UNIVERSITÀ DI TRENTO

Student: **Francesco Laiti**
Course: **Autonomous Software Agents**

Academic year: **2021/2022**
Final assignment

1 Introduction

The house proposed is a Scandinavian house, a particular house style inspired by Nordic houses. It is simple, efficient, and environmental-friendly with its 3 solar panels. This type of house is a trendy structure for new houses, due to its minimalism.

The house is designed to be a *smart home*, focusing on the user experience, the automation of the devices inside the home, and helping residents for a better life. Not all the devices are supposed to be smart, but just the most important ones that provide a great experience for the residents.

This project aims to implement user-friendly devices, based on real products available on the market and functions already applicable.

The simulation developed in this project shows different interactions between the agents, how they control devices, their plannings, and intentions. More details will be provided in the next sections.

2 House description and blueprint

The house is a two-story house composed of 4 rooms on the ground floor and 4 rooms on the first floor, for a total of 8 rooms available.
The house is subdivided in:

- **Ground floor**:
  - the **entrance** gives access to all the rooms of the ground floor, except for the kitchen, and connects the ground floor and the first floor throw the stairs.
  - the **garage** gives access only to the external part of the house and the entrance.
  - the **living room** is accessible from the entrance only.
  - the **kitchen** is reachable only from the living room.
- **First floor**:
  - the **hallway** gives access to all the rooms on the first floor. It is connected to the ground floor throw the stairs.
  - the **bedroom** is reachable from the hallway and the bathroom.
  - the **bathroom** is reachable from the hallway and the bedroom.
  - the **baby room** is accessible from the hallway and the bedroom, but it is possible to close the door for a future where the two rooms can become independent (teenage children for instance).

The blueprint of the house is shown below (blue objects are windows and green objects are devices):

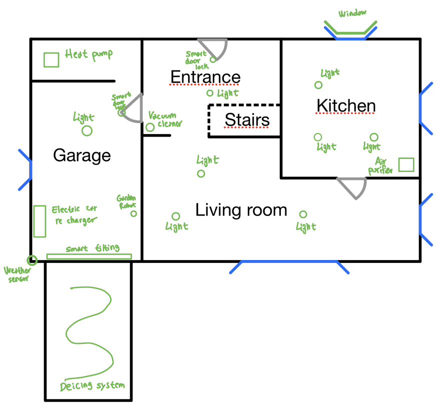


Figure 1: Blueprint of *Ground floor*

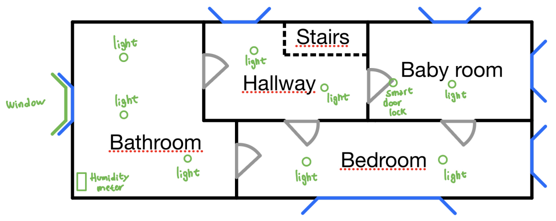
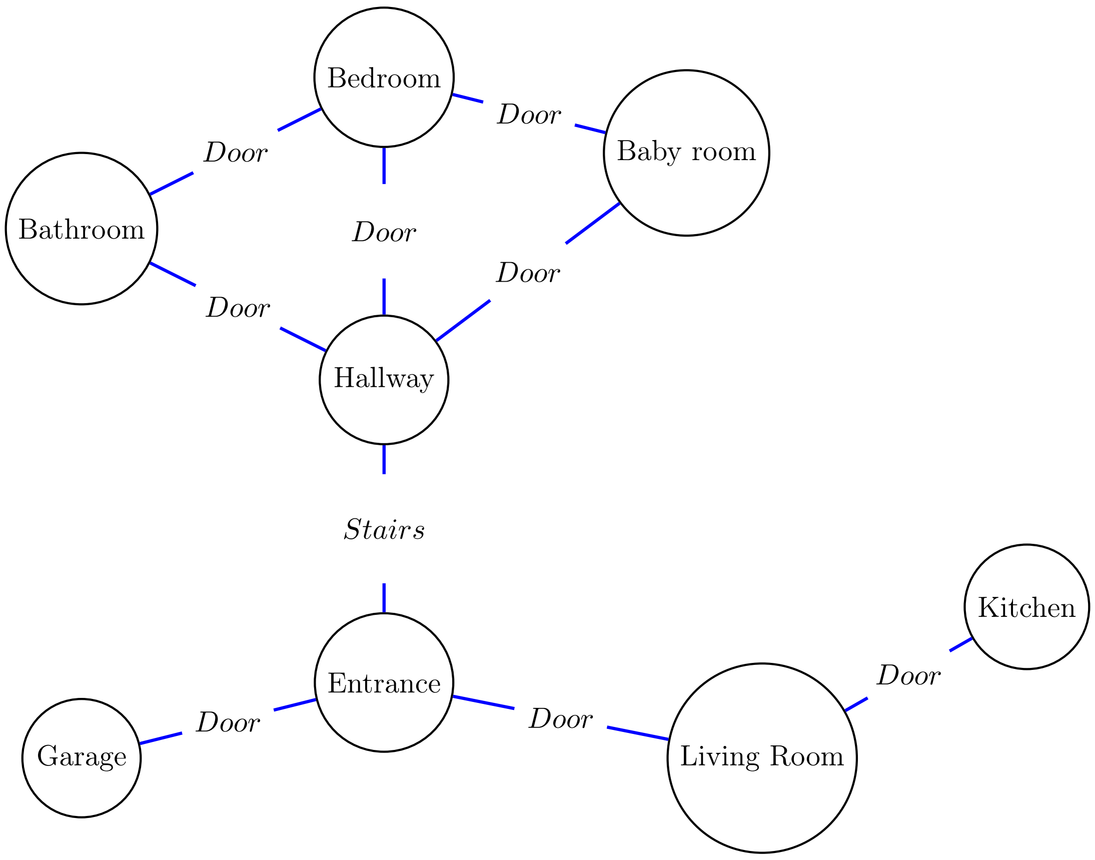


Figure 2: Blueprint of *First floor*

For a clear and intuitive connection of the rooms, I created a graph of the connection of the rooms:



I made 2 simulations to test out how the project works.
One simulation considers only a one-day scenario (from midnight to the midnight of the next day), the other simulation is only for showing how agents behave and interact with the world and devices.
Some assumptions are taken for the simulation. As an example, all the doors are supposed to be open during the day.

3 Rooms

These are the rules valid for every rooms:

- All the rooms have two overhead lights, unless otherwise stated;
- All windows are supposed to be manual, unless otherwise stated;
- Three solar panels are installed on the roof;
- The devices available in each room are described in the next section.

Entrance. This room is located on the ground floor, in the north part of the house. The openings include the main entrance door, one door for the garage, a door to give access to the living room, and the stairs to the upper floor. There are 2 devices available: one smart lock on the main door, and one light.

Garage. The garage is located on the ground floor, in the west part of the house. It is used to park the car and it is installed the heat pump. One window is available but it is not smart because the cars are supposed to be electric and so carbon-free. The openings include access to the street by car or by foot and one door connected to the entrance. There are 8 devices available: the heat pump, one light, one car charger, one smart tilting, one garden robot with its base, one deicing system outside, one door lock, and one weather sensor.

