-
        .print("Turning OFF the Led in Arduino!");
        .act(ledOn).

+port(Port,Status):
Status = off | Status = timeout <-
        .percepts(close);
        .print("It's not over, Mr. Anderson! It's not over!").

```

Exemplo: blink

bane | Argo

```
1   /* Initial beliefs and rules */
2   serialPort(ttyACM0).
3
4   /* Initial goals */
5
6   !start.
7
8   /* Plans */
9
10  +!start:
11  serialPort(Port) <-
12          .print("Ah, Mr. Anderson, I see you are as predictable in this world as you are in the other.");
13          .port(Port);
14          .percepts(open);
15          .limit(1000).
16
17  +ledStatus(on) <-
18          .print("Turning ON  the Led in Arduino!");
19          .act(ledOff).
20
21  +ledStatus(off) <-
22          .print("Turning OFF the Led in Arduino!");
23          .act(ledOn).
24
25  +port(Port,Status):
26  Status = off | Status = timeout <-
27          .percepts(close);
28          .print("It's not over, Mr. Anderson! It's not over!").
29
```

Exemplo: blink

bane | Argo

```
1   /* Initial beliefs and rules */
2   serialPort(ttyACM0).
3
4   /* Initial goals */
5
6   !start.
7
8   /* Plans */
9
10  +!start:
11  serialPort(Port) <-
12          .print("Ah, Mr. Anderson, I see you are as predictable in this world as you are in the other.");
13          .port(Port);
14          .percepts(open);
15          .limit(1000).
16
17  +ledStatus(on) <-
18          .print("Turning ON  the Led in Arduino!");
19          .act(ledOff).
20
21  +ledStatus(off) <-
22          .print("Turning OFF the Led in Arduino!");
23          .act(ledOn).
24
25  +port(Port,Status):
26  Status = off | Status = timeout <-
27          .percepts(close);
28          .print("It's not over, Mr. Anderson! It's not over!").
29
```

Exemplo: blink

bane | Argo

```
1   /* Initial beliefs and rules */
2   serialPort(ttyACM0).
3
4   /* Initial goals */
5
6   !start.
7
8   /* Plans */
9
10  +!start:
11  serialPort(Port) <-
12          .print("Ah, Mr. Anderson, I see you are as predictable in this world as you are in the other.");
13          .port(Port);
14          .percepts(open);
15          .limit(1000).
16
17  +ledStatus(on) <-
18          .print("Turning ON  the Led in Arduino!");
19          .act(ledOff).
20
21  +ledStatus(off) <-
22          .print("Turning OFF the Led in Arduino!");
23          .act(ledOn).
24
25  +port(Port,Status):
26  Status = off | Status = timeout <-
27          .percepts(close);
28          .print("It's not over, Mr. Anderson! It's not over!").
29
```

Exemplo: blink

bane | Argo

```
1   /* Initial beliefs and rules */
2   serialPort(ttyACM0).
3
4   /* Initial goals */
5
6   !start.
7
8   /* Plans */
9
10  +!start:
11  serialPort(Port) <-
12          .print("Ah, Mr. Anderson, I see you are as predictable in this world as you are in the other.");
13          .port(Port);
14          .percepts(open);
15          .limit(1000).
16
17  +ledStatus(on) <-
18          .print("Turning ON  the Led in Arduino!");
19          .act(ledOff).
20
21  +ledStatus(off) <-
22          .print("Turning OFF the Led in Arduino!");
23          .act(ledOn).
24
25  +port(Port,Status):
26  Status = off | Status = timeout <-
27          .percepts(close);
28          .print("It's not over, Mr. Anderson! It's not over!").
29
```

Exemplo: blink

blink | Compile | Deploy

