Student: **Francesco Laiti**
Academic year: **2021/2022**
Course: **Autonomous Software Agents**
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):

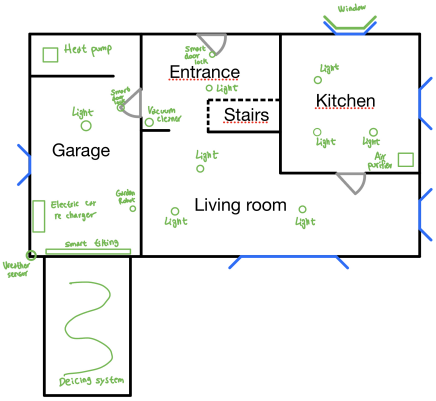


Figure 1: Blueprint of *Ground floor*

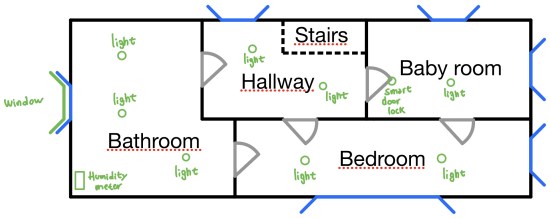
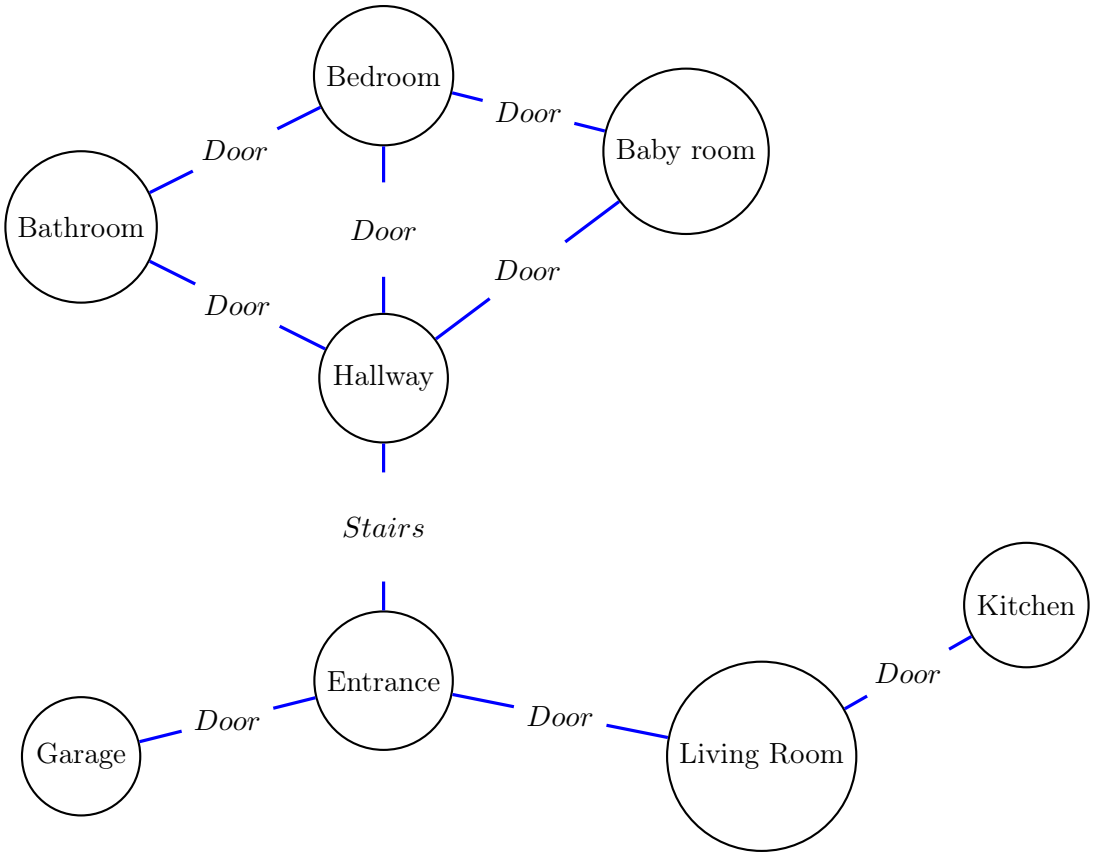


Figure 2: Blueprint of *First floor*

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.
Not all the devices are implemented with agents, someone has been considered to be manually turned on/off.
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 status 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: agents with no planning and planning agents.