Living Room. This is the biggest room in the house. It is located in the south/south-east part on the ground floor. It is used for relaxing, watching TV, and letting residents spend time together. There is a table near the kitchen to eat or play board games. The openings include access to the entrance and a door to enter the kitchen. There are 3 lights available.

Kitchen. This room is used to prepare meals during the day. It is located in the northeast of the house, also on the ground floor. There is only one opening door that gives access to the living room. There are common household appliances, like an oven, fridge, microwaves, and more. Two windows give access to natural light and change the air. There are one smart window, one smart air purifier, and three lights.

Bathroom. This room is located on the first floor in the west/south-west part of the house. It is the only bathroom in the house. It is reachable from the hallway and the bedroom throws the doors. The bathroom has a shower, two washbasins, a toilet, and one hairdryer. There are 3 lights, one humidity meter near the shower, and one smart window.

Hallway. The hallway is located on the first floor and it connects the first floor with the ground floor. It gives access to all the rooms on the first floor (the bedroom, the bathroom, and the baby room). There are two lights and one vacuum cleaner with its base.

Bedroom. The room is located on the first floor and in the south/south-east part of the house. The bedroom has one queen bed with a closet and a desk. One big window illuminates the environment. There are two lights available in the room.

Baby room. This is the last room of the house, located on the first floor in the east part. This room hosts one bed and one desk. It is possible to close the door that gives to the parent's bedroom when the child becomes independent. The openings include access to the hallway and the bedroom. There are one light and one smart lock door on the hallway door.

4 Devices

Infrared camera. It can monitor the scene inside the room. It can use the infrared to detect movement during the night or inside dark areas. The statuses are 2: `move_detect` that is triggered when something is moving inside the room, `no_move_detect` when there is no movement inside the room. The actions of the device are 3: `switchOn()` to turn on the infrared on during the night, `switchOff()` to turn off the infrared, and `movementDetected()` to alert and take the appropriate actions for that event. When the infrared is on the consumption is 300 W, otherwise it is 100 W.

Smart door lock. It is used to lock and unlock the main door in the entrance and the door from the baby room to the hallway. It provides more security for the house and in particular for the baby, in order not to risk fall him down the stairs when his parents sleep. There are two states available: `lock` and `unlock`. This device can perform two actions: `lock()` to lock the door and `unlock()` to unlock the door. This is a low-power device, the consumption is considered nil.

Light. Lights provide illumination to the room. Each room has at least one light. There are 2 statuses: `on` and `off`. The 2 simple actions are `turnOn()` and `turnOff()`. Lights are considered all LEDs, so they consume 10 W each.

Electric car charger. It provides energy to charge only one car. It is preferable to use the energy from the solar panel. The charger can detect when the electric car is fully charged. It can supply a maximum of 7 kWh power but it is necessary to have energy from the solar panels, otherwise, it provides 3.5 kWh as exit power. There are 3 possible states: `plugged_max` let charge the car at maximum power for a fast charging, `plugged_min` let charge the car at lower power but it saves energy and leaves more energy to other devices, and `unplugged` let notify when the car is not plugged in. The actions performed by the device can be `rechargeMaxPower()` to boost the charging power, `rechargeMinPower()` to charge the car with 3.5 kW, and `stopRecharge()` to stop the consumption of energy when there is no car plugged in. The device consumes 7 kW at maximum power or 3.5 kW in low charging mode.

Smart tilting. It gives access to the street and it is necessary to close the garage. It can be controlled manually by a remote controller or by an agent. It can be `open` or `close`. There are 2 actions available: `openTilting()` to open the tilting and `closeTilting()` to close the tilting. The consumption is considered nil.

Deicing system. It provides a heating system to melt the snow and ice in cold seasons for an agile exit with the car. It can be `on` or `off`. It can be activated only if the temperature outside is less than 0° Celsius. The actions performed are `turnOn()` and `turnOff()`. It is a high-power device, consumption is considered 2 kW.

Heat pump. It provides hot water for the shower and other tasks for the house. It is controlled by the heat pump agent. It can increase the temperature of water up to 80° Celsius when the solar panel provides energy and the car charger is not plugged and all the windows are closed, otherwise it decreases the temperature to 50° Celsius. The statuses are `turn_on` to wake up the pump, `turn_off` to turn off the pump, `high_temp` to set the temperature of the heat pump 80° degrees, and `low_temp` to set the temperature of the heat pump 50° degrees. It can do these actions: `turnOn()` and `turnOff()` the pump, `highTemp()` and `lowTemp()` to respectively increase the temperature and decrease the temperature of the pump. The consumption is an average of 6 kW during the `high_temp` phase, otherwise, it is considered to consume 3 kW. The water used daily is an average of 15 L.

Window. Smart windows can only be open in vasistas mode by the agent. It can be set on `open` or `close`. There are two actions: `openWindow()` or `closeWindow()`. Its consumption is considered nil.

Smart air purifier. It is used to clean and purify the air in the kitchen. It integrates a sensor that detects chips or burning smell. There are three statuses: `turn_on` to turn on the purifier, `turn_off` to switch off the device, and `dirty_air` to trigger the house agent to open the kitchen window. Three possible actions: `switchOn()`, `switchOff()` and `dirtyAirDetected()` to perform the correct action when it is detected smell. It consumes an average of 500 W when it is in action.

Humidity meter. It gives the state of the humidity inside the room. Two possible statuses: `high_humidity` when the humidity is over 85%, otherwise the state is `low_humidity`. Three actions performed: `humidityDetected()` is performed when the humidity goes over the threshold, `humidityNotDetected()` when the humidity is low, and `checkHumidity()` to constantly control the humidity value. It is considered a low-power device. No consumption value is provided.

Solar panel. It provides green energy to the house and allows using high power consumption devices at the same time. This is a simple device where two statuses are possible: `green_energy_available` when the sun is available and `no_green_energy_available` when it is night or cloudy. The two basic actions are `energyAvailable()` when the solar panels provide green energy and `noEnergyAvailable()` when the solar panels are off.

Vacuum cleaner. It is used to clean the first floor. It is controlled by the vacuum cleaner agent. It has an autonomous behaviour to clean the rooms. It has a battery, a garbage collection and a water tank inside the robot. The device can be set: `turn_off`, `turn_on`, `clean` when is cleaning a room, `move` when it has to move from a room to another (notice that the rooms has to be adjacent to move the robot), `return_to_base` means the robot is returning to the base if it cleaned all the rooms, `in_base` when the device is on and ready to be used, and `full_garbage` when the waste collection bag is full. The actions that can be performed by the device are several: `turnOn()`, `turnOff()`, `move()`, `clean()` if the room is dirty, and `returnToBase()` when it has to return to the base. It uses 150 W to fully charged the device and it consumes an average of 0.2 L and 10% of battery for each room. The water and the waste collection bag have to be manually reset by the residents. In this case, when the vacuum cleaner is empty from garbage and full fill with water, the vacuum cleaner is set on `turn_off`.

