

Introduction to Distributed and Embedded Multi-Agent Systems

Nilson Mori Lazarin

Centro Federal de Educação Tecnológica (CEFET/RJ)

Universidade Federal Fluminense (UFF), Brasil

<https://semeso.github.io/2024/minicursos/>



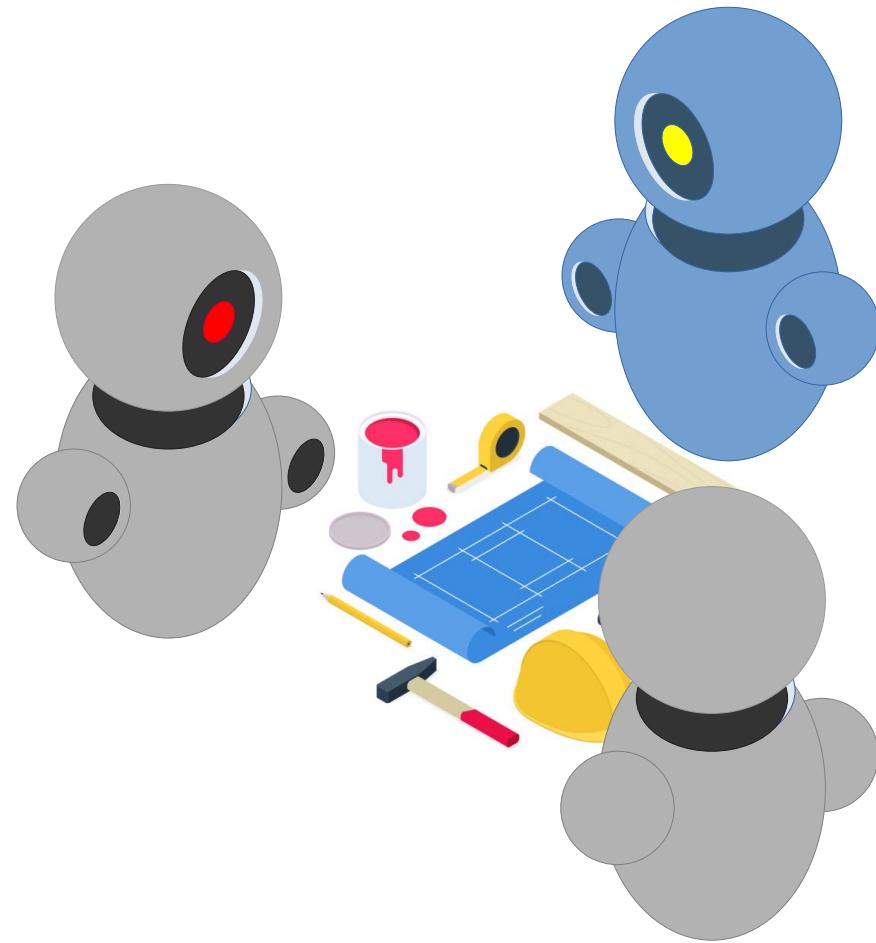
UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

ALTO VALE

CENTRO DE EDUCAÇÃO SUPERIOR
DO ALTO VALE DO ITAJAÍ



INTRODUCTION



Who I am?



Prof. Me. Nilson Lazarin

Cefet/RJ - UFF

 **Cognitive Hardware on Networks (Chon)
Research Group**
<https://chon.group>

Where we have been?

This workshop is part of:

- **Undergraduate course**
 - CEFET/RJ Maria da Graça and Nova Friburgo
- **Graduate course**
 - M.Sc. and Ph.D. at UFF



It was **presented at**:

IFSP, UFPel, UFRJ, UFSC, UnB, UTFPR and Estácio;

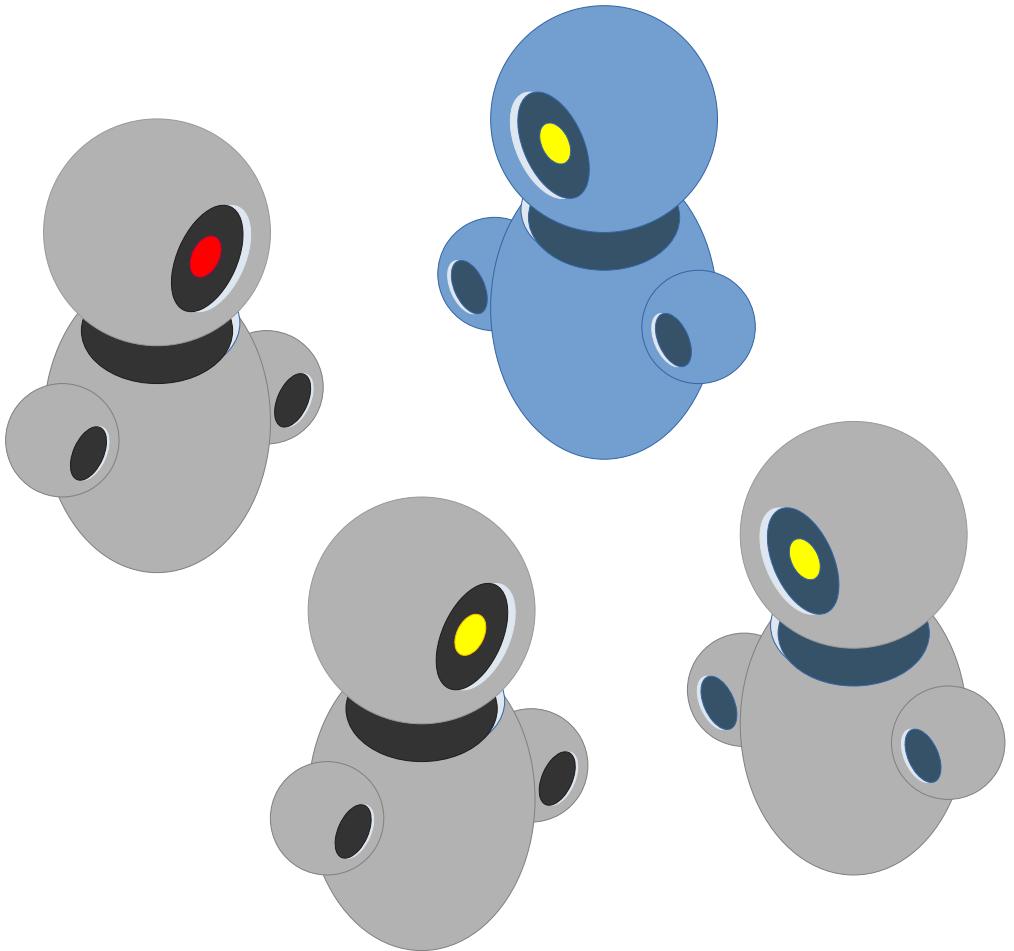
<https://workshops.chon.group>

Goals

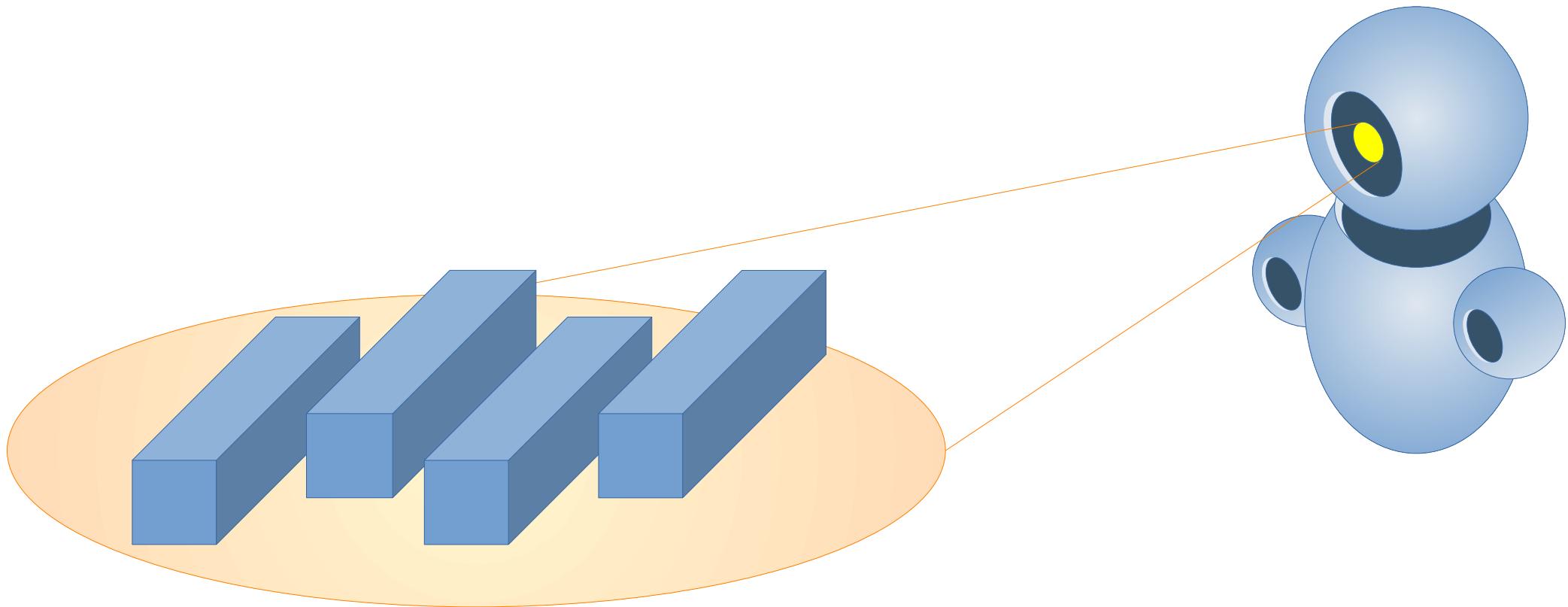
This workshop goals are:

- **To experience** the development process of **Embedded MAS** using BDI-agents and Jason.
- to learn how to **collaborate** using IoT in **distributed** scenarios.

THE AGENT ORIENTED APPROACH

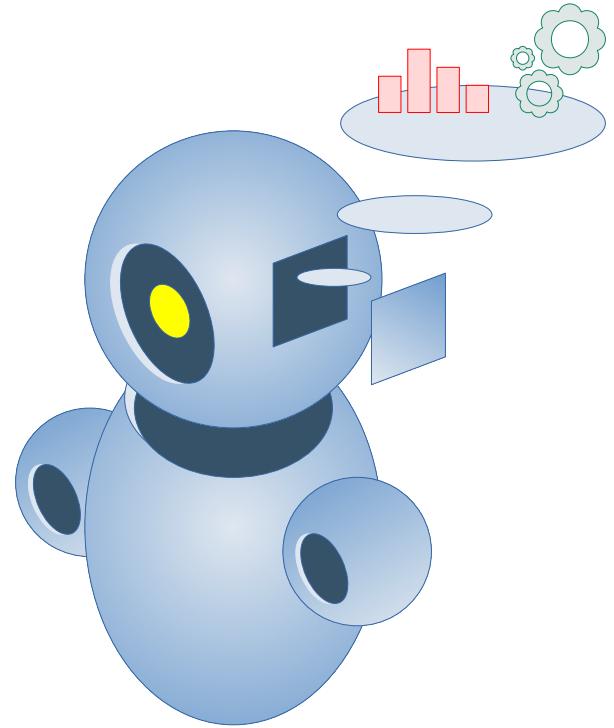
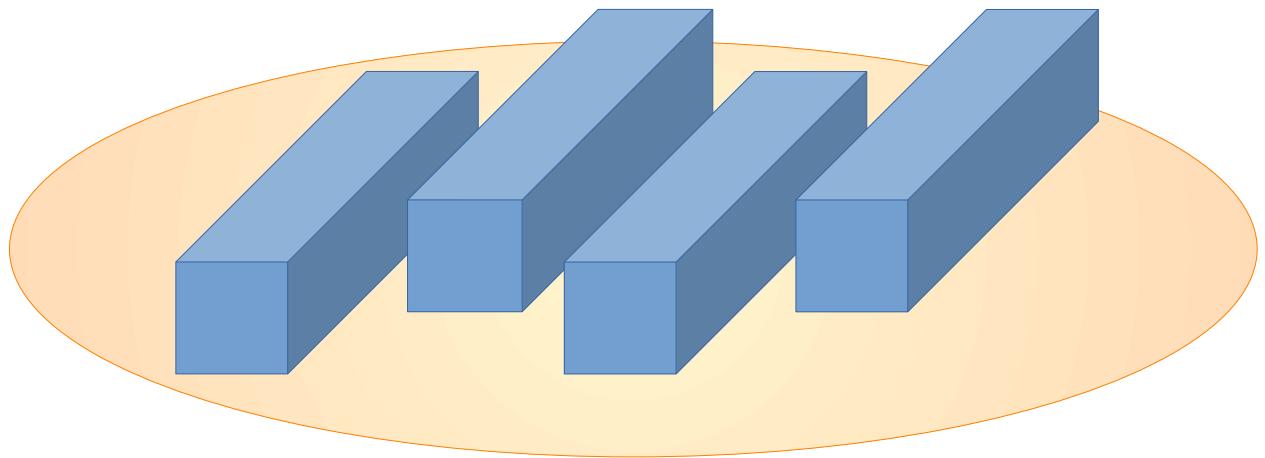


Agent



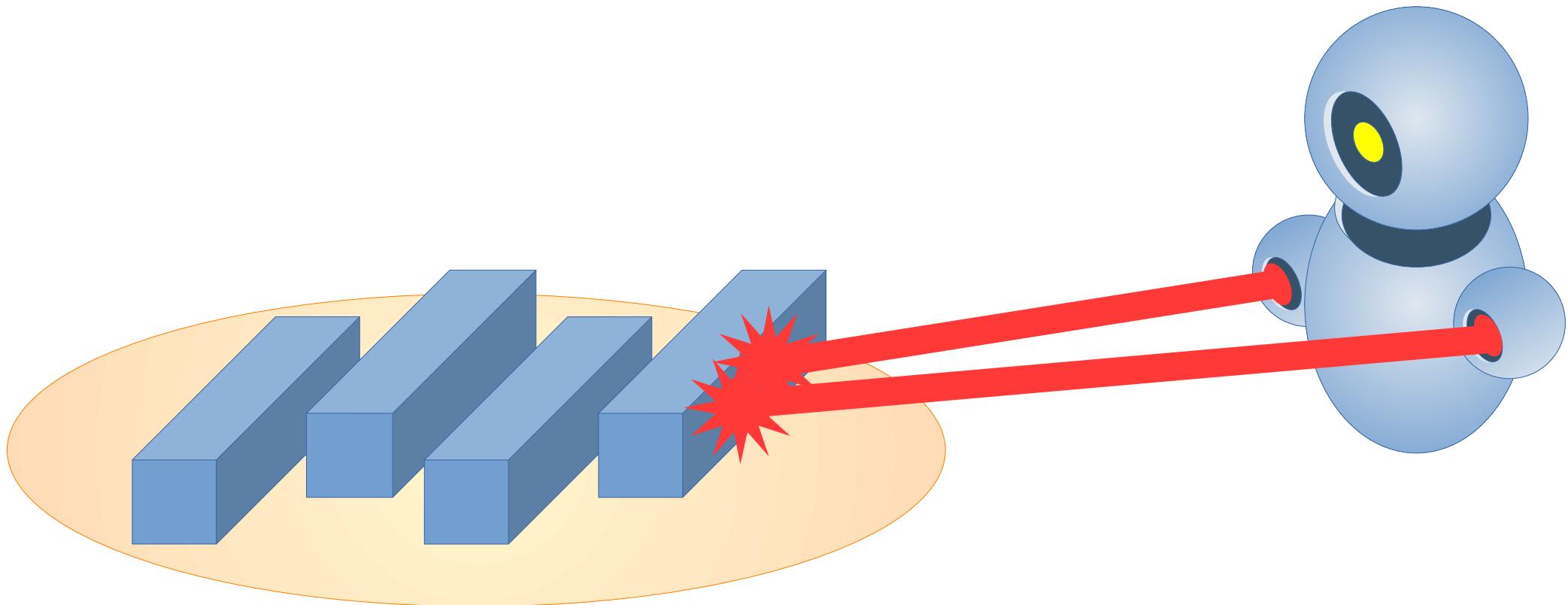
WOOLDRIDGE, Michael. Intelligent Agents. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Cambridge, MA, USA: MIT Press, 1999. p. 27–77
MICHEL, Fabien; FERBER, Jacques; DROGOUL, Alexis. Multi-Agent Systems and Simulation: a Survey From the Agents Community's Perspective.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

Agent



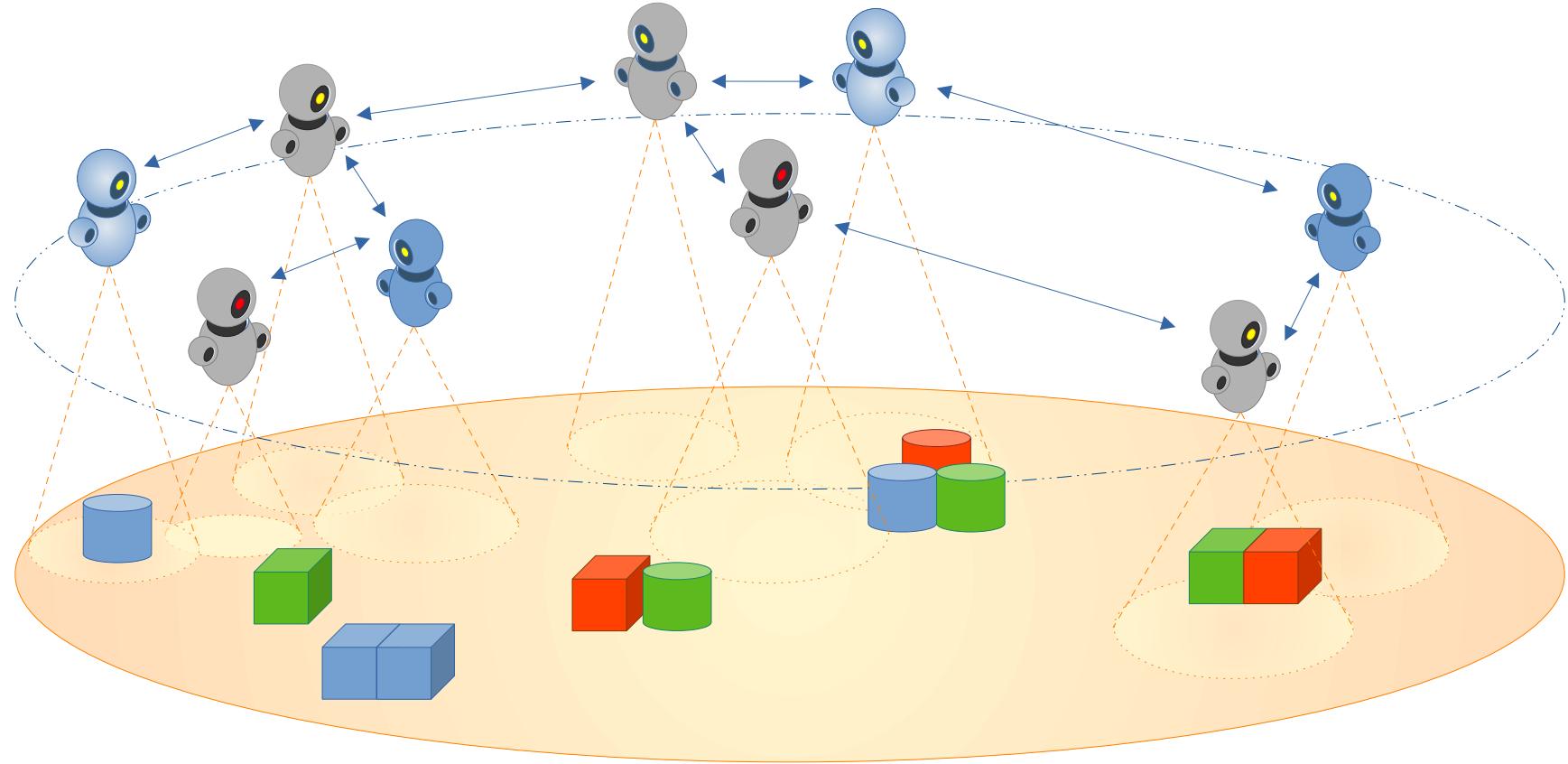
WOOLDRIDGE, Michael. Intelligent Agents. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Cambridge, MA, USA: MIT Press, 1999. p. 27–77
MICHEL, Fabien; FERBER, Jacques; DROGOUL, Alexis. Multi-Agent Systems and Simulation: a Survey From the Agents Community's Perspective.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

Agent



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

Multi-agent System

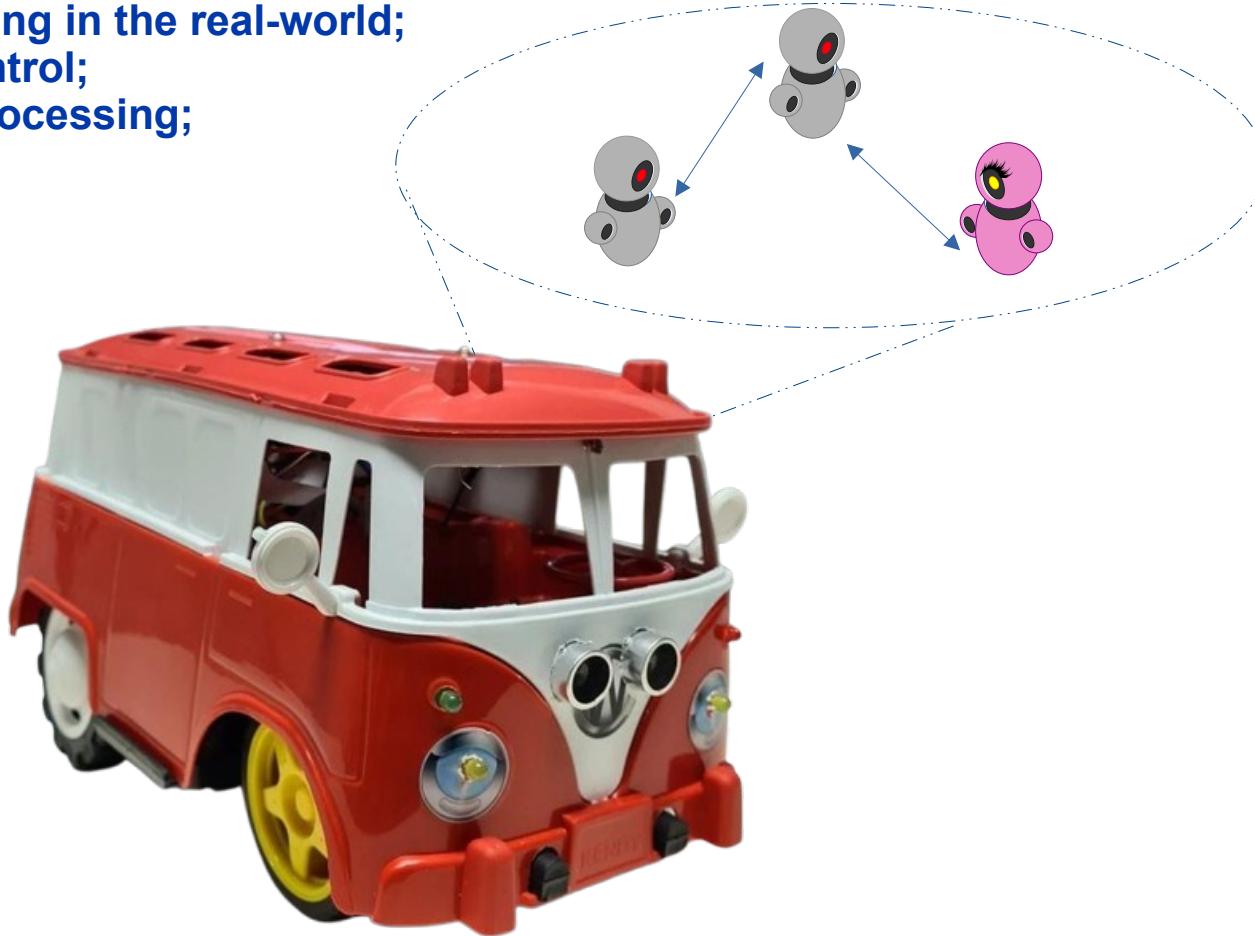


WOOLDRIDGE, Michael. Intelligent Agents. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Cambridge, MA, USA: MIT Press, 1999. p. 27–77

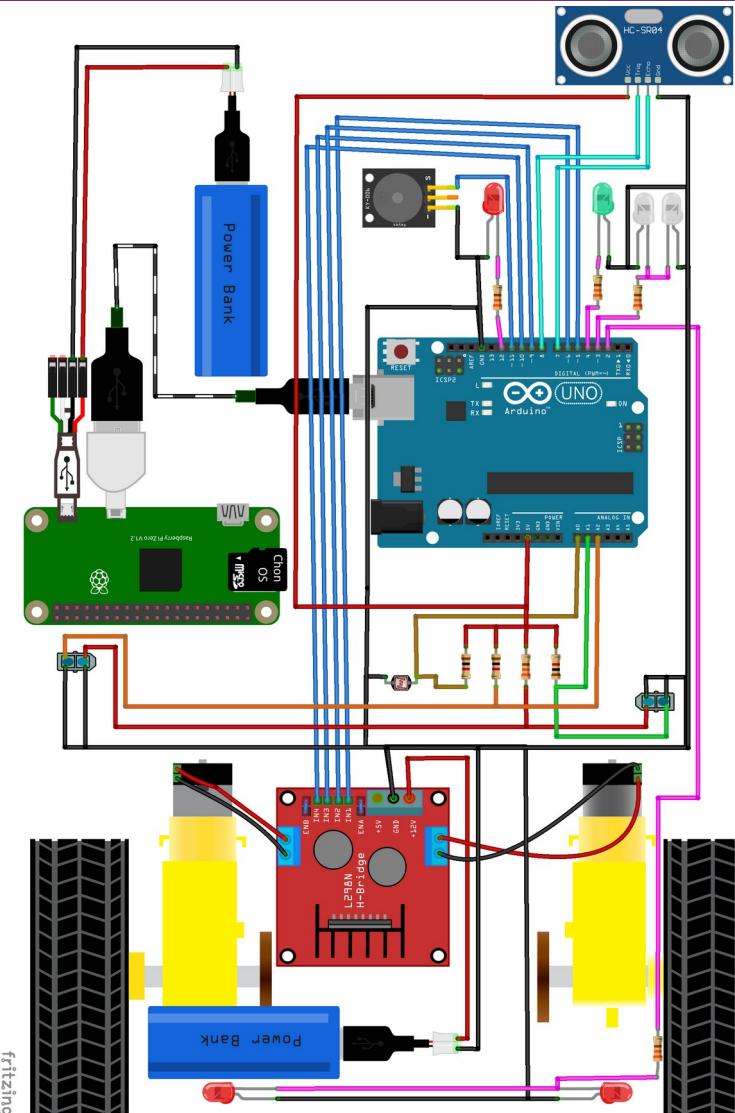
Embedded Multi-agent System

Attributes:

