Final Project

Autonomous Software Agents - UniTN 2021/2022

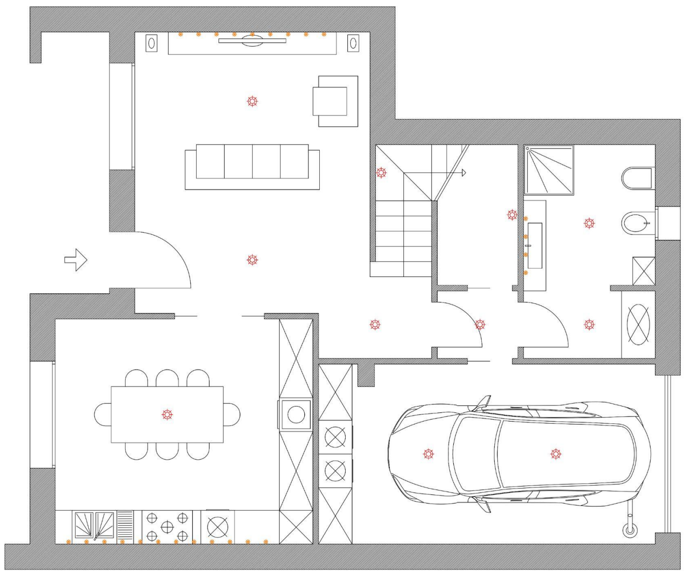
Filippo Momesso - Github Repo

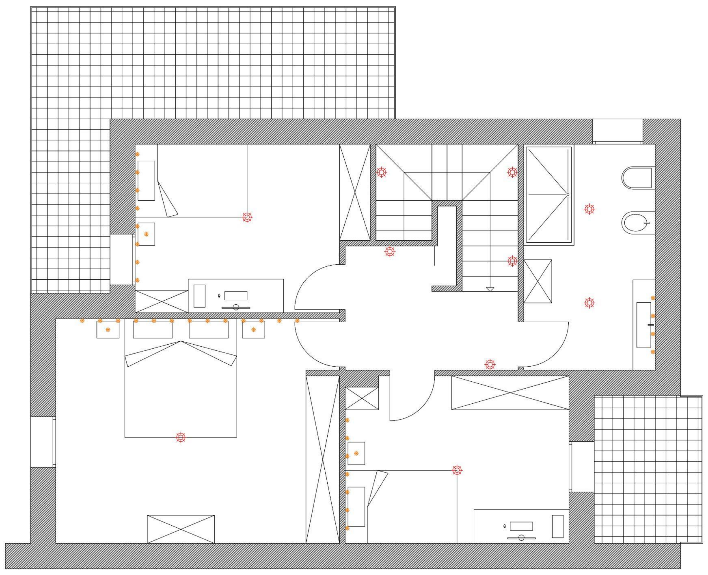
Introduction

The smart house domain I imagined combines the latest technologies and an agent oriented approach in order to help the residents in having the most convenient and comfortable home, while being in accordance with ecological standards of these days by maximizing the use of renewable energy and minimizing waste.
These goals are seeked through an agent oriented architecture which allows to automate the tedious operations in the house without the need to programmatically define all the possibilities, leaving space to adapt the behavior on the basis of the situation.
In addition to the smart devices installed in the house, which are not autonomous, I devised four agents: the house agent, the security agent, the vacuum cleaner agent and the mop-bot agent. Those will cooperate in order to fulfill the goals above mentioned.
The simulation I implemented is written in Javascript and based on the framework Autonode.js that professor Robol provided. Several scenarios and different situations have been implemented to show how my agents are able to interact with the world and the devices. Agent communication and coordination is implemented and some of the agents have planning capabilities in order to intelligently operate in a dynamic shared environment.

House description and blueprint

The house is a roughly 140 square meters terraced house located in a residential area of a medium sized town. The ideal residents are an upper-middle class family of 4 people. It is displaced on two floors for a total of 5 rooms, 2 bathrooms and a garage.
At the ground floor there is the kitchen, the living room, a guests' bathroom and the garage. From the main entrance we get into the living room, which is separated from the kitchen by a sliding door. Between the living room and the bathroom there is a small hallway which allows access to both the garage and an utility room in the space under the staircase. Note that it is not possible to go from the kitchen to the guests' bathroom without passing through the living room. All spaces are also separated by doors.
Stairs lead to the upper floor where there are three separate bedrooms, one of which is the master bedroom, and a bathroom.
All the rooms have plenty of natural light due to wide windows, especially in the kitchen and the living room.
In order to fulfill energy independence and have a low environmental footprint the house is endowed with solar panels to generate electricity, with a backup energy storage to assure the highest possible independence from the network even during night or cloudy days.





