2023 - 2024

INSA rooms management

Mohammed Amine Hatibi
Abdelmajid Anka Soubaai

Département Génie Électrique et Informatique
135, Avenue de Rangueil
31077 Toulouse Cedex 4

Dépôt Git : `https://github.com/Abdel211/Insa_room.git`

Table of contents

1 Introduction

As part of the comprehensive service architecture initiative, we were tasked with conceiving and implementing a web application aimed at efficiently overseeing the management of rooms within the INSA environment. This undertaking necessitated a seamless integration of software services, sensors, and actuators into the application's framework. The sensors play a crucial role by collecting data, and the subsequent analysis of this data informs decision-making processes to determine the optimal actions to be executed through the actuators. This dynamic integration of technology ensures a sophisticated and responsive system that enhances the overall management and functionality of the INSA rooms.

2 Scenarios for Application Development

To conceptualize the development of the application, we have envisioned distinct scenarios, presented herein as user stories:

User	User Story
User 1	As a student or teacher, I need to know which room is used or not in order to be able to use it for studying or giving extra classes.
User 2	As an administrator, I want the lights to turn on or off automatically if movement is detected or not in order to save energy if no one is in the room.
User 3	As an administrator, I want the alarm to trigger if movement is detected late at night when no one is studying or having class in order to ensure safety.

Table 1: User Stories

Additional Functionalities description :

- **Student Room Reservation:** In my role as a student, I desire the ability to reserve a room to facilitate collaborative work with my classmates.
- **WiFi Management for Energy Conservation:** As an administrator, I aim to have the capability to deactivate the Wi-Fi when a room is unoccupied, contributing to electricity conservation.
- **Climate Control for Comfortable Studying:** In the capacity of an administrator, my goal is to ensure a conducive studying environment for students. To achieve this, I wish for the air-conditioning/heating system to activate when temperatures become too hot or cold, ensuring optimal comfort during study sessions.

3 Microservice Description

The project is built upon a microservices architecture implemented with Spring Boot, fostering independent collaboration among its components. Operating in a local environment (localhost), each microservice is assigned a unique port, facilitating streamlined testing on the same host. The application comprises four distinct microservices: a sensor, two actuators, and a controller designed to facilitate seamless interaction.

Sensor:

- **MovementDetection (Port: 8082)**

Actuators:

- **LightControl (Port: 8081)**
- **AlarmControl (Port: 8083)**

Controller:

- **Controller (Port: 8080)**

4 Command list

Project Name	**Command**	**Description**
Controller	`http://localhost:8080/Controller/isAutolightactivated/`	Enable checking the status of automatic light control to determine activation or deactivation
	`http://localhost:8080/Controller/setAutolightControl/`	Allow enabling or disabling automatic light control

	`http://localhost:8080/Controller/getRoomUsed/`	Allow checking whether a room is currently in use using motion detection, indicating whether the room is occupied or not
	`http://localhost:8080/Controller/isAutoAlarmActivated/`	Allow verifying the status of automatic alarm control, indicating whether it is activated or deactivated
	`http://localhost:8080/Controller/setAutoAlarmControl/`	Activation or deactivation of automatic alarm control
	`http://localhost:8080/Controller/runAuto`	Trigger the execution of automatic actions, including automatic light control, automatic alarm control, and detection of movement, providing a summary of the executed actions in the response
MovementDetection	`http://localhost:8082/MovementDetection/getDetection/`	Allow checking the current state of motion detection.
	`http://localhost:8082/MovementDetection/setDetection/`	Enable or disable motion detection
	`http://localhost:8082/MovementDetection/getNbDetection/`	Allow retrieving the number of motion detections that have occurred
	`http://localhost:8082/MovementDetection/setNbDetection/`	Set the number of motion detections in the system

LightControl	`http://localhost:8081/LightControl/isON/`	Allow checking the current state of the light
	`http://localhost:8081/LightControl/setON/`	Set the state of the light.
AlarmControl	`http://localhost:8083/AlarmControl/status`	Allows checking the current status of the alarm
	`http://localhost:8083/AlarmControl/setStatus`	Set the status of the alarm system

5 Demonstration

This section will provide a demonstration showcasing the microservices implemented in this project. We will illustrate the functionalities of automatic light control and alarm control in response to detected movement.

To simulate each microservice, users are required to run the applications of the microservices, sensors, actuators, and the controller as Java applications within the Eclipse IDE. This simulation will allow users to interact with and observe the behavior of the microservices, gaining a comprehensive understanding of their features and how they collaborate within the system.

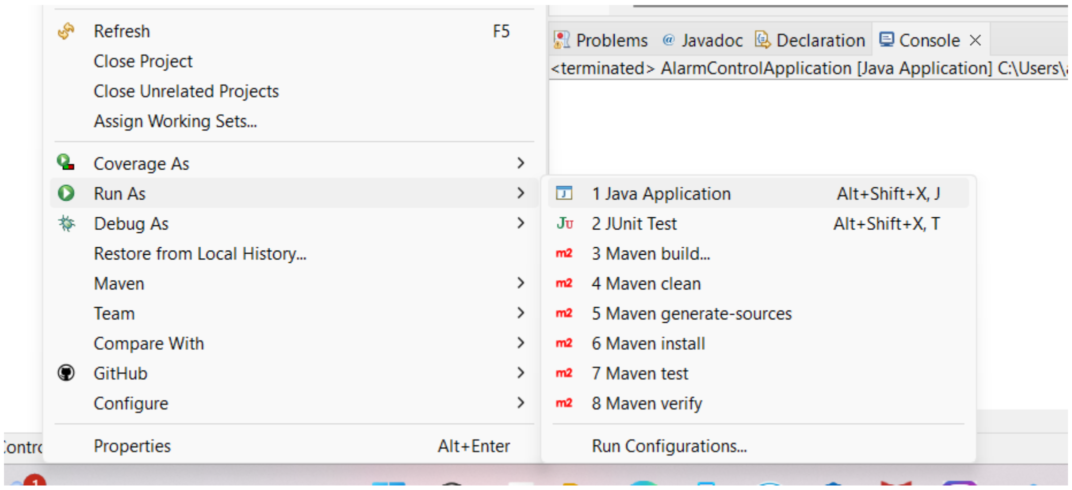


Figure 1: A Visual Guide: Running Our Application

5.1 User 1 : Room Usage

This feature is straightforward to set up and activate. It enables us to verify the occupancy status of a room through movement detection.