- 1) Hosted in a cyber-physical system with all sensors and actuator;
- 2) Acting and perceiving in the real-world;
- 3) Without remote control;
- 4) Without external processing;

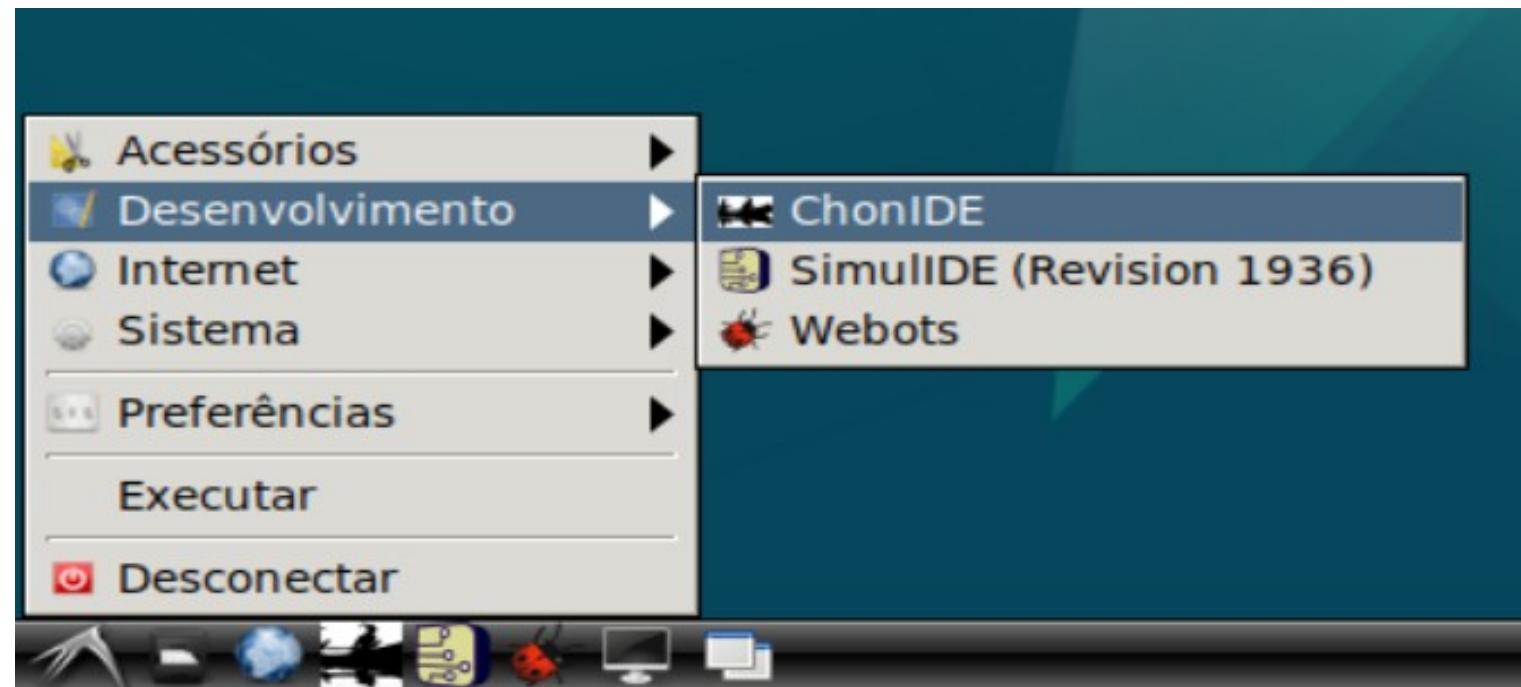


Low-cost 2WD prototype model for Embedded Multi-agent Systems <https://bots.chon.group>



THE DEVELOPMENT TOOLS

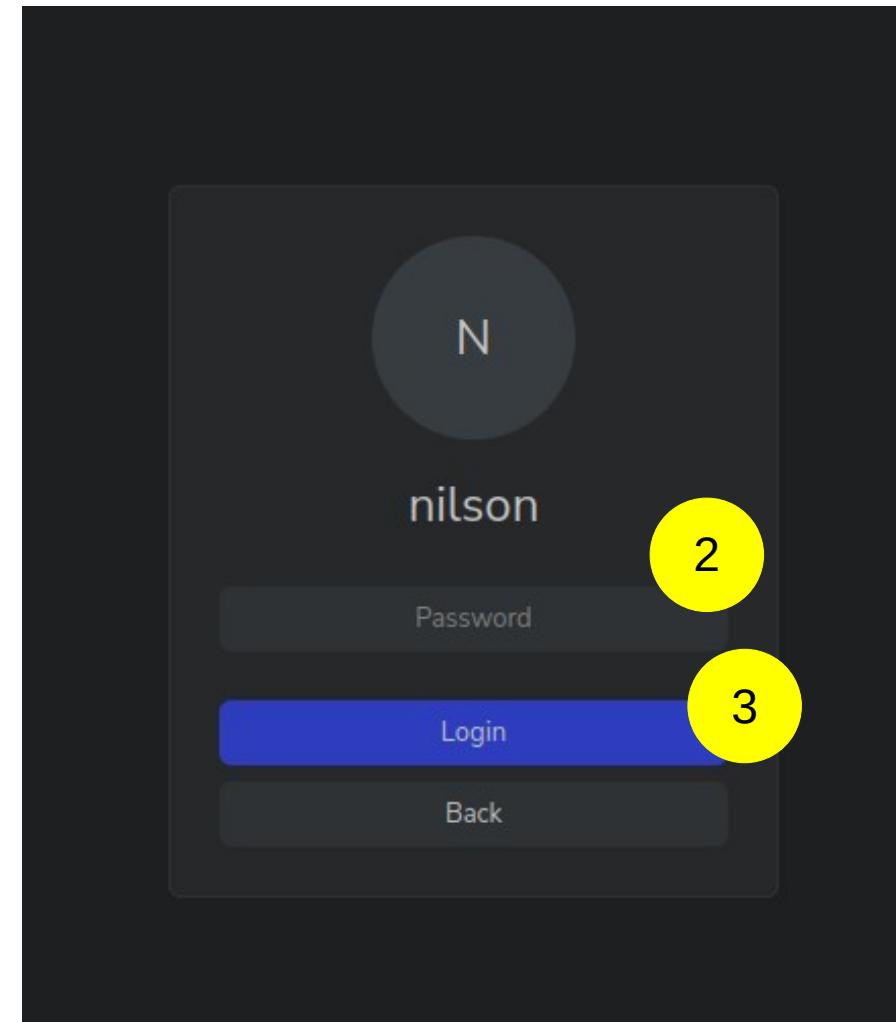
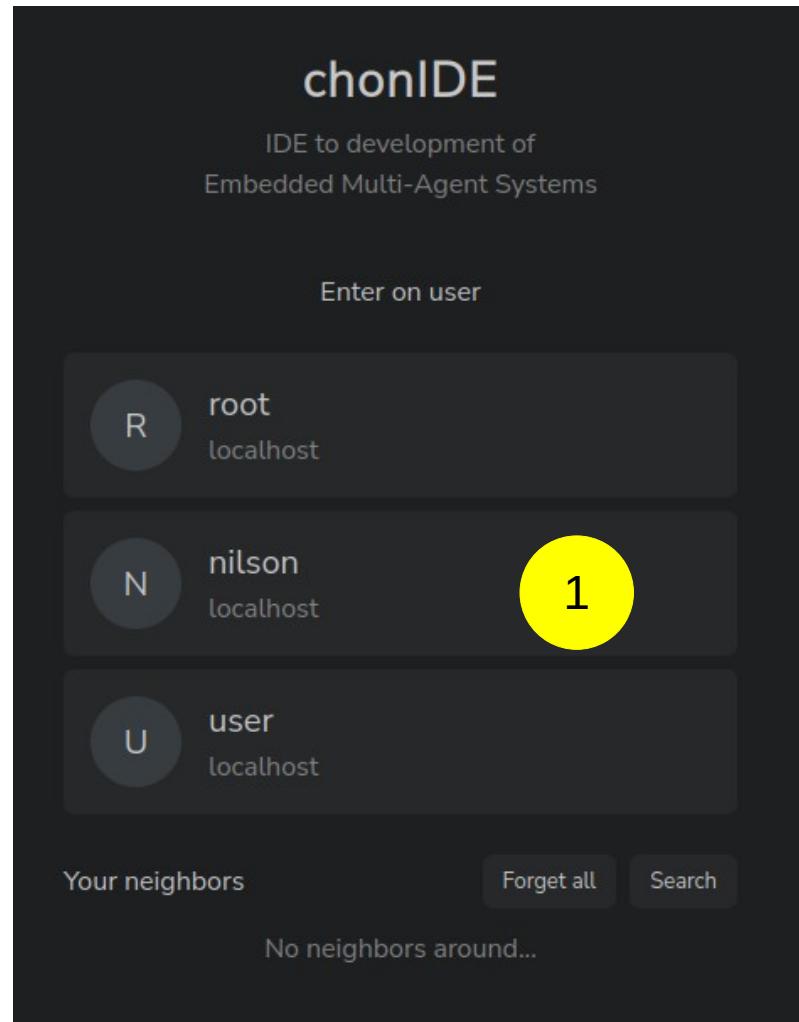




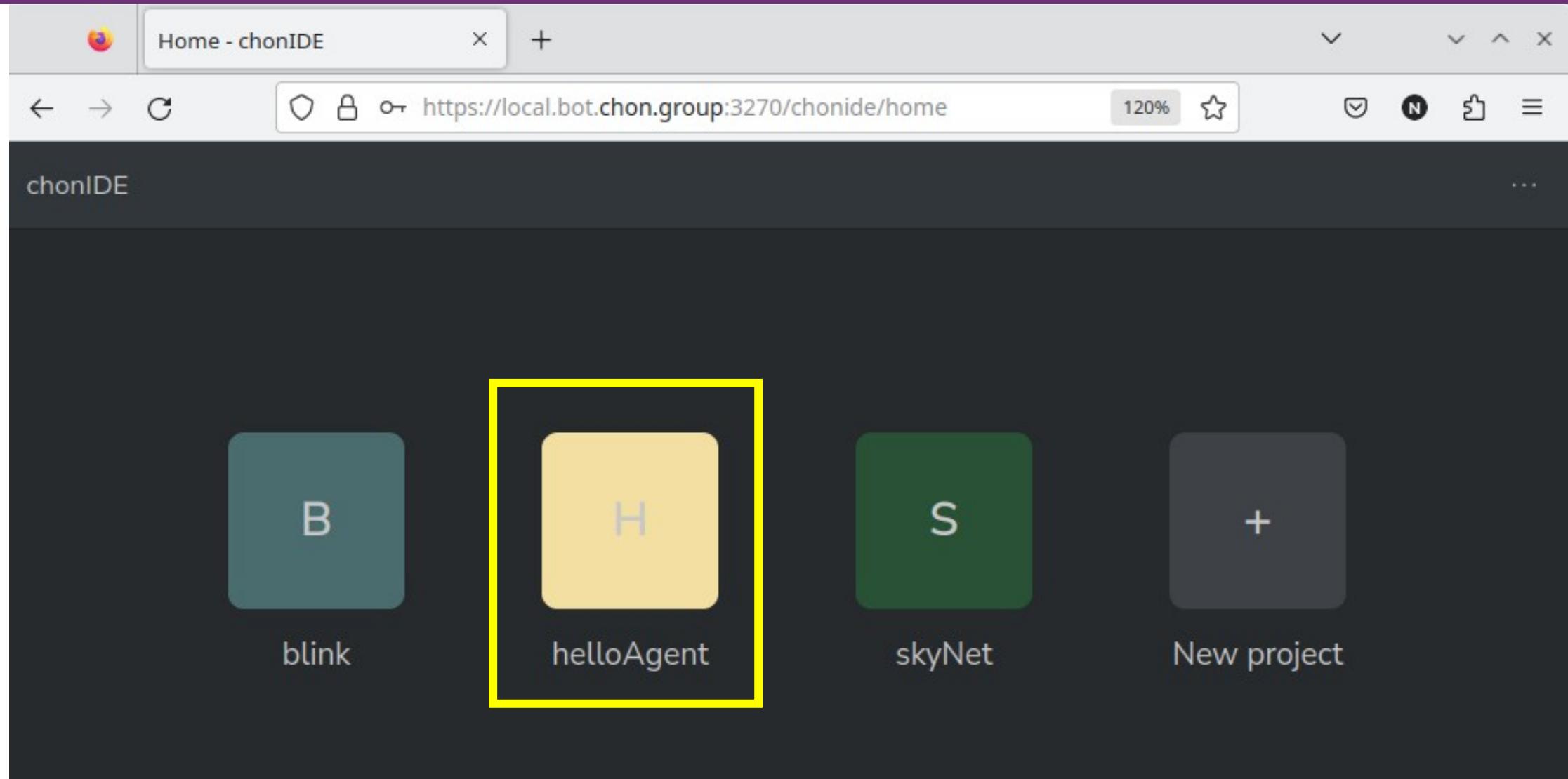
How to install?

<https://github.com/chon-group/chonIDE/wiki>

ChonIDE: login



ChonIDE: helloAgent



ChonIDE: helloAgent

chonIDE ⌂

Explorer

Multi-Agent System

Agents

smith

Firmware

1

2

3

4

5

6

7

helloAgent ✓

```
!start.  
  
+!start <-  
.print("Never Send A Human To Do A Machine's Job!").
```

Agent Tracer

```
=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED  
set. 28, 2024 7:48:05 PM group.chon.jasonembedded.Main main  
INFORMAÇÕES: Version 24.7.23 (Argo+Hermes+Mailer+Velluscinum)  
Runtime Services (RTS) is running at 127.0.1.1:37375  
Agent mind inspector is running at http://127.0.1.1:3272  
[smith] Never Send A Human To Do A Machine's Job!
```

ChonIDE: helloAgent

The screenshot shows a ChonIDE interface with a purple header bar containing the title "ChonIDE: helloAgent". Below the header is a toolbar with a play button, a red square, and a blue circle. A yellow circular badge in the top right corner displays the number "7". The main window has a dark background. At the top left, there's a browser-like tab bar with two tabs: "helloAgent - chonIDE" and "Jason Mind Inspector -- Web V". The URL bar shows "local.bot.chon.group:3272". On the left side, a sidebar titled "Agents" in red contains the entry "- smith". To the right, the main content area is titled "Inspection of agent smith" in large red text. It features three sections with horizontal lines: "- Beliefs", "- Annotations", and "+ Status". At the bottom, there are links for "hide annotations" and "plan library". The bottom right corner of the main window has some small icons and text.

helloAgent ✓

7

helloAgent - chonIDE Jason Mind Inspector -- Web V

← → ⌛ local.bot.chon.group:3272

Agents

- smith

by Jason

Inspection of agent **smith**

- Beliefs

- Annotations

+ Status

[hide annotations](#) | [plan library](#)

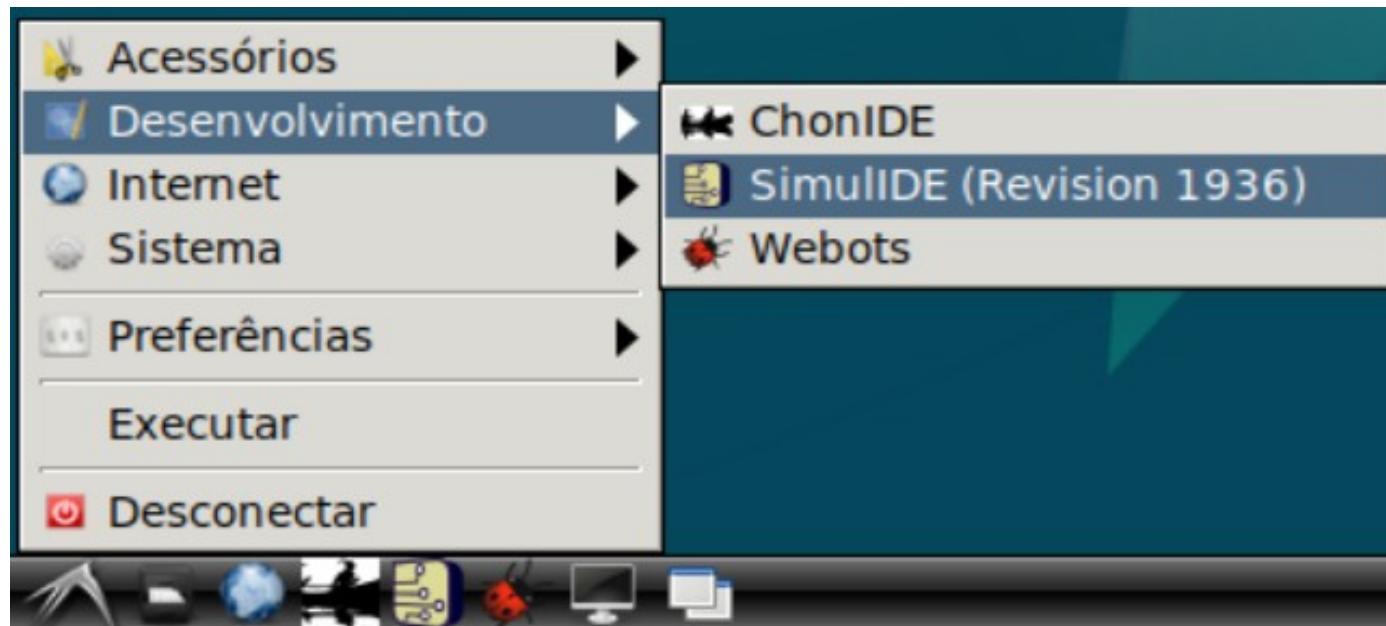


SimulIDE Circuit Simulator

by Chon Group



How to install?
<https://github.com/chon-group/dpkg-simulide>



SimulIDE

1

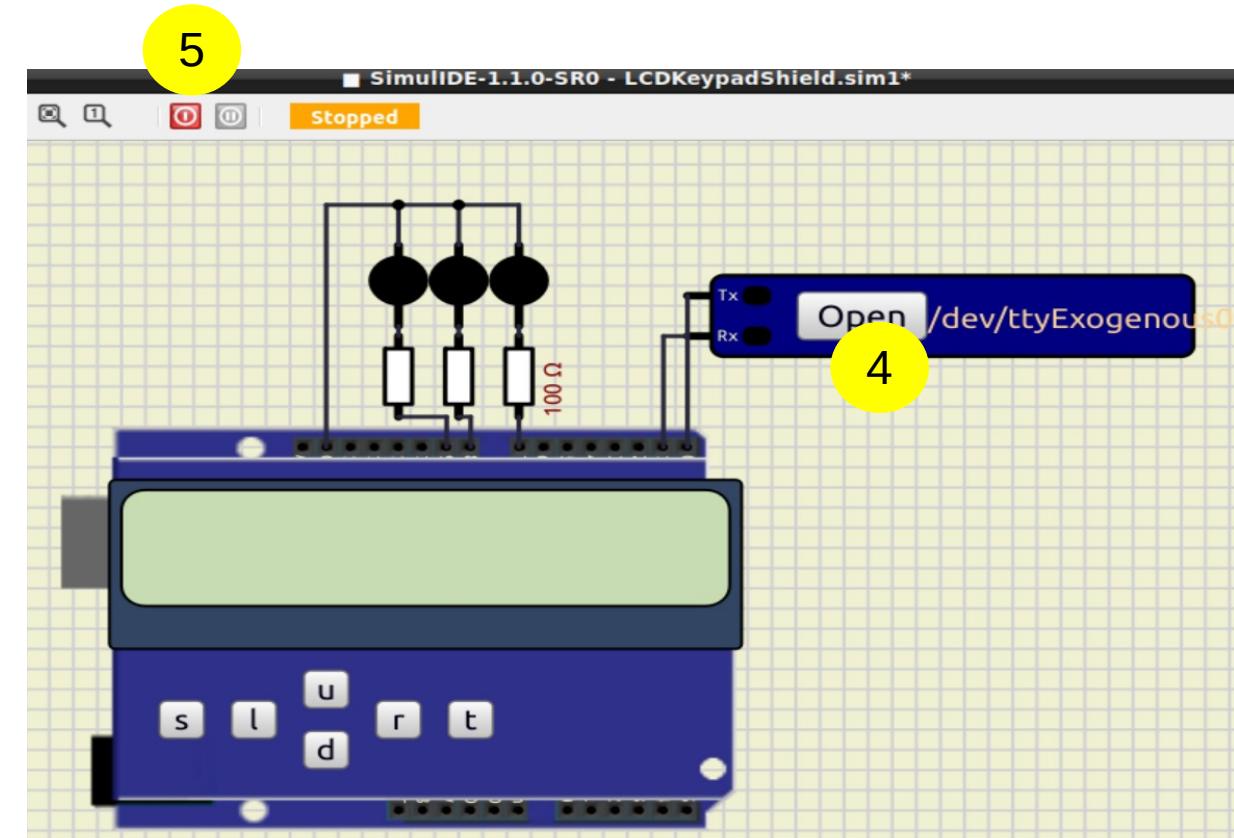
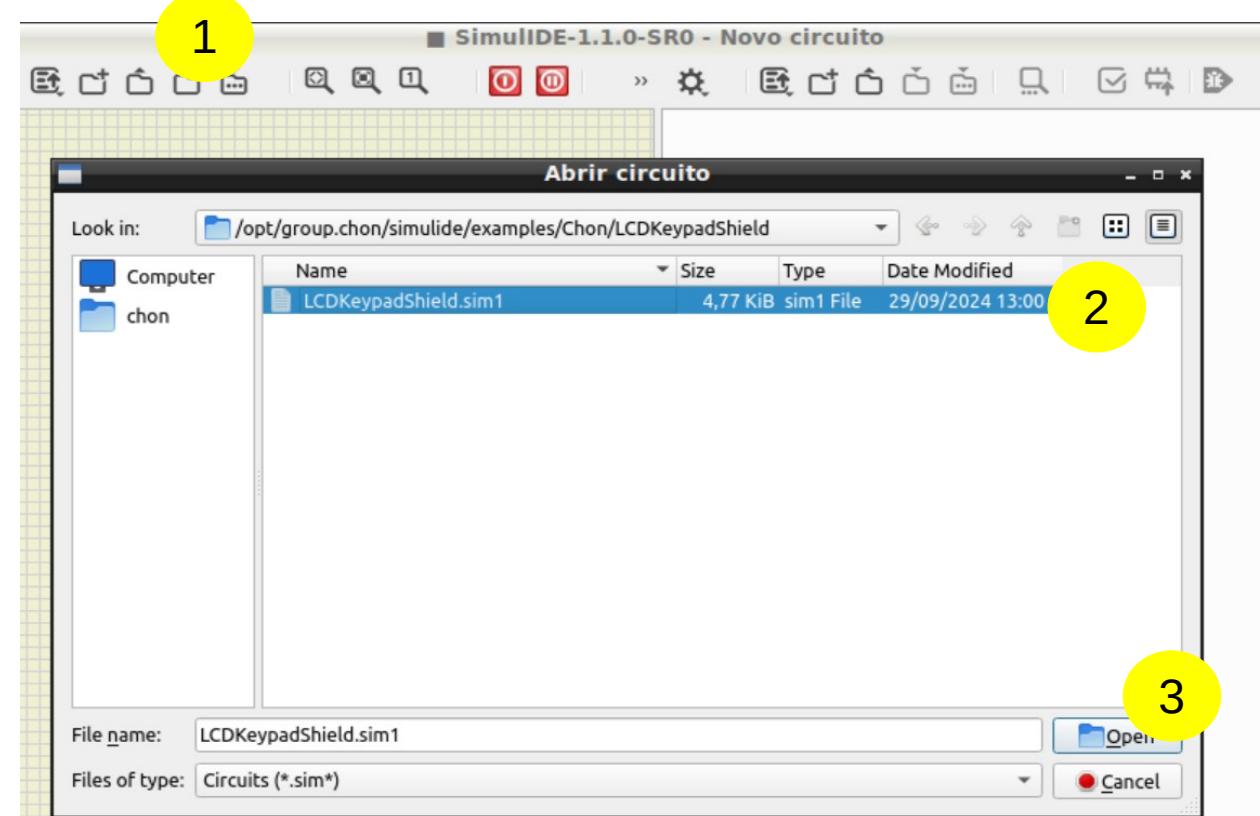
2

3

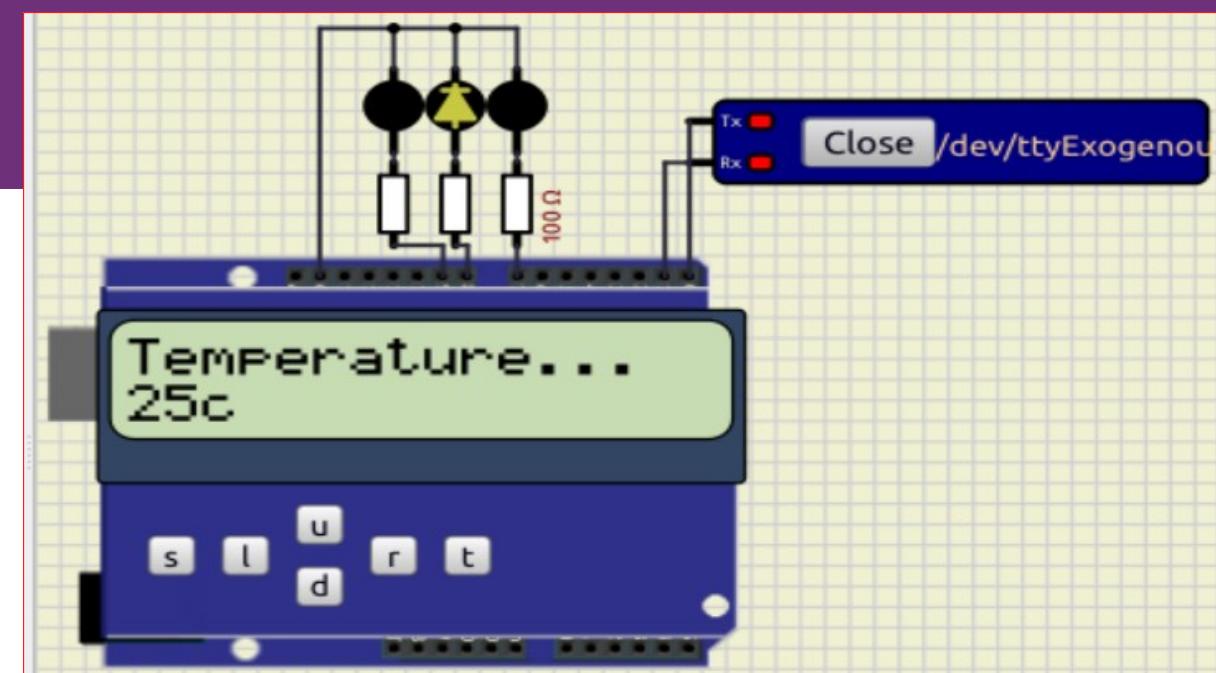
The image shows the SimulIDE interface divided into three main sections:

- Left Panel (Component Library):** A tree view of available components. The "Fontes" category is currently selected, showing components like Fixed Voltage, Gerador de pulso (Clock), Wave Generator, Voltage Source, Current Source, Controlled Source, Battery, Rail, and Terra (0 V). Other categories like Medidores, Interruptores, Passivos, and Ativos are also listed.
- Middle Panel (Simulation Area):** Displays a breadboard simulation of an Arduino Uno. A blue Arduino Uno board is placed on the grid, connected to a black breadboard. On the breadboard, there is a blue digital-to-serial converter module labeled "Open /dev/ttyExogenous". The Arduino pins are labeled with their names (e.g., RX, TX, 3.3V, GND, 5V, RST, A0-A5, D0-D13, Aref, SDA, SCL). The status bar at the bottom provides simulation information: Simulation Time: 00:00:00 s 000 ms 00, Target Speed: 100.0, Real Speed: 000.0, Engine Load: 000.0, Loading hex file: /opt/group.chon/simulide/examples/Chon/Blink/Blink.hex, Firmware successfully loaded, and Circuit Loaded.
- Right Panel (Code Editor):** Shows the source code for the "Blink.ino" sketch. The code initializes a Javino agent, sets up pin 13 as an output, and starts the loop where it runs the Javino agent. It includes notes about serialEvent() and a note about the Arduino version.

