Sapienza Università di Roma

Department of Computer, Control and Management Engineering Antonio Ruberti

EnviroMate: the social robot which raises awareness among students about recycling

Elective in Artificial Intelligence (HRI-RA)

Master program in Artificial Intelligence and Robotics

Professors:	**Students:**
Giuseppe De Giacomo	Ciarlo Alessia 1796690
Luca Iocchi	Garbetta Alessandro 1785139
Fabio Patrizi	Lavalle Leonardo 1838492

All authors have contributed to the project in an equal way.

Academic Year 2022/2023

Contents

1 Abstract

EnviroMate is a social and interactive robot that promotes environmental awareness and recycling by interacting with children and students in elementary and middle schools. Thanks to its various features, it helps children understand the materials that make up waste through an interactive game; it enables older students to read articles about current environmental conditions; using its cameras to recognize objects, it assists children in finding the path to the nearest available bin within their school. All of this is supported by robot animations and direct interaction through sight, voice, and touch. The reasoning aspect involves finding the nearest and not full trash bin to guide the child, as well as calculating the shortest path in an environment with different features (random closure of doors within the school) using PDDL and Q-learning with Reward Shaping.

2 Introduction

2.1 Environmental issues and importance of environmental education for tomorrow's adults

Environmental issues are defined as disruptions in the normal functioning of ecosystems and are considered globally serious when it is predicted that the ecosystem cannot recover to its original state or will reach a point of collapse. These problems can be caused by humans (human impact on the environment) or they can be of natural origin. Today, according to a report by *Earth.Org* updated in January 2023 [1], some of the major environmental problems of our time are caused by the immense impact of humanity on nature. In fact, at the top of the list of the *15 biggest environmental problems of 2023*, we can find: global warming from fossil fuels, food waste, plastic pollution (Figure 1) and deforestation. The *World Wildlife Fund* (WWF) [2] identifies several environmental threats that affect the Earth, such as deforestation, effects of climate change, illegal fishing (and over-fishing), waste pollution and unsustainable agriculture.

Although environmental problems may seem insurmountable, there are many actions that can be taken at personal, national, and global levels to contribute. Some solutions often involve sustainable practices, public awareness, education and technological innovations.

2.1.1 Environmental education in school

Environmental education is increasingly playing a crucial role in society, as it raises people's awareness about the numerous benefits that even small gestures can bring: actions that are simple to incorporate into daily life but are fundamental to aiding

environmental recovery. Some examples include: encouraging the reduction of our carbon footprint, saving primary resources like water, reusing waste, and practicing proper waste separation to enable effective resource recycling.
Since environmental problems, as seen in the studies mentioned earlier, are still increasing despite adults' awareness, the introduction of proper environmental education from elementary schools becomes essential, also directly incorporating it as a subject into the curriculum. This would help children to understand the natural world and the biodiversity, to respect the habitats of all living organisms and to recognize the long-term consequences of human actions on the environment. This connection with nature would promote responsible behavior towards the environment from a young age.

2.2 Social Robots

A social robot is a robot capable of interacting with humans and other robots. They can come in various forms (including the anthropomorphic one) and sizes, and are developed using artificial intelligence. They are often equipped with various sensors, cameras, microphones and other technologies so that they can respond to touch, sounds and visual cues much like a human would.
Despite their lack of consciousness, social robots are already being used in some parts of the world to provide companionship and emotional support to the elderly [3], to provide information in highly frequented places such as airport [4], and for children educational purposes through play [5]. They also collaborate in warehouses and factories for transporting goods and are used as research and development platforms, allowing researchers to study how humans interact with robots and to make further advances in the field.

Figure 1: Plastic pollution image from Earth.org.

2.3 EnviroMate

Guided by these considerations and driven by a strong interest in environmental issues, our idea for this project was to develop a robot called **EnviroMate**, aimed at raising awareness among children and young individuals about these topics.
Regarding **HRI** (Human-Robot Interaction) we use our software on the physical social and humanoid robot Pepper [6] which is of the *NAOqi* robots family [7].

EnviroMate

EnviroMate is placed within elementary and middle schools and is equipped with various features that allow optimal interaction with both age groups:

- Its primary goal is to engage students in conversations and activities related to environmental conservation and sustainability. Through an educational game for elementary school children (*Super Recycling Bros Game*) and interactive news for middle school students, EnviroMate will provide knowledge about concepts such as recycling, energy conservation, biodiversity, and the impacts of human actions on the planet.

- Pepper robot, with a friendly design and anthropomorphic features that make it recognizable and appealing to children, is equipped with a front touchscreen interface that displays the interactive visual content described above.

- Additionally, it has the ability to answer questions about the type of trash bin where the child should throw items into. By looking at the object through its front camera, it can provide information about its composition and it can find the shortest path to the nearest not full bin of the indicated type. This information will also be displayed on the touchscreen tablet.

- To enhance its engagement potential, the robot also incorporates visual and voice recognition technology. Using its sonars, it detects the presence of a student and it uses vocal interaction from the beginning of communication and throughout all the engagement time. Additionally, most of the actions done and the phrases spoken by *EnviroMate* are accompanied by gestures that make it friendly and as "human-like" as possible (Figure 2). Regarding movement speed, the gestures differ in the interaction with younger and older children, to avoid accidentally hurting younger ones through unwanted contacts.

RA (Reasoning Agent) part is provided by the EnviroMate's ability to keep track of the trash bin fill level in the school and, most importantly, by its object recognition

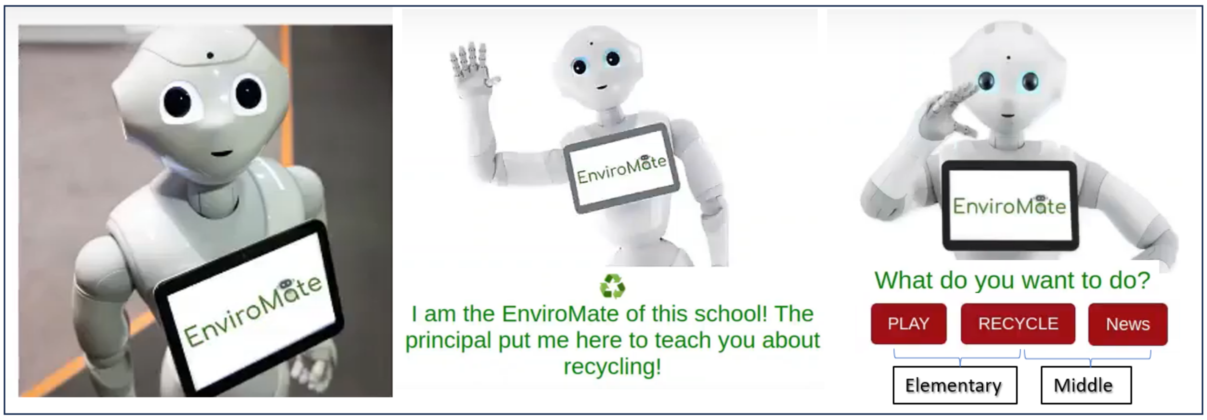

EnviroMate
EnviroMate
I am the EnviroMate of this school! The principal put me here to teach you about recycling!
EnviroMate
What do you want to do?
PLAY
RECYCLE
News
Elementary
Middle

Figure 2: The EnviroMate Pepper robot.

and calculation of the shortest path to reach the nearest bin. This is achieved using both planning with PDDL and Reinforcement Learning with Q-learning coupled with Reward Shaping [8]. The complex school environment was created using the Mazelab library [9] and was designed to ensure that every time there is a different scenario, thanks to the presence of 3 different doors that are randomly closed every time the EnviroMate has to give directions to a child.

3 Related works

The desire of creating autonomous robots capable of interact with humans is a key objective for both the research community and industry. That's why social robots are an emerging field of robotics that is rapidly gaining popularity. As a multidisciplinary field, its advancement will only appear by a wise and cautious combination of robotics, artificial intelligence and social sciences. Over the past decades, much progress have been made and the sci-fi dream of having robots as companions in our lives has never been so close (see Figure 3), but we still have a huge path ahead of us.

3.1 Human-Robot Interaction

Social robots, as already mentioned, are increasingly becoming more sophisticated and capable of adapting to individual human behavior. In many areas, cooperation between humans and robots is no longer a pretext or a "stage" on which to test new technologies, but has become a real requirement and the benefits that come with it, considerable. The most useful and immediate employment of a social robot, especially in everyday life, is as an assistant: the health care [10], [11] and elderly care where robots provide companionship and assistance in daily activities, or the educational field [12] are the ones where HRI is having the greatest success. Probably because requirements and capabilities of robot movements are not so demanding as in other application areas and

the social interaction aspect, even if complex and certainly improvable, is relatively easy to carry on in an acceptable way.
Our project focuses on a human interaction with a didactic aim: teach and sensitise students about environmental issues, which are today at the centre of public debate and beyond, and in particular about recycling. First and foremost we were guided by the importance and the beneficial impact that an initiative like this can actually have on the society and on the young generations, and for this reason we decided to pursue this direction of HRI application. We've been inspired specifically by two very recent papers which dealt with the topic ([12], [13]). The first one ([12], 2020), attributes the present pollution problem to the lack of empathy for learning any ecological and environmental skills. The authors recognized, although robotics in education is increasing, a lack of interest towards developing devices designed to teach children how to be environmentally conscious, and in particular, how to recycle. Their first prototype, *Smart Trash Junior* (see Figure 4), is a mechatronic trashcan that uses vision recognition to identify recyclable objects like our *EnviroMate* Pepper robot can do, and enters into a dialogue to educate elementary schools children how to recycle. Their robot, having been developed to be able to interact only with a specific type of users, lacks of the ability to behave differently accordingly to the context in which it acts and the target user with whom it is interacting. *EnviroMate* does it by having different interactions and socializes with two type of users: elementary and middle school kids. Actually, in recent works [14], they highlighted the importance of adapting robot behavior and appearance to match users' personalities and cultural backgrounds showing that the adaptation of a robot's behaviors to the human's personality profile makes interaction more engaging. In [14], making a step further, they achieved the goal by developing a Real-Time Robot Personality Adaptation based on Reinforcement Learning.

Figure 3: Robot waiter in a coffee shop in Tokyo, Japan.