5.1.1 Detection of a movement

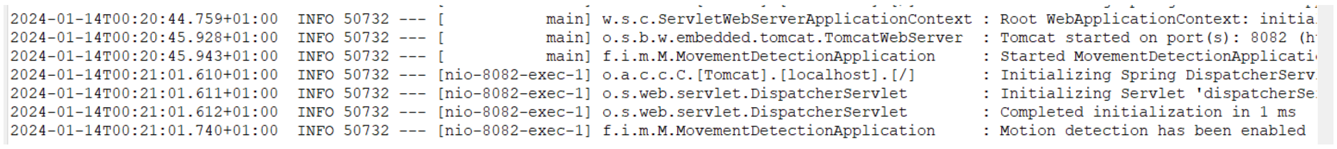
Let's begin by simulating movement:

- To simulate movement, start by accessing the following **URL:**

  `http://localhost:8082/MovementDetection/setDetection/?detection=true`

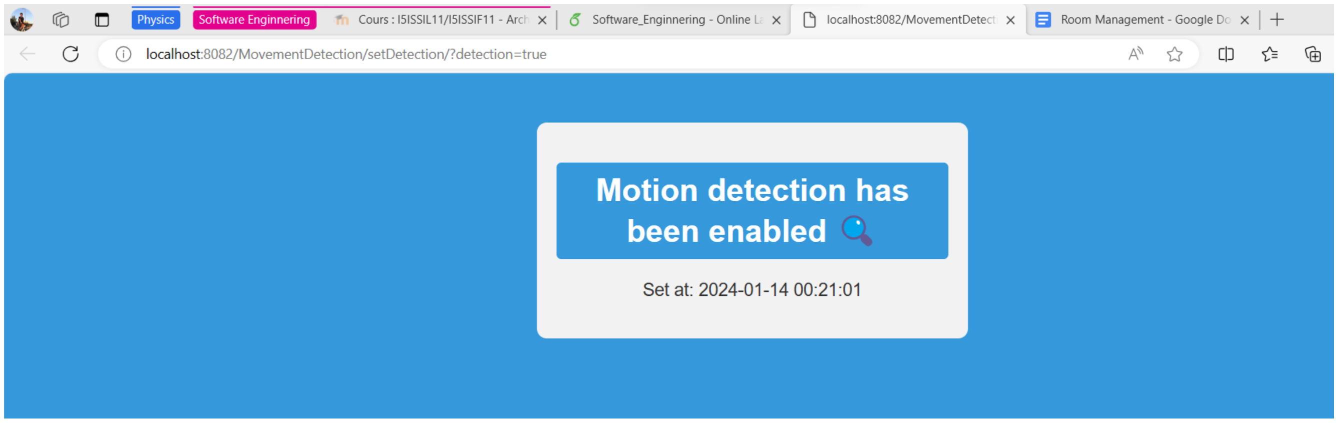
When you click on this link, you will be redirected to a page confirming that movement detection has been triggered. To verify this, inspect the logs in the Eclipse development environment, where you should see messages indicating the triggering of movement detection.

Additionally, the figures below illustrate the logs displayed in Eclipse (Figure 2) as well as the result of the URL, confirming the movement detection (Figure 3).



```
2024-01-14T00:20:44.759+01:00  INFO 50732 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initia
2024-01-14T00:20:45.928+01:00  INFO 50732 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8082 (h
2024-01-14T00:20:45.943+01:00  INFO 50732 --- [           main] f.i.m.M.MovementDetectionApplication     : Started MovementDetectionApplicati
2024-01-14T00:21:01.610+01:00  INFO 50732 --- [nio-8082-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServ
2024-01-14T00:21:01.611+01:00  INFO 50732 --- [nio-8082-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherSe
2024-01-14T00:21:01.612+01:00  INFO 50732 --- [nio-8082-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
2024-01-14T00:21:01.740+01:00  INFO 50732 --- [nio-8082-exec-1] f.i.m.M.MovementDetectionApplication     : Motion detection has been enabled
```

Figure 2: Logs in Eclipse indicating the triggering of movement detection.


Physics
Software Enginnering
Cours : I5ISSIL11/I5ISSIF11 - Arch
Software_Enginnering - Online L
localhost:8082/MovementDetect
Room Management - Google Do
localhost:8082/MovementDetection/setDetection/?detection=true