```
#include <Javino.h>
Javino javino;

void serialEvent(){
 /*
 * The serialEvent() function handles interruptions coming from the serial port.
 *
 * NOTE: The serialEvent() feature is not available on the Leonardo, Micro, or other ATmega32U4 based boards.
 * https://docs.arduino.cc/built-in-examples/communication/SerialEvent
 *
 */
  javino.readSerial();
}

void setup() {
 javino.start(9600);
 pinMode(13,OUTPUT);
}

void loop() {
 if(javino.availableMsg()){
  if(javino.getMsg() == "getPercepts")javino.sendMsg(getPercepts());
  else if(javino.getMsg() == "ledOn") ledOn();
  else if(javino.getMsg() == "ledOff")ledOff();
 }
}

/* It sends the exogenous environment's perceptions to the agent. */
String getPercepts(){
  String beliefs =
    "resourceName(myArduino);"+
    getLedStatus();
```

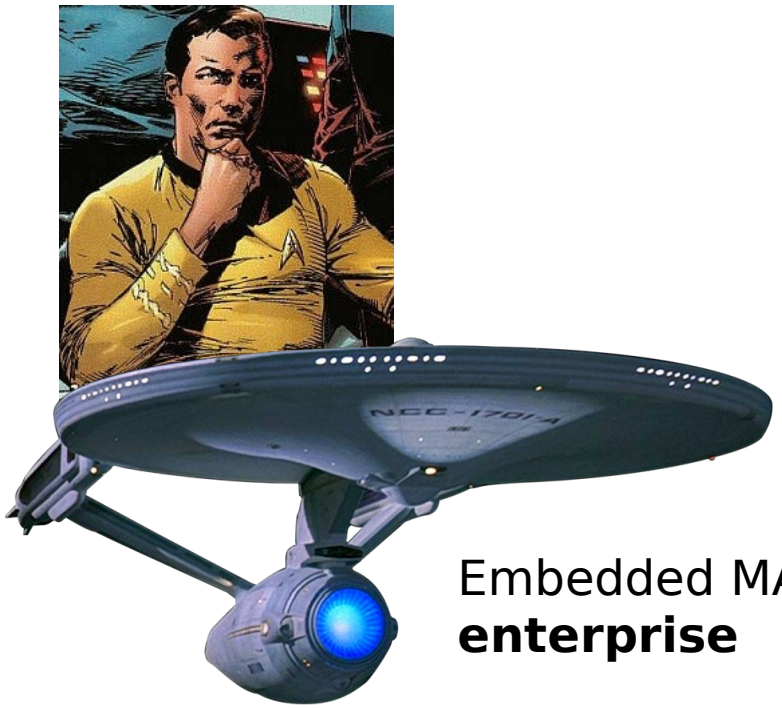
Exemplo: Argo

Exemplo: Argo

Embedded MAS
enterprise

Exemplo: Argo

agent **kirk**

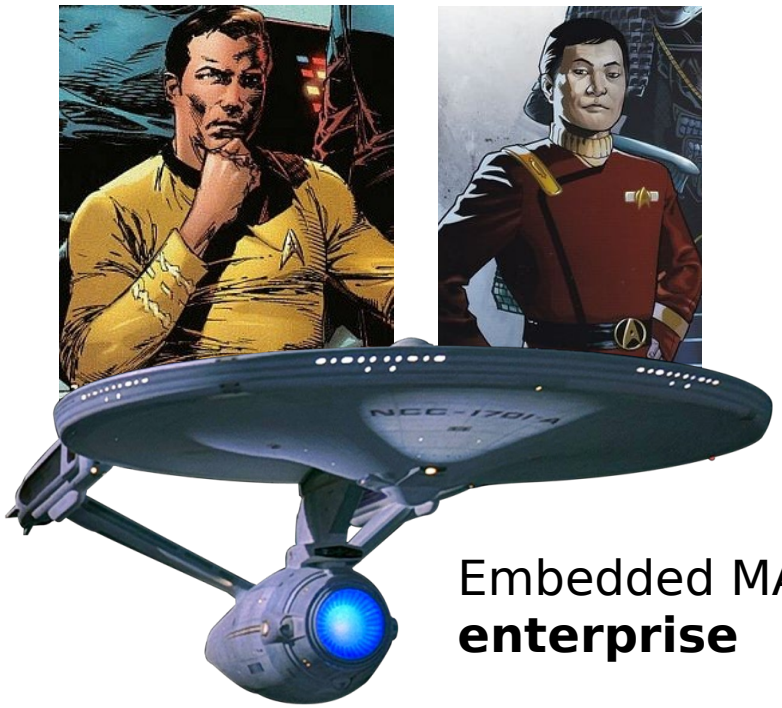


Embedded MAS
enterprise

Exemplo: Argo

agent **kirk**

agent **sulu**



Embedded MAS
enterprise

Exemplo: Argo

Exemplo: Argo

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()

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.").
17
```