SimulIDE



SimulIDE



chon@chonide: ~

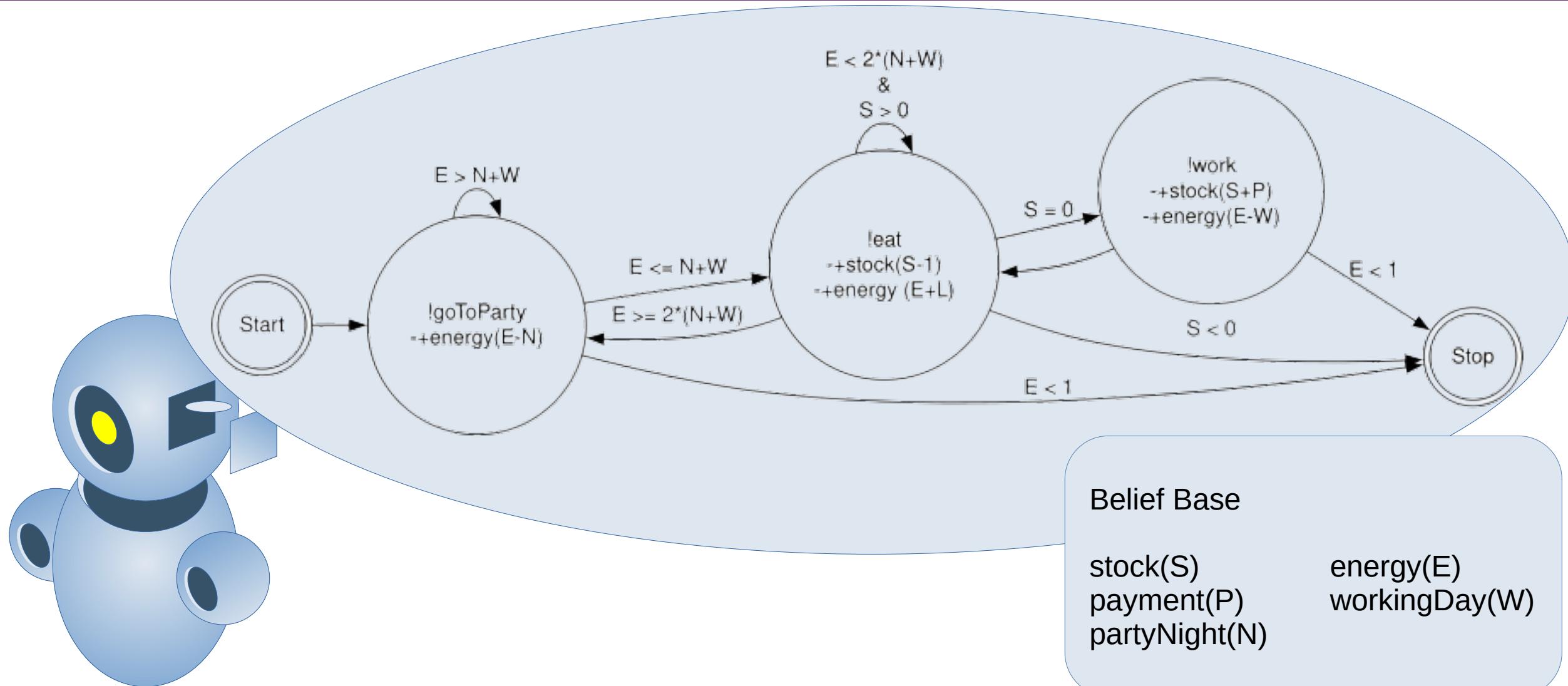
Arquivo Editar Abas Ajuda

```
chon@chonide:~$ javino /dev/ttyEmulatedPort0
javino@ttyEmulatedPort0$ write yellowAlert
javino@ttyEmulatedPort0$ request getPercepts
temperature(25);rainLast24hrs(0);humidity(50);
javino@ttyEmulatedPort0$
```

MY FIRST SINGLE AGENT



My First Agent



My First Agent: Code

```
1  /* Initial beliefs and rules */
2  stock(0).          // This is a Belief that represents the pizza stock.
3  payment(+2).        // ... represents how many pizzas does Giacomo get by a working day?
4  energy(5).          // ... represents the Giacomo life energy.
5  workingDay(-1).     // ... represents how much energy does Giacomo lose in a working day?
6  partyNight(-2).     // ... represents how much energy does Giacomo lose on a party night?
7  lifeParameters(S,E,P,W,N) :- stock(S) & energy(E) & payment(P) & workingDay(W) & partyNight(N). //this is a RULE!
8
9  /* Initial goals */
10 !startGiacomoLife.
11
12 /* Plans */
13 +!startGiacomoLife <- !startClock; !goToParty.
14 +!goToParty: lifeParameters(S,E,P,W,N) & (E+(N+W)>1) <- -+lastParty(0); -+energy(E+N); .wait(1000); !goToParty.
15 +!goToParty: lifeParameters(S,E,P,W,N) & (E+(N+W)<=1)<- !eat.
16 +!eat: lifeParameters(S,E,P,W,N) & (E<2*(-1*(N+W))) & S>0 <- -+stock(S-1); -+energy(E+1); .wait(1000); !eat.
17 +!eat: lifeParameters(S,E,P,W,N) & (E>=2*(-1*(N+W))) <- !goToParty.
18 +!eat: lifeParameters(S,E,P,W,N) & S=0 <- !work.
19 +!work: lifeParameters(S,E,P,W,N) <- -+stock(S+P); -+energy(E+W); !eat.
20 +!startClock <- +lastParty(0); !!tickingClock.
21 +!tickingClock <- .wait(100); ?lastParty(Time); -+lastParty(Time+100); !tickingClock.
22
23 +energy(E): E < 1 <- .print("Giacomo died of hunger!"); .stopMAS.
24 +stock(S): S < 0 <- .print("Without food!"); .stopMAS.
25 +lastParty(Time): Time > 10000 <- .print("Giacomo died of sadness!"); .stopMAS.
```

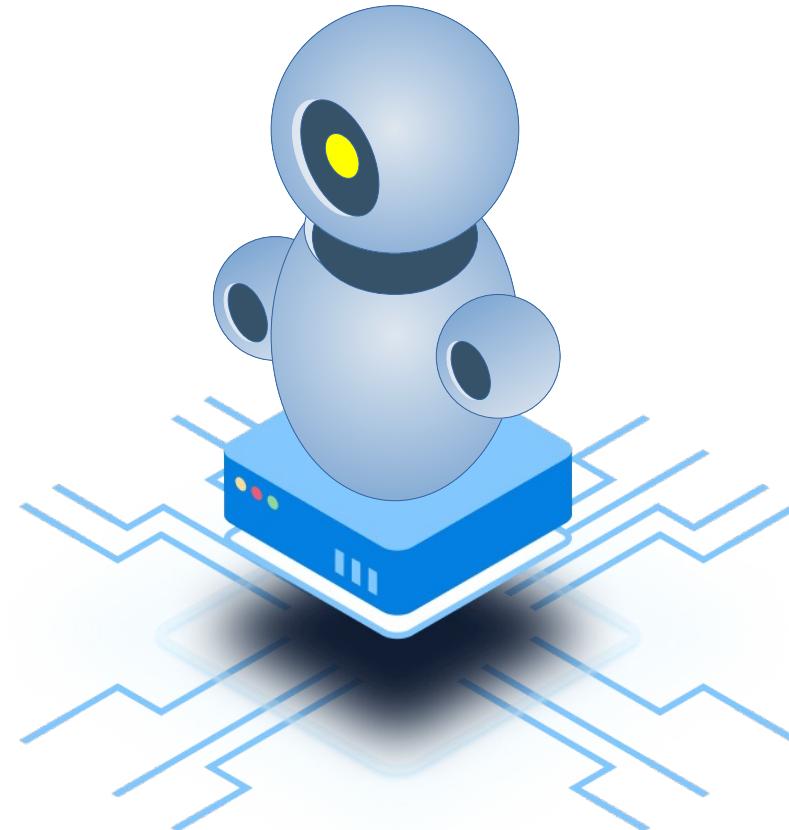
<https://github.com/chon-group/distributedAndEmbeddedAI/blob/main/workshops/semeso/2024/projects/giacomoAgent.chon>



My First Agent: Code

initial beliefs -->	energy(5). day(sunny). myPhone("555-111111").	belief plan -->	+day(D) <- .print("Today is ",D).
a rule -->	happy :- day(D) & D=sunny & money(M) & M=veryMuch.	belief plan --> mental notation search in mind (?) new goal	+day(D): D=saturday & time > 18 <- +hungry; ?pizzaPhone(Number); !callTo(Number).
initial goals -->	!start. !callTo("555-123321").	contantion plan ->	-!start <- .print("without belief myPhone(T) in mind").
achievement plan -->	+!start <- .print("hello world").	JASON stdlib → https://jason-lang.github.io/api/jason/stdlib/package-summary.html	
achievement plan -->	+!start: myPhone(T) <- .print("Call me:",T).	Introdução ao Jason → https://www.researchgate.net/publication/228828757_Introducao_ao_desenvolvimento_de_sistemas_multiagentes_com_Jason	

EMBEDDED MAS



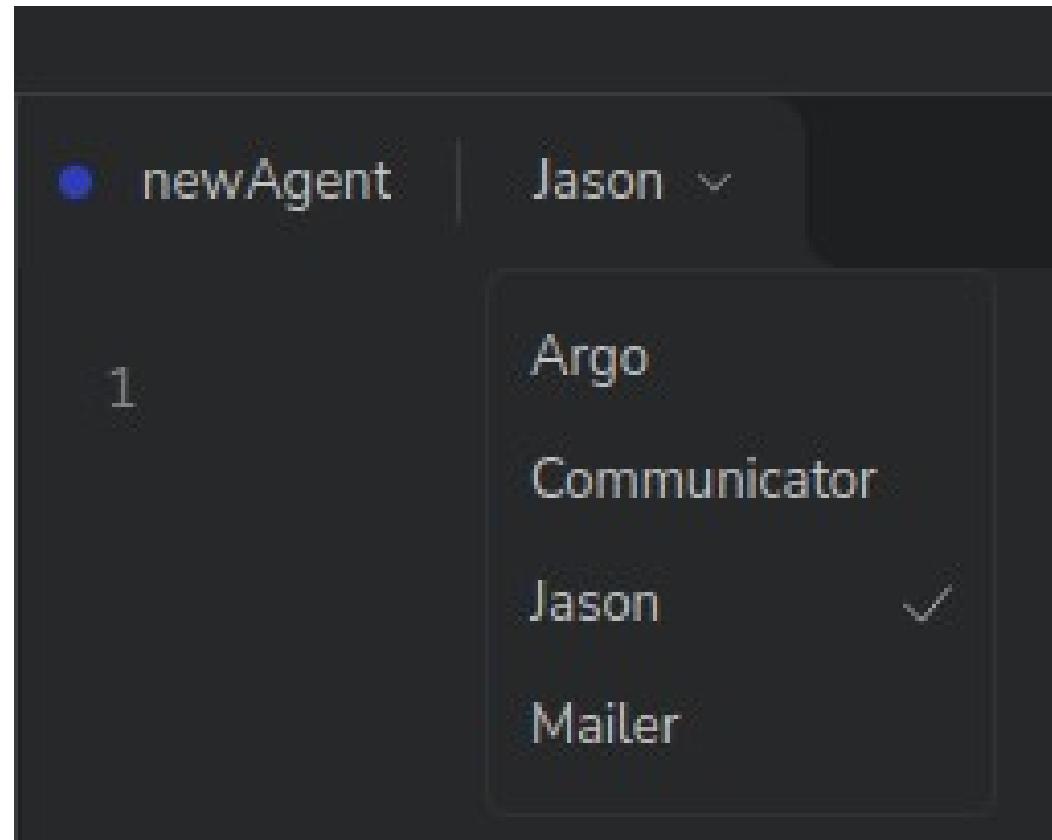
Jason Embedded

Jason Embedded is a spin-off Version of Jason for IoT and Embedded MAS:

- **Autonomy** as a characteristic an Embedded MAS provides to a device that does not need external architectures to control and manage it.
- **Communicability** is the agent's ability to communicate with other agents in its MAS or another MAS.
- **Mobility** as the agents' ability to move from one MAS to another.
- **Fault Tolerance and Adaptability** are defined as the ability of an agent to identify if a physical resource is absent or not answering commands and to overcome this situation by modifying its goals.
- **Distributed Ledger Technologies** technologies to facilitate the agreement between agents.

Jason Embedded

- **Jason Embedded** also provides **new types of agents** to control hardware devices and to communicate with agents hosted in other MAS.



Jason Embedded

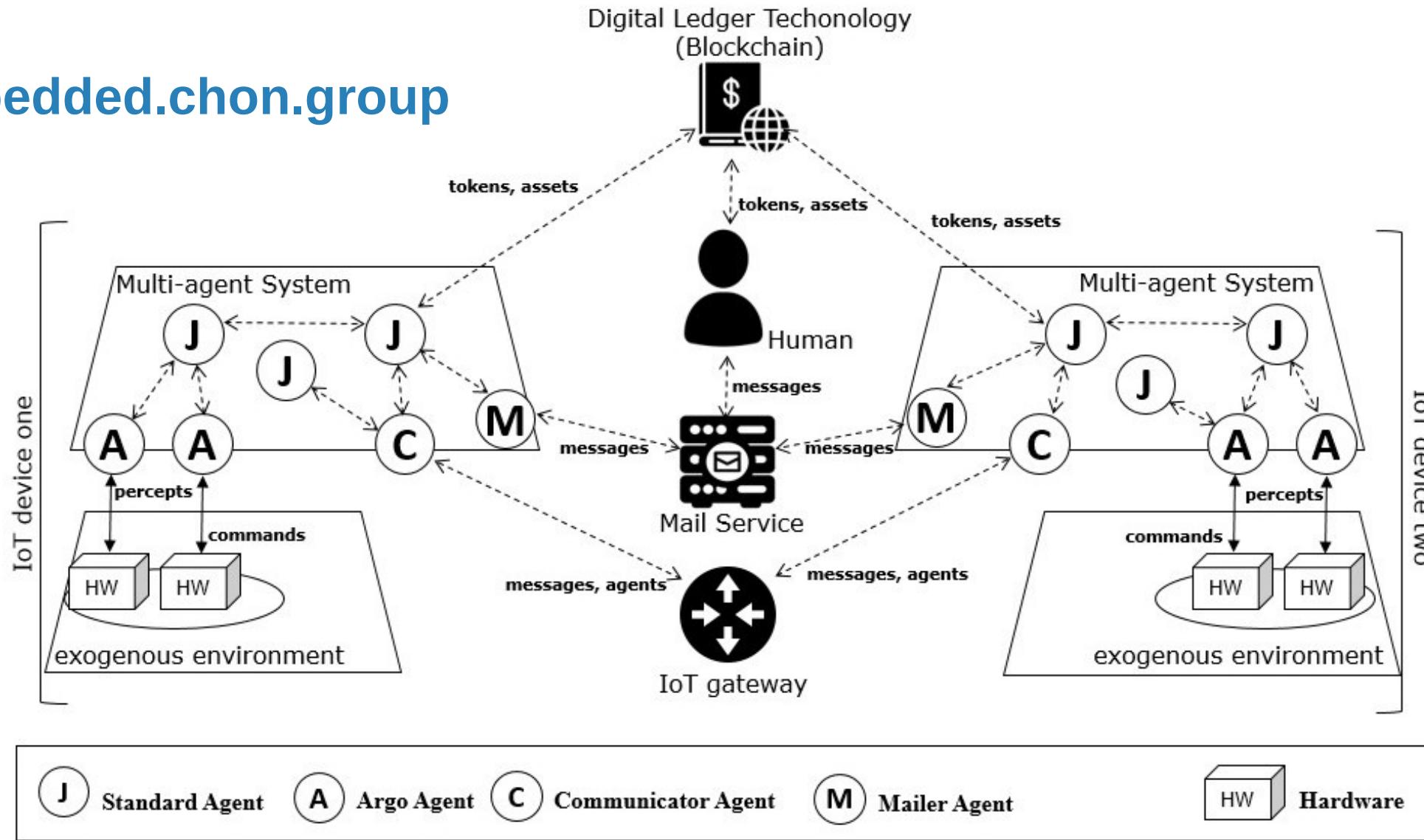
- **Jason Embedded** also provides **new types of agents** to control hardware devices and to communicate with agents hosted in other MAS.
- **ARGO**: is a customized agent architecture built on the Jason framework that extends Jason's standard agents by adding the **ability to control microcontrollers** [Pantoja et al., 2016].
- **Hermes (Communicator)** : is another customized agent architecture built on Jason's Standard agent by adding the **ability to communicate** with agents from **other MAS**, which also have Communicator agents [Pantoja et al., 2018].
- **Mailer** : agent architecture based on the Jason framework to enable secure and simplified **human-agent interaction** [Lazarin et al., 2024].

Jason Embedded

- **Mobility** is based on **bio-inspired protocols** [Jesus et al., 2021].
 - The protocols simulate natural behaviors that could be explored in collaborative tasks using IoT devices.
 - One or more agents or even the whole MAS could move from one device to another to take control of the destination device (**Predation**) or to use it as a temporary non-hazardous relationship (**Mutualism** and **Inquilinism**) if both parties accept the protocol.
- **Middleware** that allows cognitive agents to manipulate Tokens and NFT in blockchain network [Lazarin et al., 2023].

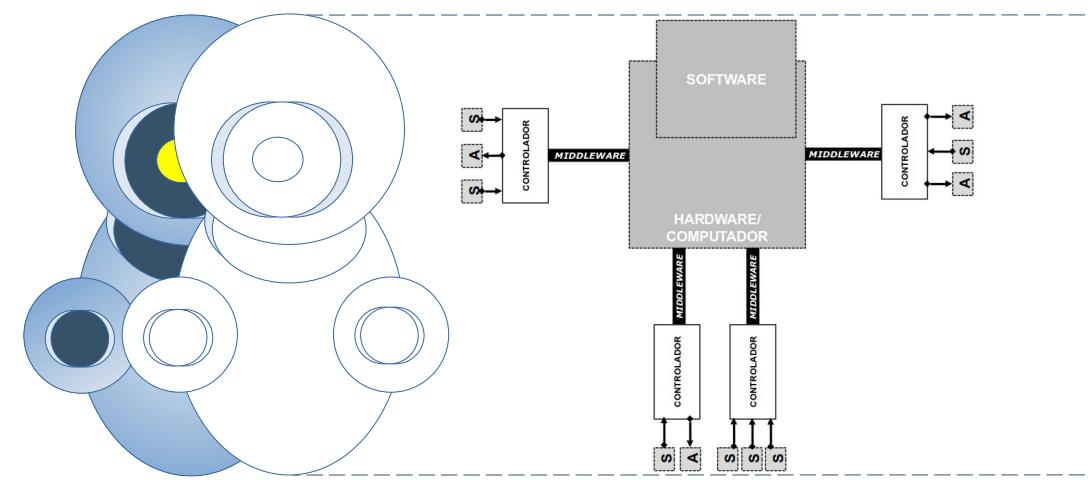
Jason Embedded

<https://jasonembedded.chon.group>



ARGO AGENT

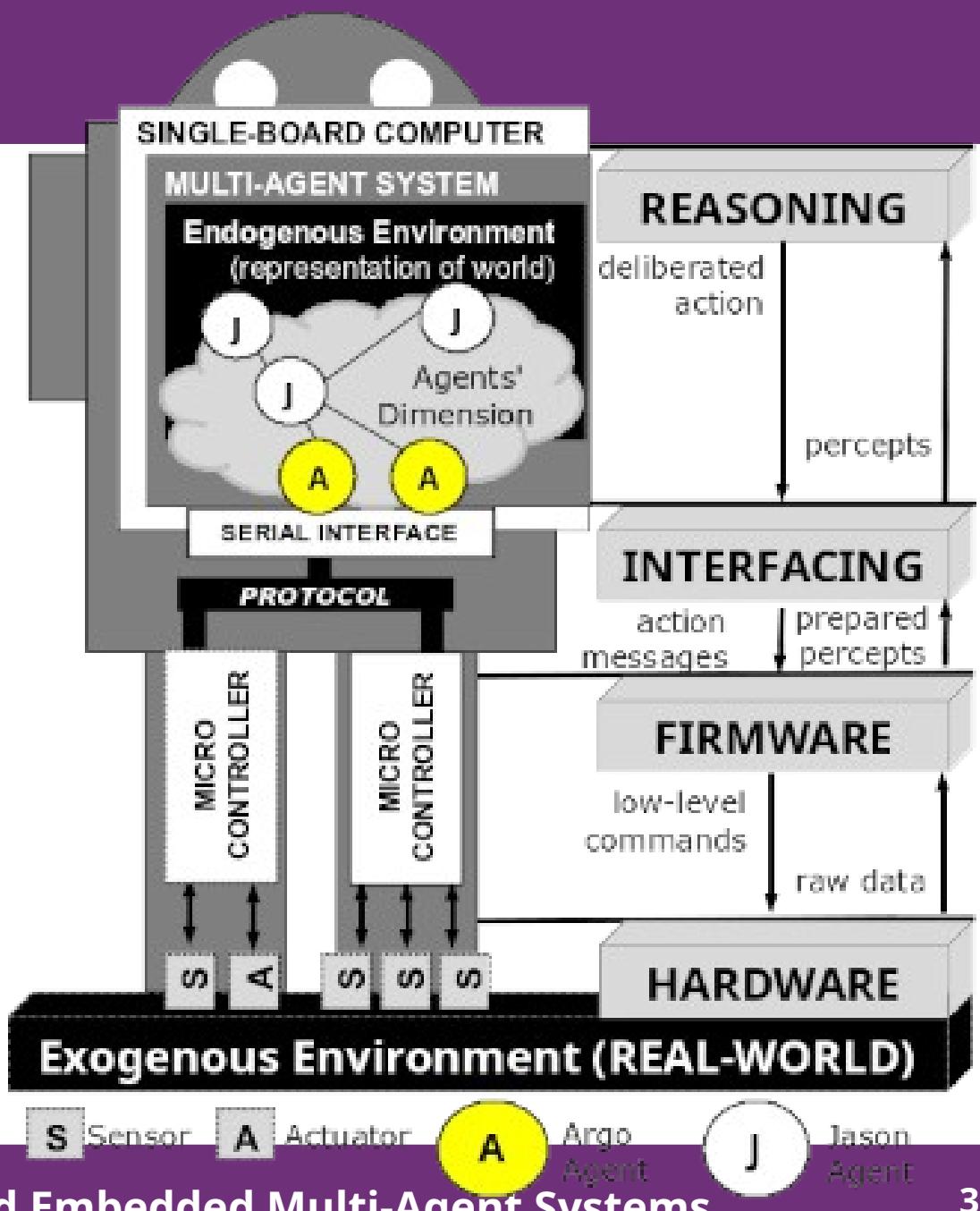
<https://argo.chon.group>



Argo for Jason

ARGO is a customized agent architecture that **interfaces** sensors and hardware actuators.