Motion detection has been enabled 🔍

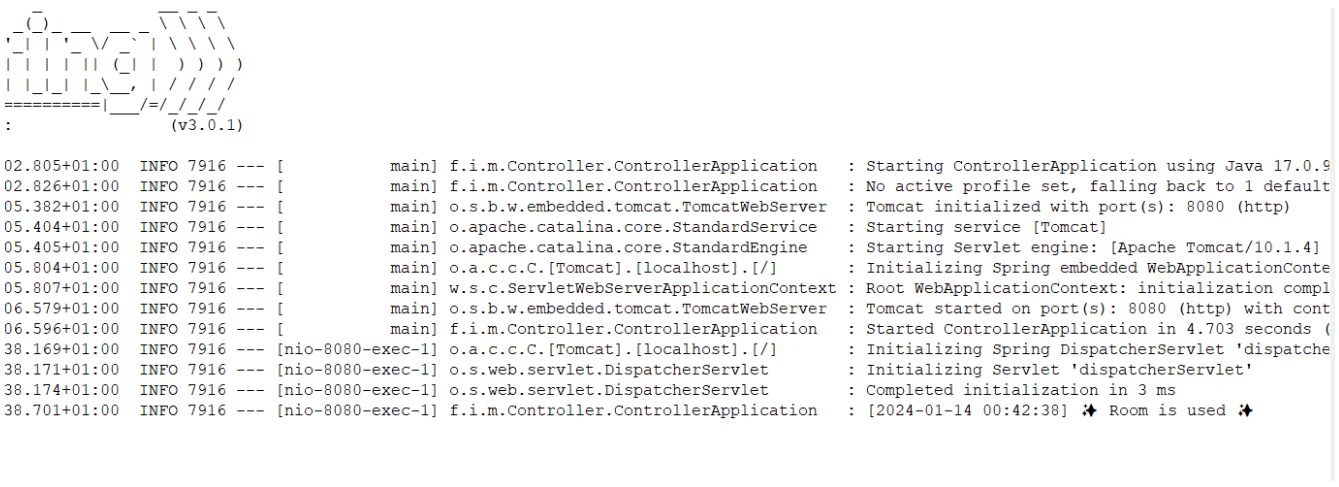
Set at: 2024-01-14 00:21:01

Figure 3: Detection of movement completed successfully.

To further inspect the state of the rooms after simulating movement, users should visit the **URL:** `http://localhost:8080/Controller/getRoomUsed/`.

The expected output from this URL should indicate that the room is in use, reflecting the simulated movement.

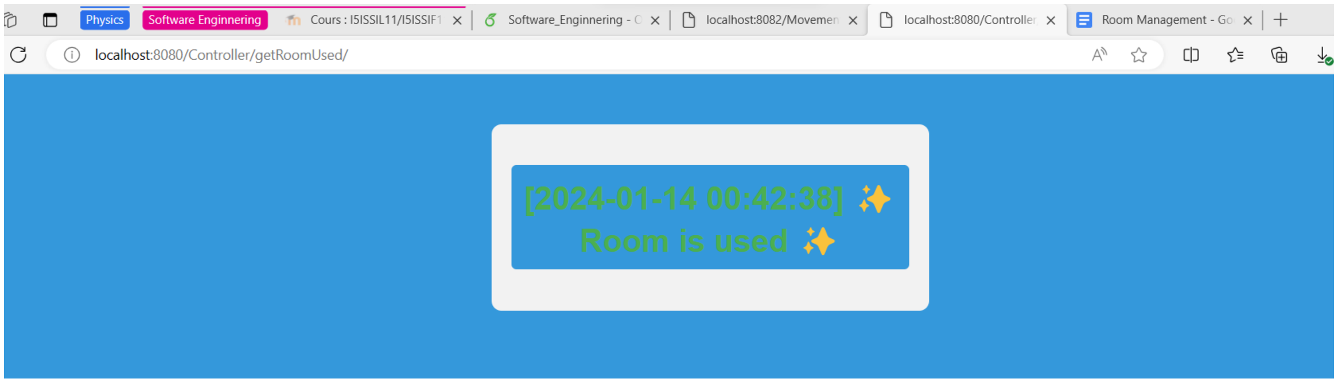
Two figures, Figure 4 and Figure 5, provide visual representations of the system's response. Figure 4 illustrates the logs in Eclipse, showcasing the status of the room after a recorded movement. Figure 5 presents the output from the specified URL, confirming the room's occupancy status.



```
 _          __ _ _
_(_)_ __  __ _ \ \ \ \
'_| | '_ \/ _` | \ \ \ \
| | | | || (_| |  ) ) ) )
| |_|_| |_\__, | / / / /
==========|___/=/_/_/_/
:                (v3.0.1)

02.805+01:00  INFO 7916 --- [           main] f.i.m.Controller.ControllerApplication   : Starting ControllerApplication using Java 17.0.9
02.826+01:00  INFO 7916 --- [           main] f.i.m.Controller.ControllerApplication   : No active profile set, falling back to 1 default
05.382+01:00  INFO 7916 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
05.404+01:00  INFO 7916 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
05.405+01:00  INFO 7916 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.4]
05.804+01:00  INFO 7916 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationConte
05.807+01:00  INFO 7916 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization compl
06.579+01:00  INFO 7916 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with cont
06.596+01:00  INFO 7916 --- [           main] f.i.m.Controller.ControllerApplication   : Started ControllerApplication in 4.703 seconds (
38.169+01:00  INFO 7916 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatche
38.171+01:00  INFO 7916 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
38.174+01:00  INFO 7916 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 3 ms
38.701+01:00  INFO 7916 --- [nio-8080-exec-1] f.i.m.Controller.ControllerApplication   : [2024-01-14 00:42:38] ✨ Room is used ✨
```

Figure 4: Logs in Eclipse displaying the status of the room.



localhost:8080/Controller/getRoomUsed/

[2024-01-14 00:42:38] ✨
Room is used ✨

Figure 5: Output from the URL indicating the current room occupancy status.

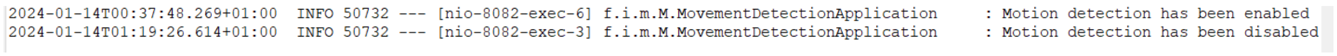
5.1.2 No movement Detected

To simulate the absence of movement, initiate the process by accessing the following **URL:**