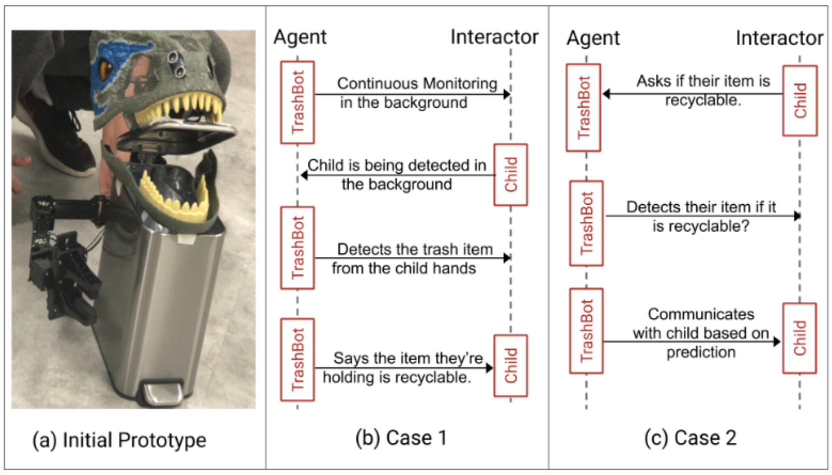

(a) Initial Prototype
Agent
Interactor
TrashBot
Continuous Monitoring in the background
Child
Child is being detected in the background
TrashBot
Detects the trash item from the child hands
TrashBot
Says the item they're holding is recyclable.
Child
(b) Case 1
Agent
Interactor
TrashBot
Asks if their item is recyclable.
Child
TrashBot
Detects their item if it is recyclable?
TrashBot
Communicates with child based on prediction
Child
(c) Case 2

Figure 4: Trash Can Junior project, initial prototype and interaction flow examples.

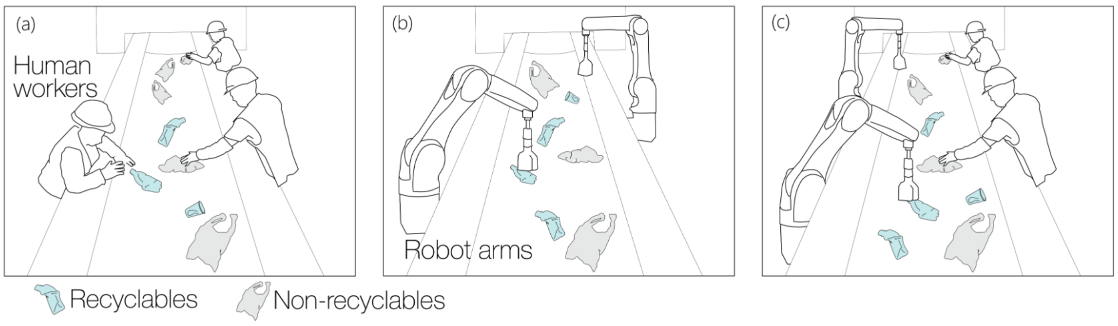


Figure 5: Illustration of Human-Robot collaboration in a recycling task.

In the other paper [13] (2022), is instead emphasised the power of human-robot collaboration. While in our project EnviroMate helps the students, in this paper is shown how much human involvement in a recycling industrial task (see Figure 5) has a remarkable impact on robots accuracy. Regardless of who helps whom, human-robot interaction is always profitable as long as it is done sensibly and for the right reasons. In fact, efforts in designing robots capable of physical collaboration with humans have been prominent. Research by Dautenhahn et al. ([15], 2005), explored collaborative scenarios involving robots and humans working together on shared tasks. *EnviroMate* is instead a students assistant trying to raise recycling awareness among them.

Our primary focus was to make *EnviroMate* as socially acceptable as possible, that is: following social norms, detect verbal and not verbal social signals and having a human-like behaviour while talking and moving body parts. Pepper itself was build for this purpose and therefore this features have been sufficiently accomplished with no so much effort. The way Pepper was developed derives from years of research: "*Design Sociable Robots*" [16] by Breazeal dates back to 2002 and introduced the concept of socially assistive robots capable of engaging in natural dialogues with humans. In [10] is analyzed how the humor, if used by the robot, can positively affects user's behaviour and perceptions. *EnviroMate* wasn't specifically designed to have sense of humour, but

a lot of attention has been paid to the words to be said with the aim of entertaining rather than boring the user.
Another important aspect we took into account from the beginning was establishing the starting of the interaction, the "handshake", if we can call it in this way, between the robot and the human. In literature there are many works which deal with the topic but from different perspectives. In [17] is highlighted the importance of detecting the eye contact with the user to acknowledge the start of the conversation. In [18], the robot is a bartender and the aim of the paper is to recognize the intention of the customers. Detecting whether a customer would like to order is essential for the service encounter to succeed. All signals (strong and weak) produced by the customers in front of the bar counter were deeply analyzed and the robot has to be able to distinguish between customers intending to order, chatting with friends or just passing by. In [19] paper, they propose an approach for social robots that attracts the attention of a target person depending on its current visual focus of attention. Based on the detected attention, the robot should decide if to interrupt or not the person activity and start an interaction. When starting a new interaction, the robot must be careful about person's "living space", a very important component in everyday social life which needs to be respected. In HRI we talk about *proxemics*, that means how far away should the robot be from the user when talking. And it is thanks to Pepper sensors (lasers and sonars) that in our project we respect proxemics rules and decide when the interaction starts.

3.2 Reasoning Agents

Knowledge Representation and Planning are the foundations of AI field and they have been studied since the very beginning. A reasoning agent is an agent, that could eventually be a robot, that has the capability to perform logical thinking, inference and decision-making based on available knowledge. This knowledge can be complete or, as it happens in real scenarios, not complete. The actions which the agent perform in the environment can be deterministic, that means we know which is the exact outcome, or not deterministic, where is the "environment" which decides the final outcome and therefore the agent has not power in determining it. We can interpret the last scenario as a "game arena" and since many real-world planning problems have a non-deterministic behavior owing to unpredictable environmental conditions, lots of effort has been put towards FOND (Fully Observable Not Deterministic) planning [20].

In our project, *EnviroMate* has the utility to indicate to the user which is the shortest path to a particular recycle bin. For this simple task, in a fully observable and deterministic domain in which basically a plan from a starting position to a goal position is computed, a classical PDDL planner (see Section 5.3.3) is more than enough. A relatively recent trend in the field, or better, a trend of which researchers are recently getting more and more excited about, is the combination of Knowledge Representation with Reinforcement Learning. As Micheal Littman said at the IJCAI Conference in 2015: "*Coming up with rewards in MDPs is too difficult! We need help from KR!*". We are realizing that Reinforcement Learning usually does not scale up well to large problems (real-world ones) where processes are not-markovian and engineering rewards becomes very complicated. The use of KR in support of RL can have many reasons, including the possibility to allow a more efficient learning, i.e. learning the optimal policy faster. It is exactly in this scenario that it is positioned [8][1]. Published in 2021, the paper uses reward shaping, that is a well-established method to incorporate domain knowledge in Reinforcement Learning by providing the learning agent with a supplementary reward. The authors proposed a novel methodology that automatically generates reward shaping functions from user-provided Linear Temporal Logic formulas. Another work which uses LTL_f to bias the RL agent's exploration is [21]. A similar approach is [22], where has been proposed a reward shaping method that generates the additional functions from STRIPS plans. We decided, indeed, to test the new methodology presented in [8] to compute the best policy for the already mentioned task of determine the path to the recycle bin (refer to Section 5.3.4 for more details).

4 Solution

4.1 Software architecture

The architecture proposed with our solution is shown in Figure 6. The core of the structure is the MODIM module (Multi-Modal Interaction Manager [23]), which manages the sending and receiving of each message, allowing the communication of all the components of the system, including:

1. *pepper_tools* that contains the *pepper_cmd*, with all the functions, classes and variables to initialize the robot, to control it, and manage tasks such as gestures, reading sensor logs and check the status of Pepper through the *Naoqi* module;

2. *Naoqi* is the core of the Robot, with this we simulate the presence of the robot;

3. *main_behaviour.py* that contains flow that describes the behaviour of the Pepper Robot, and where, with the *pepper_tools* (available through the MODIM client), we impose the functionalities and the responses of the robot;

4. *human_interaction_fake.py* simulates the human behaviour, position and inputs, and dialogues directly with the memory of the robot, writing and reading from it;

[1]This paper was presented by Lavalle Leonardo during the course.

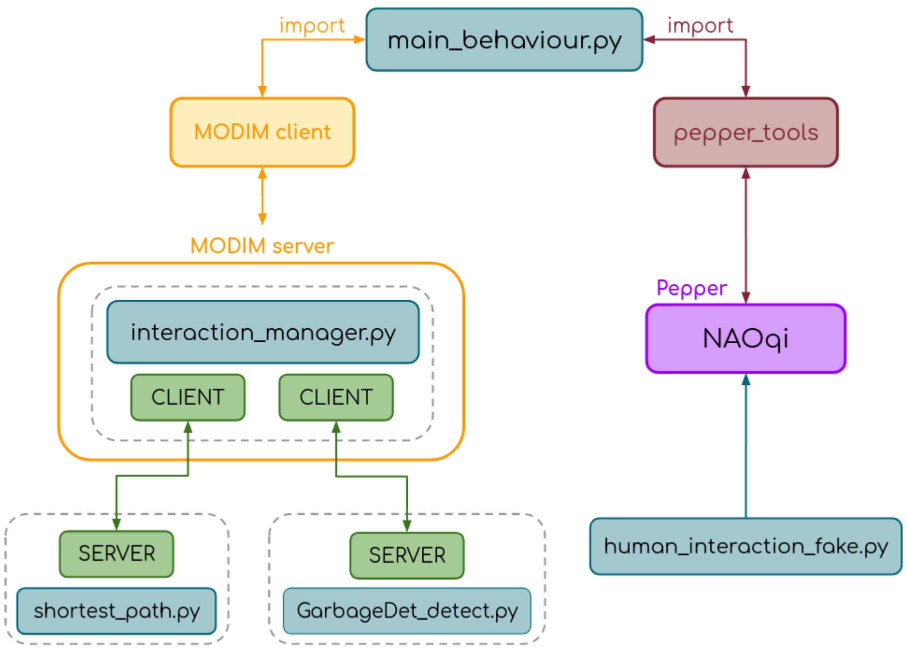


Figure 6: Software Architecture.

5. *shortest_path.py* and *GarbageDet_detect.py* are linked with the MODIM server and they are call through a request during the "Recycle" activity.

The MODIM software consists of three main components:

- *Interaction Manager* (IM), that coordinates the flow of information among the robotic modules and the human-robot interaction (GUI and speech) sub-systems;
- GUI sub-system, that is in charge of executing interaction actions (single actions exchanged by robot and user during their interplay) that involves the display of information such as texts and images. Furthermore, it captures the input from the user via the touch-screen;
- Speech sub-system which executes interaction actions involving the output of information via spoken sentences using a Text-To-Speech (TTS) component.

Given an action, the IM consults a actions database to determine the actual content of the interaction and redirects the execution to the GUI and/or the Speech sub-system. In addition, the IM receives from the GUI and Speech sub-systems inputs introduced by the user (e.g., through the touch-screen or via voice), which are transmitted to the robot.