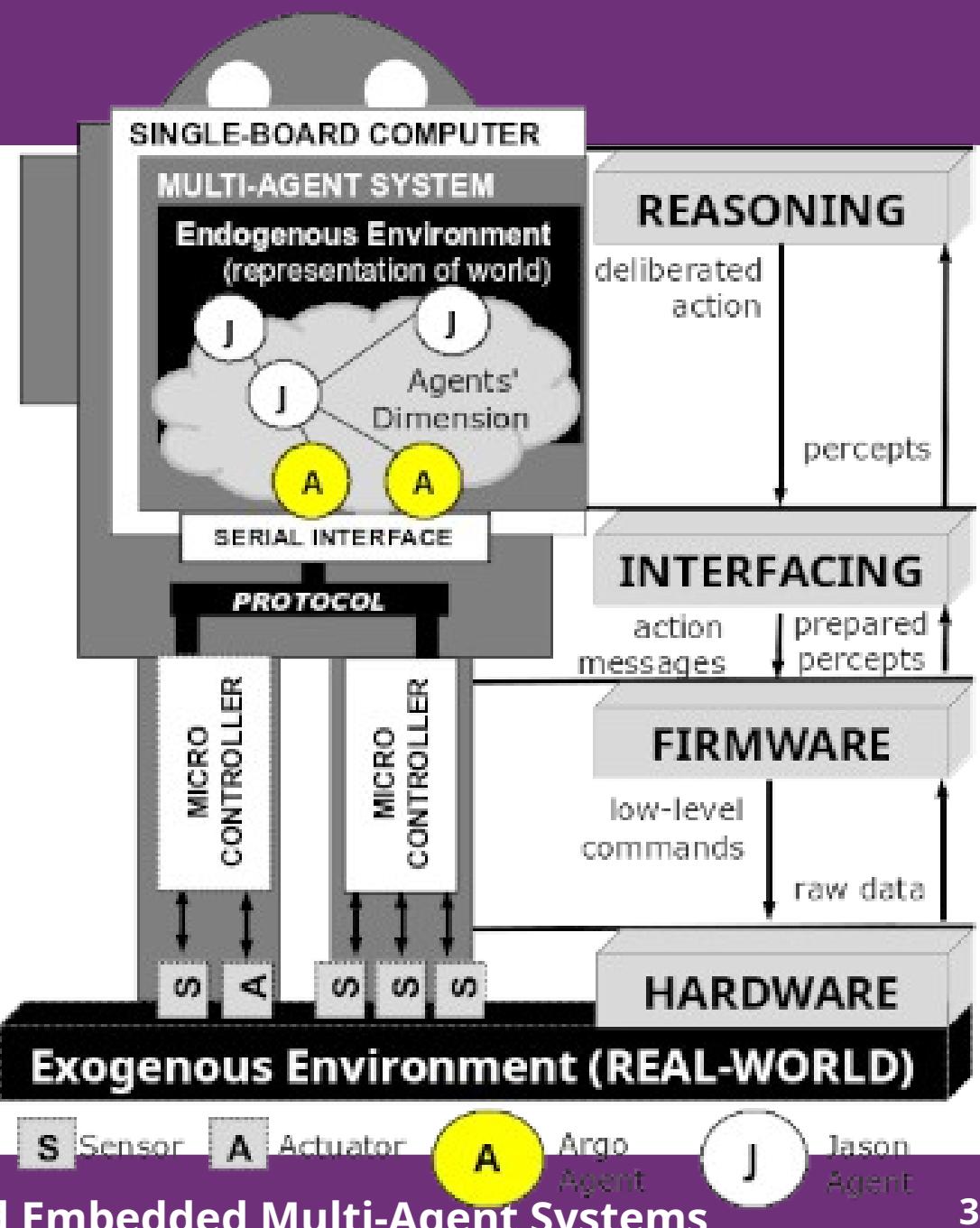
ARGO aims to be a practical architecture for **programming embedded BDI agents** using **Jason** and microcontroller boards.



Argo for Jason

The **ARGO** allows to:

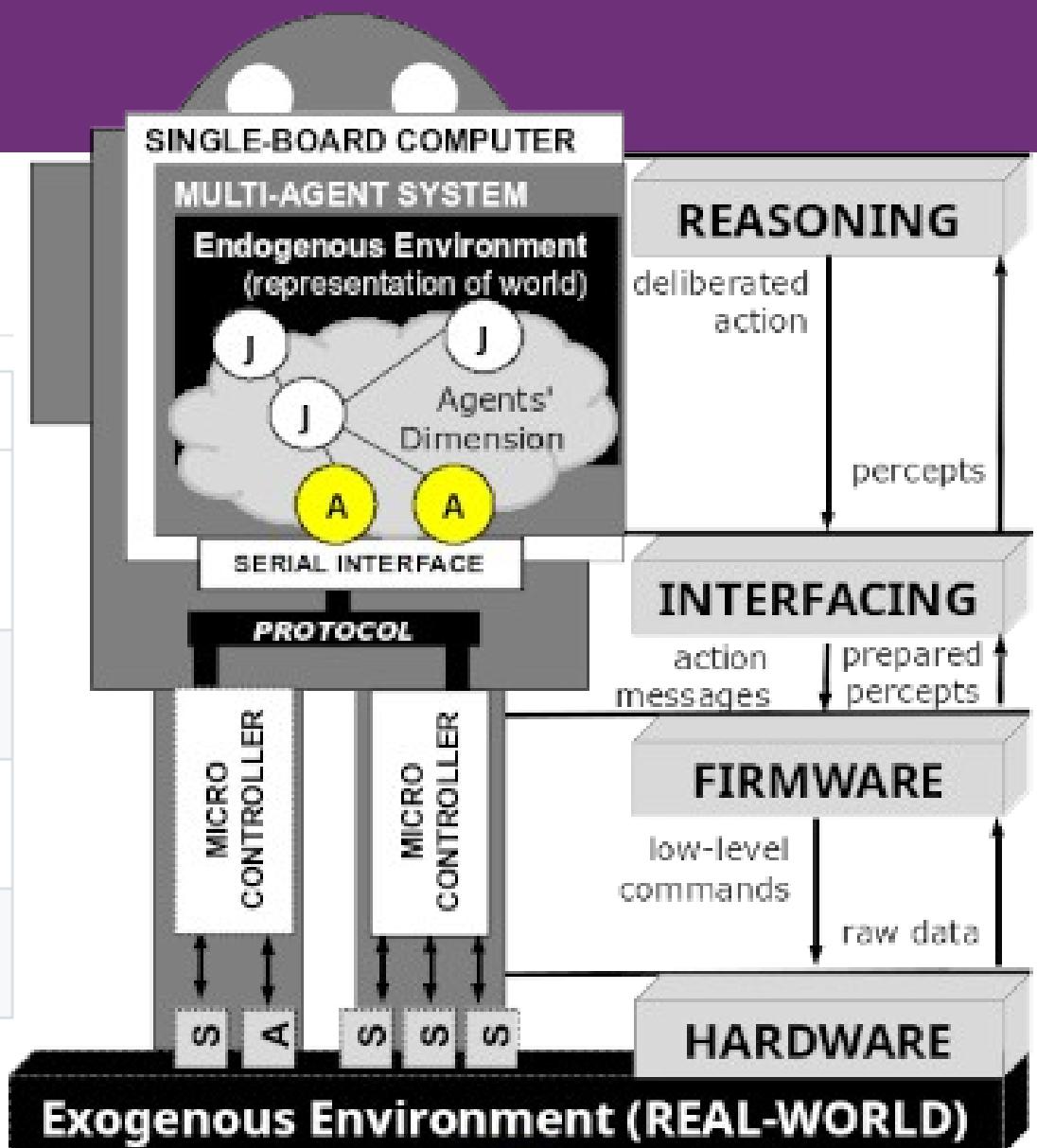
1. to direct control actuators at runtime;
2. to receive sensor perceptions within a predefined period automatically;
3. to change which devices are being accessed at runtime;
4. to communicate with other agents in Jason;
5. whether or not to perceive the real world at runtime;
6. to change perception filters at runtime.



Argo for Jason

ARGO Internal Actions

Action	Description
.argo.port(P)	defines which serial port should be used by the agent, where P is a literal representing the port identification(e.g., COM8, ttyUSB0).
.argo.limit(M)	defines the sensing interval, where M is a value in milliseconds.
.argo.percepts(open close)	decides whether or not to perceive the exogenous environment (physical world).
.argo.act(A)	sends to the hardware an action, represented by literal A , to be executed by a microcontroller;



A Argo Agent
S Sensor A Actuator A Argo Agent J Jason Agent

Example: blink

chon-group / Argo

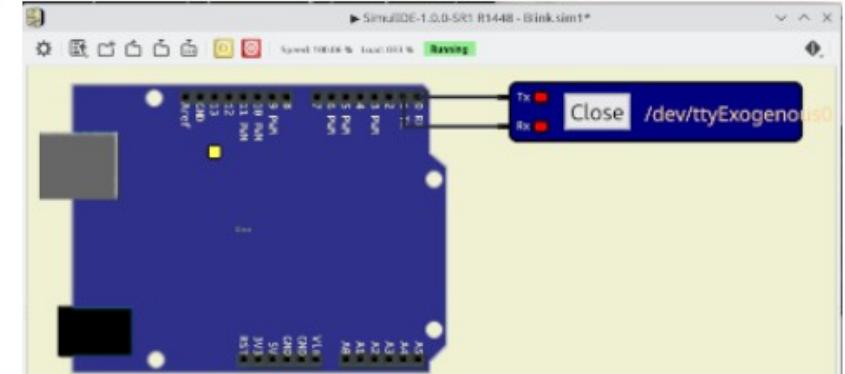
Code Issues Pull requests Wiki Security Insights Settings

Type ⌘ to search

Tools

Nilson Lazarin edited this page 19 hours ago · 1 revision

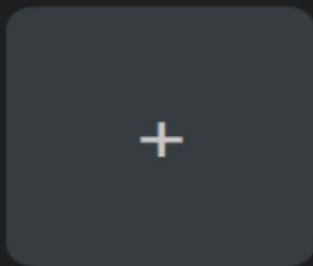
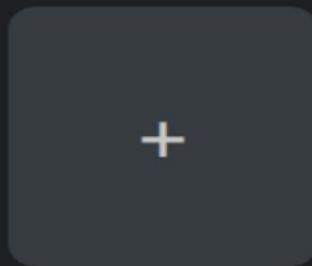
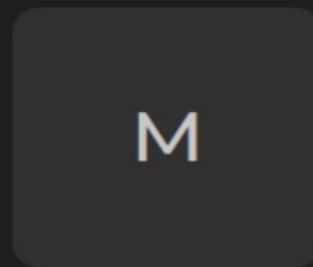
[Examples](#) [Tools](#) [Wiki](#)

Arduino	SimulIDE
	
Blink tutorial with Arduino Board	Blink tutorial with SimulIDE

Example: blink

chonIDE

...



Example: blink

The screenshot shows the chonIDE interface with the title bar "chonIDE" and "blink ✓". The main area displays a text editor with the file "blink" open. The code is written in a domain-specific language (DSL) for robotics, specifically Argo. The code includes comments for initial beliefs and rules, and a section for serial port configuration. A red box highlights the serial port configuration block.

```
/* Initial beliefs and rules */
/* Uncomment your current serial Port */
// serialPort(ttyEmulatedPort0). /* When using a Arduino Simulated! See: https://github.com/chon-group/ChonSimulator
// serialPort(ttyACM0).          /* When using a Arduino Board!      See: https://github.com/chon-group/ChonBoard
// serialPort(ttyUSB0).          /* When using a Generic Board!    See: https://github.com/chon-group/ChonFirmwares

/* 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.
    .argo.port(Port);
    .argo.percepts(open);
    .argo.limit(1000).
```

Example: blink

chonIDE ↴

blink ✓

Explorer

- bane
- Argo ▾

```
1  /* Initial beliefs and rules */
2
3  /* Uncomment your current serial Port */
4  // serialPort(ttyEmulatedPort0). /* When using a Arduino Simulated! See: https://github.com/chon-grou
5  serialPort(ttyACM0).           /* When using a Arduino Board!      See: https://github.com/chon-grou
6  // serialPort(ttyUSB0).         /* When using a Generic Board!    See: https://github.com/chon-grou
7
```

Ferramentas Ajuda

- Autoformatação Ctrl-T
- Arquivar Sketch
- Corrigir codificação e recarregar
- Gerenciar Bibliotecas... Ctrl+Shift-I
- Monitor serial Ctrl+Shift-M
- Plotter serial Ctrl+Shift-L
- Placa: "Arduino Uno"
- Porta: "/dev/ttyACM0 (Arduino Uno)"
- Obter informações da Placa
- Programador: "AVRISP mkII"
- Gravar Bootloader

Portas seriais

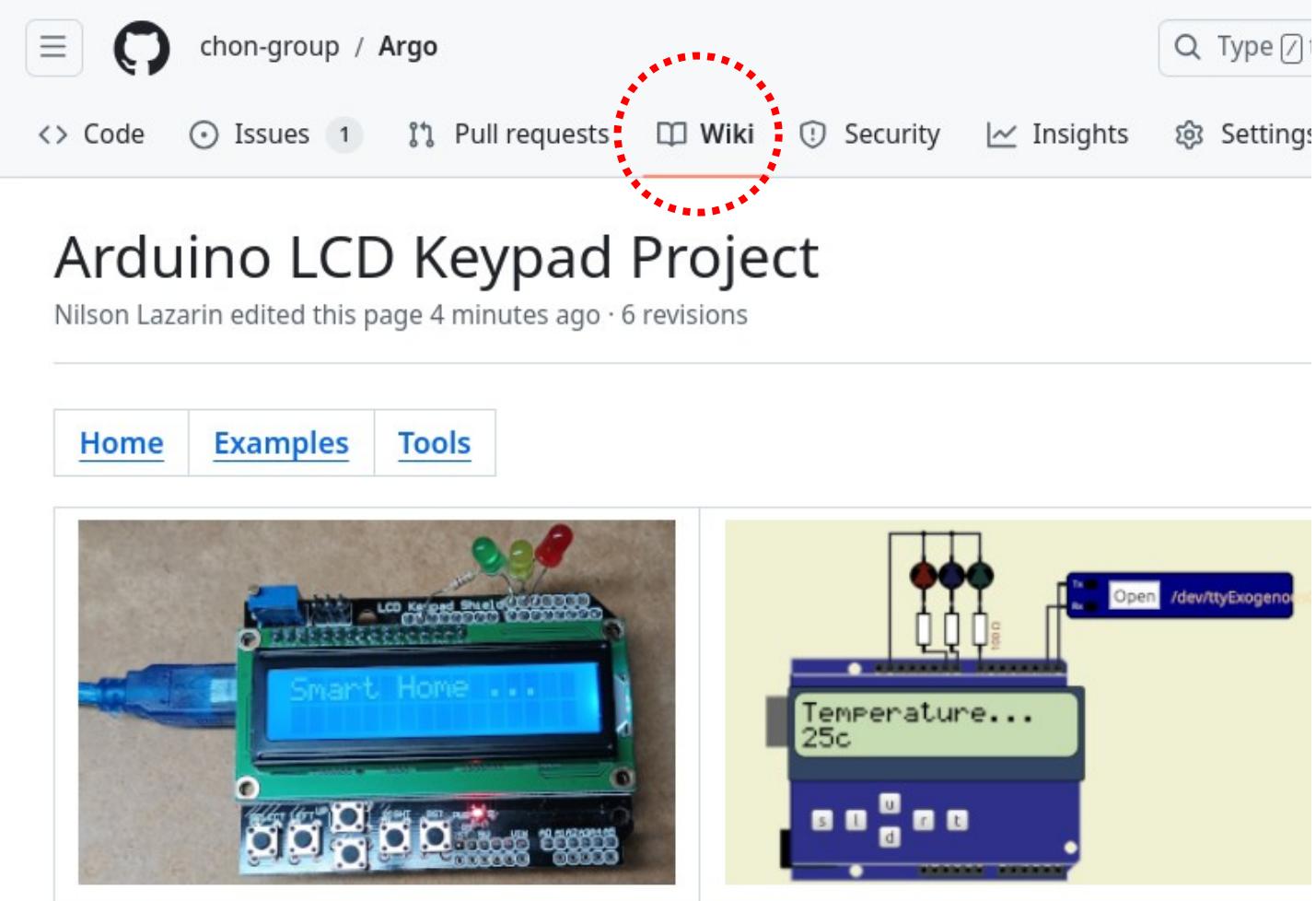
/dev/ttyACM0 (Arduino Uno)

Agent Tracer

```
[Javino] Using version 1.6.4
[bane] Ah, Mr. Anderson, I see you are as predictable in this world as you are in the other.
[bane] Turning OFF the Led in Arduino!
[bane] Turning ON the Led in Arduino!
[bane] Turning OFF the Led in Arduino!
[bane] Turning ON the Led in Arduino!
[bane] Turning OFF the Led in Arduino!
[bane] Turning ON the Led in Arduino!
[bane] Turning OFF the Led in Arduino!
[bane] Turning ON the Led in Arduino!
[bane] Turning OFF the Led in Arduino!
[bane] Turning ON the Led in Arduino!
```

⚙️

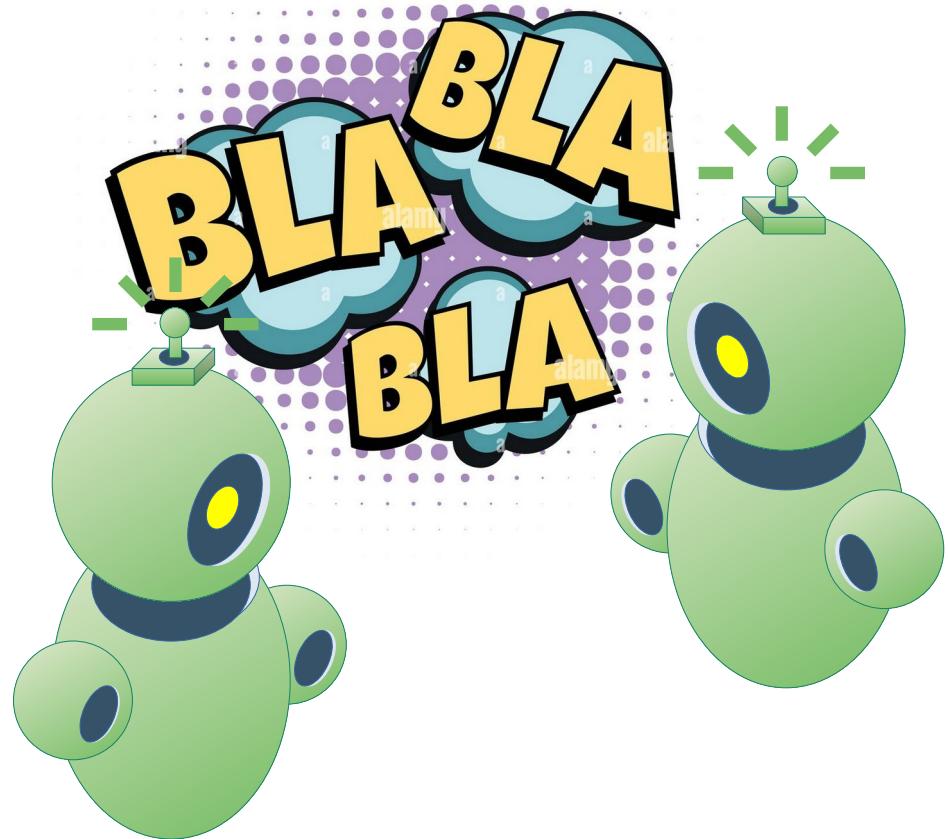
Exercise: LCD KeyPad



Adjust the ARGO agent to:

- turn on the **green LED** if the rain intensity less than 30mm;
- turn on the **yellow LED** if the rain intensity exceeds 50mm;
- turn on the **red LED** if the rain intensity exceeds 80mm;

COMMUNICABILITY



Jason Framework: Communication

Based on the theory of Speech Acts [1] and KQML [2] (Knowledge Query and Manipulation Language), a communication protocol for knowledge-based systems.



- [1] M. Bierwisch, "Semantic Structure and Illocutionary Force", em Speech Act Theory and Pragmatics, J. R. Searle, F. Kiefer, e M. Bierwisch, Orgs., Dordrecht: Springer Netherlands, 1980, p. 1–35. doi: [10.1007/978-94-009-8964-1_1](https://doi.org/10.1007/978-94-009-8964-1_1).
- [2] T. Finin, R. Fritzson, D. McKay, e R. McEntire, "KQML as an agent communication language", em Proceedings of the third international conference on Information and knowledge management - CIKM '94, Gaithersburg, Maryland, United States: ACM Press, 1994, p. 456–463. doi: [10.1145/191246.191322](https://doi.org/10.1145/191246.191322).

Jason Framework: Performatives

The performatives that are currently available for agent communication in Jason are largely inspired by KQML. We also include some new performatives, related to plan exchange rather than communication about propositions. The available performatives are briefly described below, where s denotes the agent that sends the message, and r denotes the agent that receives the message [1].

PERFORMATIVE	DESCRIPTION
tell	s intends r to believe (that s believes) the sentence in the message's content to be true;
untell	s intends r not to believe (that s believes) the sentence in the message's content to be true;
achieve	s requests that r try to achieve a state of the world where the message content is true;
unachieve	s requests that r try to drop the intention of achieving a state of the world where the message content is true;
tellHow	s informs r of a plan;
untellHow	s requests that r disregard a certain plan (i.e., delete that plan from its plan library);
askIf	s wants to know if the content of the message is true for r;
askAll	s wants all of r's answers to a question;
askHow	s wants all of r's plans for a triggering event;

[1] R. H. Bordini e J. F. Hübner, "BDI Agent Programming in AgentSpeak Using Jason", em Computational Logic in Multi-Agent Systems, F. Toni e P. Torroni, Orgs., em Lecture Notes in Computer Science, vol. 3900. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, p. 143–164. doi: [10.1007/11750734_9](https://doi.org/10.1007/11750734_9).

Jason Framework: Message Structure

.send(receiver, if, message, answer, timeout)
.broadcast(if, message)

- **receiver**
Name of the agent receiving the message (or list of recipients)
- **if**
Illocutionary force of the speech act (KQML)
- **message**
Message content
- **answer**
A term that will store the response (optional field)
- **timeout**
Timeout in milliseconds to receive a response (optional field)



<https://github.com/chon-group/distributedAndEmbeddedAI/blob/main/workshops/semeso/2024/projects/chiefGlutton.chon>

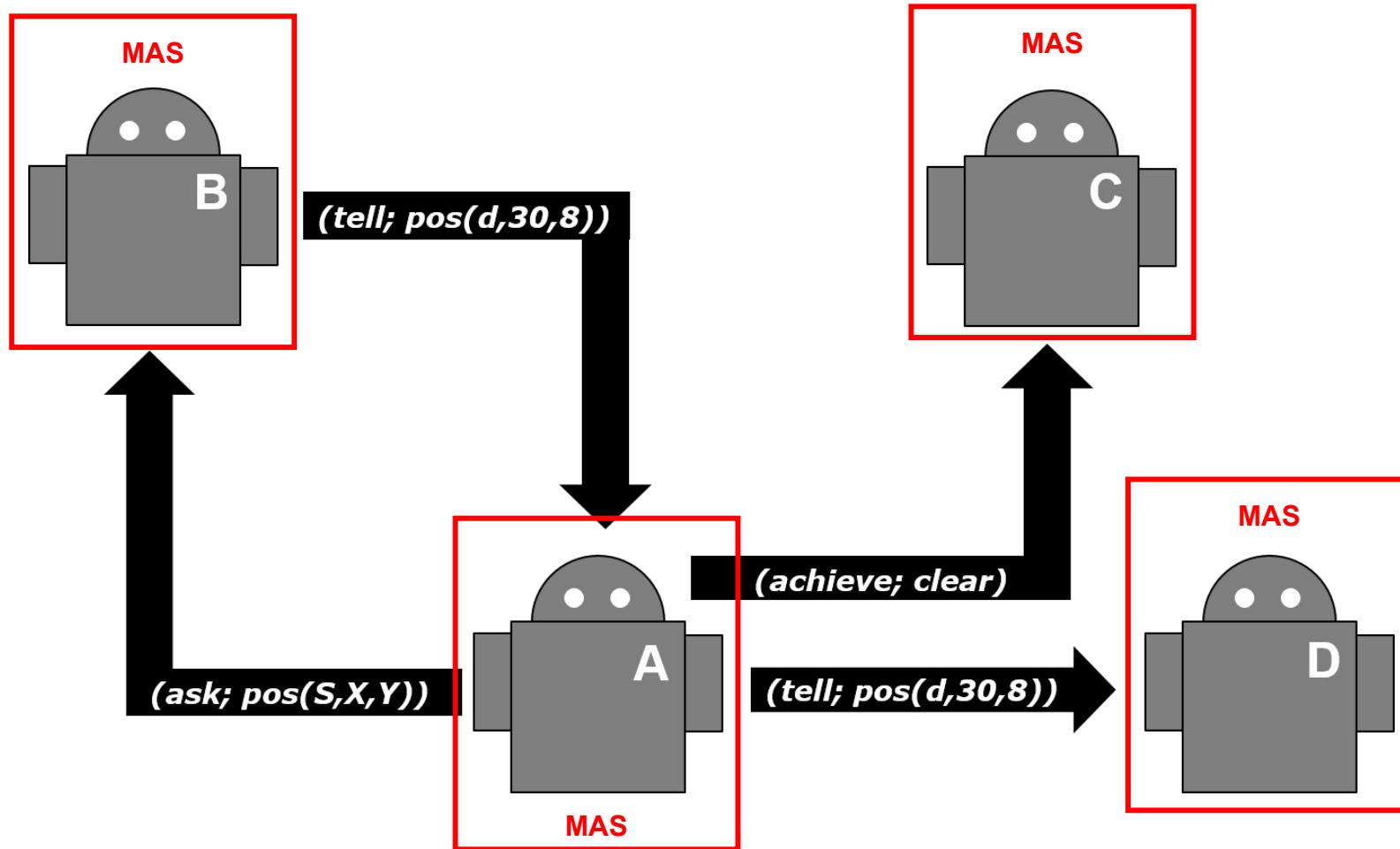


Image credits:

https://smurfs.fandom.com/wiki/Chef_Smurf

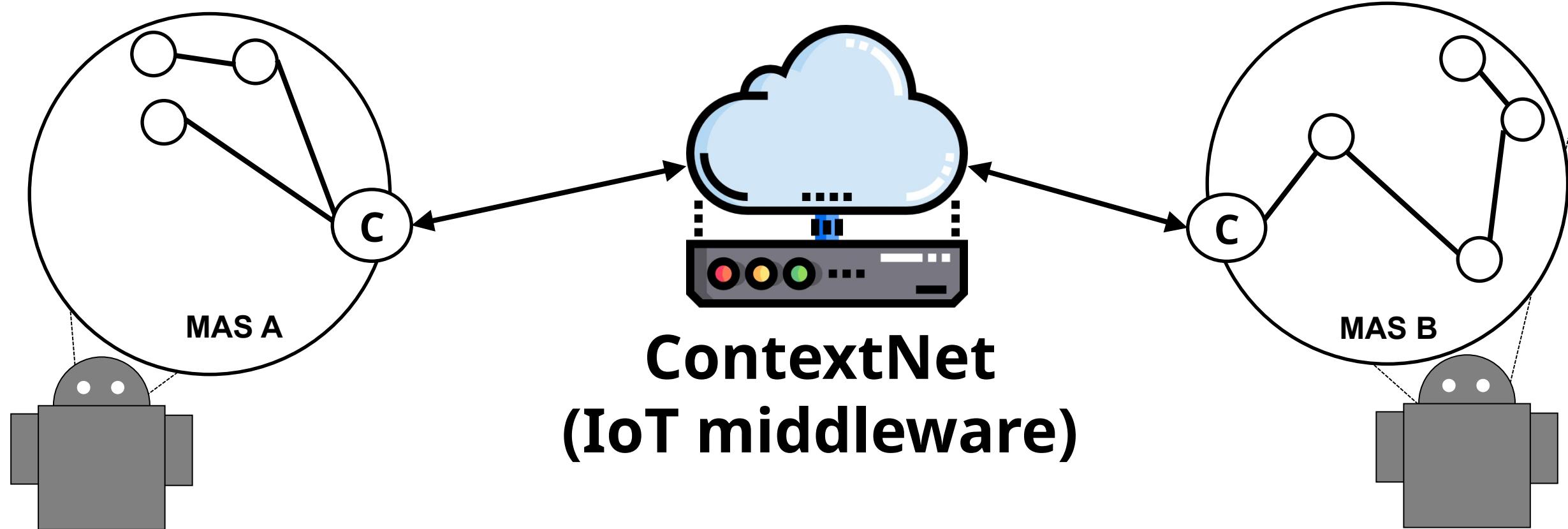
<https://smurfs.fandom.com/pt/wiki/Fominha>