`http://localhost:8082/MovementDetection/setDetection/?detection=false`

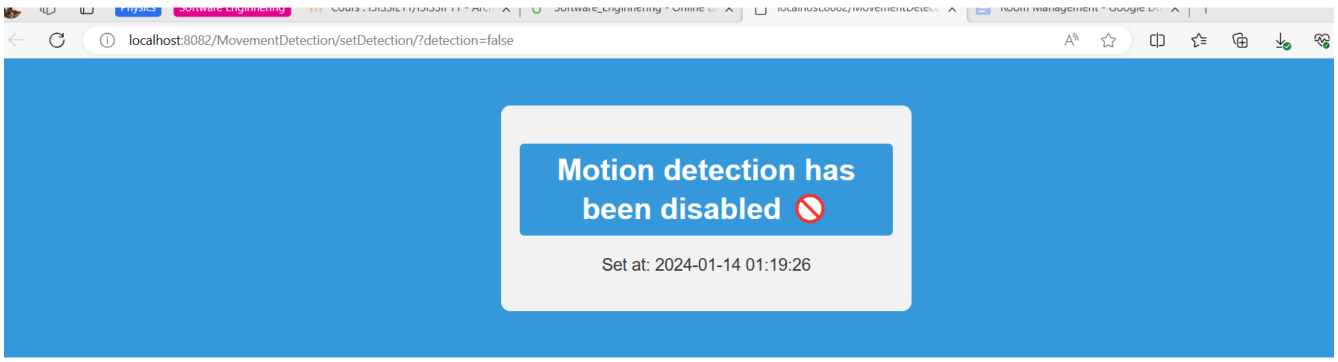
Upon clicking this link, you will be directed to a page confirming that no movement has been detected. To verify this, examine the logs in the Eclipse development environment, where you should observe messages indicating the absence of triggered movement detection.

Furthermore, the figures below illustrate the logs displayed in Eclipse (Figure 6) and the result of the URL, confirming the lack of movement detection (Figure 7).



```
2024-01-14T00:37:48.269+01:00  INFO 50732 --- [nio-8082-exec-6] f.i.m.M.MovementDetectionApplication     : Motion detection has been enabled
2024-01-14T01:19:26.614+01:00  INFO 50732 --- [nio-8082-exec-3] f.i.m.M.MovementDetectionApplication     : Motion detection has been disabled
```

Figure 6: Logs in Eclipse indicating the absence of triggered movement detection.



localhost:8082/MovementDetection/setDetection/?detection=false

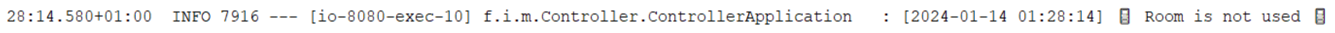
Motion detection has been disabled 🚫

Set at: 2024-01-14 01:19:26

Figure 7: Detection of movement completed unsuccessfully.

To examine the state of the rooms when there is no simulated movement, users should visit the **URL:** `http://localhost:8080/Controller/getRoomUsed/`.
The anticipated output from this URL should indicate that the room is not in use, reflecting the absence of simulated movement.

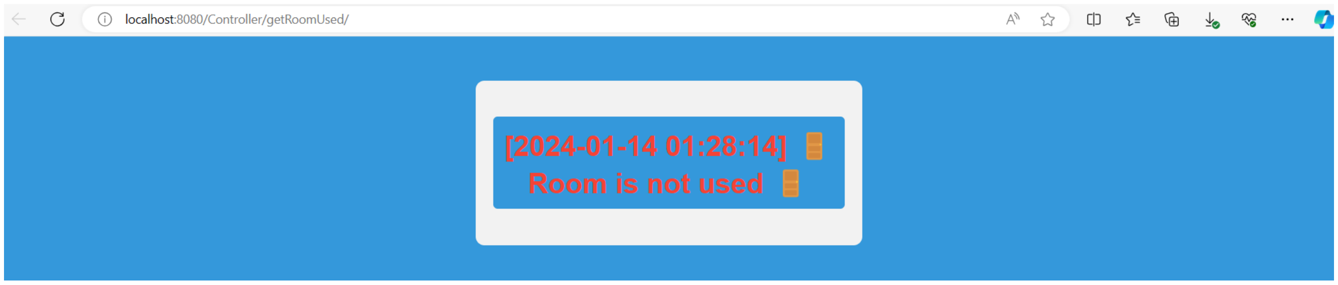
Two figures, Figure 8 and Figure 9, provide visual representations of the system's response. Figure 8 illustrates the logs in Eclipse, showcasing the status of the room in the absence of triggered movement. Figure 9 presents the output from the specified URL, confirming the room's unoccupied status.



```
28:14.580+01:00  INFO 7916 --- [io-8080-exec-10] f.i.m.Controller.ControllerApplication   : [2024-01-14 01:28:14] 🚪 Room is not used 🚪
```

Figure 8: Logs in Eclipse displaying the status of the room in the absence of movement.

[1]



[2024-01-14 01:28:14] 🚪
Room is not used 🚪

Figure 9: Output from the URL indicating the current unoccupied status of the room.

[1]Eclipse log screenshots are omitted for the other functionality for brevity. The verification process using logs has been demonstrated for the user 1.

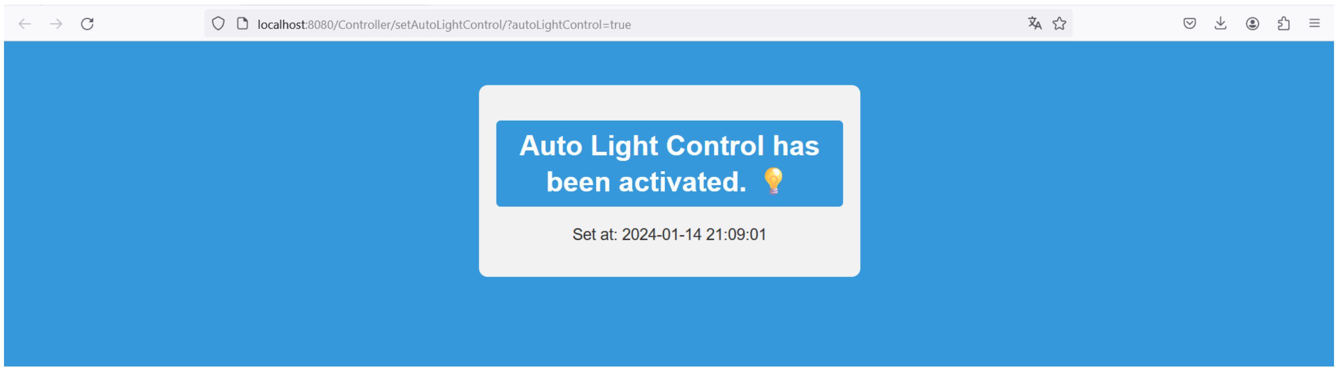
5.2 User 2: Automatic Light Control Management