Garden robot. It is located inside the garage and it is used to cut the grass outside. It is controlled by the garden robot agent and it has a battery that can be charged by the base and it has a map of the placement of the field. For simplicity, it is assumed that the fields has the same name of the rooms. The device can be set: `turn_off`, `turn_on`, `cut` when is cut a field, `move` when it has to move from a field to another (notice that the fields has to be adjacent to move the robot), `return_to_base` means the robot is returning to the base if it has cut all the fields, `in_base` when the device is on and ready to be used. The actions that can be performed by the device are several: `turnOn()`, `turnOff()`, `move()`, `cut()` if the field is not cut, and `returnToBase()` when it has to return to the base. It uses 180 W to fully charged the device and it consumes an average of 18% of battery for each field. The water and the waste collection bag have to be manually reset by the residents. It doesn't have a garbage collection because the grass cut into small pieces is used as fertilizer. It is important to mention that the robot can perform the task only if the weather is sunny or cloudy and the temperature is above zero degrees.

Weather sensor. It monitors the temperature and the current weather outside the house. This device is essential for the correct behavior of some devices, like the robot garden, the solar panel, and the deicing system. It has two variables where it is stored the temperature, `temperature`, and the weather condition, `forecast_today`. There are two actions available: `changeTemp()` to perform actions according to the temperature, and `changeForecast()` to perform actions according to the actual weather. The temperature and the weather are changed manually during the simulation.

5 Metrics

The power of the meter of the domestic utility is set to be 5 kW during the night, and during the day, if the solar panels are available, the total power is 15 kW, otherwise it is 9 kW. The consumption of water and electricity is traced during the simulation. At the end of the day, it is provided a report of the daily utility consumption of the house.

Cost of electricity. The cost of the electricity is an average of 0,26 €/kW. The energy produced by solar panels can be sold for 0.10€/kWh, so it is more convenient to use the energy, if necessary, during the day when the solar panels are available.

Cost of water. The water costs an average of 1.37 €/m^3.

6 People and agents

People. Residents in the house include John and Mary and their baby, Mario. People can be in one room at a time, or out of the home. Mary is out-of-house from 9.00 to 16.00 Monday-Friday, while John works from 8.30 to 18.30 and he uses the electric car.
People have two main actions: `moveTo()` to let move people inside the house and `resetVacuumCleaner()` to empty the vacuum cleaner from garbage and full fill with water.

Agents. Agents assist residents by taking autonomous decisions, while still being responsive to residents' behaviors. In this project I implemented two types of agents: no planning agents and planning agents.

No planning agents

- The **house agent** is an agent with autonomous behaviors and it has the ability to communicates with other agents. It has several devices to manage, listed here:
  - **Electric car charger.** The agent manages the car charger device inside the garage to monitor if the car charger requires to charge the electric car;
  - **Weather sensor.** The agent controls the weather sensor to check if the temperature or the weather outside the house has changed;
  - **Smart tilting.** The agent controls the opening and closing of the smart tilting by receiving a message from other agents;
  - **Humidity meter.** The agent controls the humidity level inside the bathroom and, in case the humidity is over 85%, the bathroom window has to be open;
  - **Smart window.** The agent has the ability to open or close a window by receiving a message from other agents;
  - **Smart door lock.** The agent closes the three doors with the smart lock at 10.30 p.m. and it can receive a message from the infrared camera agent to close the garage door;
  - **Solar panel.** The agent observes changes of the weather sensor, and in case of a sunny day the solar panel are set to active, otherwise it is set to inactive;
  - **Deicing system.** The agent observes changes of the weather sensor, and in case the temperature goes below 0° the deicing system is enabled.
- **Smart air purifier agent.** The agent has the task to check if the air in the kitchen is dirty and, in case of smell, it turns on automatically the smart air purifier device and it opens the

kitchen window by sending a message to the house agent. It has two intentions: one to observe the environment and update its internal states, and one to send a message if the internal state of the agent change. I chose to create a new `Observable()` variable, as suggested by Professor Robol, to store the state of the device inside the agent and to use the `notifyChange()` and `notifyAnyChange()` functions;

- **Infrared camera agent.** The agent has to monitor the garage and, if a movement is detected, it sends a message to the house agent to lock the door of the garage. It has two intentions that works similar to the smart air purifier agent.

Planning agents
These agents control specific devices and they have knowledge of the world with belief states declared at the beginning of the simulation. These belief states can be changed during the simulation by observing the changes of the external world.
There are three planning agents implemented in this project: the vacuum cleaner agent, the robot garden agent and the heat pump agent.

- **Vacuum cleaner agent**. This agent has to perform a plan to successfully achieve a defined goal. In the simulation, I ask the agent to clean all the rooms and the agent has to turn on the vacuum cleaner device, exit from its base and plan a way to clean and move it on the floor. At the end it has to return to the base located in the hallway and turn off the device. It will provide a log with some info about the water, battery and garbage levels.
  The desired goal can be defined in a variable, and it needs to contain the rooms that we want to clean. I used two intentions, one to achieve the goal by passing the goal and the device name instantiated in the simulation, and one to observe the state of the water and garbage of the vacuum cleaner device, that can be changed by the residents. Then, if the cleaning goal succeeded, the goal to switch off the device is given to complete the task. The agent observes the state of the garbage and the water to update its internal knowledge.

- **Garden robot agent**. This agent controls the garden robot device. In this scenario, I ask the agent to cut some fields around the house, and the agent has to turn on the garden robot, exit from its base and plan a way to cut and move the device to achieve the final goal. At the end it has to return to the base located in the garage area and turn off the device. It will provide a log with info of the battery level. There are three intentions implemented: one is used for the sensing part, to update its internal beliefs based on the observation on the external world (for instance, if the weather sensor detects the weather is rainy, the garden robot cannot be used); one for managing the open/close of the smart tilting by communicating with the house agent to open the tilting when the garden robot device changes its `tiltingToBeOpen` state; and one to achieve goals. I had to set first the goal to switch on the device because the tilting at the beginning is not open and the agent's belief is `not tilting open`, and, because it is a precondition to exit from the base and the online solver creates a plan before the tilting is open, I had to subdivide the goal in sub-goals.

- **Heat pump agent**. This agent has to manage the heat pump to avoid waste of energy and avoid cutting the power. In this scenario, I ask the heat pump agent to bring the temperature of the pump to the maximum (80°). It has to turn on the heat pump device, reach the 50° and then reach the 80°. It has to monitor the state of three devices: the solar panel, the car charger, and the window (for the simulation I chose to observe only the kitchen window, but it can be extended to all windows). I implemented an intention to observe the external world and change the internal beliefs.