Since the main focus of the project was not to realistically simulate the smart home but to have an environment in which to develop and test some agents, not all the features described above have been implemented.
A part I wanted to model but could not due to its huge complexity and the limited time available is the one related to renewable energy, storage and consumption. I would definitely implement it in a further improvement of this project.
Moreover some assumptions have been made, such as not dealing with doors inside the house or having a single light per room.

Rooms

This section contains a detailed description of the abstract idea of the main rooms of the house. Not all appliances, lights or devices described have been actually implemented. The devices I chose to implement are described in the section "Devices".

In an actual implementation of such a system, having a single point of failure is not the best choice, therefore each room should be equipped with a wall mounted terminal which allows the resident to interact with the house domotic system, when the agents' proactive decisions and behaviors are not what the user wants. The terminal therefore works as a backup controller for the house from which it is possible to manually control lights, climate, entertainment systems and appliances.

Kitchen

The kitchen is located beside the living room, with a wide double sliding-door facing towards it. It has a large window on the main facade of the building, which allows lots of natural light to enter the room. There is a main central light fixture above the table and three more separated lights above the kitchen countertop and the stovetop. As for the household appliances there is a fridge, a dishwasher, an oven, a microwave oven and a coffee machine.

The kitchen is used by residents for cooking and eating the everyday meals. An example scenario would be breakfast where the residents in different times of the morning enter the kitchen and prepare themselves breakfast. Depending on the hour, shutters should automatically raise or light should turn on when a resident comes in and the smart coffee machine should be ready to operate.

Living room

Living room is located beside the kitchen and includes the main entrance of the house. It contains a sofa and a TV with a Hi-Fi audio system. Four different light sources are present on the walls and the ceiling, which allow for different levels of illumination from dim to bright depending on the scenario (relax time, casually watching TV, cinema-like movie watching). Similarly to the kitchen the living room has a wide window which allows lots of natural light entering the room.

An example use case scenario would be the house agent understanding that the residents want to enjoy a movie and so lower the shutters and turn on the correct light setting for the situation.

Master bedroom

The master bedroom is located upstairs, on the front side of the house, with a window facing the front yard. It has a wide wardrobe, a double bed and a dresser in front of it. On the countertop a main light and on the nightstands two smaller lights.

Garage

The garage is located on the back of the house, and has an automatic up-and-over door. It opens automatically when a paired car approaches it thanks to the action of the security agent (it has all the necessary security measures which I won't discuss and take as granted). There are two large lights on the countertop to allow the necessary illumination for the room.

Connecting the garage to the hallway between the living room and the bathroom there is a sliding door.
The garage is endowed with a wall charger which can automatically connect to the electric car charging port through a sort of robotic arm. (someone actually made a working one Tesla Automatic Charger). In this way the user does not have to deal with remembering to connect the vehicle to the charger.
In the garage there is a washing machine and a dryer.

It is used by the residents to park and charge the electric car and to store supplies and to do the laundry.

Devices

Electric car

The electric car device can be driven by a person or charged by the house agent. Charging can be fast, requiring 4.7 kW and 1 hour, or slow, requiring 2.5kW and 6 hours. Typically `fastCharge()` is used during the day and `slowCharge()` during night. When the car is parked in the garage and a Person executes the `drive()` action, the car waits for the garage door to open and then exits the garage. When approaching the garage to park the car waits for the garage door to open and then it can be parked inside.

Vacuum cleaner

The vacuum cleaner device is the physical device controlled by the vacuum cleaner agent and it is used to vacuum the floor of the house, cleaning it from dust and debris. It is assumed that it can `move()` between two adjacent rooms (even climb the stairs). It has an action to `suck()` the floor of a room and an action to `charge()` in a charging station which is located in a defined room. Charging requires one minute per battery percentage point. Sucking requires the time defined to vacuum that specific room.

