Human and BDI-Agent Interaction via KQML messages over IMAP and SMTP

Nilson Mori Lazarin[1,2], João Pedro Bernardo de Souza[2], Carlos Eduardo Pantoja[2], Tielle Alexandre[1], and Jose Viterbo[1]

[1] Institute of Computing - Fluminense Federal University (UFF), Niterói - RJ, Brazil
`{nlazarin,tiellesa}@id.uff.br viterbo@ic.uff.br`

[2] Federal Center for Technological Education Celso Suckow da Fonseca (Cefet/RJ), Rio de Janeiro - RJ, Brazil
`joao.bernardo@aluno.cefet-rj.br pantoja@cefet-rj.br`

Abstract. Multi-agent Systems (MAS) are groups of autonomous agents that act in the same environment, competing or collaborating to achieve individual or collective goals. Such systems can be integrated into embedded devices to perform tasks that control hardware. The main challenges encountered between Embedded MAS are the connectionless protocols some IoT middleware use for agent communication and the lack of security in transferring messages over the network. Therefore, it is essential to use efficient, safe, and reliable means of communication. This work proposes a connection-oriented communication architecture using secure channels provided by network protocols in the application layer. This work contributes to the advancement of MAS by addressing the integration of email protocols and seeking the development of a customized agent architecture based on the Jason framework to enable secure and simplified human-agent interaction.

Keywords: Embedded MAS · Human-Agent Interaction · BDI model.

1 Introduction

A Multi-Agent System (MAS) is a set of computational elements, known as agents, that interact with each other. An agent can perceive the environment in which it is situated, and, through deliberation, it can act in that environment to achieve goals based on its convictions and motivations. They can also interact with other agents in a way analogous to everyday social interactions, such as cooperating, coordinating, and negotiating [21,23].

The Belief-Desire-Intention (BDI) [6] is a prominent model used in many multi-agent approaches [1,3]. In this model, *beliefs* represent the agent's knowledge and perception of the environment, self-conclusions about facts, and the obtained over communication; *desires* are the goals or objectives that the agent wants to achieve; *intentions* are the desires it is self-compromised to achieve. It proposes an approach to represent the human mind in intelligent agents, allowing them to make rational decisions and act according to their desires, goals, perceptions of the environment, or knowledge obtained over interaction with other agents [1]. It has been used to provide autonomy and intelligence in distributed and embedded systems [2]. By adding intelligence using a MAS, in addition to the controlling, detecting, and monitoring activities performed in a real-world scenario, the device gains abilities to make autonomous and proactive decisions and communication skills with other agent-based devices [5].

When BDI-based MAS is embedded into high-end IoT devices, mechanisms to provide interaction between humans and agents are interesting since, in the modern AI world, devices could be part of people's everyday lives, or maybe they could perform teamwork tasks, for example. Several initiatives focused also on the interaction between agents and humans [4,8,11]. However, there is a lack of tools and approaches concerning human and embedded MAS interaction. In the literature, human-agent interaction has been studied in many domains, such as conversational agents and voice assistants in robotics [20], over a decentralized messaging network [18], or using situation awareness approaches [17]. But, none of these are applied to embedded MAS. Besides, some of these do not consider the BDI model as a cognitive model, or those that consider it doesn't use any agent communication language, making complete interaction impossible.

Email sending can be a communication option between humans and agents existing in distributed MAS and embedded in devices. Despite not being a new human behavior, when adopting BDI, the email must be BDI compliant to be understandable to agents. Once the well-known and widely-used BDI frameworks Jason and JaCaMo use Knowledge Query and Manipulation Language (KQML) [10] illocutionary forces to represent the speaker's intention over agent communications, developing a communication architecture for agents over email protocols, considering KQML forces, can be an alternative to IoT middlewares and permissioned networks, making it easy for humans to send and obtain information from agents embedded in devices.

This work proposes a communication architecture to allow agents to communicate based on email exchange. The proposed architecture allows agents from an embedded MAS of a device to communicate with other agents from another embedded MAS on a different device. It also allows humans to interact with agents by sending and receiving emails. For this, we created a customized Jason architecture to allow this new specific type of agent to send emails. We chose the Jason framework [3] since it implements the BDI model, allows customization, and offers several solutions for embedded and distributed solutions. A proof of concept shows the feasibility of our approach. Furthermore, this work provides an alternative communication option to the embedded MAS that can be used with other solutions.

This work is structured as follows: in Section 2, related works are discussed; in Section 3, the methodology and implementation of the architecture are detailed; in Section 4, a proof of concept is presented, demonstrating the communication between humans and embedded agents, with the proposed architecture; finally, the conclusion is presented in Section 5.