4.2 Human-Robot Interaction

In Figure 7, we illustrate the flowchart regarding the Human-Pepper interaction. It is possible to divide the interaction into 3 phases:

- human-robot perception and presentation;
- main activity;
- greetings and feedback.

Initially, in the first phase, the robot is turned on and in idle state, waiting to perceive the presence of a human. When, through its sensors, it detects a presence

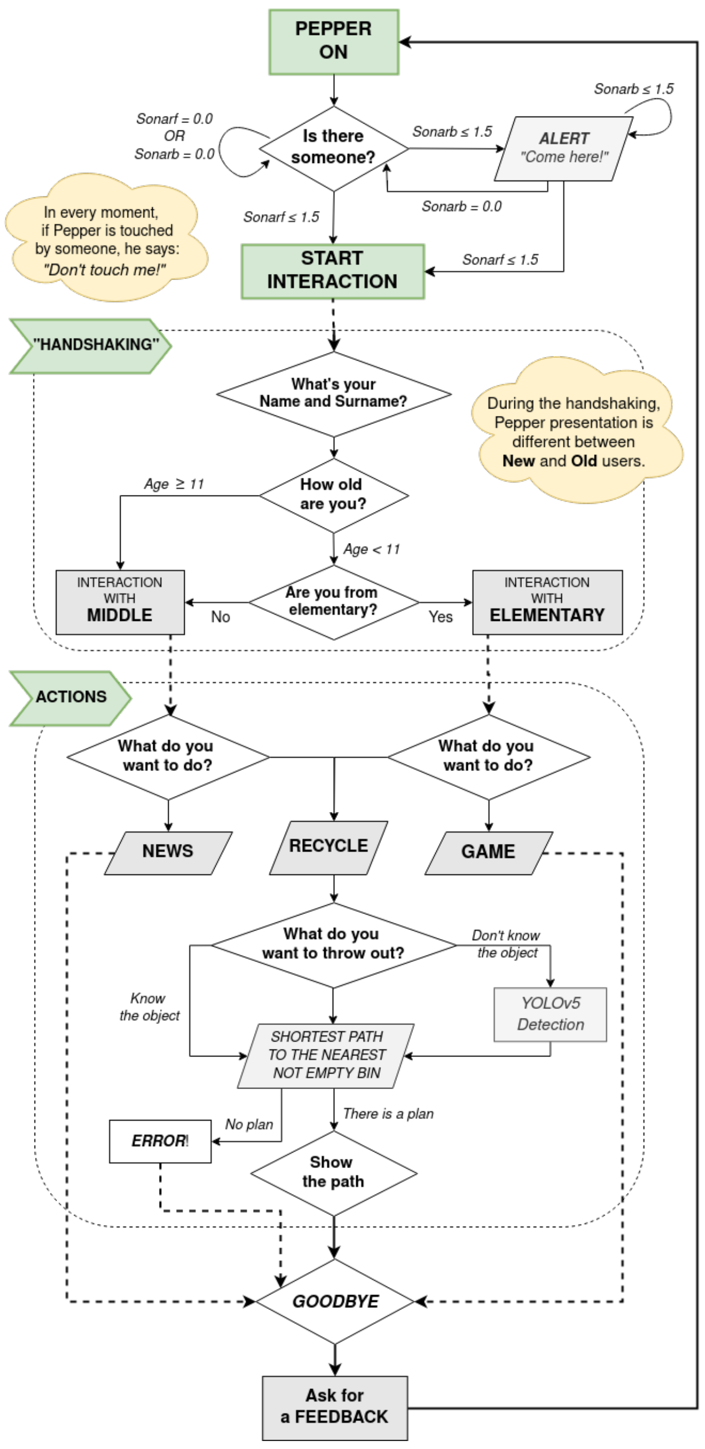


Figure 7: Flowchart of Human-Robot Interaction.

within a distance of less than 1.5 meters, the interaction begins. In case the user is behind it, it asks them to move in front of it in order to communicate. This marks the beginning of the interaction between robot and human. Pepper asks the human to provide his first name and last name, and once the response is obtained, it asks for his age. Based on the age (it has been decided to use 11 years as the cutoff age between the two educational levels), the robot decides which actions the users can perform. When a child younger than 11 years is detected, to be sure of its age, the robot asks for confirmation of the school is attending. In addition, after the presentation, through the use of a text file, the robot recognizes if the student who approached is a new user or an old user. In the latter case it remembers the level of education and it doesn't explain all its functions because the student is already aware of it.
In the second phase the flow is divided. Elementary school children can perform the "*Play*" and "*Recycle*" actions, while middle school guys can perform the "*News*" and "*Recycle*" actions. The human can communicate the choice of the action to be performed through the tablet on the chest of the humanoid robot by selecting the corresponding buttons.
When the child selects the "*Play*" activity, the game "Super Recycling Bros Game" starts. The image of a garbage is shown and the child through the tablet has to select the right category. He must give 3 correct answers to end the game; each time a wrong answer is selected, the question is repeated until the user gives the right answer.
When the middle student selects the activity "*News*" he can choose the links that will lead to web pages about ecological current issues. The student can decide whether to read one or more articles.
When the student selects Recycle, the robot asks him what he has to throw away. In this case the student can: answer specifying the object or, not knowing what to throw, ask for robot's help. In this second scenario, the object is shown to *EnviroMate* which executes garbage detection on it. Once the garbage category is identified, the robot calculates the shortest path to the first not full bucket available within the school (the specifics are in section 4.3).
Once the main action is finished, we are in the third and final phase: the robot sends a greeting message to the student, which changes according to the action chosen previously. After that it asks to the human to leave a feedback to evaluate his experience. User's evaluation is fundamental for HRI engineers and designer to improve their final product, and in general the overall performance of the interaction is very difficult to quantify. When the interaction between the child and the robot is finished the robot waits for another student.
In addition, the robot is able to handle unexpected events such as a student touching its head or hands, by telling him to stay away.

4.3 Reasoning Agents

When an elementary school child or a middle school student asks to recycle an object using the "*Recycle*" action, *EnviroMate* searches for this object in its list of known objects; otherwise, it attempts to recognize (see Section 5.3.1) the material the object is made of through the camera. Its goal at this point is to check if there are still non-full trash bins of the required type and then find the fastest path to show the student to reach one of the correct bins. To achieve this, a *grid world* was created using the Mazelab library [9] and Gym library [24]. This grid represents a school floor plan with 11 bins, 8 rooms, 9 doors, and a corridor as shown in Figure 8. To ensure

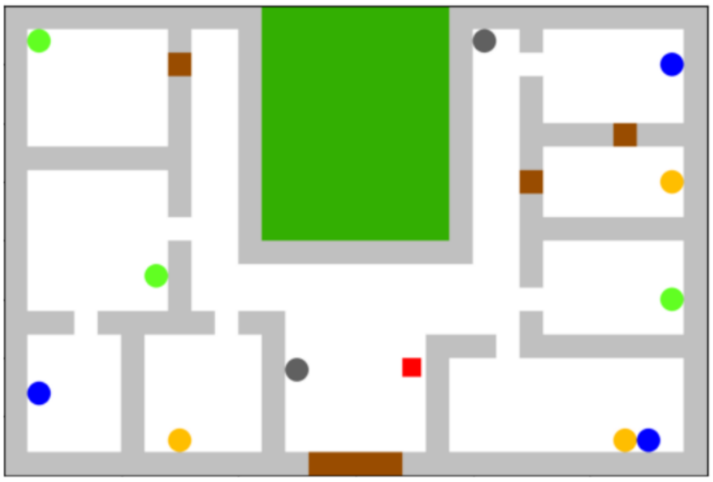


Figure 8: Map of the school.

that the environment is not always the same, every time a "*Recycle*" request is made to the *EnviroMate*, 3 out of the total doors are considered "closed". This leads to a different calculation of available bins and associated paths each time. The search for the shortest path is done using either PDDL planning or the Q-learning algorithm with the addition of the reward shaping described in [8]. Once the path is found, the *EnviroMate* will display on the tablet screen the school map with the fastest path highlighted in red to reach the first available bin relevant to the shown object. If the path is not found, because all doors to rooms containing that type of bin are closed or because all bins are full, the robot will apologize to the student and instruct them to try again later. In any case, the starting position of the child corresponds to the robot's position on the map, which is never altered.

5 Implementation

This section provides a detailed explanation of how each software component of the project was implemented: we will examine all the employed tools and libraries, and reasons behind each design choice.

5.1 Experimental Setup

All the code was developed in Linux machines, more specifically Ubuntu 22.04 LTS. We used the *pepperhri* docker image because it provides a reproducible environment and guarantees the encapsulation of all dependencies, libraries and configurations required to run our application.
Not having the physical robot at our disposal, we employed Choregraphe for robot simulation. It's a software platform developed by SoftBank Robotics which allowed us to control, test and refine Pepper robot behaviours in a virtual environment.
We mainly used Python programming language and a little of javascript and html for the web graphical interface. Even for the PDDL planning part, we used the *unified-planning* framework provided by [25] which allowed us to formalize our planning problem in python and to use many different planning engines to solve it. Moreover, as other external tools we exploited *yolov5* official repository [26] for finetuning a model for garbage detection; *mazelab* repository [9] for being able to reason over the map of the school and thanks to *pygame* [27] to visualize it; *gymnasium* library for the Reinforcement Learning part.
The two services depicted in Figure 6 (*shortest_path.py* and *GarbageDet_detect.py*), communicate with the main application through a socket python mechanism. More details about them respectively in Section 5.3 and 5.3.1.

5.2 Human-Robot Interaction

5.2.1 MODIM

Modim software is implemented in Python and the source code is available in the official repository at this link. This was the starting point to which we added other functionalities necessary for our work.
The main files of the Modim module are the *ws_server.py* and the *interaction_manager.py*. The *ws_server.py* file is responsible for creating and initializing all the structures necessary for the environment. Through the use of *tornado* library, web sockets are created to allow the communication between servers and clients; while *threading* library lets to run processes concurrently. Inside the *ws_server.py* file there is also the definition of the *DisplayWS* class that provides all the functions (called in *executeModality*) for managing the GUI such as: inserting/deleting images, texts and buttons. Additional functions have been defined for the management of buttons in the form of images and for buttons that redirect to external links, used respectively for the "*Feedback*" and "*News*" actions.
Regarding the *interaction manager*, it is instantiated in the *ws_server.py* file with the *init_interaction_manager()* function.