No planning agents

- The **house agent** is an agent with autonomous behaviors. It communicates with other agents. It has several devices to managed:
  - **Electric car charger.** The agent managed the car charger inside the garage to monitor if the car has to be charge or if it can enable the fast charging;
  - **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 opens the tilting from 7.00 a.m. to 9.00 a.m. to facilitate the exit of John with his car;
  - **Smart window.** The agent can open the window of the bathroom if the agent of the humidity meter send a message;
  - **Smart door lock.** The agent close the two doors with the lock door at 22.30.
- The **device-specific agents** are two, listed here:
  - **Smart air purifier agent.** It has the task to check if the air in the kitchen is dirty and, in case of smell, it turns on automatically and it opens the kitchen window by sending a message to the house agent, the agent that controls the window.
  - **Infrared camera agent.** It has to monitor the garage and if a movement is detected it send a message to the house agent to lock the door of the garage.

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 states can be changed during the simulation according to the behavior of the other devices and agents.
There are four planning agents implemented in this project: the vacuum cleaner agent, the robot garden agent, the heat pump agent and the deicing system agent.

- **Vacuum cleaner agent**. This agent has to perform a plan to successfully achieve a defined goal. In this case, I ask the agent to clean all the rooms, and the agent has to turn on the vacuum cleaner, exit from its base and plan a way to clean and move on the floor. At the end it has to return to the base located in the hallway and turn off the device. At the end it will provide a log with some info of the water, battery and garbage levels.

- **Garden robot agent**. This agent controls the garden robot inside the garage. In this scenario, I ask the agent to cut all the 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. At the end it will provide a log with some info of the battery. It is important to say that this agent communicate with the house agent to open the tilting when the robot has to cut the grass and close the tilting when the robot has returned to the base. The agent has internal beliefs that can be changed by the other devices. For instance, if the weather sensor detects that the weather is rainy, the garden robot cannot be used.

- **Heat pump agent**. This agent has to manage the heat pump to avoid waste of energy and avoid cutting the power. It has the following tasks to take in mind:

  - if the solar panels are available and the electric car is not plugged, the heat pump is set to `high_temp` to satisfy the necessity of the house for the shower and for the floor system and other tasks to be ready for more demand of hot water,
  - if at least one window is open, the temperature go down and so the `low_temp` is set. This is performed because it is not necessary to have more hot water if the windows are open, otherwise, it is a waste of heating,
  - if the car charger is plugged, the heat pump is set on `low_temp`. The car charger requires lots of energy, and the combination of all the devices at the maximum power could cut the power of the house.

- **Deicing system agent**. This agent has a small planning with respect to the other planning agents. It can turn on the deicing system only if the temperature outside is below zero, otherwise it cannot switch on the system.

7 Implementation

- **Sensor and agent perception**. In this project I implemented two different agents, no planning agent and planning agent. The first one perceives the changes of the environment with the functions `notifyChange()` if it has to check a single change of a particular status, or `notifyAnyChange()` to check any changes of statuses of a device. During the simulation some statuses are changed manually (temperature, humidity, weather condition...) to see how the system reacts to these changes.
  The planning agents are based on perceptions encoded in belief states, that are initialized at the beginning of the scenario. During the simulation these belief states could change according to the behavior of the other devices;

- **Agent acting in a shared environment**. The agents take autonomous decisions and every agent has an autonomous behavior. 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;
- **Agent interaction and coordination**. The `PostMan` and the `MessageDispatcher` enable the communication between two agents. In particular, I implemented these functions in the two device-specific agents and the main house agent to exchange messages and trigger different functions of the devices. For instance, if the `dirty_air` signal is triggered, the smart air purifier agent sends a message to the main house agent to open the kitchen window.

7.1 Scenarios

I created 2 scenarios, one for the simulation of one day at the Scandinavian house and one to show how intentions and planning methods work.