kirk

```
1   /* Initial beliefs and rules */
2
3   /* Initial goals */
4   !start.
5
6   /* Plans */
7
8   +!start <-
9           .print("This is Comma
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
```

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).
11
```

latest state 2 1 clear history

Inspection of agent **sulu**

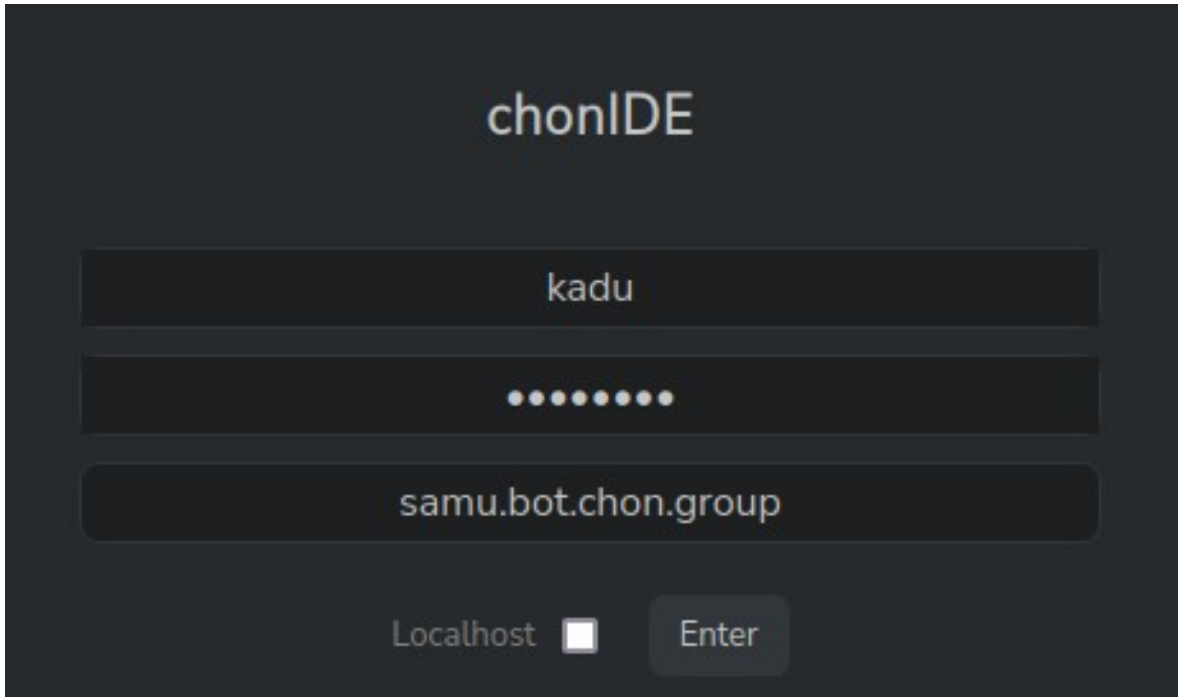
- **Beliefs**

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)].

ATO 2: ESCOLHA DO SEU BOT

Escolha dos Bots