Mop-bot

The mop-bot device is the physical device controlled by the mop-bot agent and it is used to deep clean the floor of the house. It is assumed that it can `move()` between two adjacent rooms (even climb the stairs). It has an action to `clean()` the floor of a room with a detergent solution and an action to `charge()` it in a charging station which is located in a defined room. Charging requires one minute per battery percentage point. Cleaning requires the time defined to clean that specific room.

Fridge

The Fridge device maintains an overall status of supply shortage. State can be `full`, `half` or `empty`. Actions that can be done include `takeFood()` and `refillFood()`. The status of the fridge is updated automatically when actions are performed. The house agents notifies the residents when fridge status becomes `half` or `empty`.

Dishwasher

The dishwasher's load state can be `full`, `half` or `empty` and working status can be `washing` or `idle`. Action that can be done is `start()`. The load state and working status is updated automatically. The house agent can start the dishwasher through the action `start`. Prerequisites are: `full AND NOT washing`.
A `loadDishes()` helper function is defined to simulate the action of loading dishes by a person.

Lights

Each room of the house has one light to provide illumination if the shutters are closed or if it is night, which we consider between 19.00 and 7.00. Lights are controlled by the house agent or manually.
For each light in the house its status is either `on` or `off`. Actions are `switchOn()` and `switchOff()`. Average consumption is 4W.

Shutters

Rooms with windows are provided with electric shutters for security purposes and for preventing unwanted light during night. Shutters are controlled by the security agent and automatically opened during the day, between 7.00 and 22.00 and closed if nobody is in the house. For each shutter the status is either `up` or `down` and actions are `moveUp()` and `moveDown()`.

Garage Door

The garage door is a simple device which can be `open` or `closed`. Related actions are defined.

Thermostat

The thermostat controls each room temperature independently. The `temperature` state is mapped into discrete integers. Action available is `setTemperature()` which allows to set a temperature for a specific room, bounded between 16 and 28 degrees. Heater or air conditioning devices are started accordingly (not implemented, just simulated).

Coffee machine

The coffee machine can be either in state on, `off` and actions available are either `switchOn()`, `switchOff()` and `makeCoffee()`. Can be controlled manually or by the house agent.

Metrics

Electricity

An overall electricity consumption count is tracked but not used to perform decisions or execute actions. A future update of the project could exploit it to make the agents behave in order to minimize consumption.

Vacuum time

Vacuum cleaner requires:

- 25 minutes for kitchen;
- 20 minutes for kitchen and garage;
- 15 minutes for master bedroom;
- 10 minutes for bedrooms and bathrooms;
- 5 minutes for hallways, stairs and utility room;

Mop time

Mop requires:

- 30 minutes for kitchen;
- 25 minutes for kitchen and garage;
- 20 minutes for master bedroom;
- 15 minutes for bedrooms and bathrooms;
- 10 minutes for hallways, stairs and utility room;

Dishwasher time and energy consumption

Both the dishwasher and the washing machine take 2h for a full cleaning cycle and have a power of 2000 W.

People and Agents

This section presents intelligent and autonomous entities in the house, including people and agents.

People

Two people live in the house: Bob and Alice. People can be in the house or out of the house.

Alice works out of the house from 9.00 to 19.00 monday-friday and goes to work by foot.
Bob goes to work by car and is out from 8.00 to 18.00 monday-friday.
Bob does the grocery shopping every other day before coming home from work.
Usually they are at home on Saturday and Sunday.
They all drink coffee for breakfast and they wake up 1 hour before they leave the house.
A Person has an `in_room` property to track which room is in, `temperatureFeeling` property to track their feeling about the temperature of the room they are in and a `wantsToHaveCoffee` property to track their desire to have coffee.

Implemented actions for each Person are:

- `moveTo()` to move from a room to another adjacent room;
- `eatBreakfast(), which` sets the desire for coffee, in order to have the house agent run the coffee machine, `takeFood()` from the fridge, eat and `loadDishes()` in the dishwasher.
- `doShopping()` which buys food and when the person is back home `refillFood()` in the fridge.
- `setCold()`, `setHot()` and `setOkTemperature()` to manifest to the house agent how they are feeling about the temperature of the room they are in.