The planning part is created with PDDL language using an online solver. The vacuum cleaner agent and the garden robot agent has similar PDDL actions. I preferred to create two different scripts to be more clear and differentiate the two agents. I defined 6 vacuum cleaner PDDL actions:

1. **Move.** Move the agent's device from a room to another by performing the `move()` action;
2. **SwitchOn.** Switch on the agent's device by performing the `turnOn()` action;
3. **SwitchOff.** Switch off the agent's device by performing the `turnOff()` action;
4. **Clean.** Clean the given room by performing the `clean()` action;
5. **ReturnToBase.** Move to the room where the base is located by performing the `checkStatusBase(true)` action;
6. **ExitFromBase.** Perform the `checkStatusBase(false)` action.

The garden robot PDDL actions are similar, but they rely on different preconditions. To turn on the device, the weather has to be sunny and the temperature needs to be over 0°. The tilting has to be open to exit from the base.

The heat pump PDDL actions are defined as:

1. **SwitchOn.** Switch on the agent's device by performing the `turnOn()` action. It doesn't have particular preconditions;
2. **SwitchOff.** Switch off the agent's device by performing the `turnOff()` action;
3. **setTempHigh.** Raise the temperature to 80° by performing the `highTemp()` action. It perform it only if the solar panel is available, the kitchen window is close and the car charger is not working;
4. **setTempLow.** Reduce the temperature to 50° by performing the `lowTemp()` action.

7 Implementation

- **Sensor and agent perception**. In this project I implemented two different agents, no planning agent and planning agent. The house agent, the smart air purifier agent and the infrared agent doesn't have planning capabilities (no planning agents). They have only a partial representation of the external world, but they acquire the information they need to achieve their task successfully. The planning agents are based on perceptions encoded in belief states, that are initialized at the beginning of the scenario. They also have variables created as `Observable` that can be observed and used to perform specific actions. During the simulation, these belief states could change according to the behavior of the other devices. These agents need to observe the changing of the states of devices to update their internal knowledge to perform correctly tasks. The devices involved, necessary for the agent to act correctly in the real world, are passed as reference to the agent. With the intention of the agent, it is possible to monitor the changes of particular variable of a device and declare or undeclare facts into the belief set of the agent.
  I used the functions `notifyChange()` (if it has to check a single change of a particular variable), or `notifyAnyChange()` (if it has check any changes of variables of a device) to observe the changes of the device's variables. During the simulation some statuses are changed manually (temperature, humidity, weather condition...) to see how the system reacts to these changes.
  As an example, the garden robot agent needs information from the weather sensor to update its internal knowledge according to the temperature and the weather condition. An agent that rely on different devices is the heat pump agent, where it has to observe changes from the window (for simplicity I added only the kitchen window, but it can be scalable to all the smart windows), the car charger and the solar panel, which in turn observe the weather meter parameters;

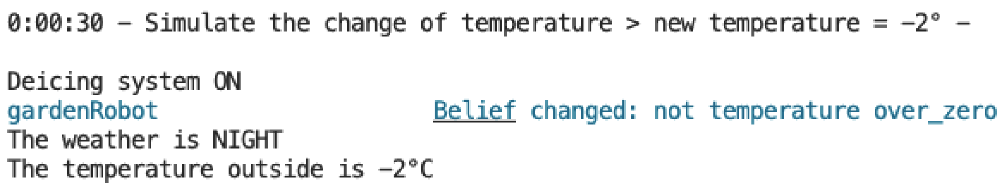
- **Agent acting in a shared environment**. The agents take autonomous decisions and every agent has an autonomous behavior and a specific device to control. Some agents do actions when they receive messages from other agents. Planning agents changed their beliefs state according to other devices that change the condition of the environment, that is shared among different agents. If the agent cannot achieve the goal with the current knowledge, it can retry to achieve the goal with the *RetryGoal* and *RetryFourTimesIntention* intention if the belief state of the agent changes during the simulation. As an example, the garden robot device has to open the tilting to operate outside the house. When the garden robot device requests to open/close the tilting, it updates a Boolean variable `tiltingToBeOpen`, and the garden robot agent perceives the change value of the variable. The smart tilting is a device of the house controlled by the house agent. To successfully open/close the tilting, the garden robot agent sends a message to the house agent to communicate its intention to do an action, in this case to open/close the tilting when it needs;
- **Agent interaction and coordination**. The agents can assign goals to each other to achieve specific actions. The `PostMan` and the `MessageDispatcher` enable the communication between two agents. I implemented this communication to control devices that belong to different agents. As an example, if the `dirty_air` signal is triggered, the smart air purifier agent sends a message to the house agent to open the kitchen window. Another example, mentioned above but useful also in this section, is the communication between the garden robot agent and the house agent to open/close the tilting, otherwise, in a real world scenario the robot garden device could not achieve its desired goal because it cannot reach the outside of the house.

7.1 Scenarios

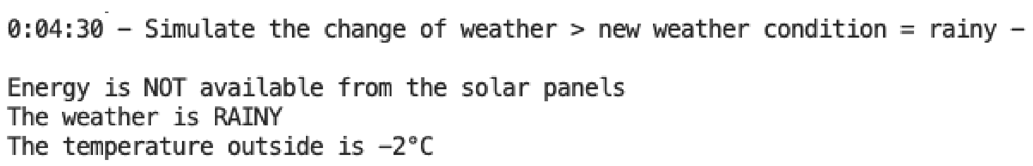
I created 2 scenarios, one to simulate a day at the Scandinavian house to see how the changes to the external world affect the agent's beliefs and one to show how intentions and planning methods work.

Scenario #1: the day simulation
During this scenario, I simulated a day by changing the weather condition, the temperature, the humidity level and the reset of the vacuum cleaner to show how the belief changed across the agents and what happen when some values changed. Some devices will perform actions based on the hour.
At the end of the scenario, I reported a sum up of the consumption of the house with the costs defined in the section *Metrics*. This trace of the consumption is made possible by the sum of the power or water consumption during the simulation of each device. Notice that in this scenario I don't want to achieve specific goal from planning agents (in the next paragraph there will be a good attention on that).

As first example, I changed the temperature perceived by the weather sensor and the beliefs state of the garden robot agent successfully changed. As expected, the deicing system turns on when the temperature is below 0°:

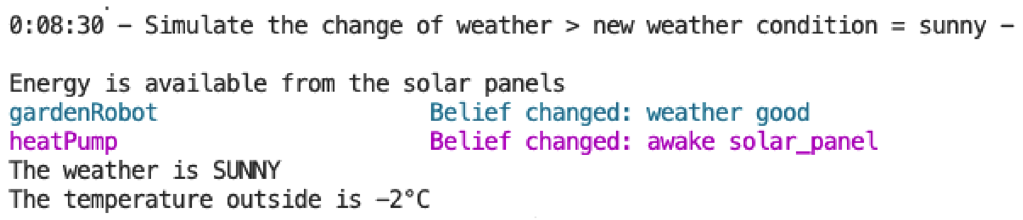

```
0:00:30 - Simulate the change of temperature > new temperature = -2° -

Deicing system ON
gardenRobot                   Belief changed: not temperature over_zero
The weather is NIGHT
The temperature outside is -2°C
```