Collaborative MultiAgent System



Hermes Agents: Server Communication

Public Server: skynet.chon.group - UDP Port: 5500

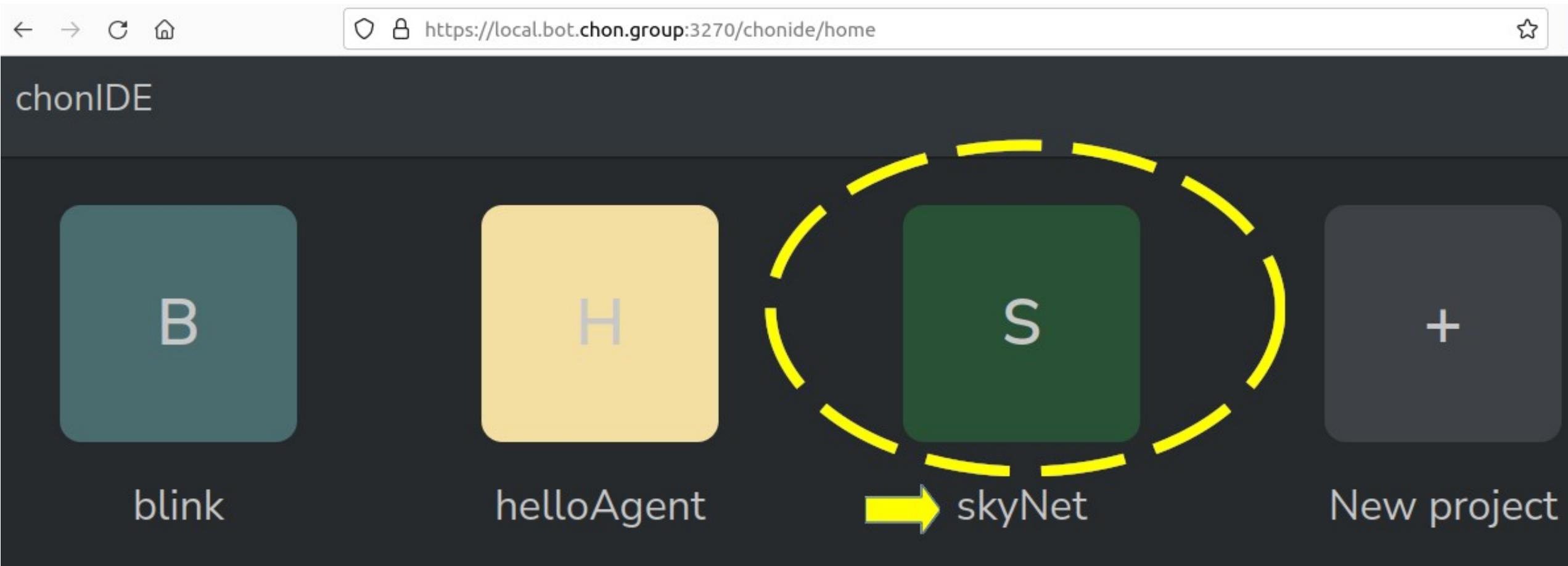


**ContextNet
(IoT middleware)**

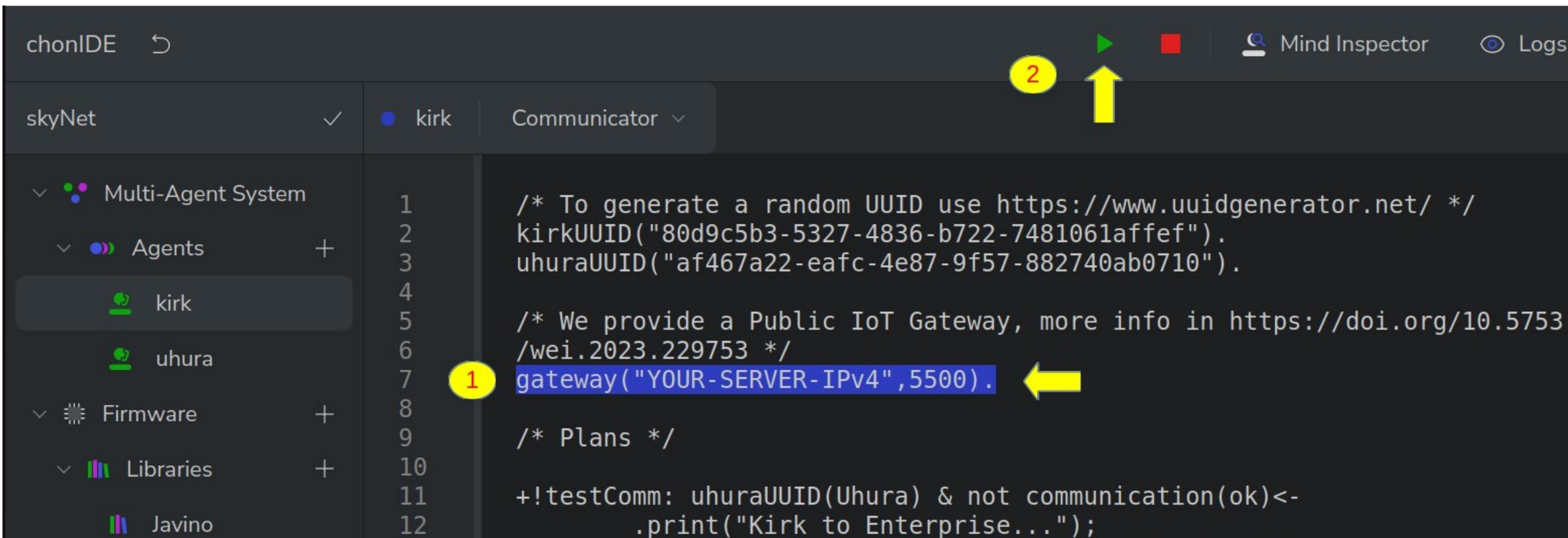
Hermes Agents: Internal Actions

Hermes	<code>.hermes.configureContextNetConnection("X", G, E, U)</code>	Configures an ContextNet network. Where <i>X</i> is a string that represents an network name; <i>G</i> is a literal that represents the FQDN or the network address of an gateway; <i>E</i> is a number that represents the network port of an gateway; <i>U</i> is a literal that represents the identification of the device in the network.
	<code>.hermes.connect("X")</code>	Joins at "X" ContextNet network.
	<code>.hermes.sendOut(D, f, M)</code>	Dispatches a message to another MAS. Where <i>D</i> is a literal that represents the identification of the recipient MAS; <i>f</i> is a illocutionary force (<i>tell</i> <i>untell</i> <i>achieve</i> <i>unachieve</i>); <i>M</i> is a literal that represents the message.
	<code>.hermes.moveOut(D, b, A)</code>	Carries over the agents to other MAS. Where <i>D</i> is a literal that represents the identification of the recipient MAS; <i>b</i> is a bio-inspired protocol (<i>inquilinism</i> <i>mutualism</i> <i>predation</i>); <i>A</i> is one, all, or a set of agents (i.e., all, agent or [agent ₁ , agent ₂ , agent _n])
	<code>.hermes.disconnect("X");</code>	Disconnects at "X" ContextNet network.

Hello SkyNet



Hello SkyNet



chonIDE

skyNet ✓ ● kirk Communicator

Multi-Agent System

Agents +

kirk

uhura

Firmware +

Libraries +

Javino

```
/* To generate a random UUID use https://www.uuidgenerator.net/ */
kirkUUID("80d9c5b3-5327-4836-b722-7481061affef").
uhuraUUID("af467a22-eafc-4e87-9f57-882740ab0710").

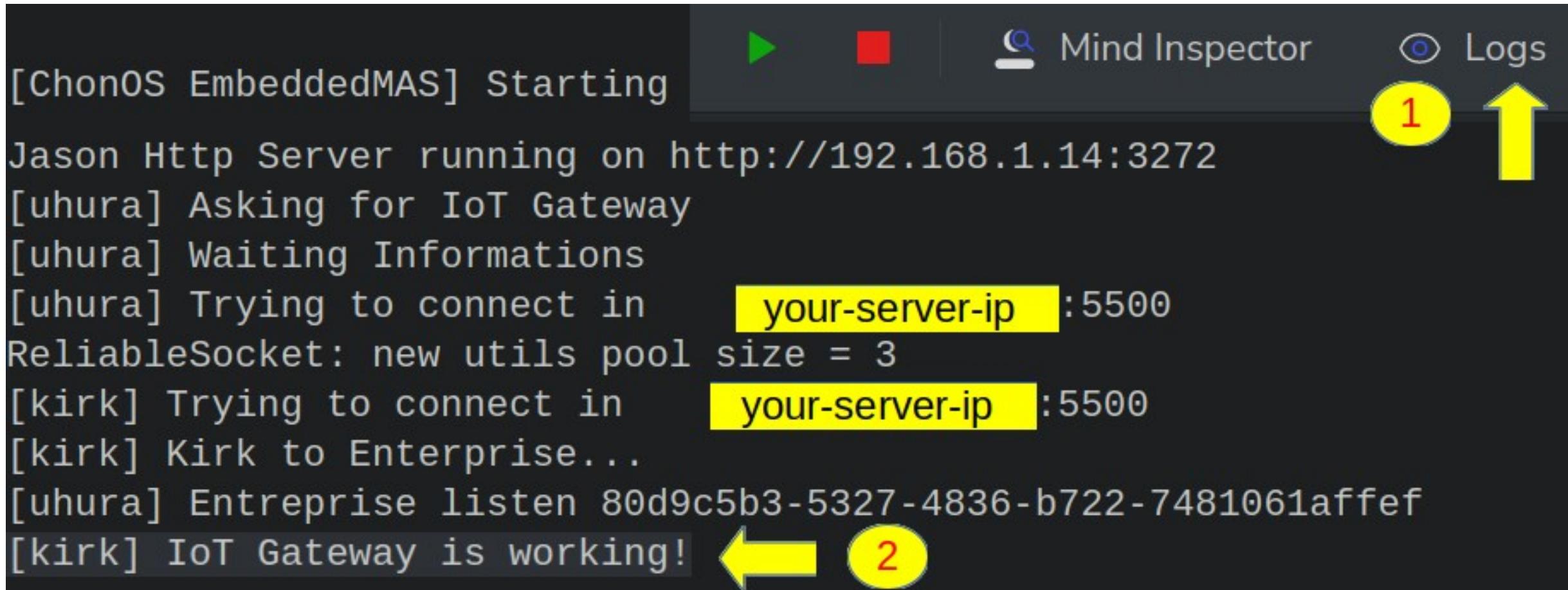
/* We provide a Public IoT Gateway, more info in https://doi.org/10.5753
/wei.2023.229753 */
gateway("YOUR-SERVER-IPv4", 5500). ←

/* Plans */

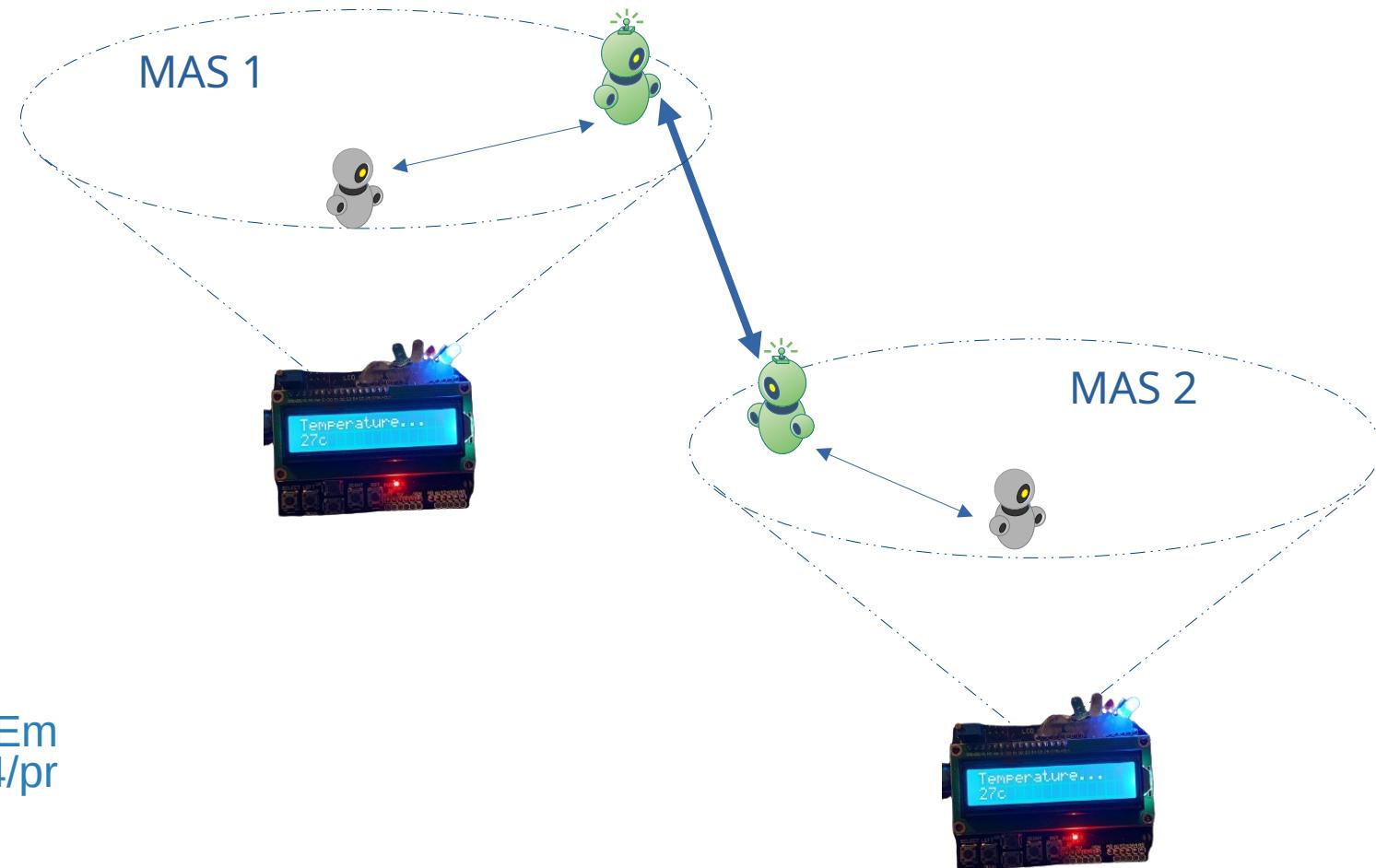
+!testComm: uhuraUUID(Uhura) & not communication(ok)<-
    .print("Kirk to Enterprise...");
```

Hello SkyNet

```
[ChonOS EmbeddedMAS] Starting
Jason Http Server running on http://192.168.1.14:3272
[uhura] Asking for IoT Gateway
[uhura] Waiting Informations
[uhura] Trying to connect in      your-server-ip :5500
ReliableSocket: new utils pool size = 3
[kirk] Trying to connect in      your-server-ip :5500
[kirk] Kirk to Enterprise...
[uhura] Enterprise listen 80d9c5b3-5327-4836-b722-7481061affef
[kirk] IoT Gateway is working!
```



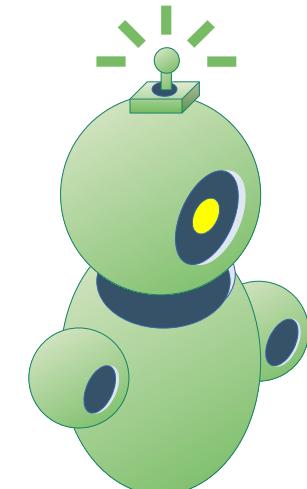
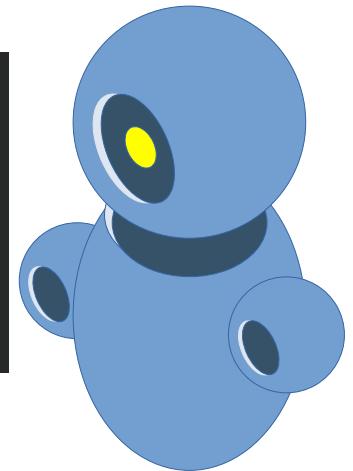
Example: MAS Communication



<https://github.com/chon-group/distributedAndEmbeddedAI/blob/main/workshops/semeso/2024/projects/hermesAGENT.chon>

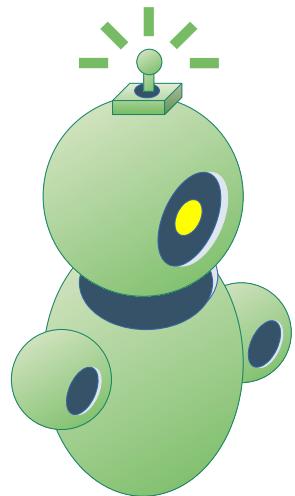
Example: MAS 1

```
+rainLast24hrs(R): R > 50 <-
    .print("New perception-> rainLevel: ",R," mm");
    .concat("Attention rain level is ",R," mm", RainLevelMessage);
    .send(agentHERMES,achieve,forward(RainLevelMessage)).
```



```
+ !forward(Message) <-
    ?neighborUUID(N);
    .print("Forwarding a message to ", N);
    .sendOut(N, achieve, showMessage(Message)).
```

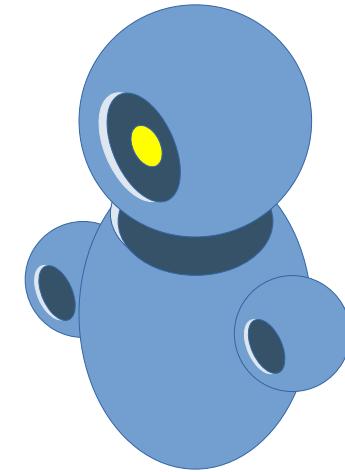
Example: MAS 2



```
+ ! showMessage(Message) [source(Device)] <-
    .print("I received the following message: ", Message);
    .send(agentARGO, achieve, infoLCD(Message)).
```



```
+ ! infoLCD(Message) <-
    .random(R); .wait(2000*R);
    .argo.act(Message).
```



Simple Exercise

Adjust to:

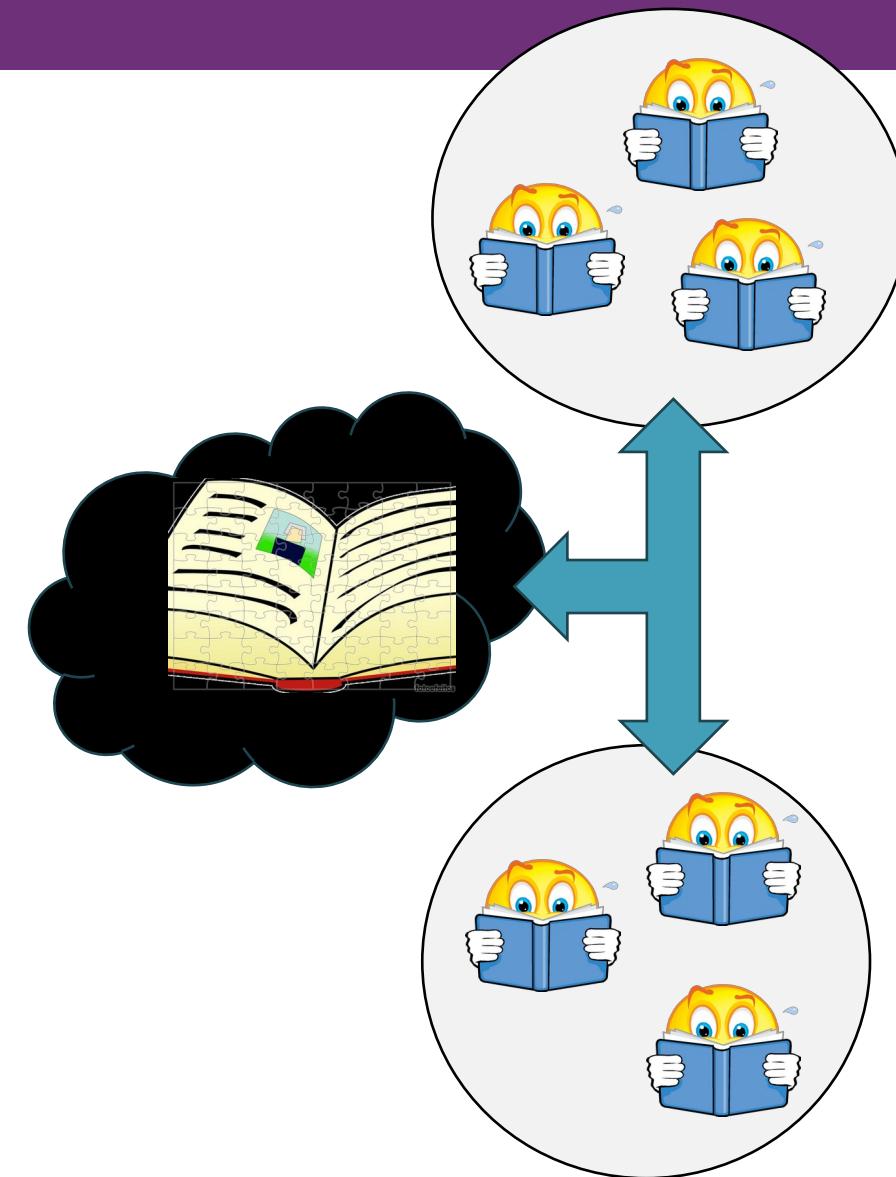
- turn on the **green LED** in **MAS 2** if the rain intensity less than 30mm in **MAS 1**;
- turn on the **yellow LED** in **MAS 2** if the rain intensity exceeds 50mm in **MAS 1**;
- turn on the **red LED** in **MAS 2** if the the rain intensity exceeds 80mm in **MAS 1**;

VELLUSCINUM

<https://github.com/chon-group/Velluscinum/wiki>



MAS + DLT



- Can facilitate the agreement between agents
Taking what is registered in the Ledger as accurate;
- Can be helpful to:
Manage trust relationships;
Open MAS;
Distributed scenarios.

Objective

Using digital assets in the relationships between cognitive agents:

the transfer of funds;

registration of ownership of artifacts;

declaration of promises or agreements.

Allowing the agents to manipulate assets and wallets **directly in the DLT**.

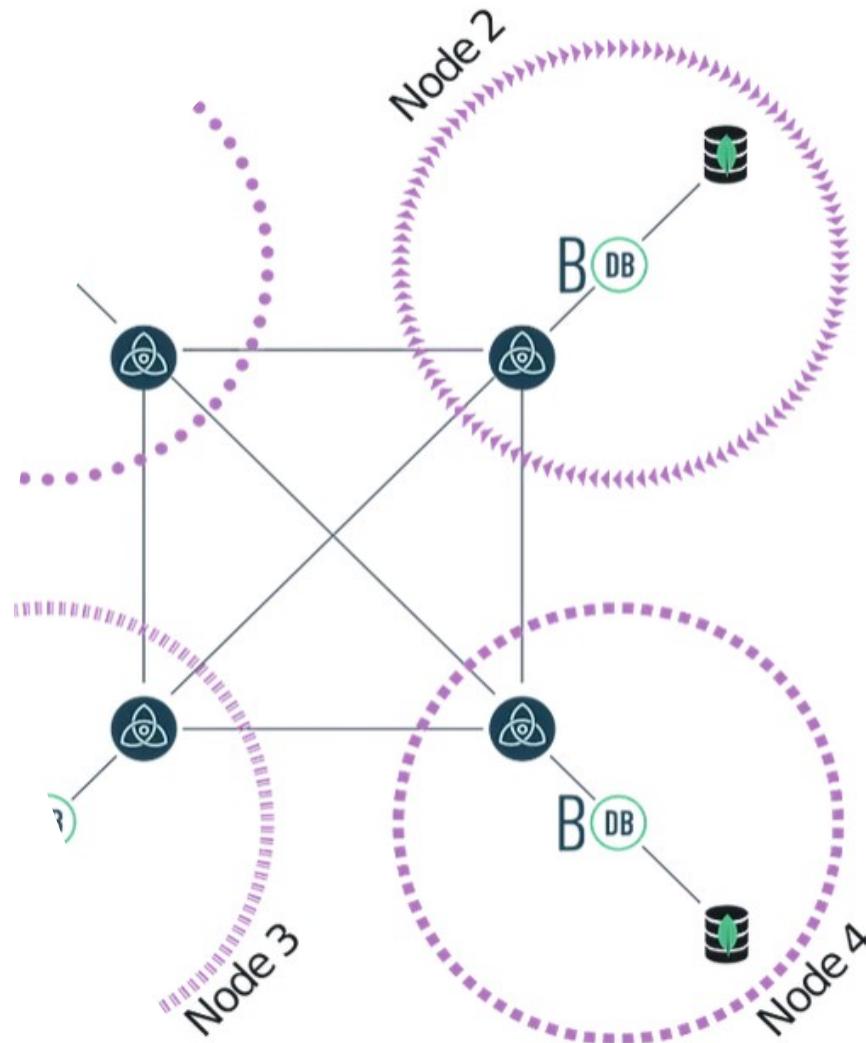


Theoretical foundation

- A DLT can be considered an append-only decentralized database because it provides a storage mechanism.

However, compared with a traditional database, its **performance is much lower** because it has a **low download rate** and a **high latency** [13].

New approaches **based on distributed databases** have been used to improve the performance of permissioned DLT.