To effectively utilize the automatic light control system, follow these steps:

Activation of Automatic Light Control

1. Activate the automatic light control system in the controller by accessing the following URLs:

   - `http://localhost:8080/Controller/setAutoLightControl/?autoLightControl=true`
   - `http://localhost:8080/Controller/isAutoLightControlActivated/`

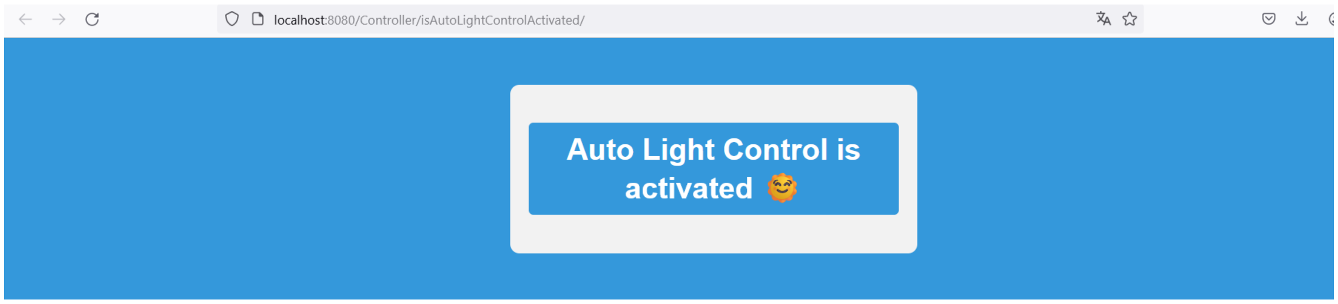


localhost:8080/Controller/setAutoLightControl/?autoLightControl=true

Auto Light Control has been activated. 💡

Set at: 2024-01-14 21:09:01

Figure 10: Activating automatic light control.



localhost:8080/Controller/isAutoLightControlActivated/

Auto Light Control is activated 😊

Figure 11: Output after activating automatic light control.

Initial Light State Check

1. Verify the initial state of the light (Assuming no movement has been detected) by accessing the URL:

    - `http://localhost:8081/LightControl/isON/`

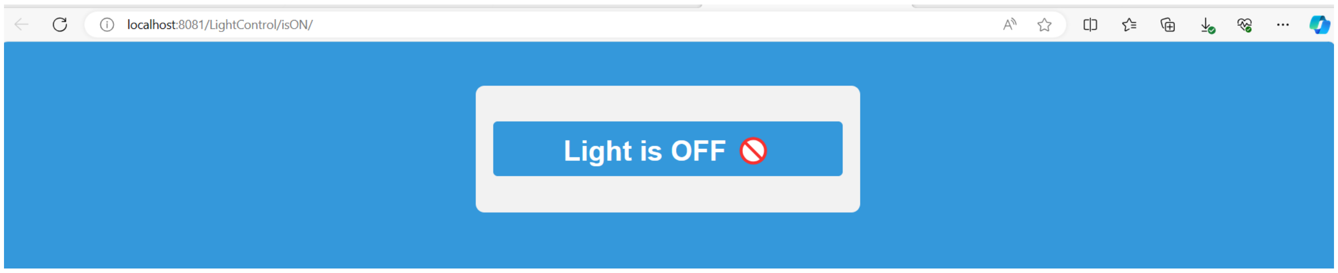


Figure 12: Initial light state check.

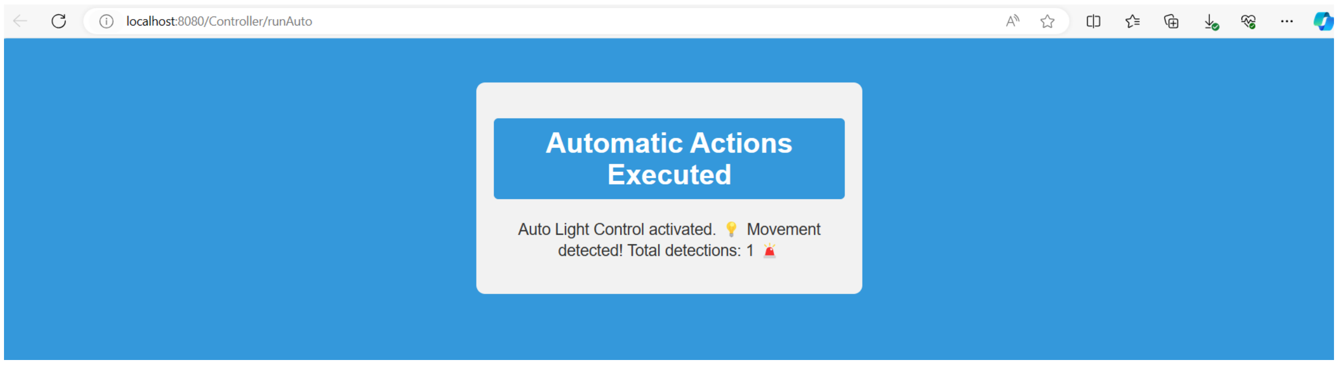
Simulating Movement Detection and Checking Light State