2 Related Work

The ContextNet middleware [9] has been used to exchange information between MAS embedded into high-end IoT devices. Bio-inspired protocols can preserve the integrity and knowledge of the MAS in case of hardware failure by transferring all agents to another known MAS, using the middleware to negotiate the terms for moving agents from one system to another [22]. In this case, both devices must be connected simultaneously to communicate. Another example is a solution that explores communication between Embedded MAS and SUMO simulator, where several IoT gateways, including the ContextNet, face a high dependency on proprietary software and unsecured network infrastructure [7]. However, this middleware uses the User Datagram Protocol (UDP) for data transfer, which does not guarantee the delivery and privacy of messages and does not provide human-agent interaction. Since embedded systems have hardware limitations, making it hard to dedicate many resources to guarantee communication integrity and cryptography, it can affect the reliability of information exchanged between agents on different devices or between agents and humans.

Alternatively to IoT middleware, embedded MAS can use radio frequency technology as a secondary communication option [12]. It is non-connection-oriented, enabling communication over a network of radio frequency devices managed by agents. Similar to our proposal, it tries to mitigate or substitute communication when no infrastructure is available. Communication only occurs when both devices are enhanced with radio frequency technology, and there is a range limitation for effectiveness.

Among the solutions for non-embedded MAS, the SPADE [18] middleware provides agent-agent and human-agent interaction. It uses the XMPP protocol, an open and decentralized federated protocol. It offers a programming scheme based on the Model-View-Controller (MVC) pattern that enables a human-agent interaction like a chatbot. However, as it is not a natively BDI framework (because it is necessary to add a BDI layer using an external plugin), it has limited support for KQML, making a complete interaction impossible. For example, a human cannot send new plans to an agent (*tellHow* or *unTellHow*) or ask about some of the agent's beliefs (*askOne* or *askAll*).

Finally, some works [17,15,16] address the situation awareness applied to BDI agents. It adds the ability for an agent to provide information about its internal state and its understanding of the surrounding environment. However, it allows limited intervention in the agents' dimension by presenting only an interface for goal manipulation. Compared with the related works, when using our proposed architecture, the infrastructure responsibility and security are transferred to the email provider, as the MAS designer can use any public email provider that fits their purpose; it offers another alternative to embedded MAS without relying on additional hardware components; it adheres to the KQML agent's language, making communication between humans and agents possible; finally, it allows using smartphones, desktops, or webmail applications for human-agent interaction, eliminating the need for dedicated software.

3 The Email Based Communication Architecture

In embedded MAS, there may be a need for communication between different devices due to the social nature of such systems. Communication between these systems can occur in many formats and use technologies (middleware, radio-frequency devices, Wi-Fi, Ethernet, etc.) depending on the designer's necessity and the system's domain. This paper considers the embedded MAS approach as a group of agents cooperating to manage and control a device and its resources (sensors and actuators). So, it is important to guarantee that the agents from a MAS can communicate with agents from other systems. Besides, in domains such as the Internet of Things (IoT), humans also can be part of the communication process. Therefore, in some communication scenarios, ensuring that a MAS receives the message from another source is important. Furthermore, this communication must also consider security, privacy, scalability, and practicality.

The Internet Mail Architecture (*IETF RFC 5598*) is based on protocols that allow interoperability between different systems and platforms, eliminating the need for humans to use specific communication software since, nowadays, a person with a smartphone automatically has an email account and can already communicate with a cognitive agent. Email services are public, widely established, and available globally; therefore, from the developer's point of view, it is more economical than implementing and maintaining a dedicated IoT network for agent-agent interaction in geographically distributed MAS, eliminating the need for the agent to participate in multiple permissioned networks. Furthermore, it allows end-to-end privacy for humans and agents, even if they are members of different networks. So, as it is asynchronous, public, secure, and decentralized, email communication is a candidate for an integrated alternative complementary to the existing MAS exchange message alternatives.

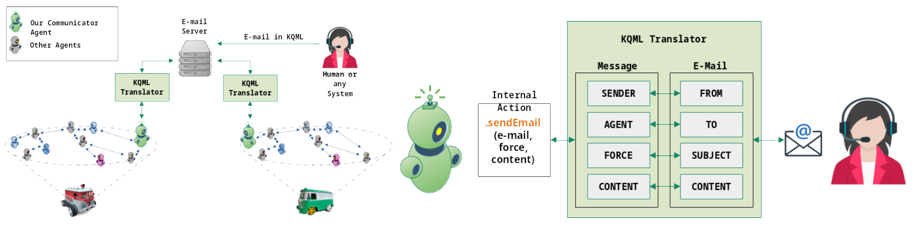
However, when considering BDI-agents, the agent-speaking language (e.g., KQML or FIPA-ACL) must be a variable in any communication mechanism. The communication is straightforward for agents speaking the same language, but a translation mechanism is necessary if they do not. That is what happens when using email; the agents need a way to understand and process the message's components and translate them into their message format. This work proposes an agent-customized architecture for communication between embedded MAS that can send messages in KQML using the email format to any other source (e.g., embedded agent, systems, and human) that access an email. Then, when the other side receives the message, it automatically translates back to KQML if it is an agent. If it is a system or a human, they must be aware of KQML or provide a translation mechanism for natural language, for example. When a human or system sends a message, it must adopt the KQML format. The translation from KQML to natural language is out of the scope of this work, and it is left as a future work.