House agent

The house agent has the task to assist the residents by making autonomous decisions in order to maximize their comfort while living in the house and making everyday activities more easy. It needs to be responsive to residents behaviors (eg. them moving in the house and needing lights to be turned on/off) and capable of interacting with other agents in the house such as the vacuum cleaner agent.
It can:

- `switchOn()` or `switchOff()` the lights when people enter/exit a room if the shutters are closed or it is night.
- Increase or decrease the temperature of a room if the person inside feels cold or hot.
- Operate the coffee machine to `makeCoffee()` if a Person `wantsToHaveCoffee`.
- Track food shortage in the fridge and notify the residents when it is half full or empty.
- `start()` the dishwasher when `full`.
- Wake up residents at their desired time by starting an alarm.
- Charge the electric car when parked, either `slowCharge()` or `fastCharge()` depending on the time of the day.

Security agent

The security agent has the task to control anti-intrusion security systems of the house such as the alarm, entrance security door, shutters, garage up-and-over door.
When nobody is at home the outside-facing doors and windows should be locked and all the shutters should be closed. The agent opens the garage door only when the person driving the car approaching the garage is an authorized person.

The security agent automatically notifies the residents (through their smartphone) and the authorities if a burglary attempt has occurred, which is when unauthorized persons enter the house.
It can:

- `moveDown()` shutters when nobody is inside the house or at night.
- `moveUp()` shutters when authorized people enter the house and it is day.
- `open()` and `close()` the garage door to let the car exit or park inside the garage.
- Notify residents and authorities when unauthorized persons enter the house (it checks if the `uuid` of the person entering is in the list of authorized persons).

Planning and device specific agents

The following devices have planning capabilities and are dedicated to the control of a single device which actually performs actions on the environment. Furthermore they can communicate between each other and the house agent in order to ask for information and fulfill goals. They have knowledge of the environment limited to their point of view and sensing capabilities, stored in a belief set of predicates.

Vacuum cleaner agent

The autonomous vacuum cleaner agent controls the vacuum cleaner device and has planning capabilities to plan how to move in the house in order to clean the rooms and go back to the charging station. The recharging base is located in the kitchen.
It has the goal to clean the house 2 times per week (just to limit the log mess, in a real scenario it would have to clean it every day).
It interacts with the house agent to ask which rooms should vacuum and which rooms have people in. Furthermore it interacts with the mop-bot agent in a master-slave pattern to order to clean the house after it has been vacuumed.
In the first deployment in the house it has to learn the topology of the house by physically exploring each room (implemented with an online-DFS algorithm).
The vacuum cleaner agents must not create discomfort for the residents, therefore it operates in a room only if nobody is in it. To perform this behavior the planning should fail if one or more people are in a room which has to be vacuumed. After failing, the agent tries to find a new plan (replanning) excluding the rooms with people inside. If for some reason when executing the `suck()` action, a person is in the room the agent is about to vacuum, the execution of the plan fails.

Mop-bot agent

The autonomous mop-bot agent controls the mop-bot device and has planning capabilities to plan how to move in the house in order to clean the rooms and go back to the charging station. The recharging base is located in the kitchen.
It interacts only with the vacuum cleaner agent to ask the status of the rooms and to accept the goal to perform the cleaning procedure. The mop-bot agent agents must not create discomfort for the residents therefore it operates in a room only if nobody is in it. The way it deals with people in rooms when cleaning is the same as described above for the vacuum cleaner agent.

Planning domain

The planning domain is common for both agents with a difference in the actions they can perform. Preconditions and effects are checked and applied on the agent's belief set. Four PDDL actions are defined:

- **`Move`**: perform the agent's device `move()`action between two rooms `r1` and `r2`.
- **`Charge`**: perform the agent's device `charge()`action.
- **`Suck`** (only for vacuum cleaner agent): perform the vacuum cleaner agent's device `suck()` action.
- **`Clean`** (only for mopbot agent): perform the mop-bot agent's device `clean()`action.

The last three actions require some time to be completed therefore are asynchronous.

Implementation

My implementation of the project is based on the framework AutoNode.js professor Robol provided us. This framework exploits Node.js event loop to simulate concurrent non-blocking operations in a shared environment where multiple agents can operate at the same time and external modifications on the environment can be performed. It implements a basic BDI architecture, where an Agent has its associated Intentions and Goals. Given a goal, the agent searches for an applicable intention in his own plan set. Different intentions could be used to achieve the same desire.
Furthermore an agent has a Beliefset to encode its fact-based partial representation of the environment.
The framework also provides planning capabilities for agents through an external PDDL planner API. So it is possible to define the planning domain made of PDDL goals and PDDL actions (which are a subclass of Intention) which is parsed into the PDDL file given to the online planner.