1. Simulate movement in the room by accessing the URL:

    - `http://localhost:8082/MovementDetection/setDetection/?detection=true`

2. Trigger the automatic actions in the controller by accessing:

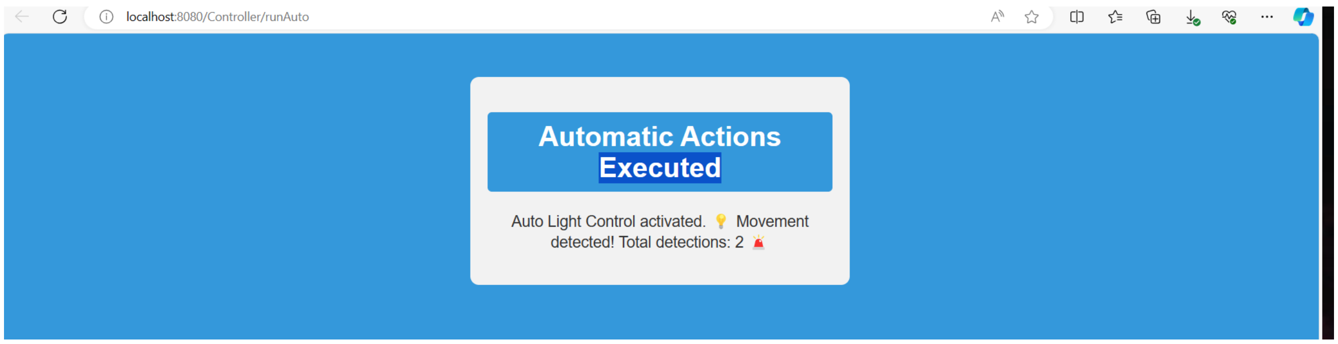
    - `http://localhost:8080/Controller/runAuto`



Auto Light Control activated. 💡 Movement detected! Total detections: 1 🚨

Figure 13: Triggering automatic actions in the controller (One detection).

Executing this URL multiple times results in an incrementation of the detection count, as demonstrated in Figure 14, which depicts two detections.


localhost:8080/Controller/runAuto

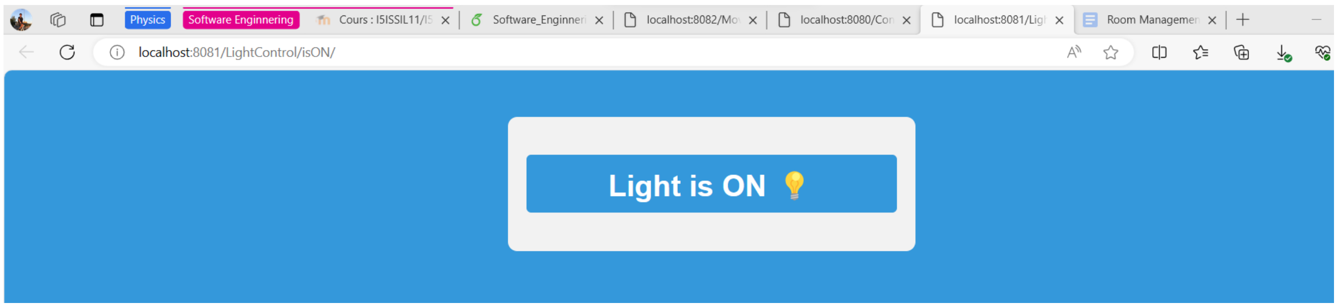
Automatic Actions Executed

Auto Light Control activated. 💡 Movement detected! Total detections: 2 🚨

Figure 14: Two detections recorded.

3. Check the updated state of the light by accessing:

   - `http://localhost:8081/LightControl/isON/`



localhost:8081/LightControl/isON/

Light is ON 💡

Figure 15: Updated state of the light after simulating movement and triggering automatic actions.

5.3 User 3: Alarm Control

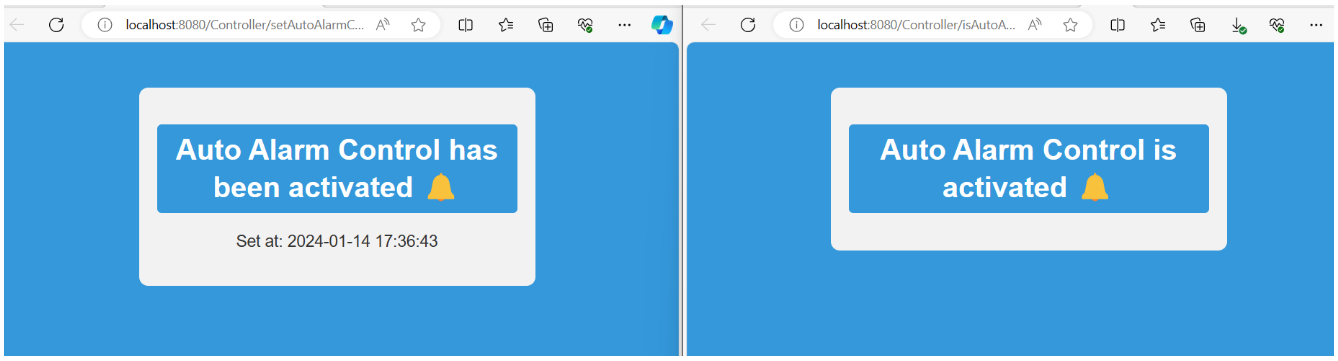
To ensure the alarm triggers only when movement is detected between 10pm and 6am, follow these steps:

5.3.1 Simulating Movement at 5pm (No Alarm Trigger)

Begin by simulating movement at 5pm, ensuring that no alarm is triggered. First, activate the automatic alarm control system in the controller:

1. `http://localhost:8080/Controller/setAutoAlarmControl/?autoAlarmControl=true`
2. `http://localhost:8080/Controller/isAutoAlarmControlActivated/`

Note: The controller code considers the local time of the computer when handling automatic alarm control activations.


localhost:8080/Controller/setAutoAlarmC...
localhost:8080/Controller/isAutoA...