As second example, I simulated a rainy day (the weather condition is initialized to `night`), and the belief state of the garden robot doesn't update because the rainy weather implies the same belief state of night weather condition, so it doesn't apply different belief states at the environment from the agent's prospective:



```
0:04:30 - Simulate the change of weather > new weather condition = rainy -

Energy is NOT available from the solar panels
The weather is RAINY
The temperature outside is -2°C
```

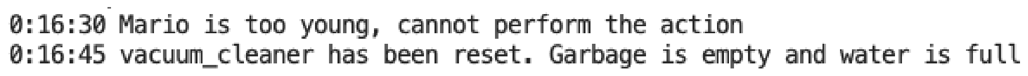
As third example, I manually changed the weather condition to `sunny`



```
0:08:30 - Simulate the change of weather > new weather condition = sunny -

Energy is available from the solar panels
gardenRobot                    Belief changed: weather good
heatPump                       Belief changed: awake solar_panel
The weather is SUNNY
The temperature outside is -2°C
```

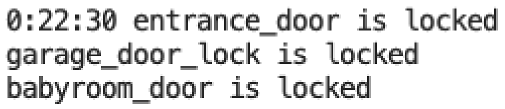
and the belief states of the garden robot and the heat pump agents have changed.

As fourth example, I reset the vacuum cleaner with Mario, but he is too young to do the job. Then I reset the device with John that he is allowed to do the task. The vacuum cleaner is already full of water and the garbage is empty, so the belief states of the vacuum cleaner agent doesn't change because the external world doesn't change with the action of John:



```
0:16:30 Mario is too young, cannot perform the action
0:16:45 vacuum_cleaner has been reset. Garbage is empty and water is full
```

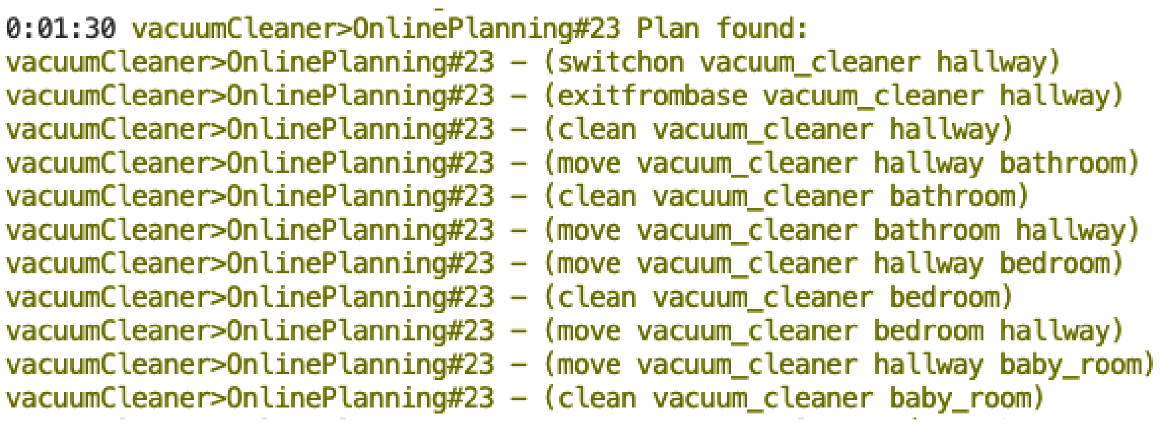
As last example, I chose to lock the door of the entrance, the baby room and the garage door at 10.30 p.m., and the task was correctly executed at that time:



```
0:22:30 entrance_door is locked
garage_door_lock is locked
babyroom_door is locked
```

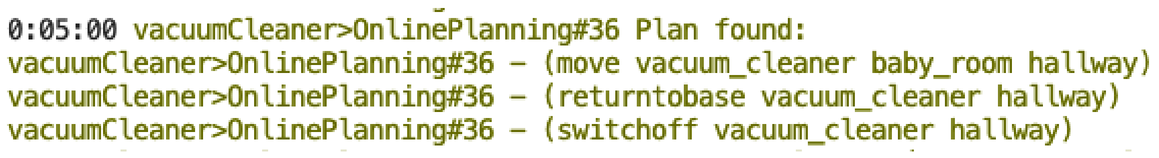
In the next part I will focus the work on planning and no planning agents.

Scenario #2: planning and no planning agents simulation
In this scenario I will show how planning and no planning agents perform. During this simulation, I want to achieve some desired goal performed by the vacuum cleaner, garden robot and heat pump agents.
After the initial declaration of all the beliefs of the agents, I want to achieve the first goal: clean all the rooms of the first floor.
I expect the vacuum cleaner to turn on, exit from the base, clean all the rooms, return to the base and turn off.
As explained in the section *People and agents*, first the online planner finds a plan to clean all the rooms and then a plan to return to the base and switch off the device.



```
0:01:30 vacuumCleaner>OnlinePlanning#23 Plan found:
vacuumCleaner>OnlinePlanning#23 - (switchon vacuum_cleaner hallway)
vacuumCleaner>OnlinePlanning#23 - (exitfrombase vacuum_cleaner hallway)
vacuumCleaner>OnlinePlanning#23 - (clean vacuum_cleaner hallway)
vacuumCleaner>OnlinePlanning#23 - (move vacuum_cleaner hallway bathroom)
vacuumCleaner>OnlinePlanning#23 - (clean vacuum_cleaner bathroom)
vacuumCleaner>OnlinePlanning#23 - (move vacuum_cleaner bathroom hallway)
vacuumCleaner>OnlinePlanning#23 - (move vacuum_cleaner hallway bedroom)
vacuumCleaner>OnlinePlanning#23 - (clean vacuum_cleaner bedroom)
vacuumCleaner>OnlinePlanning#23 - (move vacuum_cleaner bedroom hallway)
vacuumCleaner>OnlinePlanning#23 - (move vacuum_cleaner hallway baby_room)
vacuumCleaner>OnlinePlanning#23 - (clean vacuum_cleaner baby_room)
```

Figure 3: Plan to clean the rooms

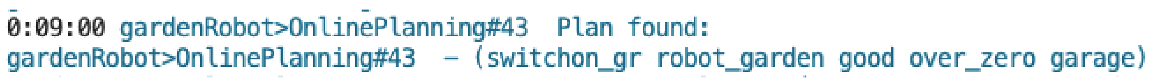


```
0:05:00 vacuumCleaner>OnlinePlanning#36 Plan found:
vacuumCleaner>OnlinePlanning#36 - (move vacuum_cleaner baby_room hallway)
vacuumCleaner>OnlinePlanning#36 - (returntobase vacuum_cleaner hallway)
vacuumCleaner>OnlinePlanning#36 - (switchoff vacuum_cleaner hallway)
```