13. McConaghy, T., Marques, R., Müller, A., De Jonghe, D., McConaghy, T., McMullen, G., Henderson, R., Bellemare, S., Granzotto, A.: Bigchaindb: a scalable blockchain database. white paper, BigChainDB (2016)

Theoretical foundation

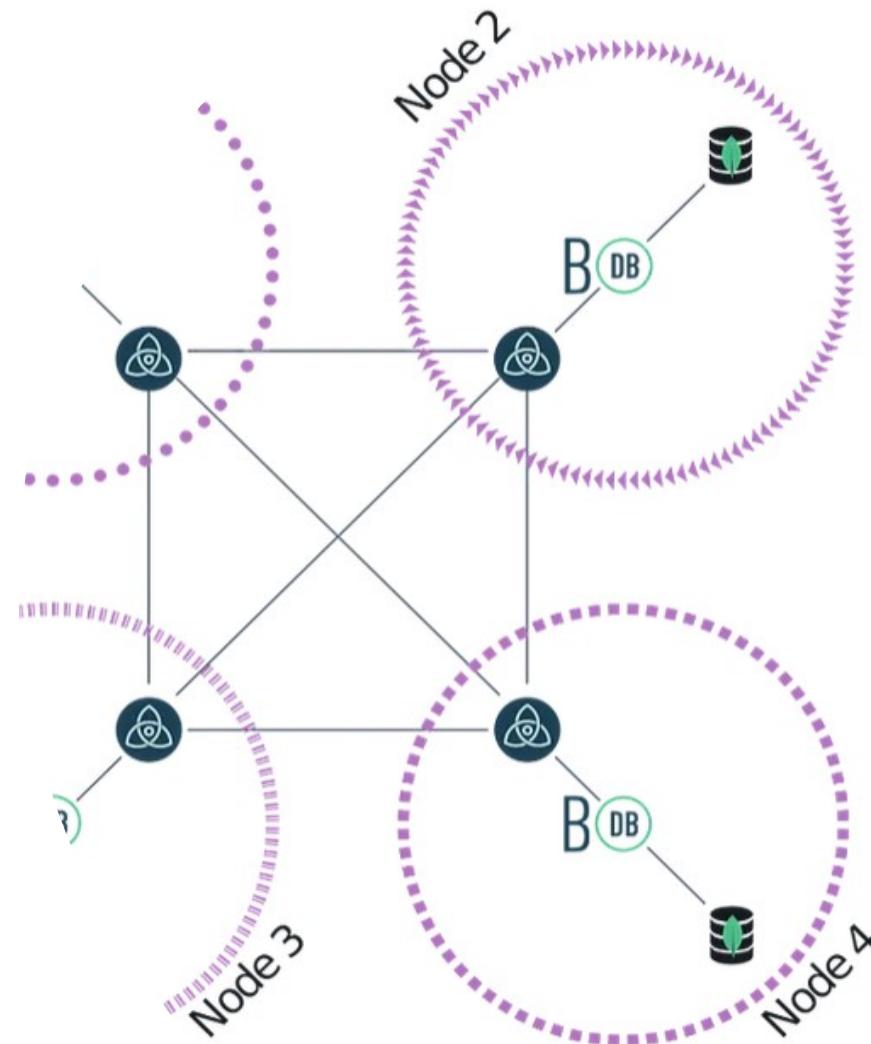
- When considering the number of transactions that a DLT can successfully execute per second:
 - DLT permissionless (e.g., Bitcoin or Ethereum)*
 - ranges between 3.45 and 4.69 TPS [24, 15];*
 - DLT permissioned (e.g., HyperLedger Fabric)*
 - vary between 4.28 and 10.51 TPS [7, 15];*
 - DLT permissioned distribution database-based (e.g., BigchainDB)*
 - varies between 50.60 and 175 TPS [7, 10].*

7. Contreras, A.F.: Benchmarking of blockchain technologies used in a decentralized data marketplace. Grado en Ingeniería Informática, E.T.S de Ingenieros Informáticos (UPM), Madrid, España (Jun 2019). <https://oa.upm.es/55775/>

10. Ge, Z., Loghin, D., Ooi, B.C., Ruan, P., Wang, T.: Hybrid blockchain database systems: Design and performance. Proc. VLDB Endow. 15(5), 1092–1104 (may 2022). <https://doi.org/10.14778/3510397.3510406>

15. Nasir, Q., Qasse, I.A., Abu Talib, M., Nassif, A.B.: Performance Analysis of Hyperledger Fabric Platforms. Security and Communication Networks 2018, 1–14 (2018). <https://doi.org/10.1155/2018/3976093>

24. Tikhomirov, S.: Ethereum: State of knowledge and research perspectives. In: Imine, A., Fernandez, J.M., Marion, J.Y., Logrippo, L., Garcia-Alfaro, J. (eds.) Foundations and Practice of Security. pp. 206–221. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-75650-9_14



Proposed Approach

Use of digital assets to supporting the BDI agents' relationship:

Indivisible assets

as property record

as promise or agreement

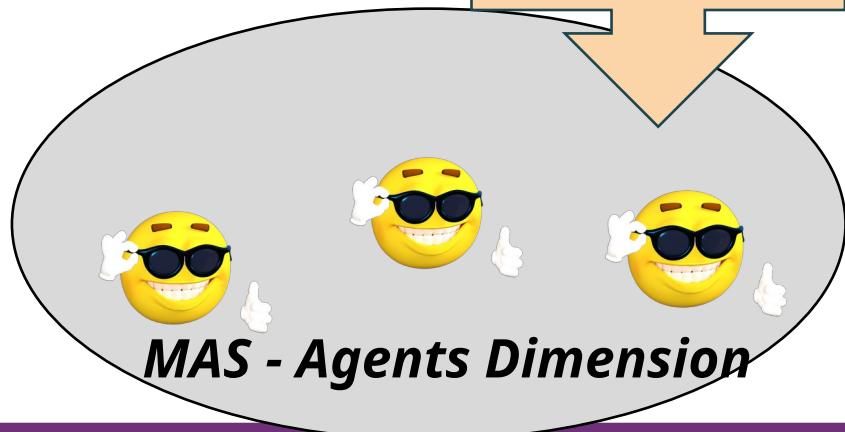
Divisible assets

as cryptocurrencies

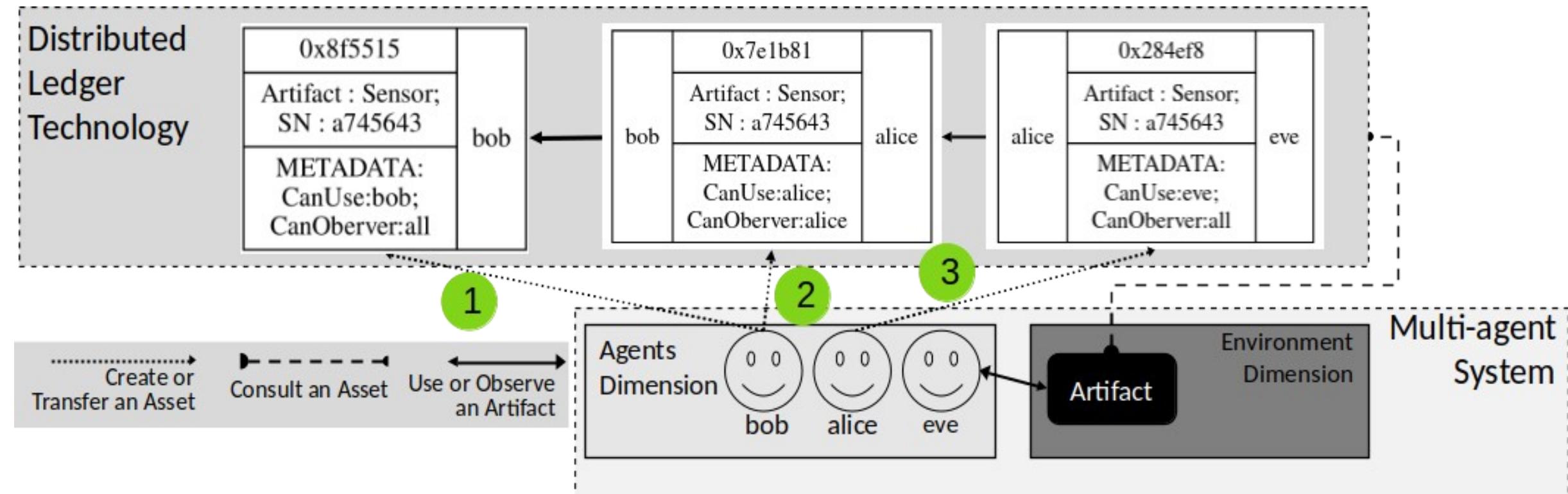
Distributed ledger technology



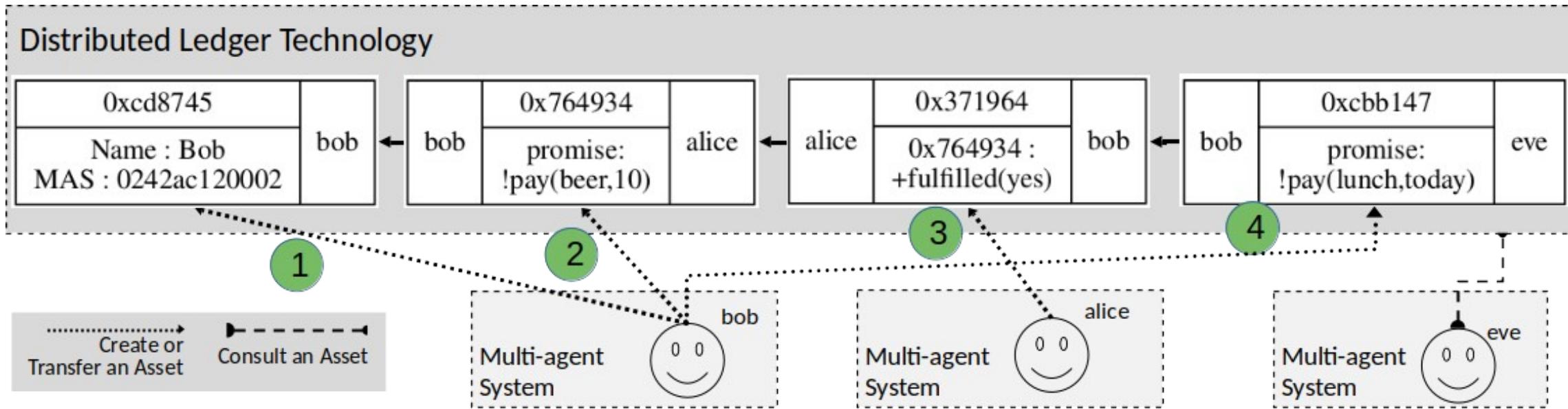
Velluscinum



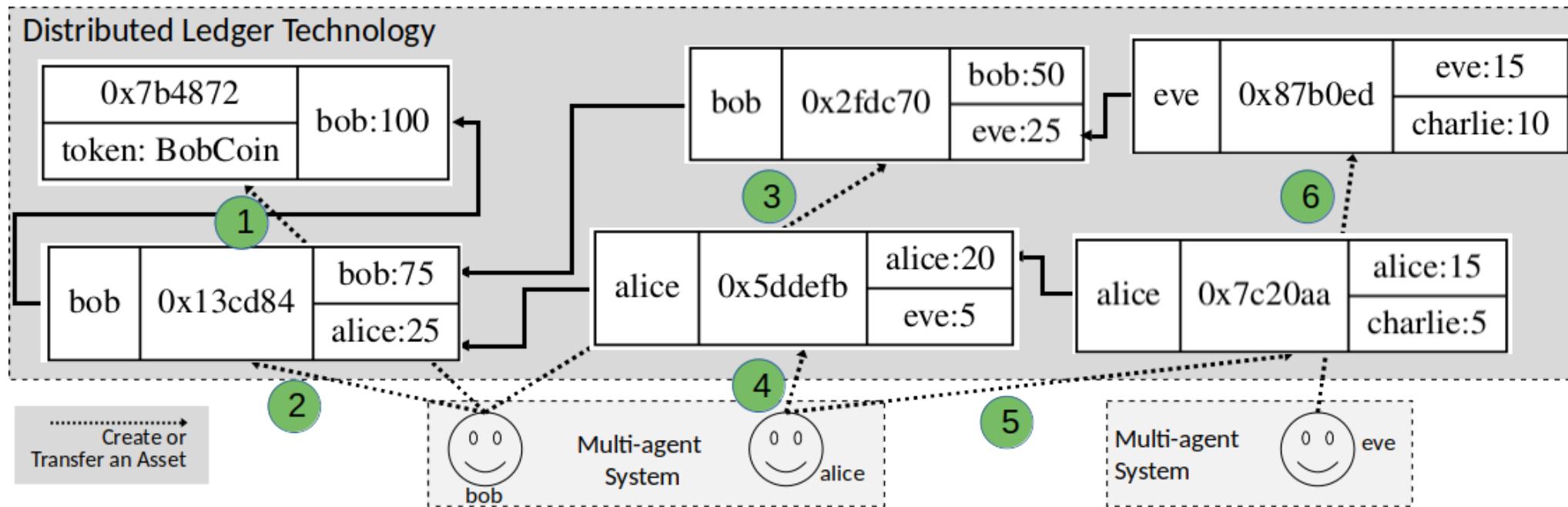
Indivisible asset such as property record



Indivisible asset such as promise or agreement



Divisible asset such as cryptocurrency



The Internal Actions provided by the Velluscinum

Internal Action	Description
<u>buildWallet</u>	Generates a digital wallet and returns a belief.
<u>loadWallet</u>	Loads (or creates if not exists) the agent's key pair into the project folder, that represents a digital wallet, and returns a belief.
<u>storeWallet</u>	Persists a belief that represents the agent's digital wallet as a key pair into the project folder.
<u>walletContent</u>	Retrieves the content of a Wallet and return a belief with a list of assets.
<u>tokenBalance</u>	Check <i>units</i> of a specific token into a wallet and return a belief.
<u>tokenInfo</u>	Retrieves the content of a Token and return a belief with a list inside.
<u>deployNFT</u>	Registers an asset and returns a belief.
<u>transferNFT</u>	Transfers a Non-Fungible-Token and returns a belief.
<u>deployToken</u>	Creates <i>n</i> units from an asset and returns a belief.
<u>transferToken</u>	Transfer <i>V</i> units of <i>C</i> and returns <i>+b(T)</i> ;
<u>stampTransaction</u>	Stamps a transaction.



<https://github.com/chon-group/distributedAndEmbeddedAI/blob/main/workshops/semeso/2024/projects/chefGlutonVellus.chon>

Image credits:

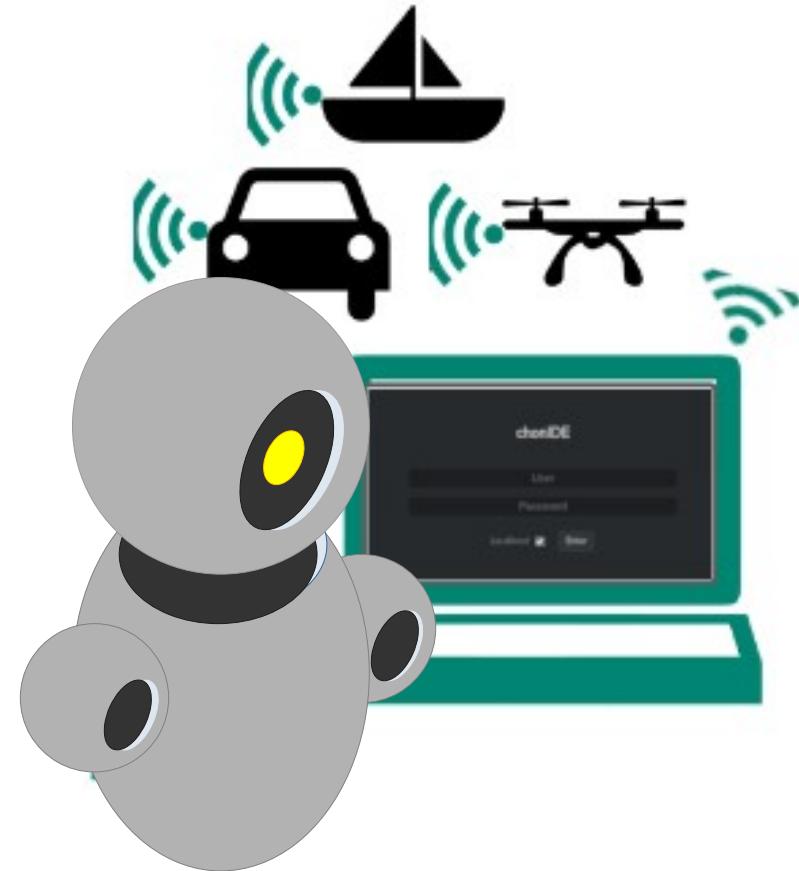
https://smurfs.fandom.com/wiki/Chef_Smurf

<https://smurfs.fandom.com/pt/wiki/Fominha>

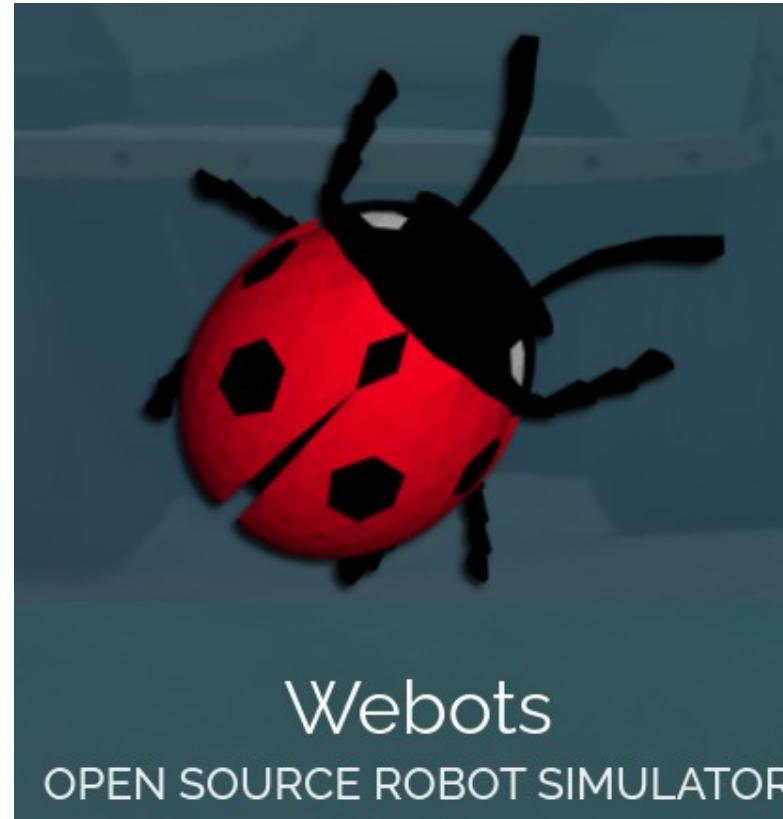
https://smurfs.fandom.com/pt-br/wiki/%C3%81vore_de_Dinheiro



OTHER TOOLS

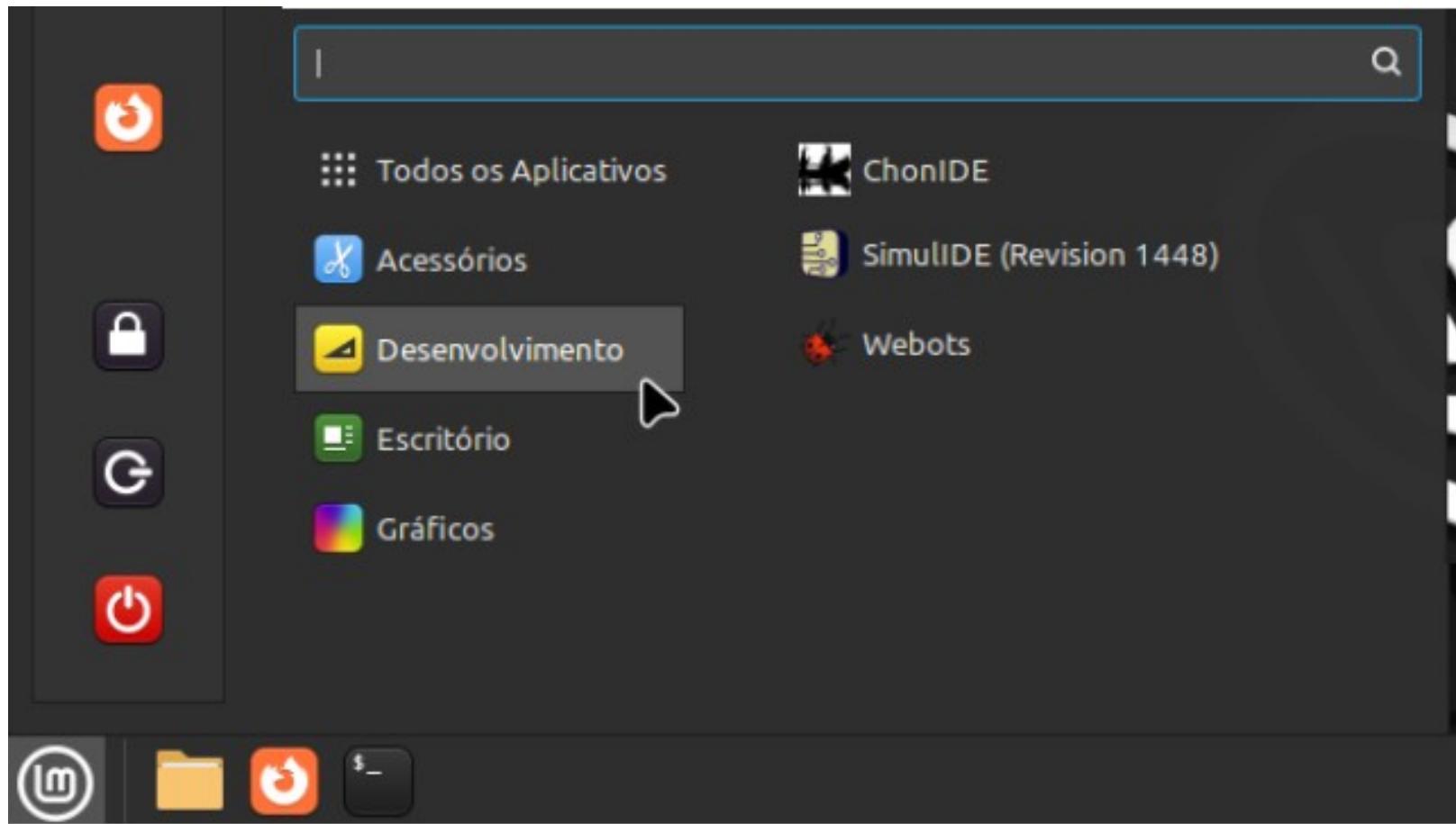


Webots



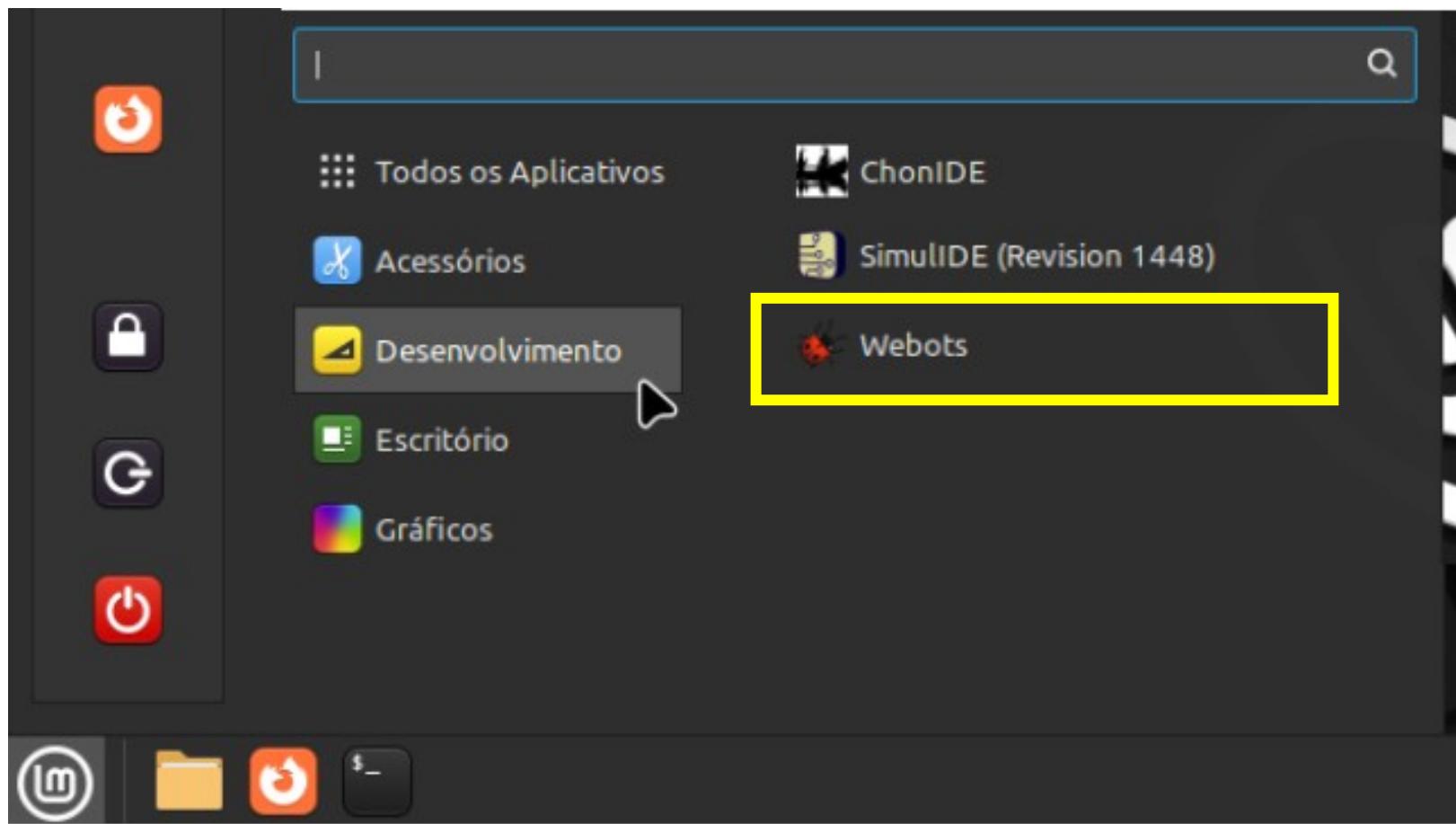
Michel, O. (1998). Webots: Symbiosis Between Virtual and Real Mobile Robots. In: Heudin, JC. (eds) Virtual Worlds. VW 1998. Lecture Notes in Computer Science(), vol 1434. Springer, Berlin, Heidelberg.
https://doi.org/10.1007/3-540-68686-X_24