Sensors and agent perception

In relation to perception and environment representation I developed two kinds of approaches.
The first is based on the agent having full control over the house, the devices and the people living in the house. The Belief Set is encoded as the actual various objects and properties of the simulated environment rather than with a fact-based approach (string predicates). The house agent and the security agent are implemented this way.
Those agents, indeed, have a reference to the House object and from it they can listen to changes in Observable properties to perform their Intentions accordingly, in order to achieve the related Goal. Therefore we can see the sensor layer as embedded in the methods to access those properties.
The reason for this choice is that I started developing those agents before the lecture in which Prof. Robol presented how to deal with a fact based representation. Furthermore, since the House Agent and the Security Agent do not have planning capabilities, having a fact-based representation would have made the code a lot more complex in the functions where it is needed to listen to changes in the state of the environment, people or devices.

This approach does not explicitly model the agent's internal knowledge as partial, however, we might still consider it as partial since the agent draws only the information it actually needs for performing its tasks.

The second approach is the one I used for the Vacuum Cleaner agent and the Mop-bot Agent in which their knowledge on the environment is encoded in the Belief set object through a fact-based representation. Those agents do not directly have a reference to the house from which to draw information.
The way those agents acquire information from the environment is through communication with other agents following an Ask-Answer pattern. In particular the Vacuum Cleaner is able to ask the House Agent to send it the cleaning status of the rooms and whether someone is in a room, for each room. Analogously the Mop-bot agent communicates only with the Vacuum Cleaner agent (since they are in a master-slave relationship) and is able to ask for the topology of the house and the above mentioned status of the rooms based on the Vacuum Cleaner agent's knowledge.
The sharing of information is implemented by directly declaring/undeclaring facts into the Belief set of the receiver to avoid dealing with parsing strings.

The predicates used are:
- `in <room>` to encode the position of the agent.
- `door <room1> <room2>` to encode the adjacency of two rooms.
- `person_in_room <room>` to encode the presence of a person in a room.
- `dirty <room>` to encode that a room is dirty.
- `sucked <room>` to encode that a room has been vacuumed.
- `clean <room>` to encode that a room is clean.
- `zero_battery` to encode that the agent's device has the battery empty.
- `full_battery` to encode that the agent's device has full battery.

Agents acting in a shared environment

The agents act in the environment to achieve the goals assigned to them. AutoNode.js implementation allows for highly dynamic environments with rapid changes in the state based both on agents actions and "manual" operations, defined by a timed schedule.
As an example, we can discuss the ability of the house agent to follow people's movements in the house and switch on or off the lights accordingly. The way this is implemented is by defining a Goal and an Intention for this task. In this intention, for each person the agent observes their location in the house and then when a change occurs, the agent calls the switchOff() method of previous room light and the switchOn() method of the current (new) room light. Each person's position is tracked in parallel by exploiting Javascript Promises and further conditional logic is used to apply the above mentioned operations when needed, for example at night or if the shutters are closed. In this way the agent is able to act proactively, by acting on the environment with the goal of accommodating the user's needs for lighting.

In a similar way the Vacuum Cleaner agent acts on the environment by modifying its state by vacuuming the floor. The Goal of completing the vacuuming procedure is posted based on a defined schedule. The related intention which is made of subgoals and other concurrent intentions are executed. The agent firstly asks the house agent the house status, in order to plan which rooms to clean, then starts the vacuuming intention which executes the plan. In this intention actions (move, suck etc.) are executed on the agent’s device which physically interacts with the world modifying its state. Once the vacuum cleaning goal is achieved, the goal of charging the device is posted, in order to find a plan to go back to the charging station and charge the device. In this way, the agent interacts with the environment and the devices in it.

Agents interaction and coordination

As already mentioned briefly agents are able to interact to exchange information and to assign goals to each other. The agents provided with this feature are the house agent, the vacuum cleaner agent and the mop-bot agent.