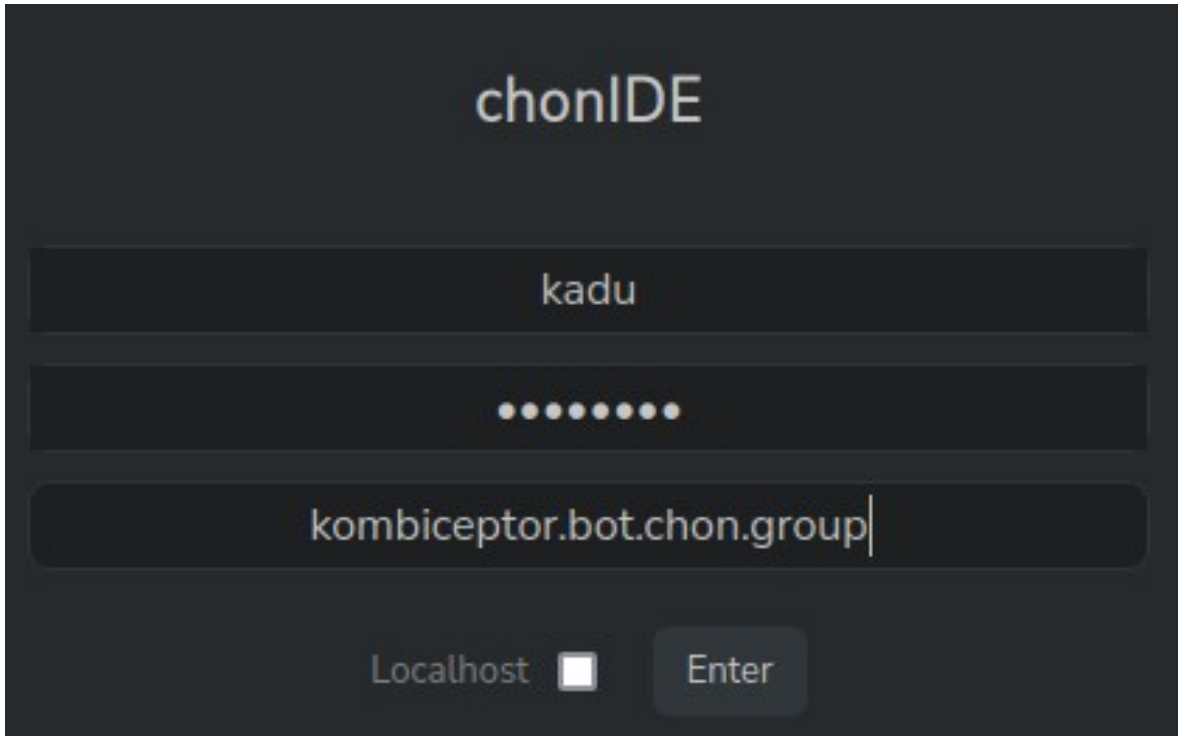
SAMU



https://sourceforge.net/p/chonos/examples/ci/master/tree/05-embeddedMultiAgent/prototypes/samu/

Escolha dos Bots

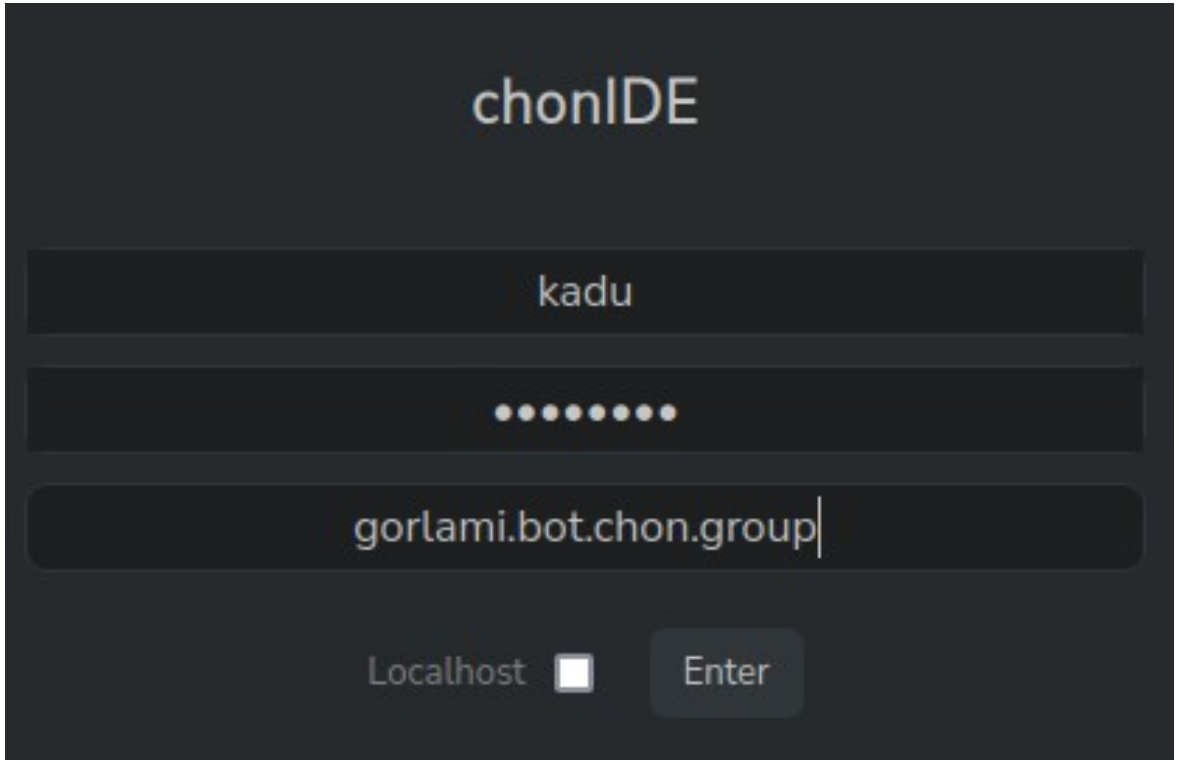
KOMBICEPTOR



https://sourceforge.net/p/chonos/examples/ci/master/tree/05-embeddedMultiAgent/prototypes/kombiceptor/

Escolha dos Bots

GORLAMI



https://sourceforge.net/p/chonos/examples/ci/master/tree/05-embeddedMultiAgent/prototypes/kombiceptor/

GORLAMI

Code | Blame | 25 lines (22 loc) · 879 Bytes