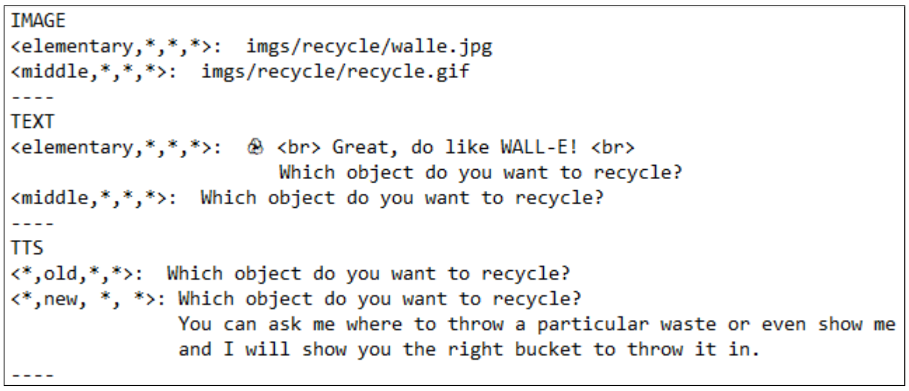
```
def init_interaction_manager(robot_port=9559):
    global robot_type, robot_initialized, im, display_ws, robot
    print "Start interaction manager"
    print "  -- display init --"
    display_ws = DisplayWS()
    display_ws.cancel_answer()
    display_ws.remove_buttons()
    print "  -- robot init --"
    robot = dummyrobot.DummyRobot()
    if (not robot_initialized):
        if (robot_type=='pepper'):
            # Connection to robot
            pepper_cmd.robot = pepper_cmd.PepperRobot()
            pepper_cmd.robot.connect(pport=robot_port)
            if (pepper_cmd.robot.isConnected):
                robot_initialized = True
                robot = pepper_cmd.robot
        elif (robot_type=='marrtino'):
            pass
    im = interaction_manager.InteractionManager(display_ws, robot)
```

In *interaction_manager.py* file, the init function, through the *init*, *index.html* and *qaws.js* files, loads the web page of the robot tablet and sets the current user profile. Also, within this file, we can find:

- the clients of the garbage detector and shortest path modules;
- the function super_recycling_game for the game execution in the task "*Play*";
- the *ask*, *execute* and *executeModality* functions to run the actions.

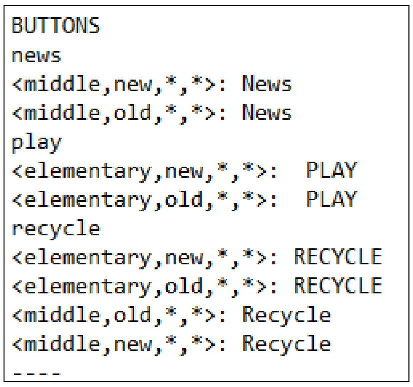
Every action that has to be performed is described in the form of text files. Each of these files, as shown in Figure 9 and Figure 10, contains actions modalities: "TEXT", "IMAGE", "TTS", "BUTTONS" and "ASR". With these keywords we manage the output (text and image on tablet and text to speech) and input (buttons on tablet and voice) of the robot.

Actions can be performed through the *execute* function, that also checks the conditions



```
IMAGE
<elementary,*,*,*>:  imgs/recycle/walle.jpg
<middle,*,*,*>:  imgs/recycle/recycle.gif
----
TEXT
<elementary,*,*,*>:  ♻ <br> Great, do like WALL-E! <br>
                        Which object do you want to recycle?
<middle,*,*,*>:  Which object do you want to recycle?
----
TTS
<*,old,*,*>:  Which object do you want to recycle?
<*,new, *, *>: Which object do you want to recycle?
               You can ask me where to throw a particular waste or even show me
               and I will show you the right bucket to throw it in.
----
```

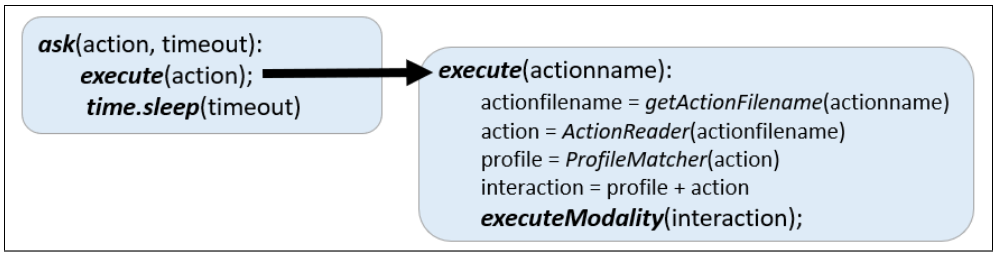
Figure 9: Example of action ("*Recycle*").



```
BUTTONS
news
<middle,new,*,*>: News
<middle,old,*,*>: News
play
<elementary,new,*,*>:  PLAY
<elementary,old,*,*>:  PLAY
recycle
<elementary,new,*,*>: RECYCLE
<elementary,old,*,*>: RECYCLE
<middle,old,*,*>: Recycle
<middle,new,*,*>: Recycle
----
```

Figure 10: Example of buttons modality in action "*Activity*".

profile present in the text file. The *profile* is defined as a tuple of 4 elements which represent: the student's school degree (to manage the branching of the flow), new and old users differentiation (for robot's message management), the language and the activity carried out (for the customization of goodbye messages). In Figure 9 and Figure 10, the use of profiles for outputs' personalization of texts, images and buttons can be seen.
Then, for all the allowed action modalities, the *executeModality* function is called as shown in Figure 11.
Actions, for which a user response is necessary, are performed with the *ask* function (Figure 11) that, after the calling of *execute* function, waits for an answer through buttons or voice with the use of *time.sleep()*.



```
ask(action, timeout):
    execute(action);
    time.sleep(timeout)

execute(actionname):
    actionfilename = getActionFilename(actionname)
    action = ActionReader(actionfilename)
    profile = ProfileMatcher(action)
    interaction = profile + action
    executeModality(interaction);
```

Figure 11: Pseudocode with the connections between *ask*, *execute* and *executeModality* functions.

At this point, with all functions described above, we have developed the *main_behaviour.py* file (in which all the HRI flow is performed). Once a client connection with the server is started, the interaction manager object (*im*) is instantiated.

5.2.2 Pepper Tools

In order to work with Pepper robot using most of its features, we also made use of the *pepper_tools* library which contains various tools for managing human-robot interactions. This repository is made up of various modules that allow us to use sensors, manage incoming and outgoing audio and perform some physical movements.

Once the connection with the robot in the *Choregraphe* simulated environment has been established, we exploit the functions (*startSensorMonitor()* and *sensorvalue()*) of pepper_tools to read the information provided by the various sensors used in our work: sonars, video cameras and touch.
Subsequently, access to the robot's memory is also used, via the *memory_service.getData()* function, to be able to temporarily retrieve information about the child's name and age before inserting them into the file in which the user's info are stored.
To ensure that communication between the *EnviroMate* and children is as natural as possible, a series of functions related to movements and gestures for the robot, to perform during interaction, have also been created. These functions have been incorporated into the *pepper_cmd.py* file, where most of the methods belonging to the *pepper_tools* library are also defined.
All the created gestures, their explanations, and the points at which they are used are explained below:

- *yes()*, used in the Super Recycling Bros Game;
- *no()*, used in the Super Recycling Bros Game;
- *notouch()*, used in the touching event;
- *hello()*, used at the beginning and at the end of the interaction;
- *thinking()*, used when the child needs to wait for the response from a server, either the server for waste object detection or the one for finding the shortest path to the correct trash bin;
- *lookhere()*, used every time the *EnviroMate* needs to point to the tablet screen (at the beginning of the game, when showing the result of object detection, or when displaying the map with the shortest path);
- *turnhead()*, used when someone is behind the robot;
- *talking()*, used in relatively long speeches to make the EnviroMate move as if it was a person that speaks;
- *talkingfast()*, first variant used in shorter speeches;
- *talkingfast1()*, second variant used in shorter speeches;
- *talkingfast2()*, third variant used in shorter speeches;
- *exultation()*, used at the end of the game;
- *sad()*, used when the feedback, at the end of the interaction, is negative;
- *normal()*, used when the feedback, at the end of the interaction, is medium;

- *happy()*, used when the feedback, at the end of the interaction, is positive.

All these functions take as input the type of school attended (elementary/middle) to ensure that the action, in the interaction with a younger child, is slowed down in movements to avoid any unintended contact that could potentially harm or frighten him. To achieve this, the *angleInterpolation()* method of the *ALMotion* service (from Naoqi library) is used because it allows setting the time over which the movement occurs through the "*time*" parameter.

5.3 Reasoning Agents

EnviroMate's reasoning part resides in the capability of computing the most conveninet path to the bin where the user has to throw a particular waste. The "most convenient" in our case stands for the nearest one. The input with which our agent starts to "reason" is the garbage type (*plastic*, *paper*, *general waste* or *organic waste*) and it can be inferred ether by the simple dialogue human-robot interaction (thanks to ASR robot's module) or, when EnviroMate doesn't recognize what the user is referring to, by a Deep Learning garbage detection model (Section 5.3.1).
Once we have the garbage type input, the agent shows the user the shortest path to the nearest not full bin of the particular type (see Figure 12). The path is calculated in two ways: (i) classical PDDL planning (Section 5.3.3); (ii) the LTL_f-based reward shaping method presented in [8] (see Section 5.3.4). Actually the path can also be computed by a Q-learning algorithm with ϵ-greedy strategy since we had to implemented it as a baseline for the just mentioned reward shaping method. Each time there's a new user who wants to know where the nearest bin is, the main application asks the *shortest_path.py* service to compute it (the method is randomly choosen every request) by giving the type of garbage as input.

```python
# 'im' is the interaction manager class of MODIM
map_goals, map_bestpath, is_path = im.shortest_path(garbage_class)
```

The socket server returns to the client a boolean value (*is_plan*) which tells if a path has been found or not, the image of the map with the bins of interest highlighted and the map with the computed path (if exists).

5.3.1 Garbage Detection

The garbage detection model is a finetuned version of YOLOv5 [26] architecture. We decided to add this feature to *EnviroMate*, since we considered it necessary in a recycling scenario (as they did in [12]).
The dataset we started from is called *trash-net* [28] and includes 2187 images split in train/test sets and divided in 7 class (*glass*, *paper*, *cardboard*, *plastic*, *metal*, *compost* and *trash*). We cleaned the dataset and reduced it both in size and in number of classes.