The communication mechanism is made of two main components: the MessageDispatcher and the Postman Intention. The former implements a way to send messages in the form of Goal objects from a sender agent to a receiver agent. To do so the received messages are saved in a stack and popped when read. The latter is an Intention which listens to changes in the stack of unread messages and posts the related goal to the agent when a new message (which is a Goal) is received.
The simplest use of this mechanism is performed by the vacuum cleaner agent posting the MopCleanProcedureGoal to the mop-bot agent to make it start the deep cleaning procedure.

This mechanism allows an agent to post whichever Goal into another agent, therefore it can be used to ask and send information by defining ad hoc goals and intentions.
To do so, I developed this pattern in which the agent asking for information sends a message which contains a goal for a “SendInformation” intention which will then be executed by the queried agent.

This strategy is used by the vacuum cleaner agent which asks the house agent for the cleaning status of the rooms in order to update its belief set before planning the cleaning procedure and by the mop-bot agent which asks the vacuum cleaner agent to update its belief set about the topology of the house and the cleaning status of the rooms.

Scenarios

I implemented several scenarios to demonstrate the various features and behaviors of the agents I developed. There are different scenarios, each one focusing on a single or small subsets of features and a complete scenario which merges everything to show how the agents behave in a complex environment. Notice that some edge-case behaviors (like having the fridge empty) are purposely left as they are to show how the system and the agents deal with them. Logs of the scenarios running can be found in `scenarios` directory in the repository, I will not report them in this document to avoid unnecessarily burdening it.

Lights and shutters scenario

This scenario is meant to demonstrate how the house agent is able to switch on and off lights following people moving in the house and the security agent is able to control shutters based on the time of the day and the presence of people in the house.
To make Bob move between rooms and out of the house, a daily schedule of actions he performs is defined. At 6:30 in the morning he moves from the master bedroom, which he is initially located in, to the kitchen by following the necessary path through the rooms. This lets us see how the lights are switched on and off by the house agent, since the shutters are still down. At 7:00 the security agent automatically moves up all the shutters. Then Bob moves between rooms on the ground floor and leaves the house. Since nobody is in the house, the security agent proactively lowers all the shutters. At 18:00 Bob comes in (the shutters are moved up) and moves between rooms. All movements before 19:00 do not trigger switching on of the lights. It is after 19:00 that the house agent starts switching on and off lights based on Bob's movements and at 22 the security agent lowers the shutters for the night.

Burglar scenario

This scenario is meant to demonstrate how the security agent is able to recognize an unauthorized individual entering the house when nobody is home and therefore starting an alarm and notifying authorities.
Bob leaves the house in the morning, then at 12:00 a burglar, (a Person which is not present in the house object), enters the house. The security agent should trigger an alarm and the burglar exit the house.

Breakfast scenario

This scenario is meant to demonstrate how the morning routine of two people Bob and Alice is managed by the agents. Features included are the wake-up alarm, the coffee making, the dishwasher loading and starting and the fridge status management and notification.
Two different alarms are set-up, one for Bob and one for Alice. After the alarm is triggered, they move from their bedroom to the kitchen where they have breakfast. Having breakfast includes taking food from the fridge, eating and drinking coffee prepared proactively by the house agent with the coffee machine it controls, and then loading the dishwasher. When the dishwasher is full the house agent automatically starts it. When the fridge is half full a notification is sent to the residents smartphones. If the fridge is empty the `eatBreakfast()` routine fails.

Vacuum Cleaner scenario

This scenario is meant to show how the vacuum cleaner and the mop-bot agents work together to achieve the goal of having the house clean, by planning their actions based on their beliefs on the environment.
Initially the vacuum cleaner has to learn the topology of the house and to build its Belief set. It does so by physically exploring each room of the house with an Online-DFS algorithm (the house can be seen as a graph where each room is a node and each door is an edge).
Each day at 00:00 some random rooms in the house are set dirty and at 8:30 in the morning the cleaning procedure Intention is started by posting the related Goal. First the vacuum cleaner agent asks the house agent to update its belief set with the cleaning status of the rooms. Then a cleaning plan is seeked. It could be that the room where Bob is located is

dirty (due to randomness) and this leads to the failure by the vacuum agent to find a plan to clean the house, as explained in section “Vacuum Cleaner Agent”. By the time Bob moves out of the house, replanning is performed and the vacuum cleaner begins to operate. After vacuuming all the necessary rooms, the vacuum cleaner agent starts the Mop-bot agent cleaning procedure and finds a plan to reach the charging station to charge its device.
As already mentioned, the MopBot Agent finds a plan to clean the vacuumed rooms and then finds a plan to reach its charging station and charge its device.