We chose to specify the ability to send/receive emails to only one type of agent to define the task assignment better. Once the embedded multi-agent system runs in single-board computers (single core and only 512 MB RAM), it's important not to overload agents with multitasking purposes. For example, an agent can have difficulties controlling a car's hardware resources (motors) if it is dealing with email messages — as humans cannot drive while using their phones. Then, the mailer agent can receive or send internal messages from other agents and send messages to a target MAS or human using email. Figure 1a shows our methodology.

By the nature of an email, the transmission of messages is asynchronous. When the MAS has access to the internet, the Mailer agent can send messages to the email provider (server). If the recipient is a mailer agent, it can access the messages by automatically checking its email inbox in every reasoning cycle after checking for other internal agents' messages (from the same MAS). It checks all possibilities, including the spam box. Once the agent reads the messages, they are marked as read on the server to avoid double reading in the next reasoning cycle. If the agent is offline or any connection problem arises, the email server stores undelivered messages and allows the Mailer agent to access them in the future when it becomes online.

The agent's message in KQML must be mapped to an email format. A Mailer agent uses an action by identifying the receiver's email, the KQML illocutionary force (e.g., *tell, untell, achieve, tellHow*, etc.), and the message's propositional content besides its email address. The agent maps the email addresses to identify the message's source and target (*from* and *to*), illocutionary force as the message's subject, and propositional content as the message's content (*body* of the message). When the agent reads the message, the opposite mapping occurs. As one can see, the agents exchange messages in KQML format. However, when humans interact with agents, they must write the message in the same language, considering the above mapping. A translation mechanism to natural language could be applied, but is out of the scope of this paper. If the message is out of format, the agent will discard the message. Figure 1b shows the mapping from a KQML message to an email.

The architecture allows humans to ask for information, send plans, or request actions to cognitive devices without needing cloud layers to store or redirect information, reducing the complexity of engineering such systems. The devices are distributed, and there is no central server for agents (each MAS can run embedded into a device) in an edge computing style. The communication occurs



(a) The methodology for embedded MAS communication using email.

(b) Mapping of a KQML message to an email and vice-versa.

Fig. 1: The Email Based Communication Architecture.

directly between the agent and the interested person, as in a common email communication between two individuals. Considering a scenario where a human leaves their home and doesn't remember whether the room light is on or off, it is possible to request the status by emailing the device that controls the lights. The answer will return the status of the agent's belief about the room's light. So, the person can reply with an achievement goal to turn off the lights. Besides, it allows humans to teach new behaviors to the embedded agents by sending emails with new plans.

3.1 The Customized Architecture Implementation

We adopt JaCaMo [1] as the target BDI platform for our architecture. The JaCaMo has Jason [3] as one of its components for creating BDI agents. Along with Argo packages [19] and fault tolerance mechanisms [14], it is possible to engineer embedded and distributed solutions. Jason implements the BDI cycle for each agent and permits the designer to customize agent architectures, modifying some behaviors of the agent cycle. As one of the first steps of an agent reasoning cycle is to check messages from fellow agents, and in the last steps, agents can act, Jason fits our purpose in offering a new architecture of agents that deals with email messages. Furthermore, actions can also be customized by creating internal actions. Then, agents send messages to email addresses using a new internal action named *.mailer.sendEmail*.

The reasoning cycle starts with the agent perceiving the environment where it is situated and checking the messages from other agents from its MAS. As of now, considering our approach, agents can exist in different MAS and communicate; these agents also need to verify if it has messages from outside sources. The same occurs at the end of the reasoning cycle: the agents send messages to other agents. In our case, we extended it to send messages as email. Figure 2 shows the modified reasoning cycle.

The email middleware was developed with the methods for sending and receiving emails. The properties address, protocol, connection port, authentication methods, recipient, subject, and content of the message must be defined at runtime. In fact, the agent instantiates these properties following the designer's coding (plans and beliefs defined at design time).

The built-in internal actions provided by the middleware are described below:

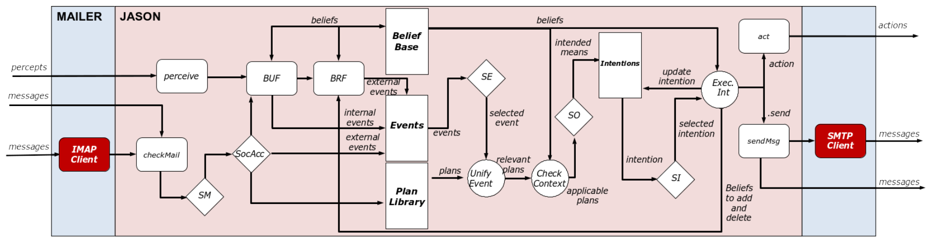


Fig. 2: The modified reasoning cycle of the Mailer agent.