Scenario #1: the day simulation
During this scenario, I simulated a day. Some devices will perform actions based on the hour. I have also changed manually some parameters (temperature, weather condition, humidity) to show how the belief changed across the agents and what happen when some values changed.
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), so the consumption are not high.
As first example, I changed the temperature perceived by the weather sensor and it successfully changed the beliefs state of the garden robot agent and the deicing system agent:

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

Energy is NOT available from the solar panels
The weather is NIGHT
The temperature outside is -2°C
garage_deice_system            Belief changed: temperature under_zero
garage_deice_system            Belief changed: not temperature over_zero
robot_garden                   Belief changed: not temperature over_zero
```

As second example, I simulated a rainy day (the weather condition is initialized to `night`), and the belief states of the solar panels and garden robot remains the same because the rainy weather doesn't apply different belief states at the environment from the agents 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, the smart tilting is open when the hour is between 7.00 a.m. and 9.00 a.m. As described in the section *Devices*, the smart tilting, when open, turn on the light of the garage, and turn off the light of the garage when it is closed:

```
0:07:00 garage_light turned on
The tilting is open
```

Figure 3: Open the tilting at 7.00 a.m.

```
0:09:00 garage_light turned off
The tilting is close
```

Figure 4: Close the tilting at 9.00 a.m.

As fourth 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
The weather is SUNNY
The temperature outside is -2°C
robot_garden                    Belief changed: weather good
garage_pump                     Belief changed: awake solar_panel
```

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

As last example, I chose to lock the door of the entrance and the baby room 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
houseAgent>smartDoorLockIntention#6 Intention success
houseAgent                  Succesfully used intention smartDoorLockIntention to achieve goal smartDoorLockGoal#6[object Object]
houseAgent>smartDoorLockIntention#7 Intention success
houseAgent                  Succesfully used intention smartDoorLockIntention to achieve goal smartDoorLockGoal#7[object Object]
houseAgent>smartDoorLockIntention#5 Intention success
houseAgent                  Succesfully used intention smartDoorLockIntention to achieve goal smartDoorLockGoal#5[object Object]
```

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 agents perform. During this simulation, I want to achieve some desired goal performed by the vacuum cleaner, garden robot, heat pump and deicing system 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. This is the snippet of code with the goal:

```
myHouse.devices.vacuum_cleaner.postSubGoal( new RetryGoal( { goal: new PlanningGoal( { goal: ['finish_clean vacuum_cleaner'] } ) } ) )
```

I expect the vacuum cleaner to turn on, exit from the base, clean all the rooms, return to the base and turn off.

```
0:01:15 vacuum_cleaner>OnlinePlanning#14 Plan found:
vacuum_cleaner>OnlinePlanning#14 - (switchon vacuum_cleaner)
vacuum_cleaner>OnlinePlanning#14 - (exitfrombase vacuum_cleaner)
vacuum_cleaner>OnlinePlanning#14 - (clean vacuum_cleaner hallway)
vacuum_cleaner>OnlinePlanning#14 - (move vacuum_cleaner hallway bathroom)
vacuum_cleaner>OnlinePlanning#14 - (clean vacuum_cleaner bathroom)
vacuum_cleaner>OnlinePlanning#14 - (move vacuum_cleaner bathroom hallway)
vacuum_cleaner>OnlinePlanning#14 - (move vacuum_cleaner hallway bedroom)
vacuum_cleaner>OnlinePlanning#14 - (clean vacuum_cleaner bedroom)
vacuum_cleaner>OnlinePlanning#14 - (move vacuum_cleaner bedroom hallway)
vacuum_cleaner>OnlinePlanning#14 - (move vacuum_cleaner hallway baby_room)
vacuum_cleaner>OnlinePlanning#14 - (clean vacuum_cleaner baby_room)
vacuum_cleaner>OnlinePlanning#14 - (cleanall vacuum_cleaner hallway bathroom bedroom baby_room)
vacuum_cleaner>OnlinePlanning#14 - (move vacuum_cleaner baby_room hallway)
vacuum_cleaner>OnlinePlanning#14 - (returntobase vacuum_cleaner hallway)
vacuum_cleaner>OnlinePlanning#14 - (switchoff vacuum_cleaner)
vacuum_cleaner>OnlinePlanning#14 - (lasttask vacuum_cleaner)
```

Figure 5: Plan of the vacuum cleaner

We can see that a plan was 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.

As second test, I want to cut the grass of the house. Here is the snippet of the code:

```
myHouse.devices.robot_garden.postSubGoal( new RetryGoal( { goal: new PlanningGoal( { goal: ['finish_cut robot_garden'] } ) } ) )
```

I expect the garden robot to turn on, open the tilting, exit from base, cut all grass, return to base and turn off, only if it is sunny or cloudy.