Webots



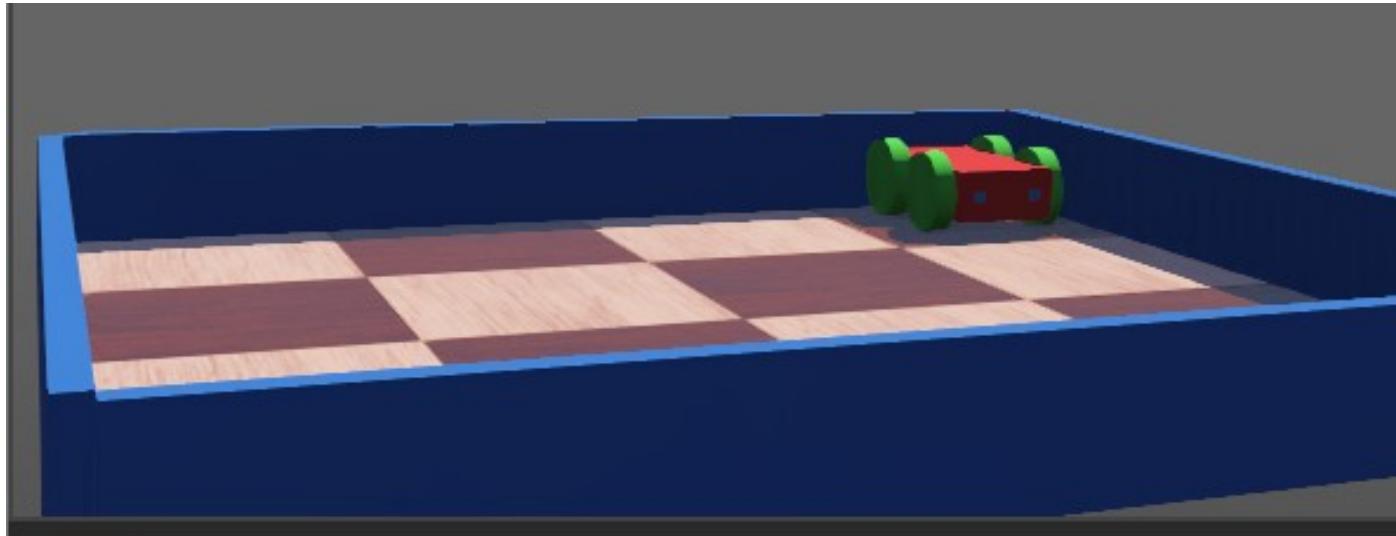
Manual de instalação
<https://cyberbotics.com/doc/guide/installation-procedure#installing-the-debian-package-with-the-advanced-packaging-tool-apt>

Webots



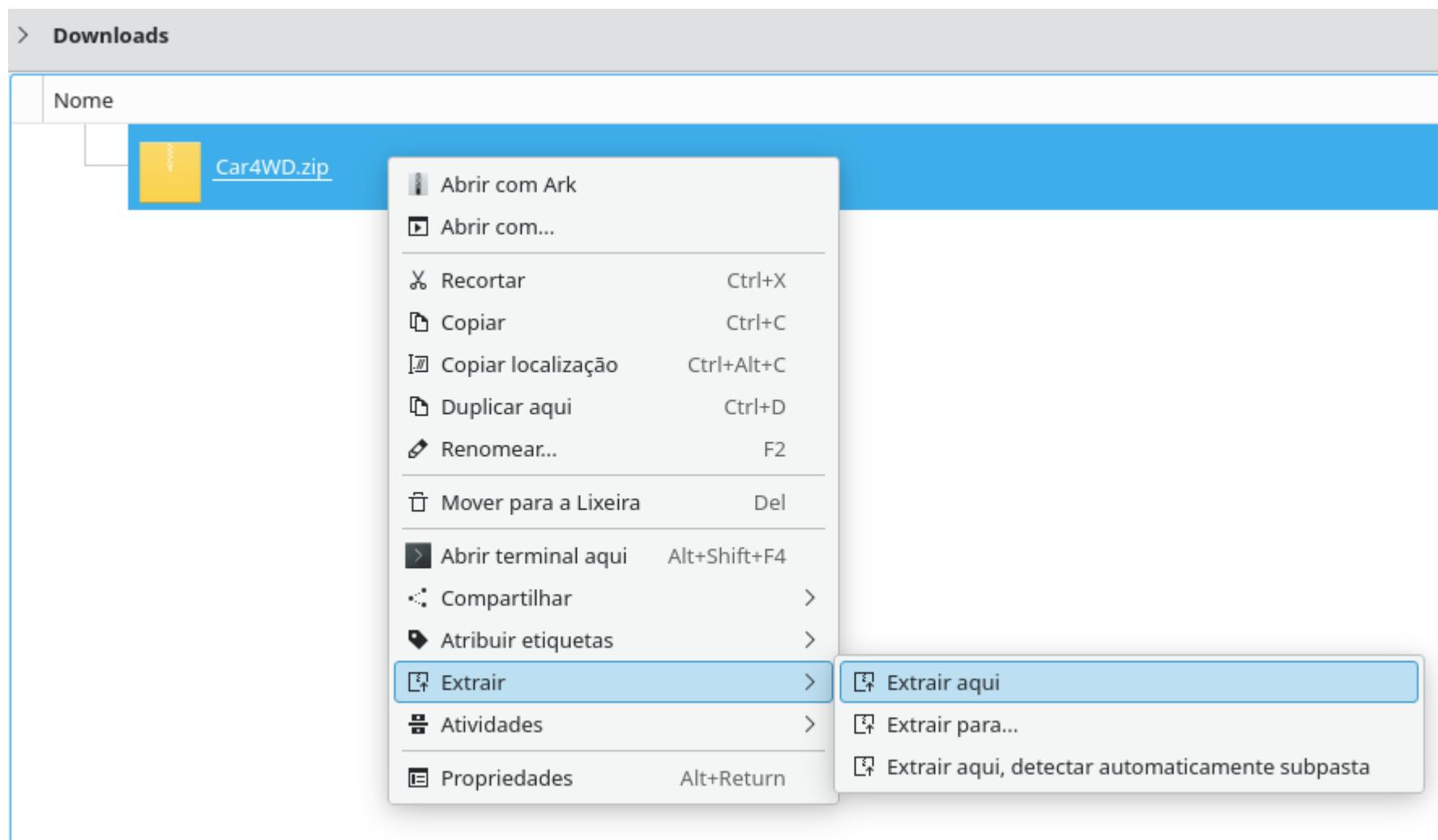
Manual de instalação
<https://cyberbotics.com/doc/guide/installation-procedure#installing-the-debian-package-with-the-advanced-packing-tool-apt>

Webots

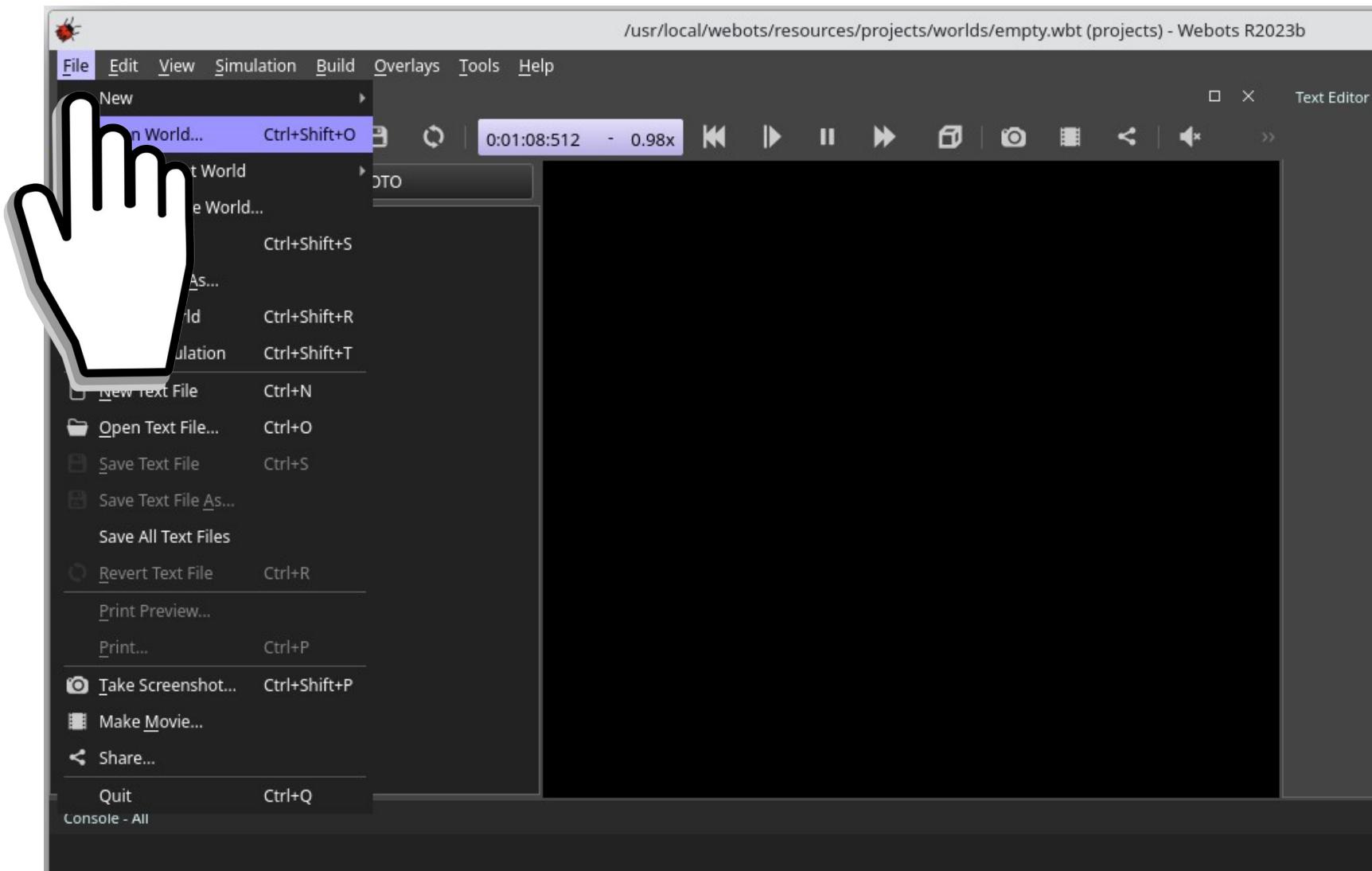


[https://github.com/chon-group/
distributedAndEmbeddedAI/blob/main/workshops/
semeso/2024/projects/Car4WD.zip](https://github.com/chon-group/distributedAndEmbeddedAI/blob/main/workshops/semeso/2024/projects/Car4WD.zip)