- **.mailer.credentials(E,K)** - set email credentials;
- **.mailer.eMailService([I,p],[O,q])** - set email provider configurations;
- **.mailer.sendEMail(D,f,*M*)** - send an email.

Where:

D	is a string that represents the destination email (e.g., *human@example.com*);
E	is a string that represents the agent email account (e.g., *agent@example.com*);
f	is a literal that represents the KQML illocutionary force;
I	is the FQDN of the input email server (e.g., *imap.example.com*);
K	is a string that represents the email password or key;
M	is a term, literal, or string that represents the message;
O	is the FQDN of the output email server(e.g., *smtp.example.com*);
p	is a literal that represents the input protocol (e.g., *imaps* or *pop3*);
q	is a literal that represents the output protocol (e.g., *stmp* or *smtpOverTLS*);

4 Proof of Concept

Two human-agent interaction scenarios using email are proposed to present the architecture's feasibility. The first considers a smart home in which the owner can request information and actions in the environment. The second one considers an autonomous vehicle where a human can request a rescue.

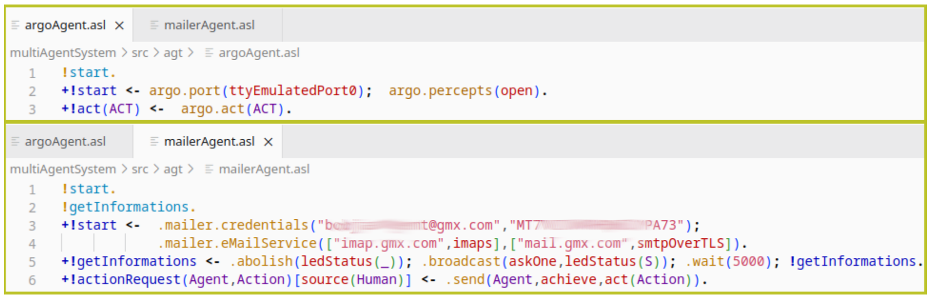
4.1 Smart Home

In this scenario, the human owner of the smart home will send an email asking about a certain perception of the environment. After receiving the response from the multi-agent system, the owner will request the execution of a certain action in the environment.

To fulfill the proposal of this scenario, a Raspberry Pi was used with an LED connected to one of its GPIO. A multi-agent system was developed with two agents responsible for communication (*mailerAgent*) and hardware management (*argoAgent*). Finally, a Python application was used to interface the reasoning and hardware layers, send percepts, and execute actions in the environment according to the determinations of the agent responsible for controlling the hardware.

In Figure 3a, the programming of the reasoning layer of the embedded multi-agent system is presented. The *argoAgent* has one initial objective (start): to execute the connection plan to the serial port (*ttyEmulatedPort0*), coupled with the interface application, and to activate the reception of perceptions from the environment. Furthermore, the agent has another plan to receive and execute requested actions in the environment (*!act*). The *mailerAgent*, in your turn, has two initial objectives: one to execute the email service configuration plan (start) through the new internal actions proposed in this work (*.mailer.credentials* and *.mailer.eMailService*) and the other to execute a plan that, every five seconds, updates a certain belief about the environment (*getInformations*). Perceptions are received from the controlling agent of the physical environment (.broadcast). Finally, the agent has another plan that allows forwarding requests for action in the environment directly to an internal agent (*!actionResquest*). The scenario was executed as follows:

1. first, the human starts running the multi-agent system. The *mailerAgent* connects to the email service, and the *argoAgent* connects to the GPIO management application;
2. the homeowner sends an email to the system with the subject *"askOne"* (KQML performative) and the content *"ledStatus(S)"*, representing that the human wants to know if the agent has the belief in your mind, as shown in Figure 3b;
3. the *mailerAgent* receives and interprets the email;

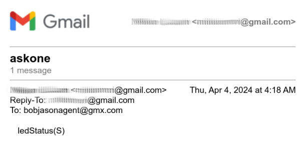

argoAgent.asl
mailerAgent.asl
multiAgentSystem > src > agt > argoAgent.asl

```
!start.
+!start <- argo.port(ttyEmulatedPort0);  argo.percepts(open).
+!act(ACT) <-  argo.act(ACT).
```

argoAgent.asl
mailerAgent.asl
multiAgentSystem > src > agt > mailerAgent.asl

```
!start.
!getInformations.
+!start <-  .mailer.credentials("b[redacted]t@gmx.com","MT7[redacted]PA73");
            .mailer.eMailService(["imap.gmx.com",imaps],["mail.gmx.com",smtpOverTLS]).
+!getInformations <- .abolish(ledStatus(_)); .broadcast(askOne,ledStatus(S)); .wait(5000); !getInformations.
+!actionRequest(Agent,Action)[source(Human)] <- .send(Agent,achieve,act(Action)).
```

(a) Reasoning layer of the Embedded MAS in the smart home.