```
1   // Agent bob in project jasonProject
2   /* Initial beliefs and rules */
3   serialPort(ttyACM0).
4   /* Initial goals */
5   !start.
6
7   /* Plans */
8   +!start : true <-
9       .print("hello world.");
10      ?serialPort(SP);
11      argo.port(SP);
12      argo.percepts(open).
13
```

```
13
14  +motor(M) <-    .print("Motor Status -> ",M).
15  +flashLight(FL) <- .print("FlashLight Status -> ",FL).
16  +light(L) <- .print("Light Status -> ",L).
17  +buzzer(B)  <- .print("Buzzer Status -> ",B).
18  +speed(S) <- .print("Speed Status -> ",S).
19  +breakL(BL) <- .print("Break Light Status -> ",BL).
20  +luminosity(Lu) <- .print("Luminosity Status -> ",Lu).
21  +distance(D) <- .print("Distance Status -> ",D).
22  +lineL(LL) <- .print("Line-following Left Status -> ",LL).
23  +lineR(LR) <- .print("Line-following Right Status -> ",LR).
24
25  +port(P,S): S=off & serialPort(SP) <- .print("Serial Port ",SP, " offline!").
```

https://github.com/chon-group/bot2WD/blob/main/reasoning/jasonProject/src/agt/driver.asl

Tarefa Argo

“Criar um agente **Argo** que ande e evite colisão (pare ou vire) com qualquer obstáculo que estiver a 20 centímetros.”

“Depois faça com que o protótipo só ande acionado por luz.”

percepções disponíveis

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

ações disponíveis

- goAhead
- stop
- goLeft
- goRight

Agradecimentos

OBRIGADO!

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