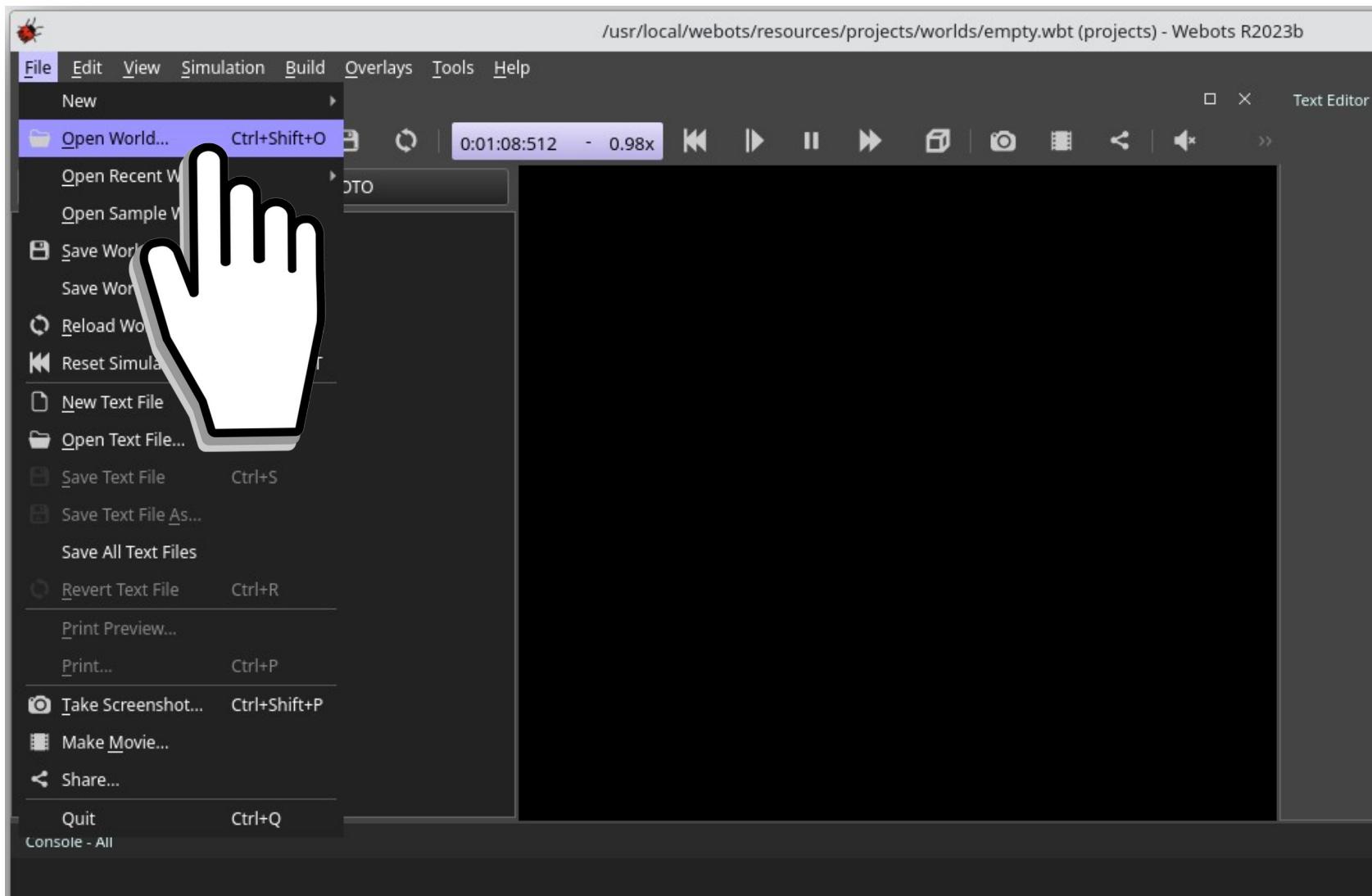
Webots



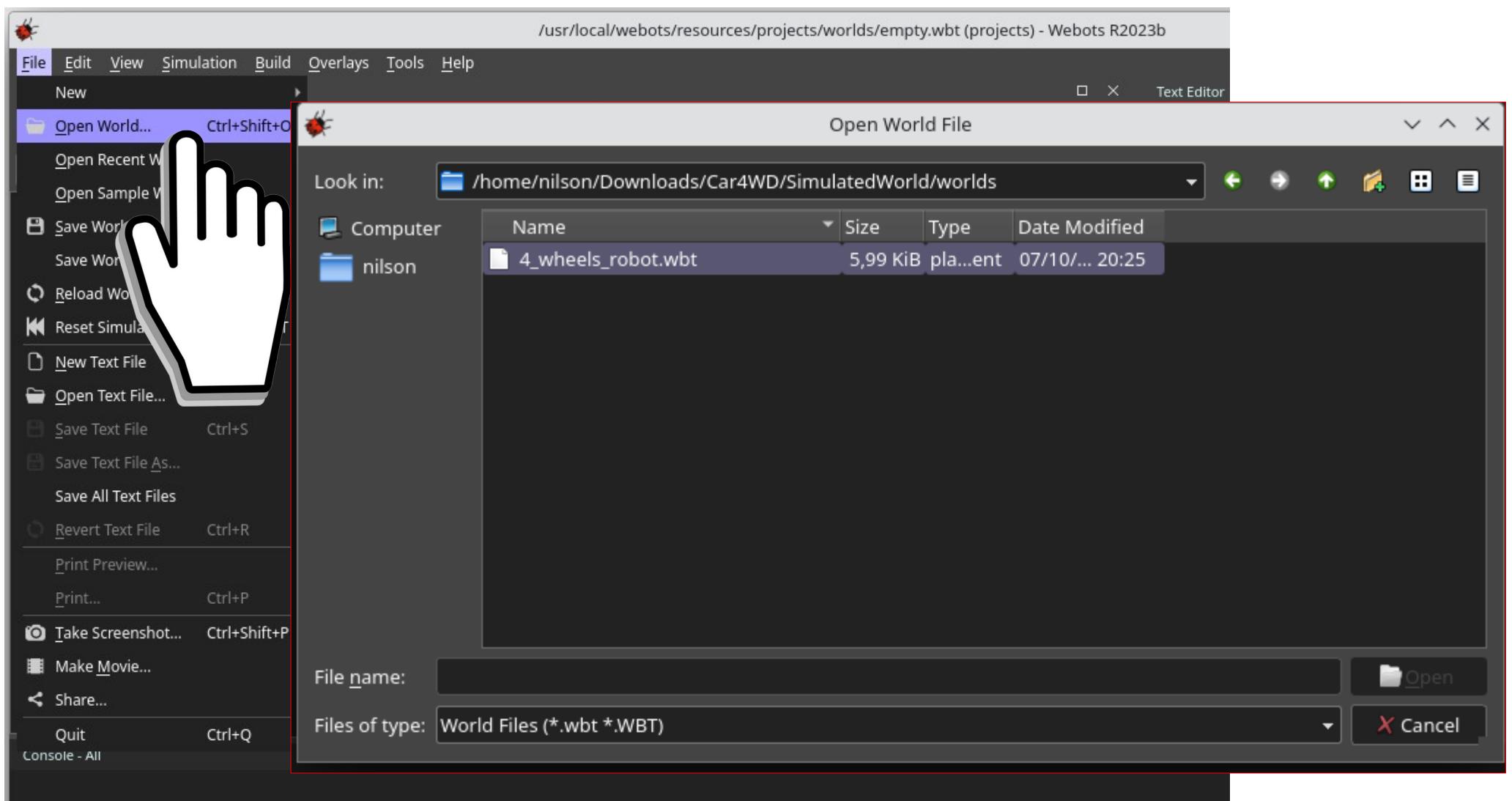
Webots



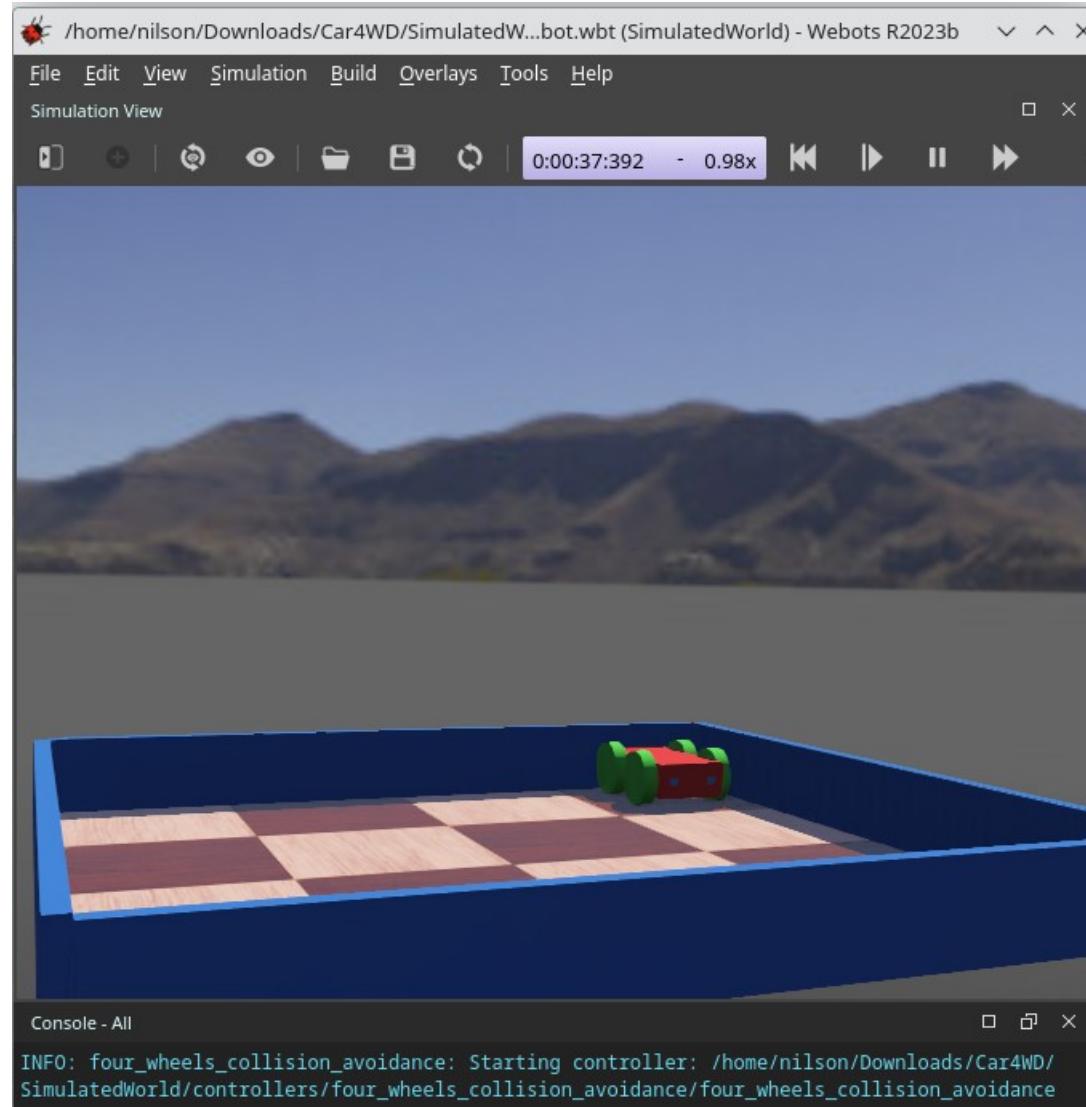
Webots



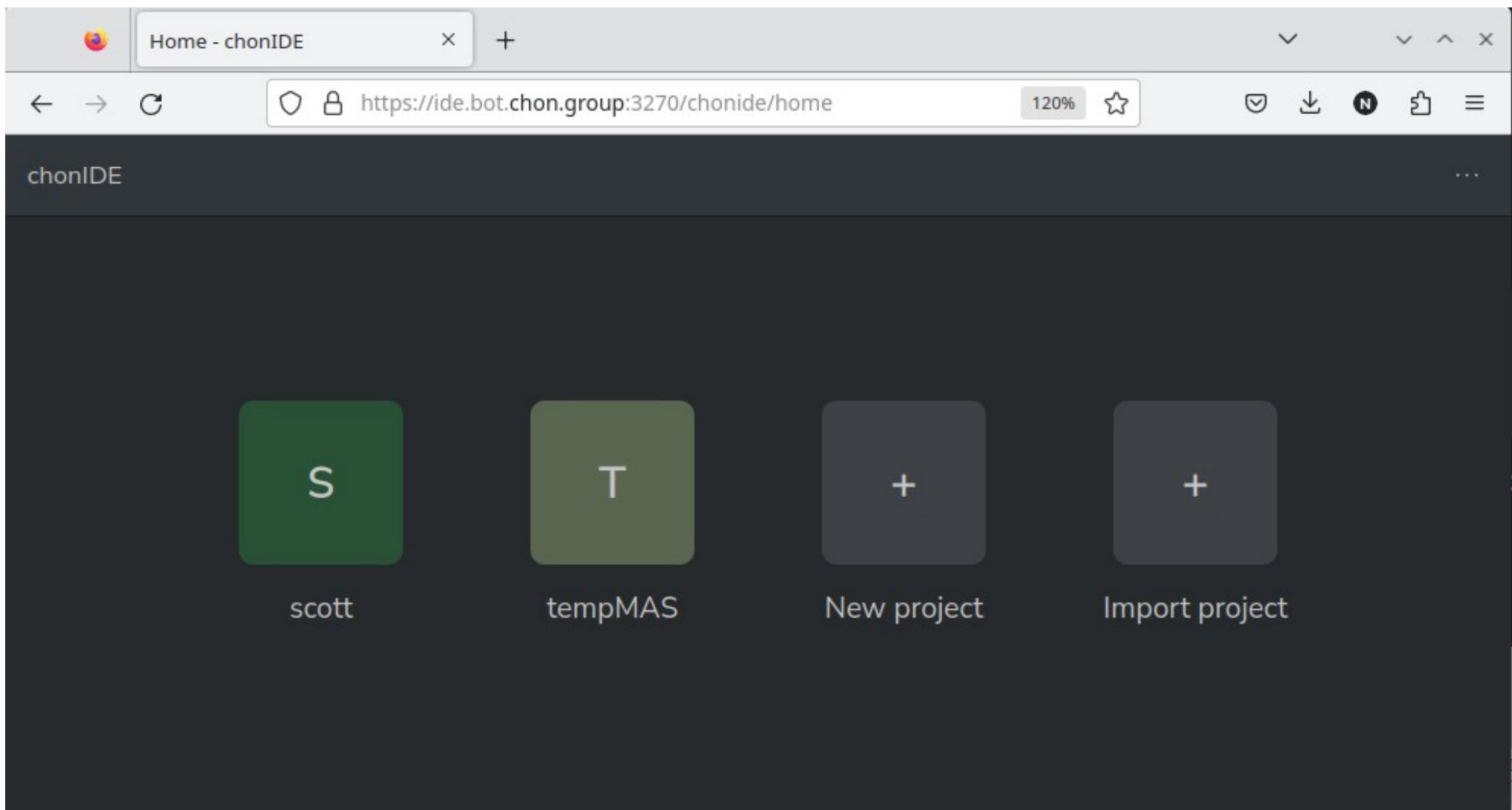
Webots



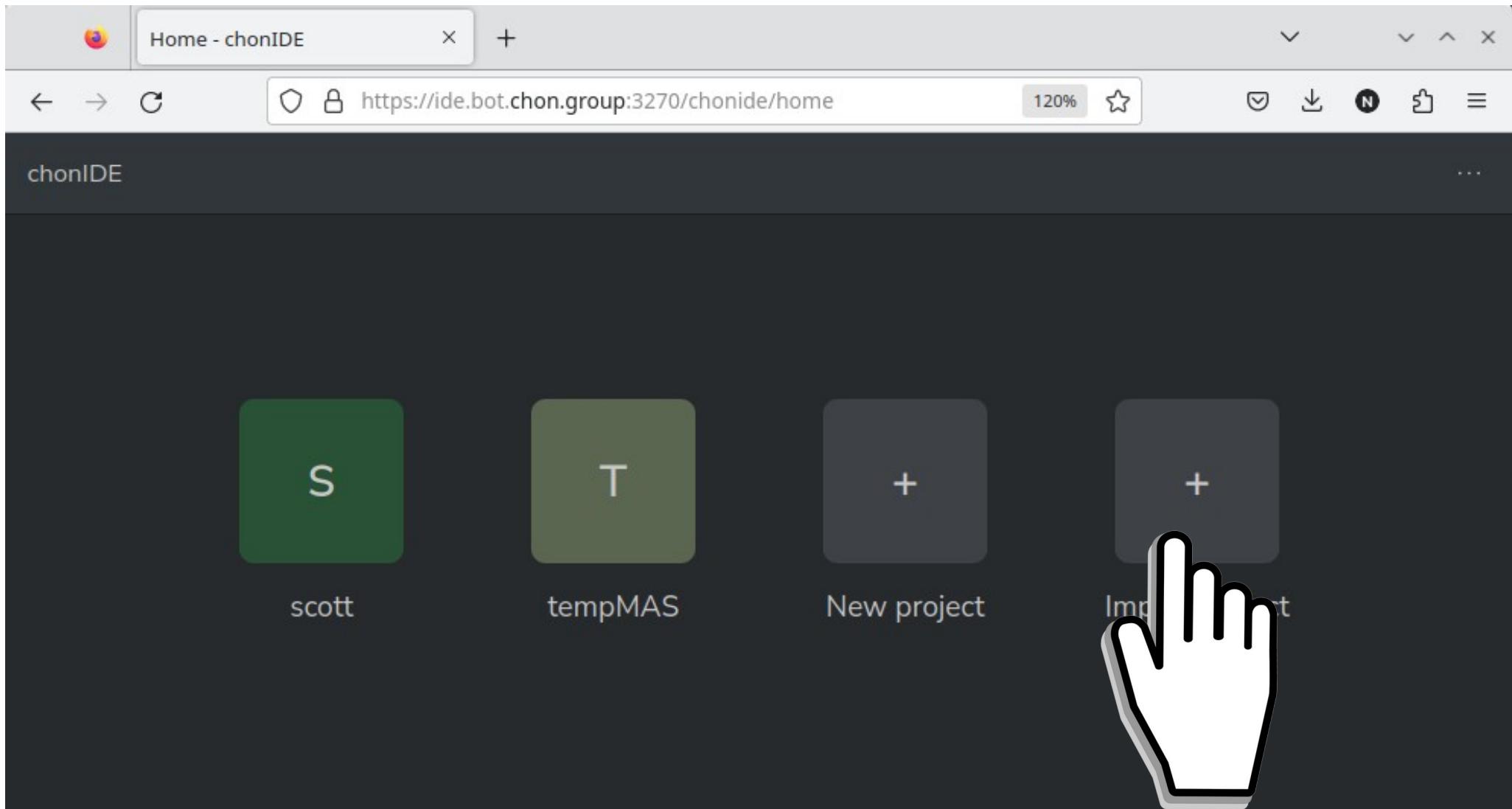
Webots



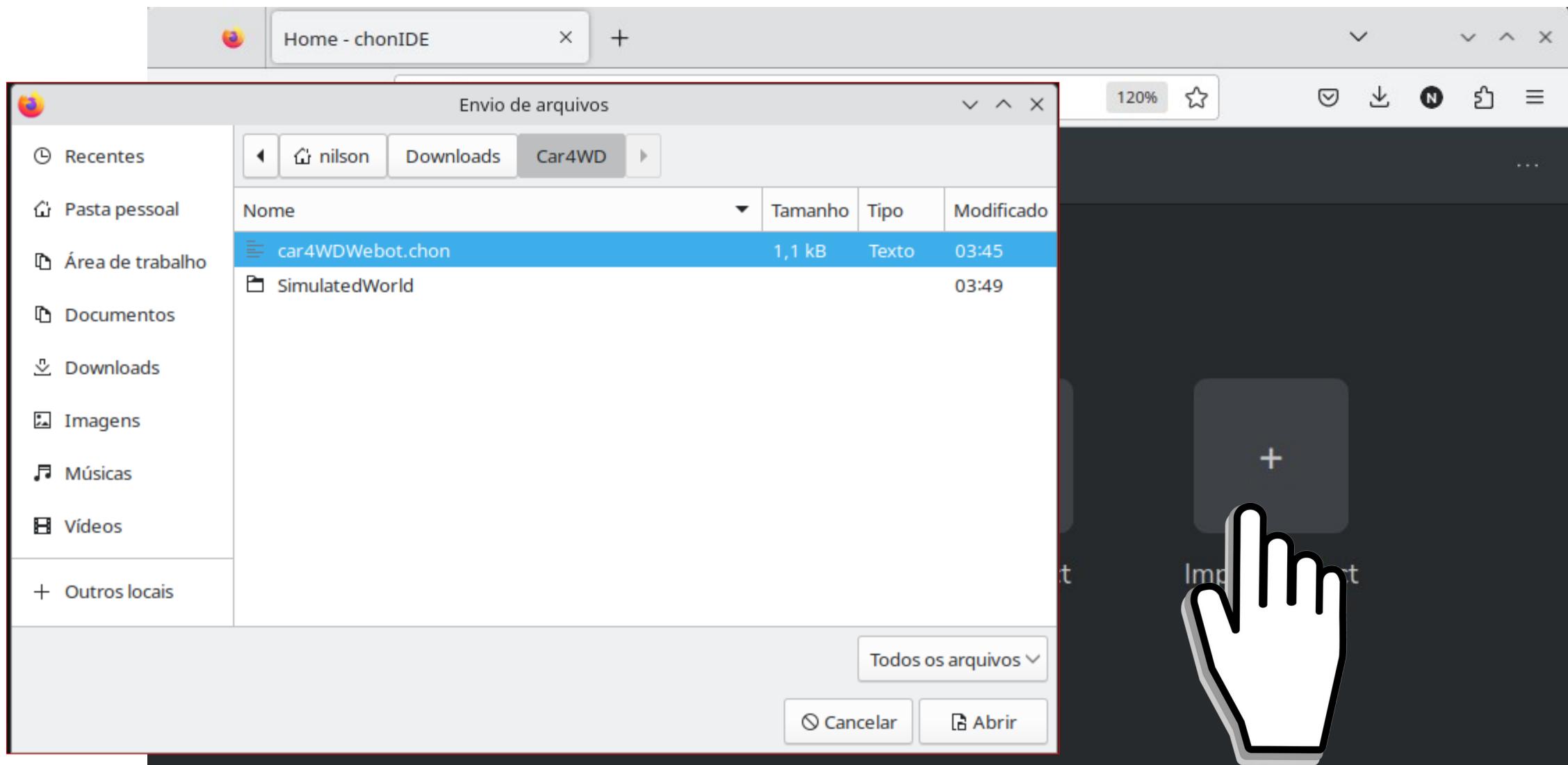
Webots



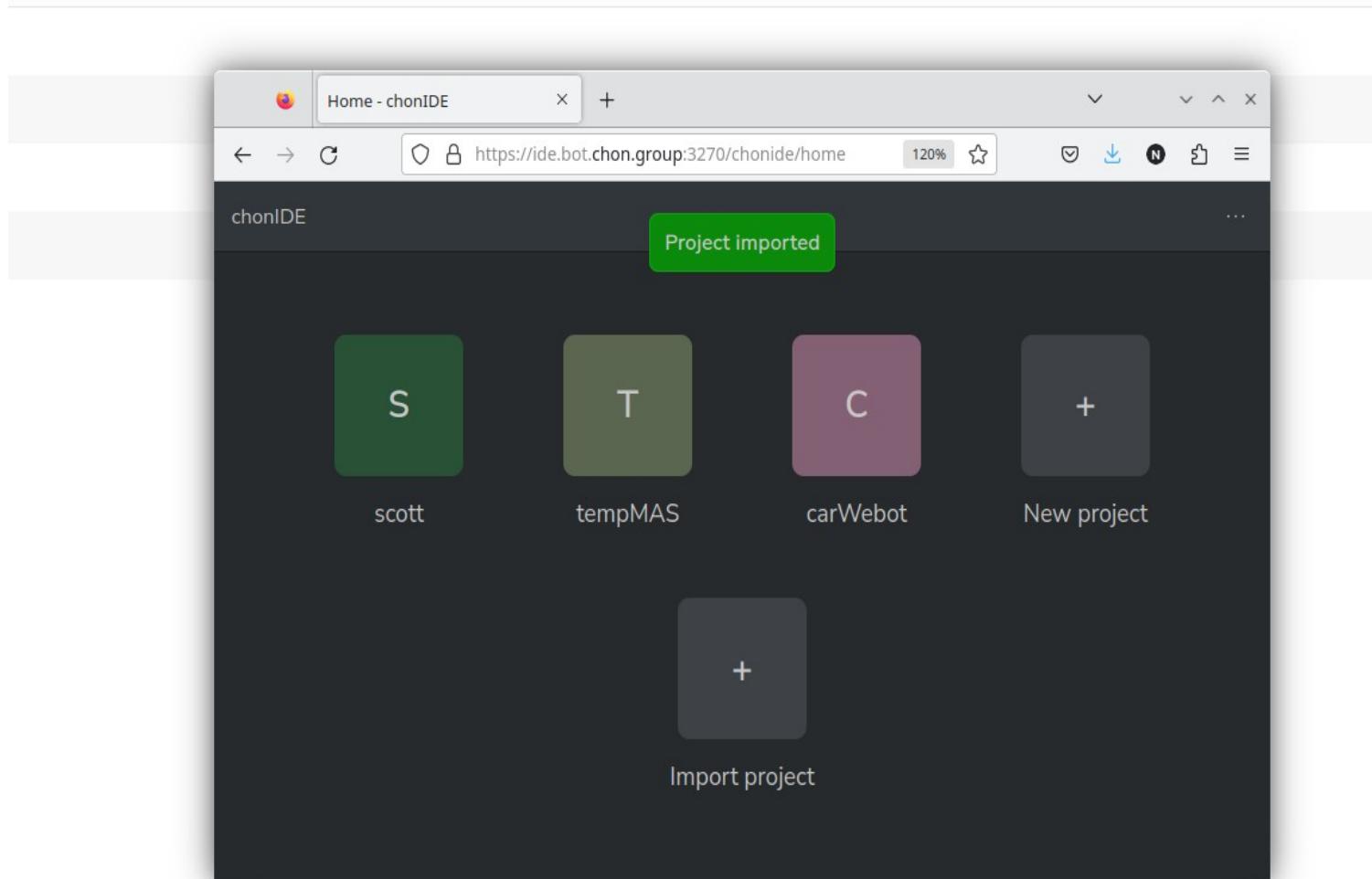
Webots



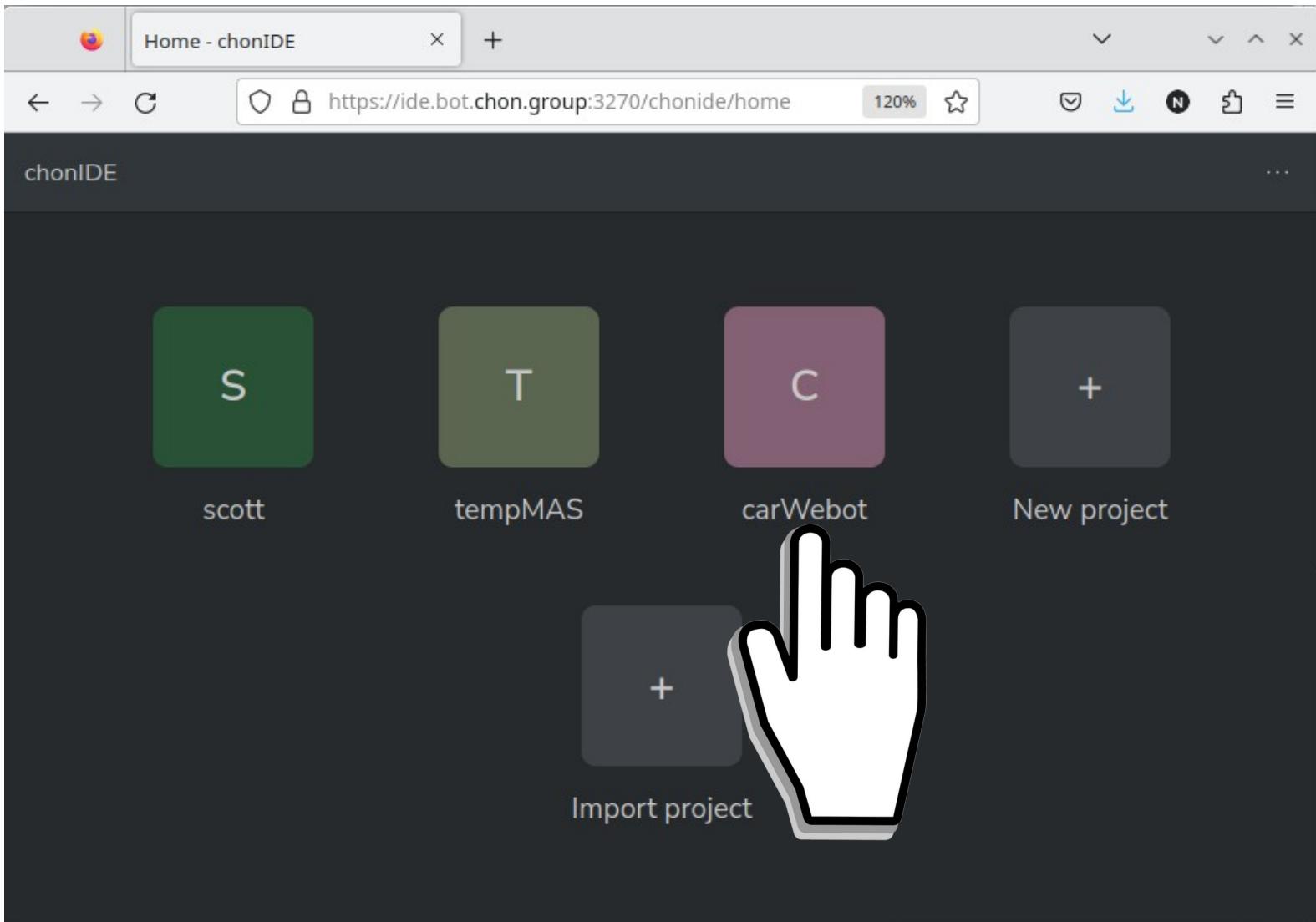
Webots



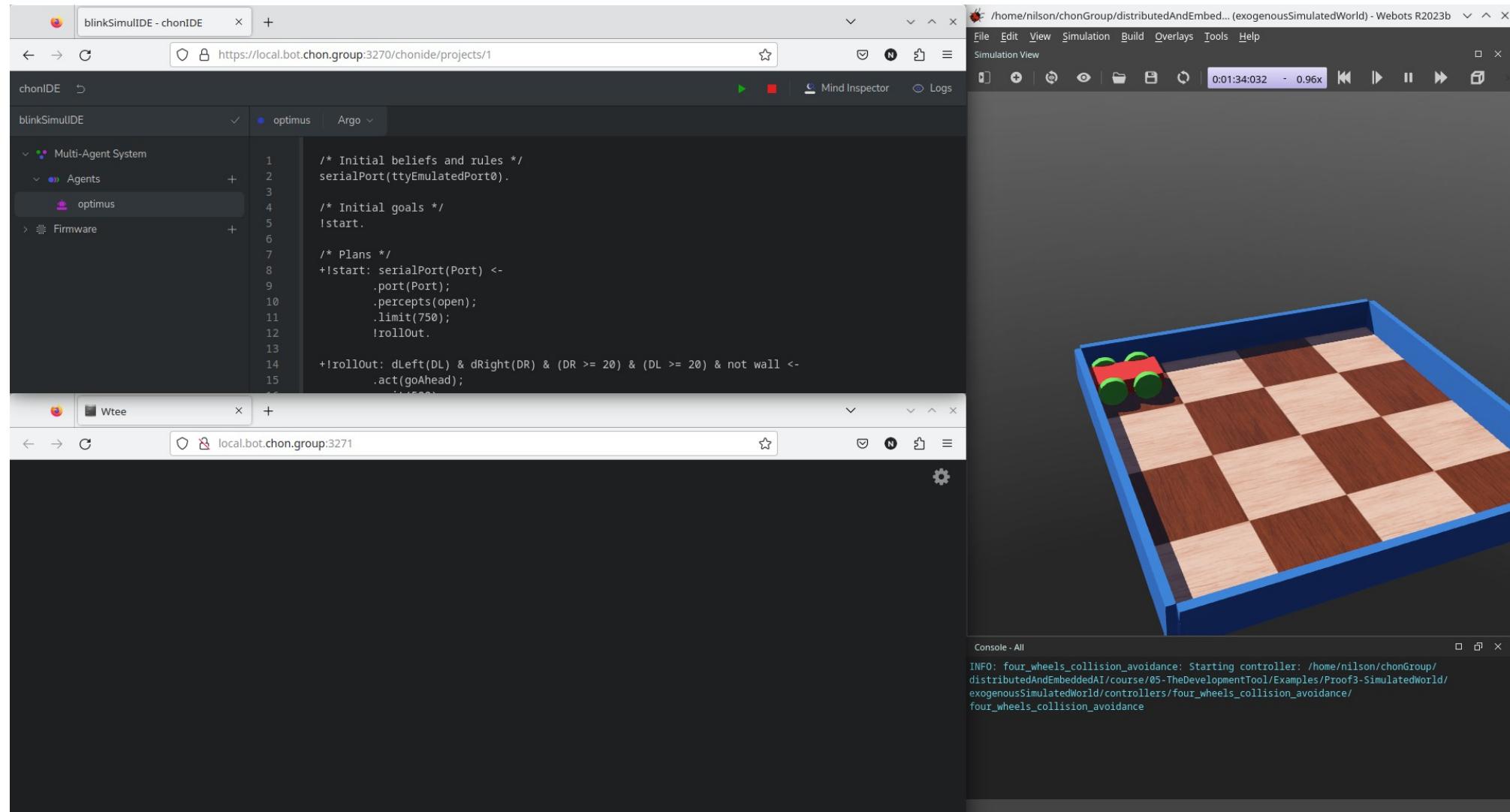
Webots



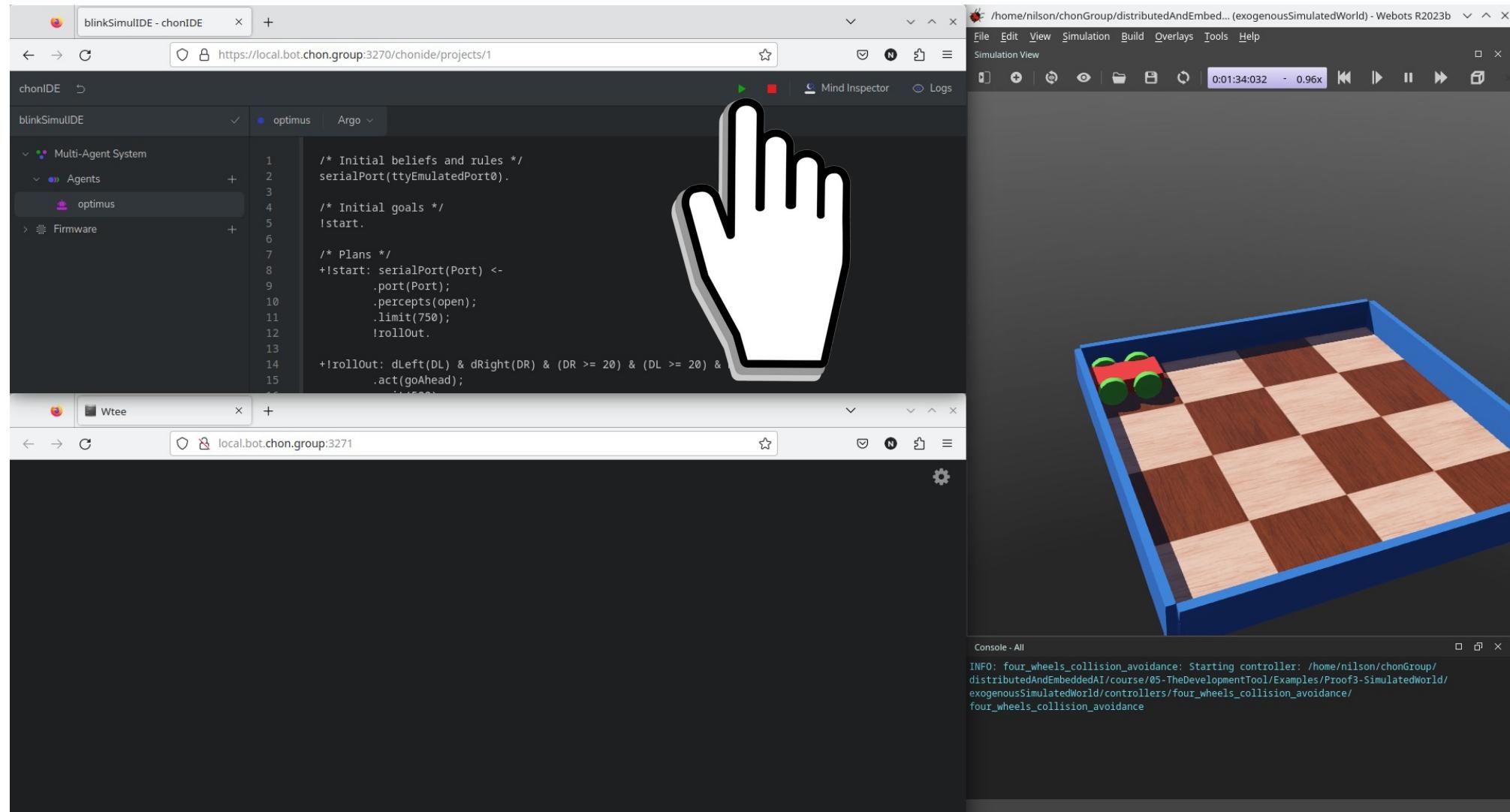
Webots



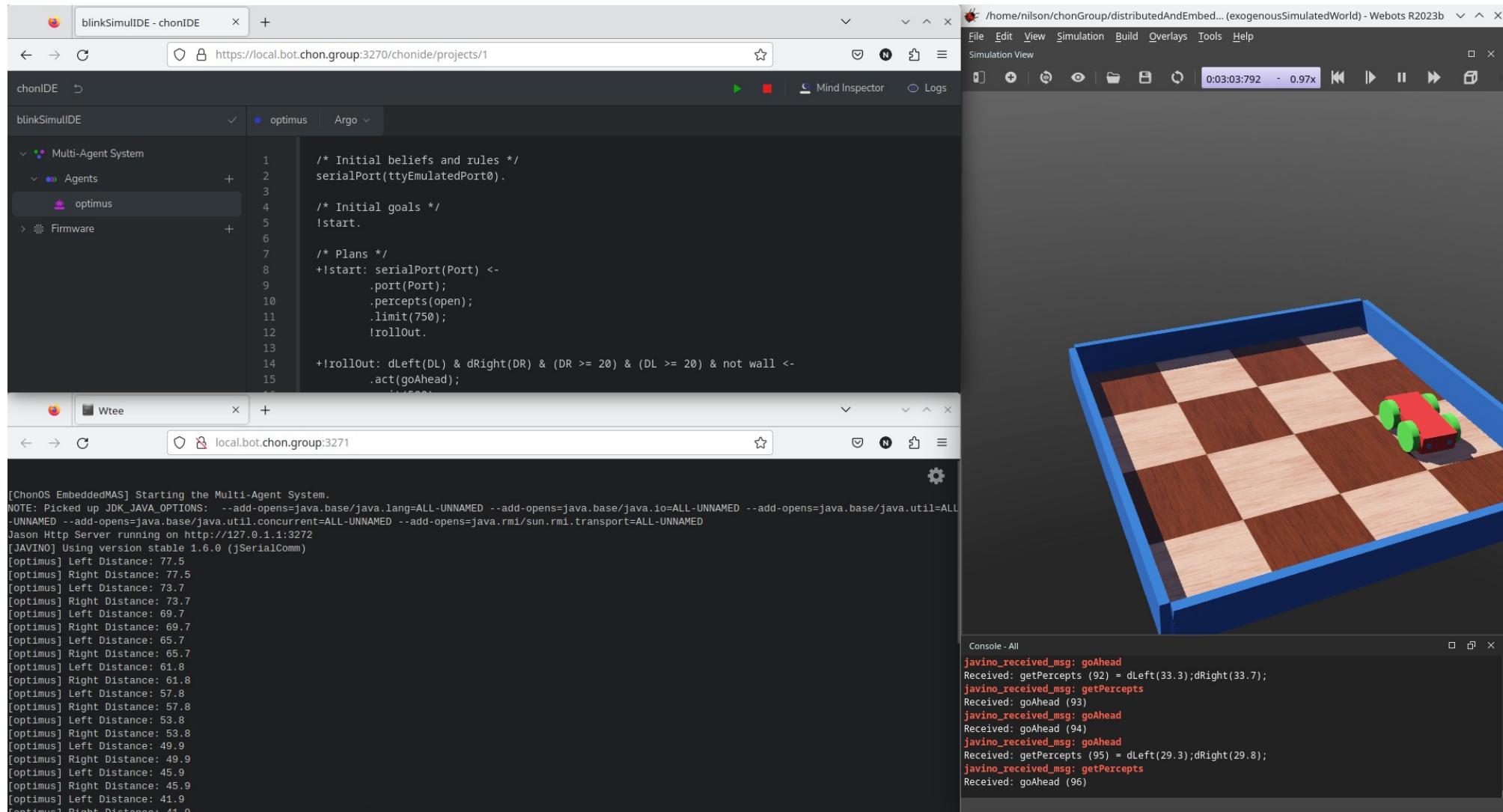
Webots



Webots



Webots



Acknowledgements

THANK YOU!

nilson.lazarin@cefet-rj.br



UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

ALTO VALE

CENTRO DE EDUCAÇÃO SUPERIOR
DO ALTO VALE DO ITAJAÍ



FAPESC

Fundação de Amparo à
Pesquisa e Inovação do
Estado de Santa Catarina



CEFET/RJ
Uned Nova Friburgo



Instituto de
Computação