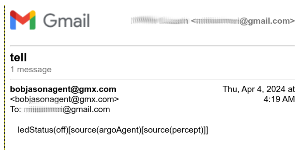
[redacted] <[redacted]@gmail.com>

askone
1 message

[redacted] <[redacted]@gmail.com> Thu, Apr 4, 2024 at 4:18 AM
Reply-To: [redacted]@gmail.com
To: bobjasonagent@gmx.com

ledStatus(S)

(b) The human is asking the embedded MAS about the *ledStatus* belief.



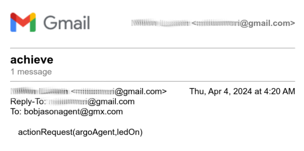
[redacted] <[redacted]i@gmail.com>

tell
1 message

bobjasonagent@gmx.com Thu, Apr 4, 2024 at 4:19 AM
<bobjasonagent@gmx.com>
To: [redacted]@gmail.com

ledStatus(off)[source(argoAgent)[source(percept)]]

(c) The embedded MAS replied to the human about the *ledStatus* belief.



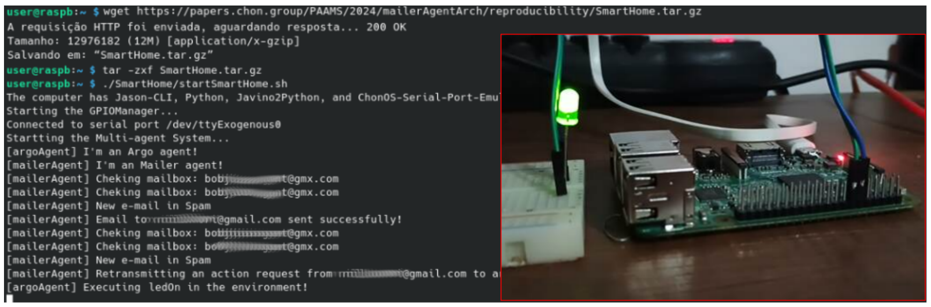
[redacted] <[redacted]i@gmail.com>

achieve
1 message

[redacted] <[redacted]@gmail.com> Thu, Apr 4, 2024 at 4:20 AM
Reply-To: [redacted]@gmail.com
To: bobjasonagent@gmx.com

actionRequest(argoAgent,ledOn)

(d) The human is requesting the execution of a plan for the embedded MAS.



```
user@raspb:~ $ wget https://papers.chon.group/PAAMS/2024/mailerAgentArch/reproducibility/SmartHome.tar.gz
A requisição HTTP foi enviada, aguardando resposta... 200 OK
Tamanho: 12976182 (12M) [application/x-gzip]
Salvando em: "SmartHome.tar.gz"
user@raspb:~ $ tar -zxf SmartHome.tar.gz
user@raspb:~ $ ./SmartHome/startSmartHome.sh
The computer has Jason-CLI, Python, Javino2Python, and ChonOS-Serial-Port-Emu
Starting the GPIOManager...
Connected to serial port /dev/ttyExogenous0
Startting the Multi-agent System...
[argoAgent] I'm an Argo agent!
[mailerAgent] I'm an Mailer agent!
[mailerAgent] Cheking mailbox: bobj[redacted]t@gmx.com
[mailerAgent] Cheking mailbox: bobj[redacted]t@gmx.com
[mailerAgent] New e-mail in Spam
[mailerAgent] Email to [redacted]@gmail.com sent successfully!
[mailerAgent] Cheking mailbox: bo[redacted]t@gmx.com
[mailerAgent] Cheking mailbox: bo[redacted]t@gmx.com
[mailerAgent] New e-mail in Spam
[mailerAgent] Retransmitting an action request from [redacted]@gmail.com to a
[argoAgent] Executing ledOn in the environment!
```

(e) The embedded MAS log.

Fig. 3: Execution of Smart Home scenario.

4. in reply to the request, the agent sends an email with the subject *"tell"* and the content *"ledStatus(off)[source(agentArgo)[source(percept)]]"*, representing that the belief that he has of the environment was received from agentArgo's perception, as shown in Figure 3c;
5. after that, the homeowner sends another email with the subject *"achieve"* and the content *"actionRequest(argoAgent,ledOn)"*, representing that the human wants the *mailerAgent* to execute the *!actionResquest* plan, forwarding *ledOn* to the *argoAgent*, as shown in Figure 3d;
6. the *mailerAgent* receives the email, interprets the message, and forwards the request to *argoAgent*;
7. finally, the *argoAgent* executes the action in the environment, as shown in Figure 3e.