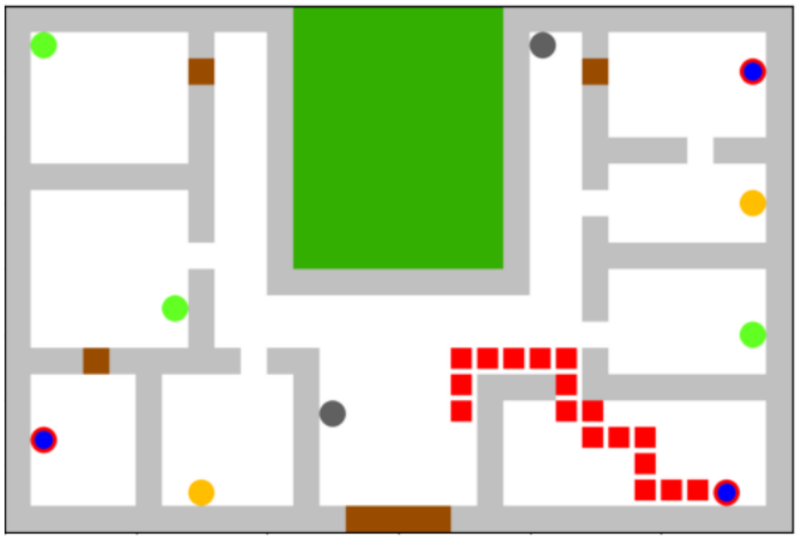


Figure 12: Shortest path (in red) to the nearest plastic (blue) bin.

The number of classes were reduced to 4: *glass* wasn't considered, the *cardboard* was encapsulated in the *paper* class and *metal* was embodied in *plastic*. We then filtered out manually all the images which were not very suitable to a school environment: for example we deleted all the images of beer cans because not allowed in such a place. We finally split the dataset producing: 1173 train set images, 50 test set images and 50 valid set images. The dataset is stored in *Roboflow* and available at this link.
We finetuned a YOLOv5 medium size model (*yolov5m.pt*) over our dataset for 20 epochs, with a batch size of 32 and using Mosaic Augmentation technique aiming to enhance its generalization power. Considering the small size of the dataset, we achieved excellent results (mAP-50 of **0.994** on the test set). It must be said anyway that the model didn't perform very well in detecting objects from other type of images. As we can see in Figure 13, the images derives from the same distribution and are relatively simple to detect because they have a white background and they are always the unique object present in the image (that's why we reached such an excellent result).

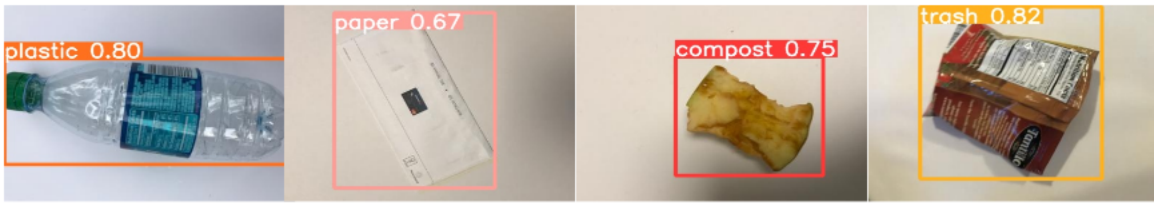


Figure 13: Sample of detected images.

By keeping in mind that this detection module would have been inserted in a social robot which is full of software components that have to be computed fastly, we executed some operations with the aim of reducing the inference time of the network. The model was already very fast (YOLOv5 architectures were designed to be computationally efficient), but we decided to adopt, at inference time, *FP16* half-precision and a *post-pruning* technique. After making some experiments, we achieved a good trade-off between inference time and performance of the model by post-pruning with a 0.3 scaling factor and by using half-precision (we reached a 10.2 ms inference speed). The confusion matrix of our best model is reported in Figure 14.

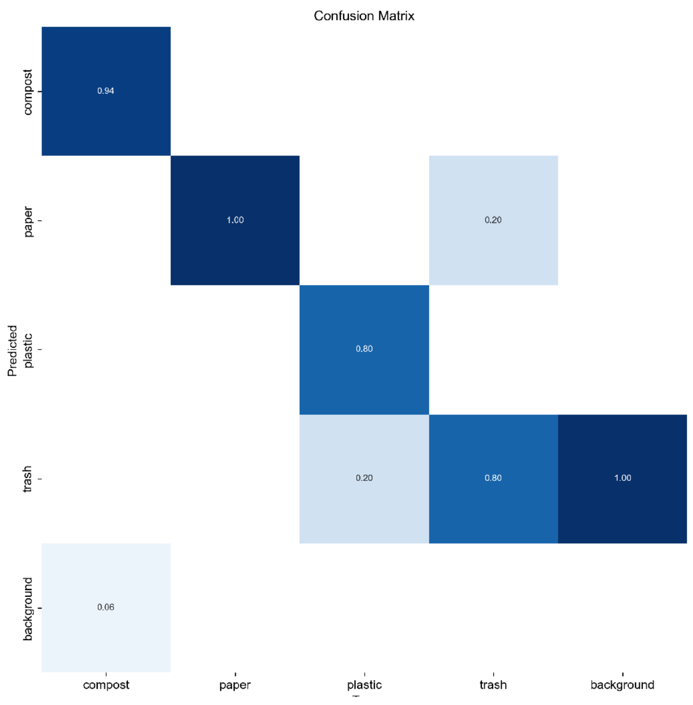


Figure 14: Confusion matrix of our best model.

Since we couldn't use an actual camera and connect it to the robot, to simulate the user who has to show to the robot the object to throw, each time a random test image is selected and the detection performed. As already anticipated, this detection process is provided by a server (*GarbageDet_detect.py*) which runs outside the docker container where the main application runs, and awaits new requests. It receives as input the path of the image to process and it returns to the client application the detected garbage type and the path of the input image with the bounding box drawn.

```
# detect_garbage is the client socket which asks for the detection
# service to GarbageDet_detect.py
garbage_class , ris_img_path = im.detect_garbage(img_path)
```

5.3.2 Grid World

To create the school environment for our work, we used the Mazelab library [9], which integrates functions from the Gymnasium library (known as Gym) [24], while the graphical map creation was done using the Pygame library [27].
In *MyEnv* class (*environment.py* file) the main functions, necessary for working within

the environment using RL algorithms, are defined including the *step() function* (essential for reward assignment), the *reset() function* (called at the beginning of each training episode), and various checks on the presence of the goal or the validity of a state-action pair. The *MyEnv* class is initialized by taking the environment structure in matrix form as input, allowing the creation of the observation space (using the *Box()* function) and the action space (using the *Discrete()* function).
The matrix representing the environment is of size 20x30 (Figure 8) and features, as previously described in the Solution section 4.3, 11 trash bins (2 for non-recyclable waste, 3 for organic waste, 3 for paper, and 3 for plastic), 8 rooms, 9 doors (3 closed randomly each time), and a hallway. *EnviroMate* robot is located at the entrance of the school, and from that point, all paths to the correct bins are calculated.
Finally, for the graphical representation of the environment that is shown on the Pepper robot's tablet, the *create_map()* (school_map.py file) function was created which uses functions from the *pygame* library. Thanks to this we can generate our surface every time and we can draw the environment on it by changing the color of the goals (bin of a certain type) and the color of the randomly closed doors. To allow understanding of the different types of trash bin present in the school, these were made in different colors similar to the real colours: yellow for paper, blue for plastic, green for organic waste and dark gray for non-recyclable waste.

5.3.3 PDDL Planning

When we faced with the task of calculating the shortest path, the natural option was to utilize *classical planning*. The goal of the reasoning agent is to find a path from an initial position to the nearest goal (we have multiple goals which represent the bins). The domain is fully observable and deterministic: the only thing that change each execution time is the doors configuration. In this way we tried to reflect the dynamism of a school environment where the doors are continuously closed and opened. The other feature is about the fullness of the bins: they are in fact constantly monitored by *EnviroMate* and if a bin is completely full is not considered as a goal where a student can recycle his waste.
The problem itself is simple because is essentially, with due differences and additions we did, a classical path planning from a starting point A to a final point B. This is the PDDL domain file we started from:

```
(define (domain school)
    (:requirements :strips)
    (:predicates (is_agent ?x) (is_bin ?x) (agent_at ?x ?y)
                 (bin_at ?x ?y) (adj ?x ?y) (throw_in_bin ?x))
    (:functions (level ?x))
    (:action move
        :parameters (?r ?from ?to)
        :effect (and (not(at ?r ?from)) (at ?r ?to))
        :precondition (and (is_agent ?r) (agent_at ?r ?from)
                      (adj ?from ?to) )
    )
    (:action recycle
        :parameters (?r ?b ?pos)
        :effect (and (throw_in_bin ?b))
        :precondition (and (is_agent ?r) (is_bin ?b)
                      (agent_at ?r ?pos) (bin_at ?b ?pos)
                      (< (level ?b) 3))
    )
)
```

As we can see the actions are only two: *move* which let the agent move in the grid world and *recycle* that can be performed only when the agent is in front of the bin and is not full. The bin is considered not full when the “level” is less than 3. “Level”, in the above code snippet referred as function, is a numeric fluent that, with respect to not numeric fluents (predicates), represents a variable that have a numerical value associated to it. We can observe in the PDDL problem file below, how the level is associated in the initialization to *bin1* (level=2), *bin2* (level=2) and *bin3* (level=1).

```
(define (problem school)
    (:domain school)
    (:objects robot bin1 bin2 bin3
            ;these are the cells of the grid world
            A1 ... I108
    )
    (:init (is_agent robot) (agent_at robot I90)
           (is_bin bin1) (is_bin bin2) (is_bin bin3)
           (=(level bin1) 2) (=(level bin2) 2) (=(level bin3) 1)
           (bin_at bin1 C9) (bin_at bin2 E39) (bin_at bin3 H12)

            ; adjacency fluents of evrey cells in the grid world
            ...

            ; doors open (3 random doors closed)
            (adj A12 I5) (adj I5 A12)
            (adj B18 I33) (adj I33 B18)
            (adj B33 C3) (adj C3 B33)
            ;(adj D4 I56) (adj I56 D4)
            (adj E3 I78) (adj I78 E3)
            (adj F13 I68) (adj I68 F13)
            ;(adj G7 I28) (adj I28 G7)
            (adj G4 H22) (adj H22 G4)
            ;(adj H7 I8) (adj I8 H7)
    )
    ; plastic bin goal example
    (:goal ( or
    (and (agent_at robot C9) (bin_at bin1 C9) (throw_in_bin bin1) )
    (and (agent_at robot E39) (bin_at bin2 E39) (throw_in_bin bin2) )
    (and (agent_at robot H12) (bin_at bin3 H12) (throw_in_bin bin3) )
    ))
)
```

The real tedious and time-consuming part of performing this type of planning in our map was first the instantiation of the grid world cells (325 in total) and then the instantiation of all the *adjacency* predicates (huge number) for letting the agent take legal moves. In Figure 15 we can see how we logically divided the rooms for a better and easier way of modelling PDDL problem. In the above PDDL code is possible to understand how a goal, actually a disjunction of goals ("*goal1 or goal2 or goal3*"), is written. In the particular example the goal is to reach a plastic bin, and a possible solution path (depends on which is the doors and bins fullness situation), can be the one in Figure 12. The capability of opening/closing the doors is easily reached through the deletion/addition of the adjacency fluents relative to the passage between rooms (reported in PDDL problem file).