Auto Alarm Control has been activated 🔔

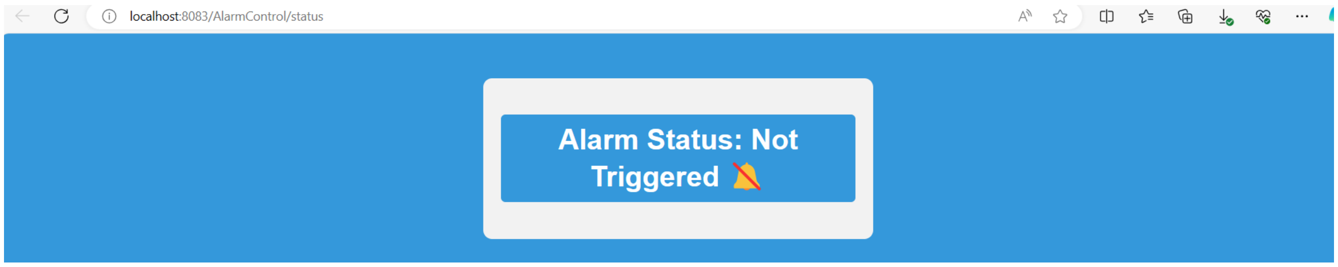
Set at: 2024-01-14 17:36:43

Auto Alarm Control is activated 🔔

Figure 16: Result of activating automatic alarm control.

Check if the alarm is currently on:

- `http://localhost:8083/AlarmControl/status`



Alarm Status: Not Triggered 🔕

Figure 17: State of the alarm at 5PM.

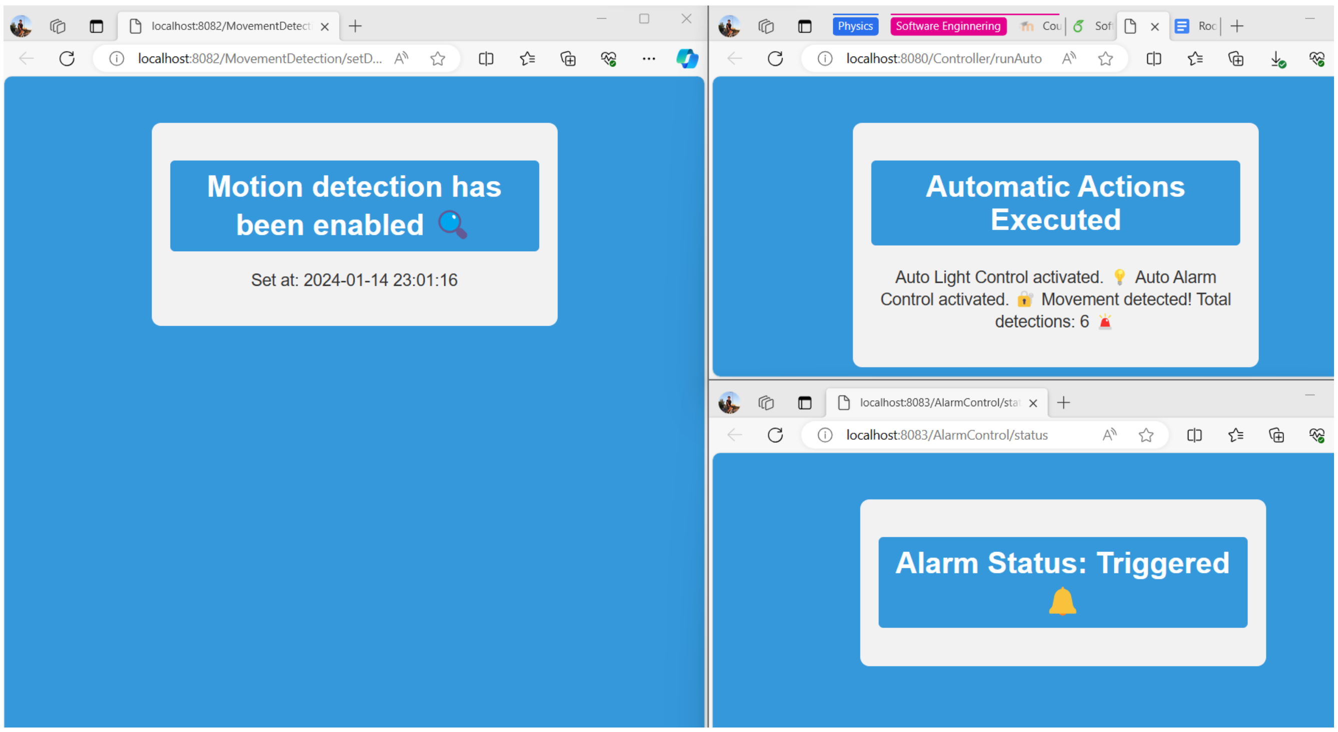
Now, simulate movement in the room and verify if the alarm is triggered with the controller:

1. `http://localhost:8082/MovementDetection/setDetection/?detection=true`
2. `http://localhost:8080/Controller/runAuto`
3. `http://localhost:8083/AlarmControl/status` Output = Alarm not triggered

5.3.2 Simulating Movement at 11pm (Alarm Triggered)

Repeat the process by simulating movement at 11pm, expecting the alarm to be triggered. Check the alarm state with the controller:

1. `http://localhost:8082/MovementDetection/setDetection/?detection=true`
2. `http://localhost:8080/Controller/runAuto`
3. `http://localhost:8083/AlarmControl/status`


localhost:8082/MovementDetection/setD...
localhost:8080/Controller/runAuto
localhost:8083/AlarmControl/status
Physics
Software Enginnering

Motion detection has been enabled 🔍

Set at: 2024-01-14 23:01:16

Automatic Actions Executed

Auto Light Control activated. 💡 Auto Alarm Control activated. 🔐 Movement detected! Total detections: 6 🚨

Alarm Status: Triggered 🔔

Figure 18: Alarm state after simulating movement at 11pm.

6 Agile Method

6.1 Room Mangement Project :

In this project, we firmly believe that adhering to an Agile methodology, such as SCRUM, is the optimal approach. By following this method, we align with the iterative cycle of conception, programming, and testing. At the conclusion of each iteration, it becomes more convenient to identify and address bugs or implement changes that have been developed, rather than deferring such actions until the final production stage, as seen in other development methods.

If this project were executed using an Agile methodology over a period of 2 quarters, we would have completed 4 iterations of 2 weeks each. We would maintain a prioritized list of objectives, a product backlog, and assign team members the roles of SCRUM Master and Product Owner.

The SCRUM Master would be tasked with ensuring the smooth execution of sprints following the SCRUM methodology. They would also be responsible for mitigating impediments and guiding other team members for continuous improvement.