```
0:01:15 robot_garden>OnlinePlanning#14 Plan found:
robot_garden>OnlinePlanning#14 - (switchon_gr robot_garden good over_zero)
robot_garden>OnlinePlanning#14 - (exitfrombase_gr robot_garden)
robot_garden>OnlinePlanning#14 - (cutgrass robot_garden garage)
robot_garden>OnlinePlanning#14 - (move_gr robot_garden garage entrance)
robot_garden>OnlinePlanning#14 - (cutgrass robot_garden entrance)
robot_garden>OnlinePlanning#14 - (move_gr robot_garden entrance kitchen)
robot_garden>OnlinePlanning#14 - (cutgrass robot_garden kitchen)
robot_garden>OnlinePlanning#14 - (move_gr robot_garden kitchen living_room)
robot_garden>OnlinePlanning#14 - (cutgrass robot_garden living_room)
robot_garden>OnlinePlanning#14 - (cutall robot_garden garage entrance living_room kitchen)
robot_garden>OnlinePlanning#14 - (move_gr robot_garden living_room garage)
robot_garden>OnlinePlanning#14 - (returntobase_gr robot_garden garage)
robot_garden>OnlinePlanning#14 - (switchoff_gr robot_garden)
robot_garden>OnlinePlanning#14 - (lasttask_gr robot_garden)
```

Figure 6: Plan of the garden robot

As before, we can see the plan is correctly created and the garden robot has successfully cut all the grass. Also, as shown down below, the tilting opens before turning on the garden robot. The same works when the robot turns off.

```
robot_garden>OnlinePlanning#14 Starting sequential step (SwitchOn_gr robot_garden good over_zero) Effect: switch_on robot_garden
robot_garden>SwitchOn_gr#15    Intention started
robot_garden                   Belief changed: switch_on robot_garden
0:01:30 garage_light turned on
The tilting is open
robot_garden turned on
robot_garden                   Belief changed: not switch_off robot_garden
```

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 knowledge of the agent has changed and it cannot cut the grass if it is raining. This is the output of the PDDL script:

```
robot_garden>OnlinePlanning#14 No plan found
robot_garden>OnlinePlanning#14 [object Object]
robot_garden>OnlinePlanning#14 Intention failed: Plan not found
robot_garden                   Failed to use intention OnlinePlanning to achieve goal PddlGoal#13[object Object]: Plan not found
robot_garden                   Error: Plan not found
    at OnlinePlanning.doPlan (/Users/francescolaiti/Downloads/src_test/scandHouse/pddl/OnlinePlanner.js:55:23)
    at processTicksAndRejections (node:internal/process/task_queues:96:5)
```

As third test, I want to put the temperature of the heat pump higher as possible. I implemented the heat pump that raises the temperature to 80 degrees only if 3 conditions are valid: the solar panels are active, windows are closed and the car charger is disconnected. This is the snipped of code of the desired goal:

```
myHouse.devices.garage_pump.postSubGoal( new RetryGoal( { goal: new PlanningGoal( { goal: ['temp_high heat_pump'] } ) } ) )
```

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 anymore 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` .

As last test, I tested the deicing system planning agent. The system turns on only if the temperature outside is less than 0 degrees. As initial belief, the temperature is over 0 degrees. This means that a plan could not be found. Then I manually set the temperature to -2 degrees from the weather sensor and the deicing system successfully found a plan. It retries the goal after the beliefs changed, thanks to the intention `RetryFourTimesIntention`. As before, to avoid lots of screenshot, I provide all the logs in the `run_planning_intention.log` file.

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

Starting with the smart air purifier, I changed his status 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#13[object Object]
houseAgent>PostmanAcceptAllRequest#13 Intention started
smartAirAgent>smartAirPurifierIntention#10 status kitchen_air_pur: dirty_air

houseAgent>PostmanAcceptAllRequest#8 Reading received message windowGoal#14[object Object]
houseAgent                    Trying to use intention windowIntention to achieve goal windowGoal#14[object Object]
houseAgent>windowIntention#14  Intention started
kitchen_window1 is open
garage_pump                   Belief changed: open window_open
```