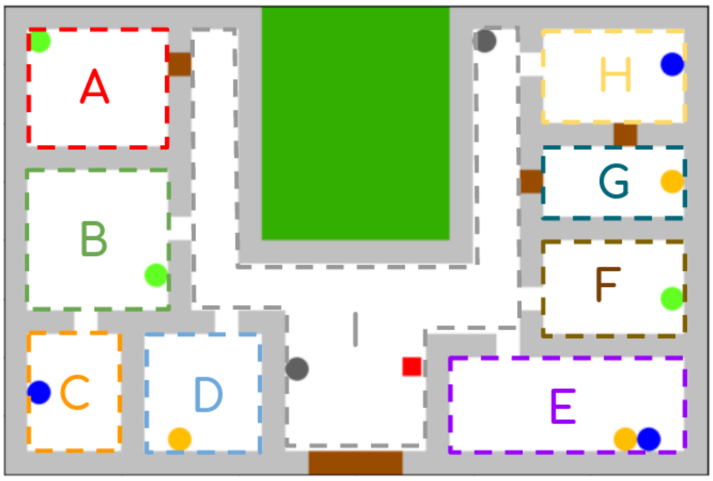


Figure 15: Division of the map in rooms for a easier PDDL modelling of the environment.

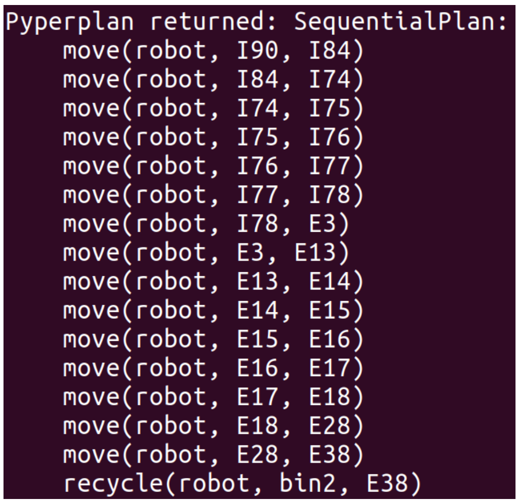
We couldn't use directly PDDL formulations in our software architecture. Fortunately it exists the *unified-planning* library, developed for the **AIPlan4EU** project [25], which "translates" the domain and the problem files in Python. There's a nice documentation where it explains how to use it, letting us to formulate our planning problem in a simple and intuitive way without worrying about which native solvers to use.

```
## SOLVE the PLAN
  ris_plan, bin_to_update = [], (0,0)
  with OneshotPlanner(problem_kind = problem.kind) as planner:
    result = planner.solve(problem)
    if result.status ==
            up.engines.PlanGenerationResultStatus.SOLVED_SATISFICING:
      plan_split=str(result.plan).split("move(robot, ")[1:]
      goal=(plan_split[-1].split(")")[0]).split(", ")[1]
```

```python
      for e in plan_split:
        ris_plan.append(e.split(",")[0])
      ris_plan.append(goal)
      # update the bin
      bin_to_update = states2coords_bins[goal]
    else:
      ris_plan = None

  return ris_plan, bin_to_update
```

As soon as the computation ends, we return the plan (if exists) and the position of the bin to update because we throw an object in it and we have to increase its fullness level by one. In Figure 16 there's an example of returned plan. It's worth mentioning that have been done a big effort, for visualizing the final path on the map like in Figure 12, in terms of mapping the PDDL object cells to the coordinates of the school grid world.



```
Pyperplan returned: SequentialPlan:
    move(robot, I90, I84)
    move(robot, I84, I74)
    move(robot, I74, I75)
    move(robot, I75, I76)
    move(robot, I76, I77)
    move(robot, I77, I78)
    move(robot, I78, E3)
    move(robot, E3, E13)
    move(robot, E13, E14)
    move(robot, E14, E15)
    move(robot, E15, E16)
    move(robot, E16, E17)
    move(robot, E17, E18)
    move(robot, E18, E28)
    move(robot, E28, E38)
    recycle(robot, bin2, E38)
```

Figure 16: One example of computed plan by the unified-planning engine.

5.3.4 Reasoning Agents and Reinforcement Learning

Reasoning Agents (RA) and Reinforcement Learning (RL) are two distinct approaches in the field of Artificial Intelligence, each characterized by unique features and methodologies. The combination of Reasoning Agents with Reinforcement Learning, leveraging the strengths of both approaches, can lead to more powerful and adaptable systems capable of addressing complex problems that require both logical reasoning and learning from interactions with the environment. This can result in solutions that are more flexible and effective even in dynamic environments.

Reinforcement Learning: Q-learning Q-learning is a fundamental algorithm in Reinforcement Learning (RL). It's a value-based approach used to learn optimal

policies for decision-making in sequential environments. In this project, the Q-learning algorithm was used as an alternative to PDDL planning, and it was later integrated with a new method of reward shaping (described in Section 5.3.4). The environment, for which the Q-table will be constructed, is the same as described in the Section 5.3.2, represented by states (school tiles in which agent can step on) and actions (orthogonal to the starting tile, defined as "*Von Neumann motion*"). Through Q-learning function, defined by us as described below, the Q-table is iteratively filled with Q-value estimates for each state-action pair. The Q-value of a state-action pair represents the expected cumulative reward the agent will receive if it starts in that state, takes the specified action, and then follows a certain policy to make decisions.
Our Q-function takes the following inputs:

- *environment* (states and actions, including initial state and current goal);
- *alpha* parameter (= 0.1);
- *gamma* parameter (= 0.9);
- initial and final *epsilon* values (from 0.1 to 0.0001);
- maximum number of *episodes* (= 500).

At this point, using the ϵ-greedy strategy to ensure that the execution first goes through the *exploration* phase (with random selection of the next action from a state) and then through the *exploitation* phase (with selection of the next action based on the one that maximizes the current reward), the table is updated by inserting a value in each state-action position with the formula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

Figure 17: Q-learning update function.

The reward value, present in the formula, is assigned as follows:

- *reward* $= -0.01$ if the current position is one of the walkable cells but not the goal;
- *reward* $= -1$ if the current position is not valid (either a wall or closed door);
- *reward* $= +1$ if the current position is the goal.

For each episode (500 learning episodes in total), the agent starts from the initial state and tries to reach the goal by gradually updating the Q-table as described above. At the end of this training process, the table will be used to calculate the optimal path to present to the student.

Reward Shaping As anticipated, we developed the rewarding shaping method explained in [8]. The paper is one of those that fits into scenario where Knowledge Representation methods help Reinforcement Learning algorithm.
Reward shaping method allows the RL agent to learn faster and the basic idea is to give small intermediate rewards to the algorithm that help it converge more quickly.

$$\boxed{Q(s,a) \leftarrow Q(s,a) + \alpha[r + \underbrace{F(s,s')}_{\text{additional reward}} + \gamma \max_{a'} Q(s',a') - Q(s,a)]}$$

Figure 18: This is reward shaping update rule in Q-learning algorithm. Notice the additional reward.

The additional reward is known as *domain knowledge*, i.e. notions the human engineers know in advance about the domain. Since reward shaping can, in general, not reach the optimal policy, we need some theoretical guarantees. These are given if the additional reward is in the form of Figure 19, where $\phi(s)$ is the potential of state s and ϕ is some heuristic measure of the value of each state.

$$F(s,s') = \gamma\Phi(s') - \Phi(s)$$

Figure 19: Potential-based Reward Shaping function.

In this case we talk about Potential-based Reward Shaping (PBRS) and we assure that optimal policy doesn't change.
In our case the domain knowledge is incorporated in the RL agent through LTL_f formulas which are then used to generate potential-based reward shaping. This type of temporal logic formulas, being quite similar to natural language, are a rich and compact way to express domain knowledge with minimum effort and indeed a easy manner to describe agent's goal. The LTL_f formula is then transformed to a DFA in order to track its satisfaction stage during the learning process and build the potential-based function. Each time, *EnviroMate* has to reach the nearest bin, so the LTL_f formula to be satisfied is $(\diamond at\_bin1) \vee (\diamond at\_bin2) \vee (\diamond at\_bin3)$ which stands for: "*Eventually reach the first bin or eventually reach the second one or eventually the third one*". The corresponding DFA (generated by LTLf 2 DFA sapienza tool) is shown in Figure 20.

Without going too much into the details, we now show how we built function $\phi_t(s_t)$ at time t and in the state s_t of the markovian decision process (MDP). For more details and better definitions please refer to [8].

$Distance(q_t)$ in our case is always 1, ω is a scale factor we set to 10 and n is a normalization function which represents the maximum initial manhattan distance to reach the nearest bin. Function h is reduced to the minimum distance that the agent,

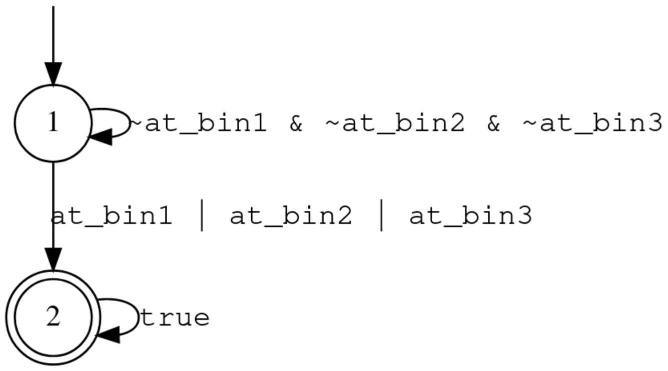


Figure 20: Corresponding DFA to the LTL_f formula.

$$\Phi_t(s_t) = -\omega \times \left[Distance(q_t) + \frac{1}{n} \times h(s_t, \phi^t_{guiding}) \right]$$

Figure 21: Definition of ϕ function for the construction of potential-based reward shaping.

at a certain instant t, has to one of the goals. See below how we implemented it in python (notice that *reward_shaping_gamma* is referred to the γ present in Figure 19):

```python
def compute_reward_shaping(agent, old_state, new_state, goal,
                           scale_factor=10, reward_shaping_gamma=1):
    # (PBRS function)
    # F = reward_shaping_gamma*phi_new - phi_old
    n = np.max([abs(g[0]-agent[0])+abs(g[1]-agent[1]) for g in goal])

    # first we compute phi_old (old state)
    min_manhattan_distance_old = np.min([abs(g[0]-old_state[0])+
                                         abs(g[1]-old_state[1])
                                         for g in goal])
    phi_old = -scale_factor * (1 + (min_manhattan_distance_old / n))

    # then we compute phi_new (new state)
    min_manhattan_distance_new = np.min([abs(g[0]-new_state[0])+
                                         for g in goal])
    phi_new = -scale_factor * (1 + (min_manhattan_distance_new / n))

    # if the state is the one of the goals --> we return 0!
    is_goal = False
    for g in goal:
        is_goal |= (g[0]==new_state[0] and g[1]==new_state[1])
    if is_goal:
        return 0

    return (reward_shaping_gamma*phi_new) - phi_old
```

The obtained results in terms of convergence speed to the optimal policy were not the ones really expected: also because the results reported in [8] were really promising. It's also true that our task is very simple compared to the ones tested in the paper and maybe in such a simple task the simple Q-learning algorithm is fast enough and is difficult to have significant improvements. In Figure 22 is quite clear how there aren't big differences in convergence speed but we are certain that both methods reached the optimal policy.