Figure 4: Plan to switch off the vacuum cleaner after cleaning the rooms

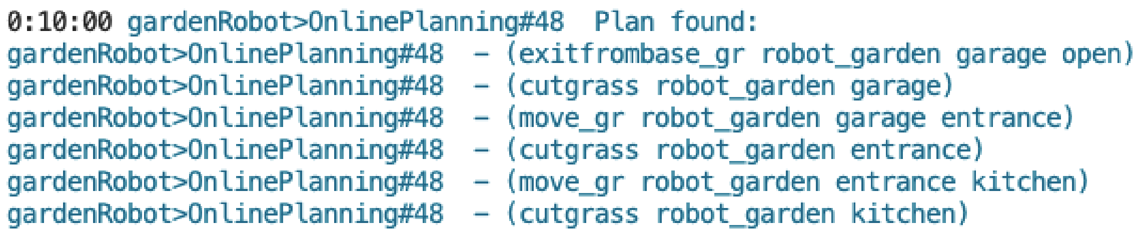
We can see that the plans were successfully found for reaching the desired goal.
The vacuum cleaner move from a room to another. Because of the structure of the house, the hallway is the room where the device can reach all the other rooms. It correctly move to the hallway to reach the other rooms but it doesn't clean the hallway every time. At the end, when it has to return to the base, it knows, from its internal knowledge, that the vacuum cleaner is inside the baby room. It plans to move from the baby room to the hallway and then turn the device off.

As second test, I want to cut some areas of the house.
I expect the garden robot to turn on, open the tilting, exit from base, cut the areas specified in the goal, return to base and turn off, only if it is sunny or cloudy.



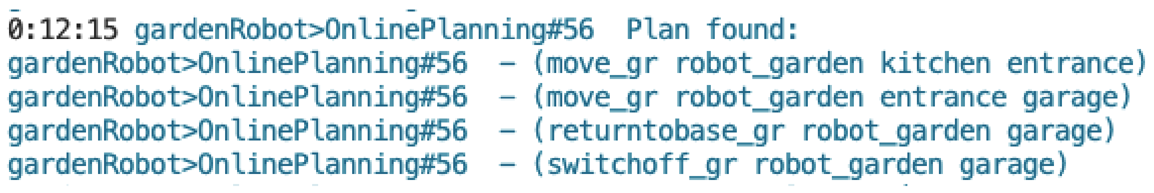
```
0:09:00 gardenRobot>OnlinePlanning#43  Plan found:
gardenRobot>OnlinePlanning#43  - (switchon_gr robot_garden good over_zero garage)
```

Figure 5: Plan to turn on the device



```
0:10:00 gardenRobot>OnlinePlanning#48  Plan found:
gardenRobot>OnlinePlanning#48  - (exitfrombase_gr robot_garden garage open)
gardenRobot>OnlinePlanning#48  - (cutgrass robot_garden garage)
gardenRobot>OnlinePlanning#48  - (move_gr robot_garden garage entrance)
gardenRobot>OnlinePlanning#48  - (cutgrass robot_garden entrance)
gardenRobot>OnlinePlanning#48  - (move_gr robot_garden entrance kitchen)
gardenRobot>OnlinePlanning#48  - (cutgrass robot_garden kitchen)
```

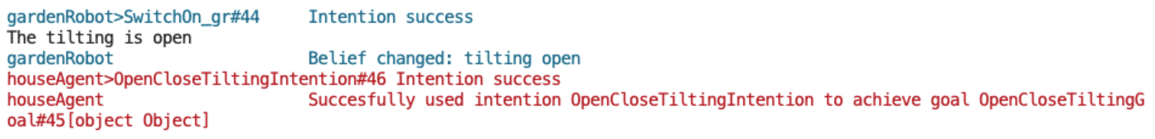
Figure 6: Plan to cut the grass of some areas



```
0:12:15 gardenRobot>OnlinePlanning#56  Plan found:
gardenRobot>OnlinePlanning#56  - (move_gr robot_garden kitchen entrance)
gardenRobot>OnlinePlanning#56  - (move_gr robot_garden entrance garage)
gardenRobot>OnlinePlanning#56  - (returntobase_gr robot_garden garage)
gardenRobot>OnlinePlanning#56  - (switchoff_gr robot_garden garage)
```

Figure 7: Plan to turn off the garden robot

As before, we can see the plans are correctly created and the garden robot has successfully cut the grass.
Also, as shown down below, the tilting opens before the garden robot exits from the base. The same works when the robot turns off. The agent correctly communicate with the house agent to achieve the goal to open or close the smart tilting.

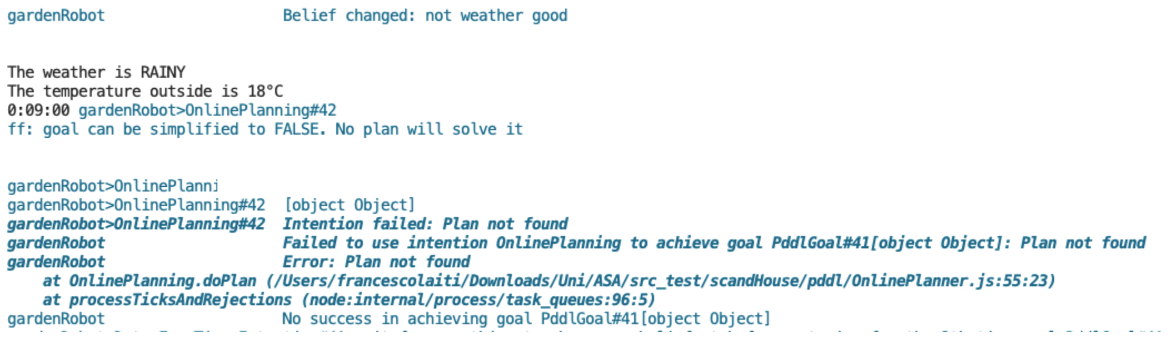


```
gardenRobot>SwitchOn_gr#44     Intention success
The tilting is open
gardenRobot                    Belief changed: tilting open
houseAgent>OpenCloseTiltingIntention#46 Intention success
houseAgent                     Succesfully used intention OpenCloseTiltingIntention to achieve goal OpenCloseTiltingG
oal#45[object Object]
```

Just to show, I changed the weather from `sunny` to `rainy`

```
myHouse.devices.meteo_sensor.forecast_today = 'rainy'
```

The plan is not found, because the belief of the agent regarding the weather has changed by observing the weather sensor and it cannot cut the grass if it is raining. This is the output:



```
gardenRobot                   Belief changed: not weather good


The weather is RAINY
The temperature outside is 18°C
0:09:00 gardenRobot>OnlinePlanning#42
ff: goal can be simplified to FALSE. No plan will solve it


gardenRobot>OnlinePlanni
gardenRobot>OnlinePlanning#42  [object Object]
gardenRobot>OnlinePlanning#42  Intention failed: Plan not found
gardenRobot                    Failed to use intention OnlinePlanning to achieve goal PddlGoal#41[object Object]: Plan not found
gardenRobot                    Error: Plan not found
    at OnlinePlanning.doPlan (/Users/francescolaiti/Downloads/Uni/ASA/src_test/scandHouse/pddl/OnlinePlanner.js:55:23)
    at processTicksAndRejections (node:internal/process/task_queues:96:5)
gardenRobot                    No success in achieving goal PddlGoal#41[object Object]
```