The Product Owner would handle relationships with stakeholders and customers who would utilize the developed application.

At the beginning of each sprint, we would conduct a sprint planning session and utilize a free version of the JIRA software for SCRUM setup.

Given that we would not be working full time every day on this project, daily SCRUMs would not be relevant. Instead, we would conduct 20-minute SCRUM reviews every three days to monitor progress, discuss encountered difficulties, assess remaining tasks, and synchronize activities.

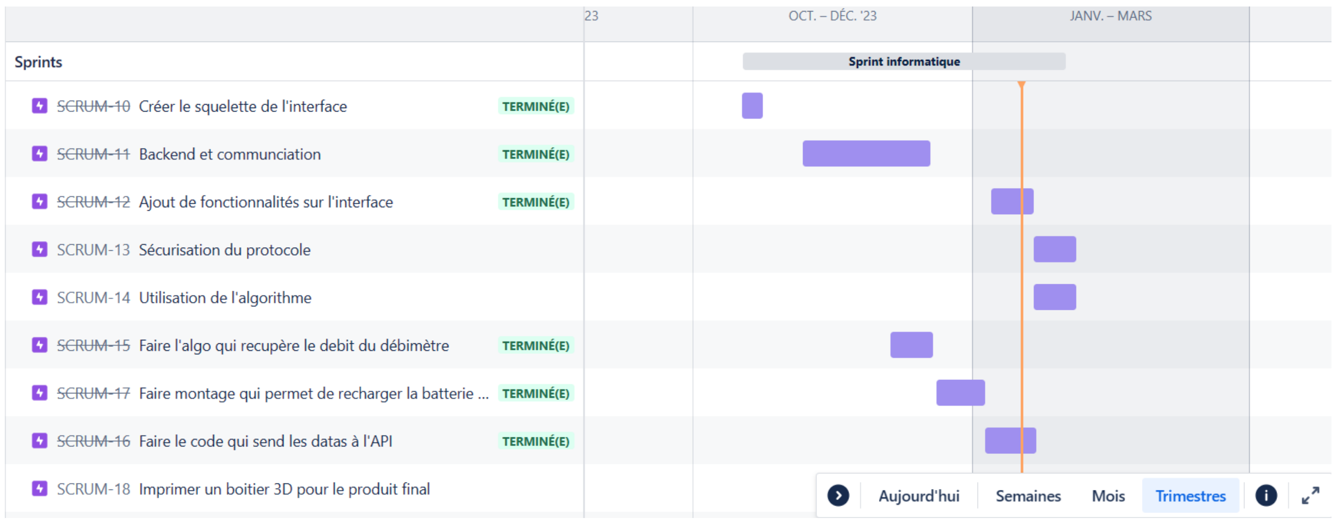
At the end of each sprint, we would conduct a thorough review to examine the work done, identify areas for improvement, and create a plan for implementing enhancements.

6.2 Water Leak Detection Project

In our water leak detection project, we adopted a Scrum approach to manage our tasks effectively. Using the JIRA software, we organized and tracked our SCRUM sprints, with our Scrum Master, who also served as our project mentor, providing essential guidance even if not present at every scrum meeting. The flexibility of our scrum cycles was noteworthy, varying from 6 days for simpler tasks to 15 days for more complex assignments, allowing us to adapt to the nature and complexity of the tasks at hand.

After each sprint, comprehensive reviews were conducted to evaluate the progress made, identifying areas for improvement and refining our strategies for subsequent sprints. Additionally, within our team, one of our pairs took on the role of the Product Owner. This individual was responsible for communicating with Mr. Gael for equipment procurement and coordinating with Mr. Garcia to address any issues encountered during the delivery process. This collaborative effort not only facilitated effective communication channels but also streamlined our workflow, ensuring a smooth interaction with external stakeholders.

Explore our project's progress on JIRA by visiting the following link:
`https://vigileak.atlassian.net/jira/software/projects/SCRUM/boards/1/timeline`



Sprints		
~~SCRUM-10~~	Créer le squelette de l'interface	TERMINÉ(E)
~~SCRUM-11~~	Backend et communciation	TERMINÉ(E)
~~SCRUM-12~~	Ajout de fonctionnalités sur l'interface	TERMINÉ(E)
SCRUM-13	Sécurisation du protocole	
SCRUM-14	Utilisation de l'algorithme	
~~SCRUM-15~~	Faire l'algo qui recupère le debit du débimètre	TERMINÉ(E)
~~SCRUM-17~~	Faire montage qui permet de recharger la batterie ...	TERMINÉ(E)
~~SCRUM-16~~	Faire le code qui send les datas à l'API	TERMINÉ(E)
SCRUM-18	Imprimer un boitier 3D pour le produit final	

Figure 19: Scrum Planning for Water Leak Detection Project

To enhance our development and deployment processes, we implemented Continuous Integration (CI) and Continuous Deployment (CD) methodologies, leveraging Jenkins as our chosen platform for automation. This allowed us to automate the building, testing, and deployment phases, ensuring a more efficient and reliable development pipeline. Explore our Jenkins setup by visiting the following link: `https://jenkins.vigileak.obrulez.fr/job/Vigileak/`

7 Conclusion

This project served as a practical application of the skills acquired during the Service Architecture course. We worked together to design and implement an architecture for an application aimed at automating the management of INSA's rooms. Each of us was assigned a distinct user story to implement and also took charge of developing the necessary microservices. [0.2cm] When navigating the intricacies of service-oriented architecture, we encountered challenges that prompted creative problem-solving and collaborative decision-making. The project not only honed our technical abilities but also enhanced our teamwork and communication skills as we synchronized efforts to achieve a cohesive and efficient solution.

Throughout the development process, we remained committed to delivering a solution that not only met the specified requirements but also demonstrated a deep understanding of the principles learned in the course. This endeavor allowed us to apply theoretical knowledge in a practical setting, reinforcing our comprehension of service architecture concepts.