Figure 7: 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. 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.house.devices.kitchen_windows, this.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:

```
0:02:00 - Infrared camera agent -

GOAL: lock the door of the garage

houseAgent                    Trying to use intention PostmanAcceptAllRequest to achieve goal Postman#13[object Object]
houseAgent>PostmanAcceptAllRequest#13 Intention started
infraredCameraAgent>infraredCameraIntention#12 status garage_camera: move_detect

houseAgent>PostmanAcceptAllRequest#8 Reading received message smartDoorLockGoal_lock#14[object Object]
houseAgent                    Trying to use intention smartDoorLockIntention_lock to achieve goal smartDoorLockGoal_lock#14[object Object]
houseAgent>smartDoorLockIntention_lock#14 Intention started
garage_door_lock is locked
```

Figure 8: 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.

I had to create a new dedicated intention `smartDoorLock_intention_lock.js` for the window because it was not possible to use the same intention with the clock condition implemented inside the `smartDoorLock_intention.js` file.

In the final 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#13[object Object]
houseAgent>PostmanAcceptAllRequest#13 Intention started
Humidity: 88%
houseAgent>PostmanAcceptAllRequest#8 Reading received message windowGoal#14[object Object]
houseAgent                      Trying to use intention windowIntention to achieve goal windowGoal#14[object Object]
houseAgent>windowIntention#14   Intention started
bathroom_window is open
garage_pump                     Belief changed: open window_open
```

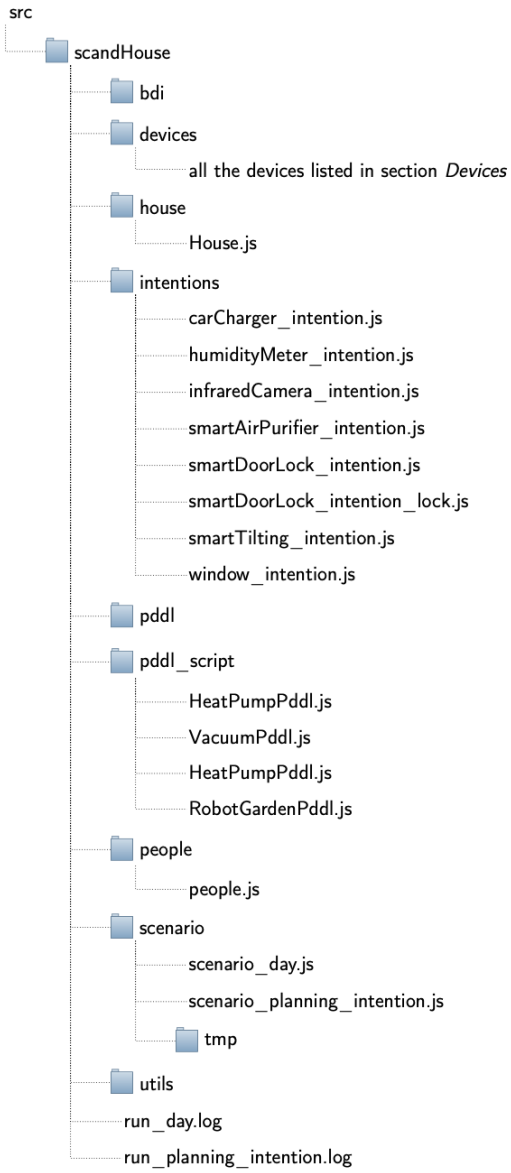
Figure 9: Log of the humidity meter

The humidity meter agent sends a message to the house agent and now the bathroom window is open.

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



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

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

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

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

Devices like the vacuum cleaner, the heat pump, the deicing system and the garden robot are extended as *Agent* class. In this way it is possible to push a planning and to use all the tools that an agent provide, for instance the belief states.

This is the link to the repository on GitHub
https://github.com/laitifranz/ASA-project-21-22

8 Improvements from the previous work

- 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.