4.2 Autonomous vehicle

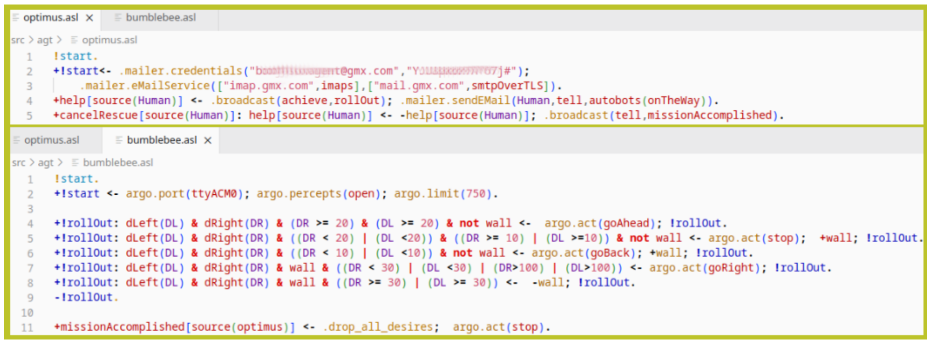
In this scenario, a human will send a belief requesting help to the multi-agent system, which will start executing a rescue plan and respond to the email, informing that help is on the way.

To meet the proposal in the second scenario, a *Generic 2WD ChonBot* [13] was built, and a system was developed with two agents. One is for communication (*optimus*), and the other is for driving the vehicle (*bumblebee*).

Figure 4a shows the reasoning layer code of the Embedded MAS. The *optimus* agent has an initial intention (*start*) to monitor an email box. Besides, it has two belief plans. The first (*help*) deals with the human help's request, sending a command to all agents in the system to execute the rescue mission (*rollout*). It also responds to humans, informing them that help is on the way, using the new internal action proposed in this work (*.mailer.sendEMail*). The last plan (*cancelRescue*) deals with canceling the rescue mission. In your turn, the *bumblebee* agent has an initial plan (*start*) to connect to the prototype's serial control port (*ttyACM0*) and activate the reception of perceptions from the environment. After this, a set of plans for executing the rescue mission (*rollOut*) is described (in this experiment, we chose the vehicle just to avoid obstacles). Finally, the agent has a plan to cancel the execution of the mission.

The scenario was executed as follows:

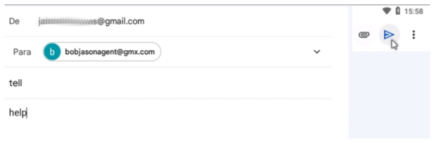
1. The multi-agent system was embedded and has been put in operation;
2. Figure 4b shows a human who emailed the embedded MAS with the subject *"tell"* (KQML performative) and content *"help"*;
3. By consulting the email box and receiving the belief, the *optimus* agent determines the execution of the mission to the other agents in the system;
4. The *bumblebee* agent started moving the vehicle, as shown in Figure 4d;
5. Finally, the optimum agent replies an email to the human, informing that the rescue execution is in progress, as shown in Figure 4c.



```
!start.
+!start<- .mailer.credentials("[redacted]@gmx.com","[redacted]7j#");
    .mailer.eMailService(["imap.gmx.com",imaps],["mail.gmx.com",smtpOverTLS]).
+help[source(Human)] <- .broadcast(achieve,rollOut); .mailer.sendEMail(Human,tell,autobots(onTheWay)).
+cancelRescue[source(Human)]: help[source(Human)] <- -help[source(Human)]; .broadcast(tell,missionAccomplished).
```

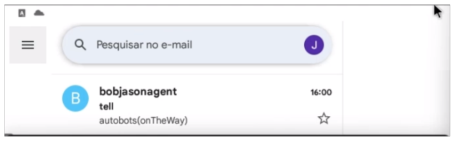
```
!start.
+!start <- argo.port(ttyACM0); argo.percepts(open); argo.limit(750).

+!rollOut: dLeft(DL) & dRight(DR) & (DR >= 20) & (DL >= 20) & not wall <-  argo.act(goAhead); !rollOut.
+!rollOut: dLeft(DL) & dRight(DR) & ((DR < 20) | (DL <20)) & ((DR >= 10) | (DL >=10)) & not wall <- argo.act(stop);  +wall; !rollOut.
+!rollOut: dLeft(DL) & dRight(DR) & ((DR < 10) | (DL <10)) & not wall <- argo.act(goBack); +wall; !rollOut.
+!rollOut: dLeft(DL) & dRight(DR) & wall & ((DR < 30) | (DL <30) | (DR>100) | (DL>100)) <- argo.act(goRight); !rollOut.
+!rollOut: dLeft(DL) & dRight(DR) & wall & ((DR >= 30) | (DL >= 30)) <-  -wall; !rollOut.
-!rollOut.

+missionAccomplished[source(optimus)] <- .drop_all_desires;  argo.act(stop).
```

(a) Reasoning layer of the Embedded MAS in the autonomous vehicle.