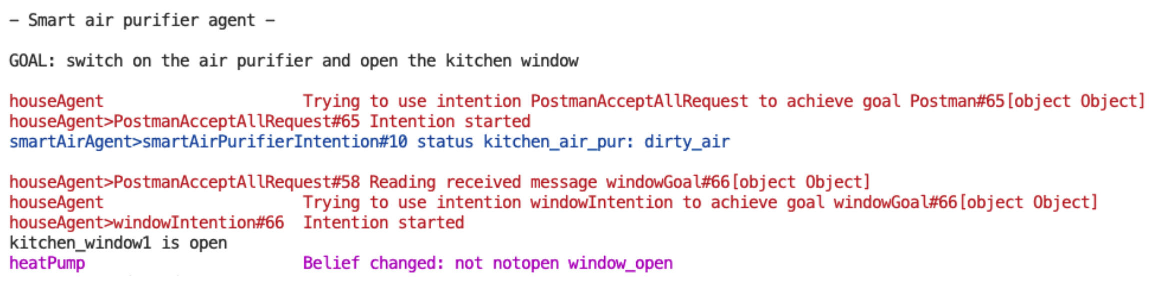
As third test, I want to put the temperature of the heat pump higher as possible (80°). I implemented the heat pump that raises the temperature to 80 degrees only if 3 conditions are valid: the solar panels are active, the kitchen window is closed and the car charger is disconnected.
At the beginning the plan is found, because the 3 precondition above are all satisfied. If I change the weather to `cloudy` the solar panels turn off and the desired goal could not be achieved, so the plan cannot be found. To avoid lots of screenshot logs in this section, more details are provided in the `run_planning_intention.log` file.
All the agents can retry the goal, if it is not achieved, for four times if the beliefs changed, thanks to the intention `RetryFourTimesIntention`.

Now I will present how the 3 no planning agents communicate each other to reach the specific goal as described in *Section 6*.

Starting with the smart air purifier, I changed his state to `dirty_air`

```
myHouse.devices.kitchen_air_pur.status = 'dirty_air'
```

and this is what happens:



```
- Smart air purifier agent -

GOAL: switch on the air purifier and open the kitchen window

houseAgent                     Trying to use intention PostmanAcceptAllRequest to achieve goal Postman#65[object Object]
houseAgent>PostmanAcceptAllRequest#65 Intention started
smartAirAgent>smartAirPurifierIntention#10 status kitchen_air_pur: dirty_air

houseAgent>PostmanAcceptAllRequest#58 Reading received message windowGoal#66[object Object]
houseAgent                     Trying to use intention windowIntention to achieve goal windowGoal#66[object Object]
houseAgent>windowIntention#66  Intention started
kitchen_window1 is open
heatPump                       Belief changed: not notopen window_open
```

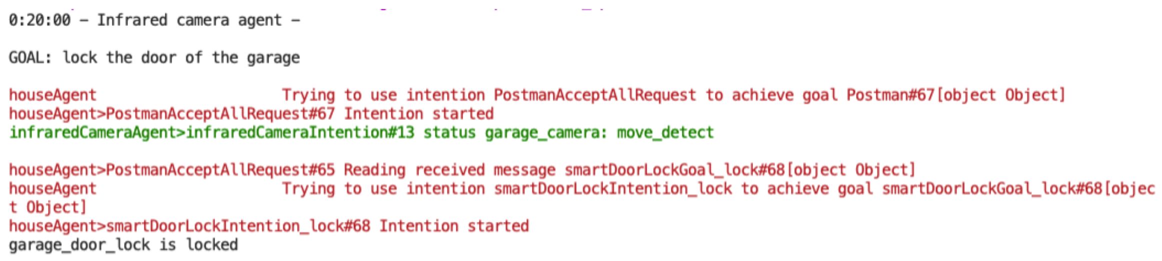
Figure 8: Log of the smart air purifier

The smart air purifier agent sent correctly the message to the house agent, and the final action is to open the kitchen window. It is worth notice that the heat pump agent changes its belief state because the window is now open. In order to send the correct state for the window, I added to the `MessageDispatcher` a new variable that contains the desired state of the window:

```
MessageDispatcher.authenticate(this.agent).sendTo('houseAgent', new windowGoal(this.agent.house.devices.kitchen_windows, this.agent.house, 'open'));
```

and I adjusted the window intention file to correctly managed the new variable. In this way I don't have to create two different intentions for the same device.

Another example is provided with the infrared camera:

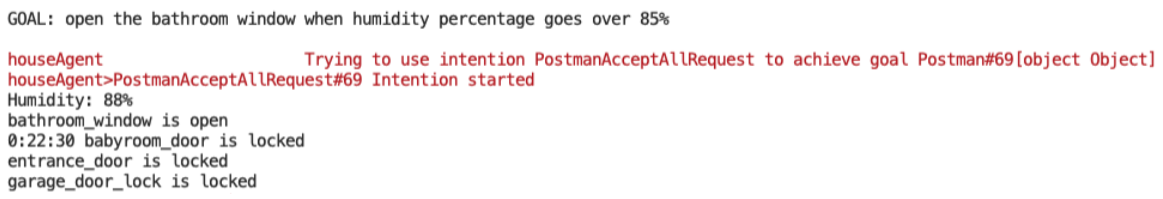


```
0:20:00 - Infrared camera agent -

GOAL: lock the door of the garage

houseAgent                     Trying to use intention PostmanAcceptAllRequest to achieve goal Postman#67[object Object]
houseAgent>PostmanAcceptAllRequest#67 Intention started
infraredCameraAgent>infraredCameraIntention#13 status garage_camera: move_detect

houseAgent>PostmanAcceptAllRequest#65 Reading received message smartDoorLockGoal_lock#68[object Object]
houseAgent                     Trying to use intention smartDoorLockIntention_lock to achieve goal smartDoorLockGoal_lock#68[objec
t Object]
houseAgent>smartDoorLockIntention_lock#68 Intention started
garage_door_lock is locked
```

Figure 9: Log of the infrared camera

As shown in the log, when I manually set a movement in the garage, the door of the garage is locked. The infrared camera agent sends a message to the house agent and the garage door is now locked.

As another example, I set the humidity value to 88% and these are the outputs:

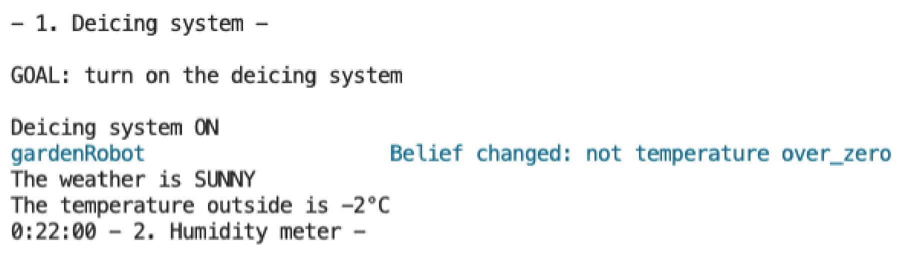


```
GOAL: open the bathroom window when humidity percentage goes over 85%

houseAgent                     Trying to use intention PostmanAcceptAllRequest to achieve goal Postman#69[object Object]
houseAgent>PostmanAcceptAllRequest#69 Intention started
Humidity: 88%
bathroom_window is open
0:22:30 babyroom_door is locked
entrance_door is locked
garage_door_lock is locked
```

Figure 10: Log of the humidity meter

The house agent observes the state of the humidity of the sensor and, because the humidity level goes over 85%, the bathroom window has to be open. The 3 doors are locked because the house agent has the task to close the doors at 10.30 p.m. automatically.