Electric Car scenario

This scenario is meant to show how the electric car related tasks are performed by the agents.
In this scenario Bob uses the Car device to go drive out the house and then parks the car in the garage at different times of the day. This is meant to show how the house agent performs fast charging if the car is parked between 6 and 18, and slow charge in the complementary time slot. This scenario also shows how the security agent is able to open the garage door when Bob gets inside the car to go out and close it when the car has exited the garage. Similarly the security agent is able to open the garage door when the car is approaching the garage and Bob, who is authorized, is driving.

Complete Scenario

The complete scenario combines all the already mentioned functionalities in one weekly schedule for Bob and Alice, based on their description in section “People”. This scenario shows how all the agents are able to successfully operate in a shared environment with multiple devices and persons, by performing intentions based on their beliefs.
Without entering in the details of the schedule which is easy to read in the code, a brief description follows.
Bob and Alice wake-up to their alarms, then have breakfast as previously described and exit the house to go to work. Bob by car, Alice by foot. In the morning of each other day (to avoid too many log messages) the cleaning agents perform their procedures. Before getting home after work, every other day, Bob does the grocery shopping and, once home, refills the fridge.
When Bob arrives at home, he feels hot and the house agent sets the temperature accordingly. After some time he feels cold and the house agent adjusts the temperature again. The two residents move between rooms to finally go into their bedrooms to sleep.

Source Code Organization

Here is a link to the Github repository containing the project. The project is coded almost entirely in Javascript summing up to roughly 3800 lines of code.
The root folder contains three directories: `scenario-logs` which contains the .log files for the different scenarios, `tmp` which contains the dynamically generated .pddl files for the online planner, and `src` which contains the actual code.

The src folder contains four sub-folders: `auto-house` which contains all the files related to my implementation of the simulated environment, devices and agents, `bdi` which provides the building blocks for the BDI architecture (Agent, Beliefset, Goal, Intention classes), `pddl` which contains classes related to the planning capabilities and `utils` which provides the Clock, Observable and some other utilities. Those last three directories and the code they contain are taken from AutoNode.js framework and left pretty much unchanged. The features I needed to add and the bugs to fix can be found as pull requests by me on professor Robol Autonode.js repository.

As said, my implementation is in the `auto-house` folder which is organized in `agents`, `devices`, `scenarios`, `sensors` and `helpers` directories which are self explanatory. I organized the code following the object oriented paradigm, exploiting inheritance in order to modularize it and make it as clean as possible.
In general each agent is defined in a dedicated class which extends `Agent` provided by Autonode.js. Similarly each device is implemented in a dedicated class.
Intentions relative to managing a specific device are defined in the device's file. Agent specific intentions that do not operate on a specific device are defined in the agent's file. This does not hold for agent specific Intentions to perform communication and the cleaning/vacuuming procedures which are defined in the `AgentIntentions.js` file. This choice is a consequence of the fact that some circular dependencies were needed and Node.js does not support them. Therefore I could not define those intentions in their agent's file. Furthermore, PDDL actions and goals to model the planning domain are defined in dedicated files in the agents folder.
In the `scenarios` folder are defined the different scenarios. In each scenario the House, people and the necessary agents and devices are instantiated while intentions and goals are added to the relative agent and poste. A timed schedule is defined to perform actions and operations on the environment at defined time.
In the `helpers` folder are defined the `MessageDispatcher` and the `Postman` Intention to implement communication between agents.
In the `sensors` folder are defined some sensor intentions I ended up not using.

Conclusions

In this project I implemented a simulated version of an autonomous house based on autonomous agents and devices. I put into practice the theoretical knowledge professor Giorgini explained to us in theory classes, fixing those concepts and increasing my understanding of Agent Oriented Software Development.
As future improvements, I would like to model and implement the part related to renewable energy, storage and consumption I initially described but due to its complexity and time limitations could not do in this development iteration. Overall I can conclude that I am satisfied with the outcome of this project.