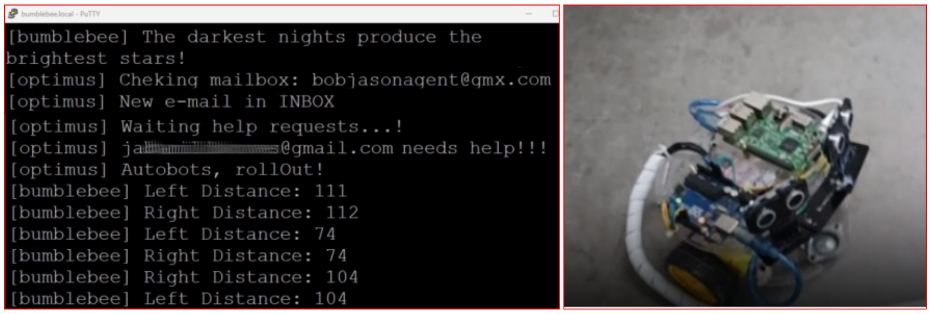
De [redacted]s@gmail.com
Para bobjasonagent@gmx.com
tell
help

(b) The human is sending a new belief to the embedded MAS.



bobjasonagent 16:00
tell
autobots(onTheWay)

(c) The embedded MAS replied to the human about the rescue.



```
[bumblebee] The darkest nights produce the
brightest stars!
[optimus] Cheking mailbox: bobjasonagent@gmx.com
[optimus] New e-mail in INBOX
[optimus] Waiting help requests...!
[optimus] ja[redacted]@gmail.com needs help!!!
[optimus] Autobots, rollOut!
[bumblebee] Left Distance: 111
[bumblebee] Right Distance: 112
[bumblebee] Left Distance: 74
[bumblebee] Right Distance: 74
[bumblebee] Right Distance: 104
[bumblebee] Left Distance: 104
```

(d) Embedded MAS log and execution.

Fig. 4: Execution of Autonomous vehicle scenario.

4.3 Reproducibility

The documentation, source code, video demonstration, and instructions for executing described scenarios can be found on the reproducibility[3] page. Besides, a web page is offered for online testing of human-agent interaction. Through it, it is possible to start a multi-agent system with a *Mailer* agent programmed to await interaction via email. Moreover, on this page, it is possible to see the system log and inspect the agent's mind. Finally, there are instructions for sending new plans to the agent and requesting some plan execution.

[3] `https://papers.chon.group/PAAMS/2024/mailerAgentArch/`

5 Conclusion

Multi-agent frameworks must provide standard and effective communicability, elasticity in communication, full and transparent integration of humans and agents, support for open systems, and the ability for agents to connect to the system independently of their running devices [18]. In this work, the forms of communication between embedded MAS were explored. The analysis revealed problems in tools, such as the lack of encryption and non-connection-oriented communication when using ContextNet [9]. To solve this, we proposed a communication architecture between humans and MAS, using the KQML [10] over SMTP and IMAP.

One of the main contributions of this work is to provide a simple and secure communication option, as email protocols do not face network blocks and also support encryption. Furthermore, this architecture offers an asynchronous communication option, improving communication availability between embedded MAS. In future work, we intend to improve the proposed architecture, enabling communication via email and IoT gateway in the same agent. Another relevant point is to explore, in the proposed architecture, the development of the *AskAll* and *AskHow* performatives.

References

1. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with JaCaMo. Science of Computer Programming **78**(6), 747–761 (2013). https://doi.org/10.1016/j.scico.2011.10.004
2. Bordini, R.H., El Fallah Seghrouchni, A., Hindriks, K., Logan, B., Ricci, A.: Agent programming in the cognitive era. Autonomous Agents and Multi-Agent Systems **34**(2) (2020). https://doi.org/10.1007/s10458-020-09453-y
3. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak using Jason. John Wiley & Sons Ltd (2007)
4. Bradshaw, J.M., Feltovich, P., Johnson, M.: Human-agent interaction. Handbook of human-machine interaction pp. 283–302 (2017)
5. Brandão, F.C., Lima, M.A.T., Pantoja, C.E., Zahn, J., Viterbo, J.: Engineering Approaches for Programming Agent-Based IoT Objects Using the Resource Management Architecture. Sensors **21**(23) (2021). https://doi.org/10.3390/s21238110
6. Bratman, M.E., Israel, D.J., Pollack, M.E.: Plans and resource-bounded practical reasoning. Computational Intelligence **4**(3), 349–355 (1988). https://doi.org/10.1111/j.1467-8640.1988.tb00284.x
7. Souza de Castro, L.F., Manoel, F.C.P.B., Souza de Jesus, V., Pantoja, C.E., Pinz Borges, A., Vaz Alves, G.: Integrating embedded multiagent systems with urban simulation tools and iot applications. Revista de Informática Teórica e Aplicada **29**(1), 81–90 (2022). https://doi.org/10.22456/2175-2745.110837
8. Dahiya, A., Aroyo, A.M., Dautenhahn, K., Smith, S.L.: A survey of multi-agent human–robot interaction systems. Robotics and Autonomous Systems **161**, 104335 (2023). https://doi.org/10.1016/j.robot.2022.104335
9. Endler, M., Baptista, G., Silva, L.D., Vasconcelos, R., Malcher, M., Pantoja, V., Pinheiro, V., Viterbo, J.: ContextNet: Context reasoning and sharing middleware