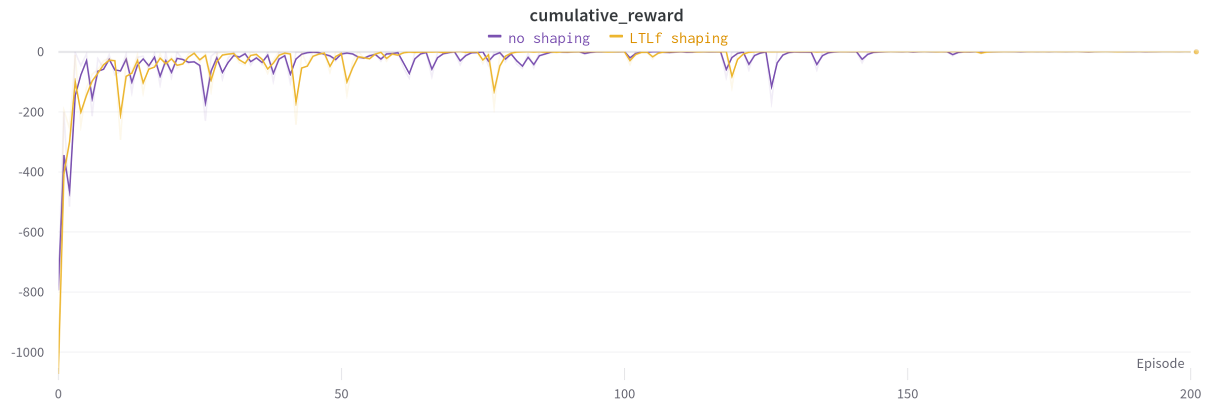


Figure 22: Plot of cumulative reward over learning episodes of Q-learning method with ϵ-greedy strategy. One with no shaping and the other with LTL_f-based reward shaping method.

We are driven to believe that this reward shaping method can express its full potential in real-world and more complex scenarios than in simpler scenarios such as ours. In situations where the algorithm without reward shaping starts to struggle to find the optimal policy.

6 Results

In this section we show the final results obtained, joining the various sections and the parts described above.
The interaction between the human and the robot can take place either through the web pages shown on the robot's tablet or through the voice (simulated with the terminal). Users answer by voice when, after a question, he will not find the buttons on the screen. The decision of what is possible to answer with buttons or voice was made so as to make the user experience and interaction between the robot and the human as natural as possible. A student uses the voice when he needs to answer simple questions in a presentation phase or, for example, to specify what he intends to throw. The buttons, however, are available when there are simple default options such as, during the game, during news reading and during feedback evaluation. To maintain the focus on personalized user experience, it's been decided to show the content of the buttons in capital letters for primary school children (from 5 to 10 years) in order

to make the writing more understandable and visible, and to make them select the buttons more easily; while for the middle school students lowercase letters are used (see Figure 26).

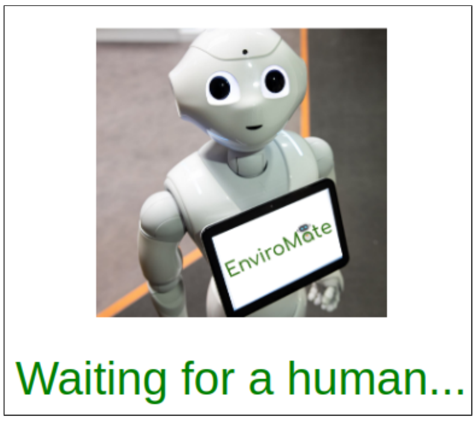


Figure 23: Tablet configuration during the *waiting* state of the robot.

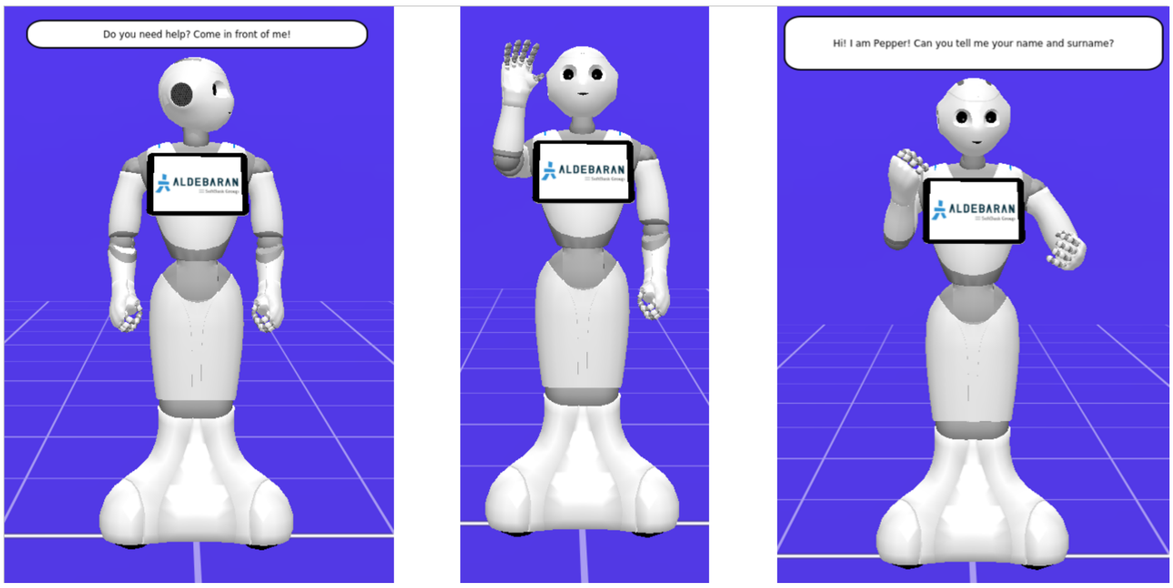

Do you need help? Come in front of me!
ALDEBARAN
Hi! I am Pepper! Can you tell me your name and surname?

Figure 24: Examples of Text-To-Speech (TTS) and gestures during "*handshaking*" phase.

Initially the robot waits for a user to approach, as shown in Figure 23. Through his sensors (sonars, front and rear), Pepper detects the presence of a human behind him or frontally. Interaction can take place in three ways:

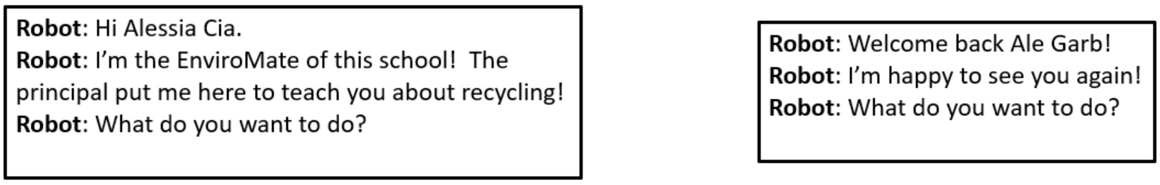
1. the user approaches the robot from the front, at a distance of less than 1.5 meters, stops in front of it for more than 5 seconds and the robot starts the interaction greeting him and asking for his personal details;

2. the user approaches the robot at the rear, at a distance of less than 1.5 meters, at which point the robot will interact with the human asking him to position

himself in front of him if he wants to interact. The question is asked every 5 seconds until the user is perceived behind;

3. the user stays behind the robot as in the second case, but, in the event that the human will move away, the robot will stop wondering who is behind him and will get back into a waiting state.

In Figure 24 are shown different gestures and vocal messages of Pepper during some stages of the "*handshaking*" phase.
Once the intention to start the interaction has been established, the robot will ask the human to present himself, explaining his name, surname and age. The robot will "understand" whether it is the first time it meets the user or not. If he has recognized the user he will greet him in a more "friendly" way, asking him what he wants to do; otherwise, Pepper will be more "formal" and will explain his role within the school. An example is present in Figure 25.

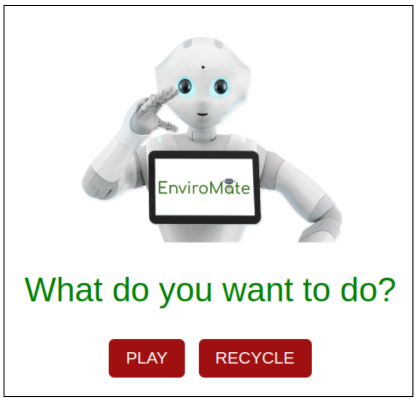

Robot: Hi Alessia Cia.
Robot: I'm the EnviroMate of this school! The principal put me here to teach you about recycling!
Robot: What do you want to do?

Robot: Welcome back Ale Garb!
Robot: I'm happy to see you again!
Robot: What do you want to do?

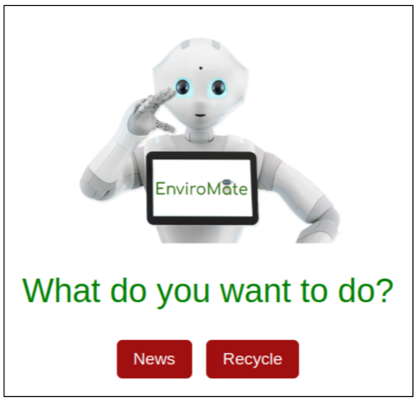
Figure 25: Different *welcome* messages from Pepper. In the left side the welcome for a new user, in the right those for a old user.