In the final example I set the temperature outside -2°. The deicing system automatically turns on and the belief of the garden robot agent changes:



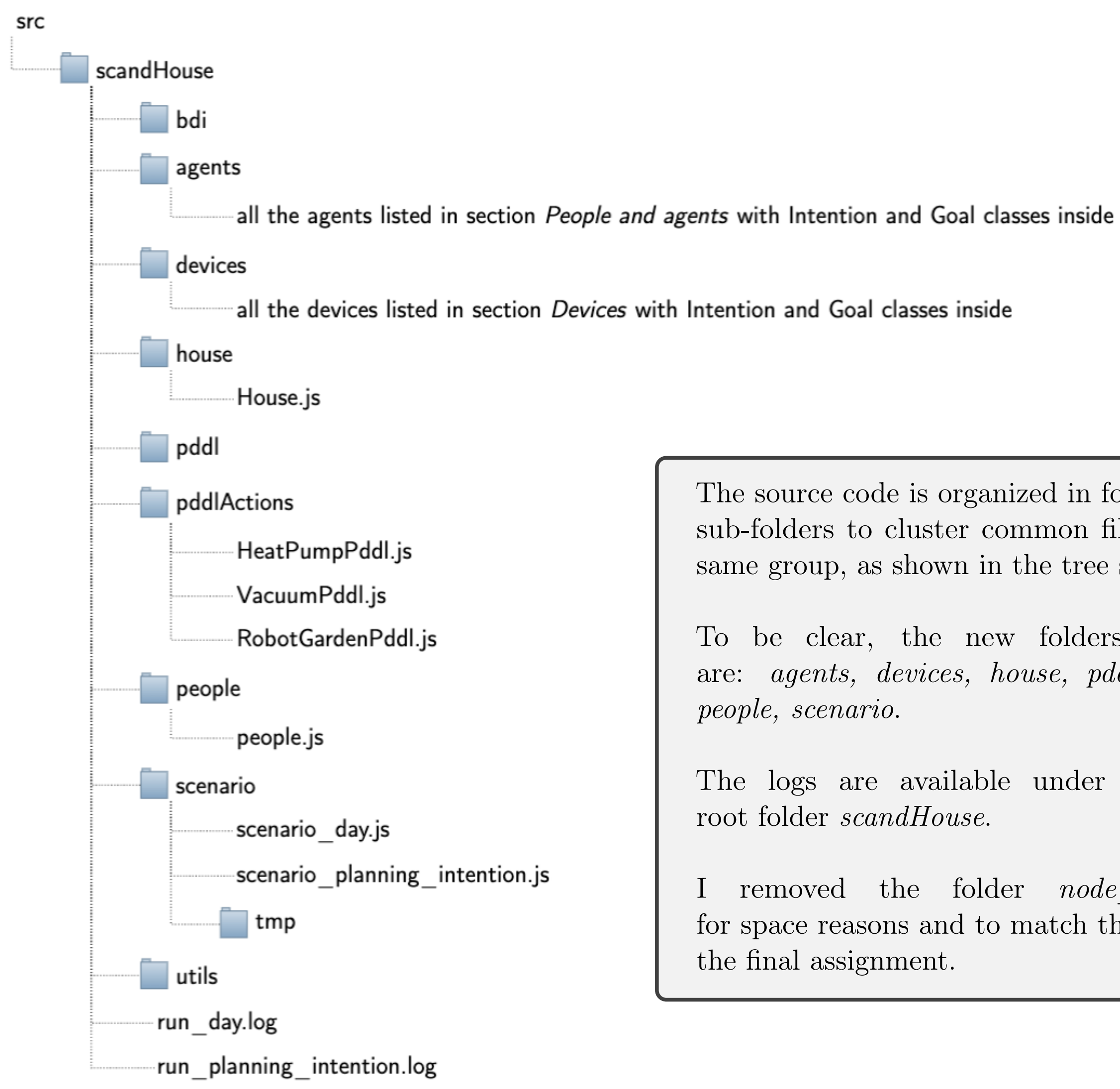
```
- 1. Deicing system -

GOAL: turn on the deicing system

Deicing system ON
gardenRobot                    Belief changed: not temperature over_zero
The weather is SUNNY
The temperature outside is -2°C
0:22:00 - 2. Humidity meter -
```

Figure 11: Log of the deicing system

More information are available in the `run_day.log` and `run_planning_intention.log` log files.

7.2 Source code organization

Below I report the structure of the code and I highlight the files modified and created by me:



```
src
└── scandHouse
    ├── bdi
    ├── agents
    │   └── all the agents listed in section People and agents with Intention and Goal classes inside
    ├── devices
    │   └── all the devices listed in section Devices with Intention and Goal classes inside
    ├── house
    │   └── House.js
    ├── pddl
    ├── pddlActions
    │   ├── HeatPumpPddl.js
    │   ├── VacuumPddl.js
    │   └── RobotGardenPddl.js
    ├── people
    │   └── people.js
    ├── scenario
    │   ├── scenario_day.js
    │   ├── scenario_planning_intention.js
    │   └── tmp
    ├── utils
    ├── run_day.log
    └── run_planning_intention.log
```

The source code is organized in folders and sub-folders to cluster common files in the same group, as shown in the tree structure.

To be clear, the new folders created are: *agents, devices, house, pddlActions, people, scenario.*

The logs are available under the sub-root folder *scandHouse.*

I removed the folder *node_modules* for space reasons and to match the rules of the final assignment.

This is the link to the repository on GitHub with the change's history of my code
https://github.com/laitifranz/ASA-project-21-22

8 Improvements from the previous work

July 2022

- Improvement of PDDL actions of the vacuum cleaner;
- Fixed different behaviour of agents;
- Implementation of the Clock inside the intentions;
- Change the bulleted list in the section *Devices*;
- Added two new devices, the weather sensor and the garden robot.

September 2022

- Fixed the agent and device implementation;
- Update the belief state of an agent according to the observation of the external world;
- Implementation of *MessageDispatcher* and *PostMan* to correctly use the device related to the agent without using shortcuts (using the house reference to manipulate directly the device inside the agent);
- Better organization of the code, in particular the *Intention classes* and *Agent classes*;
- Fixed PDDL implementation with a more generic structure and remove unnecessary classes;
- Bug fixes and stability improvements.