for large-scale pervasive collaboration and social networking. ACM, New York, NY, USA (2011). https://doi.org/10.1145/2088960.2088962

10. Finin, T., Fritzson, R., McKay, D., McEntire, R.: KQML as an agent communication language. In: CIKM '94. p. 456–463. ACM, New York, NY, USA (1994). https://doi.org/10.1145/191246.191322
11. Flathmann, C., McNeese, N., Canonico, L.B.: Using human-agent teams to purposefully design multi-agent systems. Proceedings of the HFES Annual Meeting **63**(1), 1425–1429 (2019). https://doi.org/10.1177/1071181319631238
12. Jesus, V.S.D., Lazarin, N.M., Pantoja, C.E., Manoel, F.C.P.B., Alves, G.V., Viterbo, J.: A middleware for providing communicability to Embedded MAS based on the lack of connectivity. Artificial Intelligence Review **56**(S3), 2971–3001 (2023). https://doi.org/10.1007/s10462-023-10596-z
13. Lazarin, N., Pantoja, C., Viterbo, J.: Towards a toolkit for teaching ai supported by robotic-agents: Proposal and first impressions. In: Proceedings of the XXXI WEI. pp. 20–29. SBC, Porto Alegre, RS, Brasil (2023). https://doi.org/10.5753/wei.2023.229753
14. Lazarin, N.M., Pantoja, C.E., Viterbo, J.: Dealing with the Unpredictability of Physical Resources in Real-World Multi-agent Systems. In: ICAART 2023, Revised Selected Papers. p. 48–71. Springer-Verlag, Berlin, Heidelberg (2024). https://doi.org/10.1007/978-3-031-55326-4$_3$
15. Noorunnisa, S., Jarvis, D., Jarvis, J., Rönnquist, R.: A conceptual model for human-agent teams. In: Agents and Multi-Agent Systems: Technologies and Applications 2021. pp. 17–26. Springer Singapore, Singapore (2021). https://doi.org/10.1007/978-981-16-2994-5_2
16. Noorunnisa, S., Jarvis, D., Jarvis, J., Watson, M.: Application of the GORITE BDI Framework to Human-Autonomy Teaming: A Case Study. Journal of Computing and Information Technology pp. 13–24 (May 2019). https://doi.org/10.20532/cit.2019.1004396
17. Noorunnisa, S., Jarvis, D., Jarvis, J., Watson, M.: Human-Agent Collaboration: A Goal-Based BDI Approach. In: Agents and Multi-Agent Systems: Technologies and Applications 2018, vol. 96, pp. 3–12. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-92031-3_1
18. Palanca, J., Terrasa, A., Julian, V., Carrascosa, C.: Spade 3: Supporting the new generation of multi-agent systems. IEEE Access **8**, 182537–182549 (2020). https://doi.org/10.1109/ACCESS.2020.3027357
19. Pantoja, C.E., Stabile, M.F., Lazarin, N.M., Sichman, J.S.: ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In: Engineering Multi-Agent Systems. pp. 136–155. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-50983-9_8
20. Seaborn, K., Miyake, N.P., Pennefather, P., Otake-Matsuura, M.: Voice in human–agent interaction: A survey. ACM Comput. Surv. **54**(4) (may 2021). https://doi.org/10.1145/3386867
21. Sichman, J.S., Demazeau, Y., Boissier, O.: When can knowledge-based systems be called agents? In: Proceedings of the IX Brazilian Symposium on Artificial Intelligence (SBIA). vol. 9, pp. 172–185. SBC, Rio de Janeiro (1992)
22. Souza de Jesus., V., Pantoja., C.E., Manoel., F., Alves., G.V., Viterbo., J., Bezerra., E.: Bio-inspired protocols for embodied multi-agent systems. In: Proceedings of the ICAART 2021. pp. 312–320. INSTICC, SciTePress (2021). https://doi.org/10.5220/0010257803120320
23. Wooldridge, M.J.: An Introduction to MultiAgent Systems. John Wiley & Sons, Chichester, U.K, 2nd edn. (2009)