Depending on the age of the student who appears in front of the robot, the flow be different. In the case that the child user chooses to perform the "**Play**" activity, the home screen of the fantasy game "Super Recycling Bros Game" is opened on the tablet screen, which can be started only by pressing the "START" button. At that point images (one at a time) are shown to the student, to which he must associate the right garbage category choosing from the options: "PAPER", "WASTE", "ORGANIC WASTE" and "PLASTIC", in the form of buttons (see Figure 26c). In case the child misses an answer, the same image is submitted to him until he chooses the right option. The game stops when three images are guessed. Each response is accompanied by a response from *EnviroMate*, who, by animating himself, congratulates him on the exact answer, or invites him to answer again in the event of a wrong answer. During the choice of options the robot was not moved in order not to disturb the child who has to see the image and choose the answer.

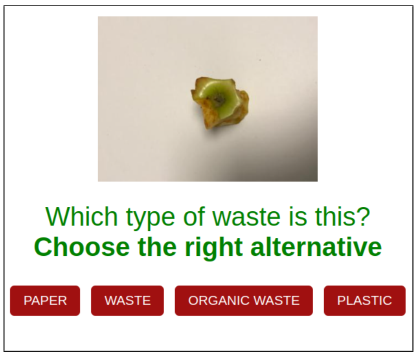
Among the activities that the middle school student can choose, after making presentations with the robot, there is "**News**". The boy through the tablet is able to choose to view some web articles about current issues related to the environment, as shown in Figure 26d. We focused on articles about: seas polluted by the presence of microplastics, pollution produced by the fashion industry and the collection and



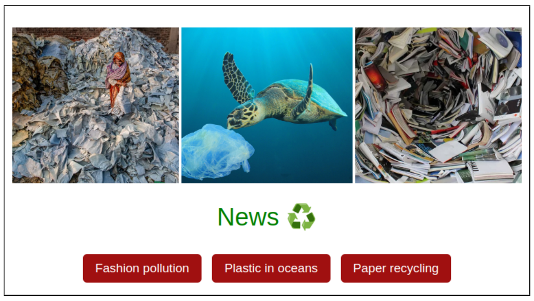
(a) Elementary student home webpage.



(b) Middle student home webpage.



(c) Super Recycling Bros Game options.



(d) "News" activity webpage articles.

Figure 26: Results of buttons personalization for middle and elementary school

recycling of paper. When the student selects an article the robot makes a small summary of the content present in the text. Once the speech is over, the robot asks the boy if he has finished reading the article or if he wants to continue browsing the page. The question is submitted cyclically every 10 seconds. As soon as it is finished, the robot requests if it wants to continue reading another article or if it wants to end its interaction.

In the event that the student chooses to perform the "**Recycle**" task, the simulated robot asks what they want to throw away. The user can respond by specifying the object to be thrown or make explicit the fact that he does not know what it is so as to ask Pepper for help to attribute the right composition of the material and be addressed to the right bin. In the first case the robot identifies the right recycling category thanks to the name said by the user, through the use of keywords. In the second one, the robot can assign the right garbage category through the use of an object detection model. Once the right category is assigned, the robot calculates the shortest path to the nearest free bin. A trash may be unreachable if it is full, or if the room in which it is present has the door closed. It first shows a map of the school containing information on the location of the robot, the bins and the doors. It also highlights the bins of the right category and after a short moment of "reasoning" shows on the tablet the path to follow to reach the basket. In case there are no empty bins, the robot isn't be able to calculate the route and, once he apologizes to the student, invites him to come back later (see Figure 27).

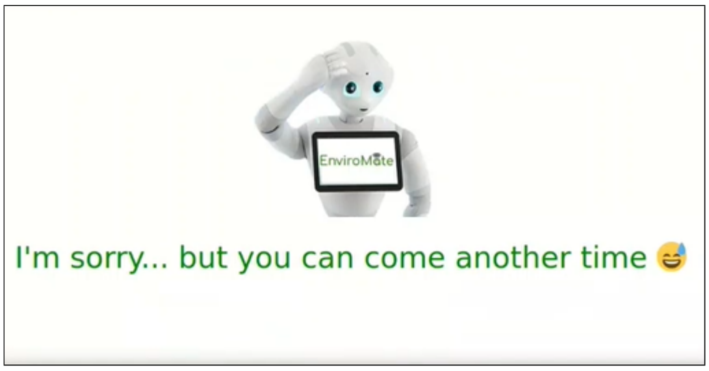


Figure 27: *EnviroMate* webpage when no plan is found.

When the main action is over, the robot greets the user, changing his greeting according to the activity they did together. Finally, the robot asks the user to leave feedback on the experience and based on the feedback received, the robot has a different reaction.
Throughout the execution of the interaction between the robot and the student the event of an *unexpected* touch of the robot's head or hands by the human was managed. The robot requests the user to not be touched and it performs the *noTouch()* gesture moving away from him. In this way we protect both the robot and the student, preventing possible damage on both sides. Furthermore, this attitude encourages respect for interpersonal spaces (*proxemics*).
To make the students' experience as safe as possible, it was decided to perform the gestures more slowly and to limit the movements when the robot interacted with an elementary school student.
In some cases, the interaction between the human and the robot suffers from some delay, these are caused by waiting for the user's response from the robot, which is put on hold to give the user time to give an answer.
The results are observable via the following link: `https://youtu.be/8hq5CO-dYBk`

7 Conclusions

The development of this project has allowed us to contemplate the potential uses of social robots in the field of environmental education, which will become increasingly necessary, starting with childhood and schools. It was very interesting to develop interaction through both verbal communication and a graphical tablet interface between the robot and a potential child, solving issues related to adapting language and movements for interaction even with very young children of elementary schools.

For the HRI part of the project, we worked diligently to make the communication natural and straightforward, ensuring the best possible experience. We aimed for the *EnviroMate* to be seen, through the eyes of a child or teenager, as a friend who is simply "smarter" and more informed about environmental issues.
Regarding RA, it was very interesting to structure the environment, the methods for working within it, the comparison between planning in PDDL and learning with RL methods, and, above all, the integration of the study on adding reward shaping to our Q-learning algorithm.
Since we have worked in a simulated environment with a virtual Pepper robot, in the future it would be interesting to test our solutions on the physical one. After completing all safety assessments for the robot and ensuring that all components are functioning correctly, it would be even more exciting and challenging to bring the robot to a school and directly testing it on the field.
*And, as EnviroMate would say: "**Remember to respect the environment!**"*

References

[1] Earth.org. 15 biggest environmental problems of 2023. 2023.

[2] WWF. Tackling threats that impact the earth.

[3] Lihui Pu, Wendy Moyle, Cindy Jones, and Michael Todorovic. The Effectiveness of Social Robots for Older Adults: A Systematic Review and Meta-Analysis of Randomized Controlled Studies.

[4] Rachid Alami Rudolph Triebel, Kai Arras. Spencer: A socially aware service robot for passenger guidance and help in busy airports. 2016.

[5] European School Education Platform. The impact of social robots on children's development: What science say.

[6] Aldebaran Robotics. Pepper tools. `http://doc.aldebaran.com/2-5/home_pepper.html`.

[7] Aldebaran Robotics. Naoqi tools. `http://doc.aldebaran.com/2-5/index_dev_guide.html`.

[8] Mahmoud Elbarbari, Kyriakos Efthymiadis, Bram Vanderborght, and Ann Nowe. Ltlf-based reward shaping for reinforcement learning. In *Proceedings of the Adaptive and Learning Agents Workshop 2021 (ALA2021) at AAMAS*, April 2021. Adaptive and Learning Agents Workshop 2021 : at AAMAS , ALA2021 ; Conference date: 03-05-2021 Through 04-05-2021.

[9] Xingdong Zuo. Mazelab: A customizable framework to create maze and gridworld environments. `https://github.com/zuoxingdong/mazelab`.

[10] Deborah Johanson, Ho Ahn, JongYoon Lim, Christopher Lee, Gabrielle Sebaratnam, Bruce Macdonald, and Elizabeth Broadbent. Use of humor by a healthcare robot positively affects user perceptions and behavior. volume 1, 11 2020.

[11] Iroju Olaronke, Oluwaseun Ojerinde, and Rhoda Ikono. State of the art: A study of human-robot interaction in healthcare. volume 3, pages 43–55, 05 2017.

[12] Marcus Arnett, Zhenyang Luo, Pradeep Kumar Paladugula, Irvin Steve Cardenas, and Jong-Hoon Kim. Robots teaching recycling: Towards improving environmental literacy of children. In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '20, page 615–616, New York, NY, USA, 2020. Association for Computing Machinery.

[13] Sruthi Ramadurai and Heejin Jeong. Effect of human involvement on work performance and fluency in human-robot collaboration for recycling. HRI '22, page 1007–1011. IEEE Press, 2022.

[14] Hannes Ritschel and Elisabeth André. Real-time robot personality adaptation based on reinforcement learning and social signals. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '17, page 265–266, New York, NY, USA, 2017. Association for Computing Machinery.

[15] Kerstin Dautenhahn, Sian Woods, Christina Kaouri, Michael Walters, Kheng Koay, and Iain Werry. What is a robot companion - friend, assistant or butler? pages 1192 – 1197, 09 2005.

[16] Cynthia Breazeal. *Designing Sociable Robots.* MIT Press, Cambridge, MA, USA, 2002.

[17] Kyveli Kompatsiari, Francesca Ciardo, Vadim Tikhanoff, Giorgio Metta, and Agnieszka Wykowska. It's in the eyes: The engaging role of eye contact in hri. volume 13, pages 1–11, 06 2021.

[18] Sebastian Loth, Kerstin Huth, and Jan De Ruiter. Automatic detection of service initiation signals used in bars. volume 4, 2013.

[19] Dipankar Das, Md. Golam Rashed, Yoshinori Kobayashi, and Yoshinori Kuno. Supporting human–robot interaction based on the level of visual focus of attention. volume 45, pages 664–675, 2015.

[20] Giuseppe De Giacomo and Sasha Rubin. Automata-theoretic foundations of fond planning for ltlf and ldlf goals. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4729–4735. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[21] Rodrigo Icarte, Toryn Klassen, Richard Valenzano, and Sheila Mcilraith. Advice-based exploration in model-based reinforcement learning. pages 72–83, 04 2018.

[22] Marek Grzes and Daniel Kudenko. Plan-based reward shaping for reinforcement learning. volume 31, pages 10–22, 10 2008.

[23] Modim: Multi-modal interaction manager. `https://bitbucket.org/mtlazaro/modim/src/master/doc/modim_userguide.pdf`.

[24] Gymnasium library. `https://pypi.org/project/gymnasium/`.

[25] Aiplan4eu. https://www.aiplan4eu-project.eu.

[26] Ultralytics. Yolov5. https://github.com/ultralytics/yolov5.

[27] Xingdong Zuo. Pygame library. https://www.pygame.org/news.

[28] Trash-net dataset. https://github.com/garythung/trashnet.