# Smart Nuclear Research Safety and Monitoring System

Group 14: Sarthak Bapat, Sumit Jadhav, and Shreyash Kulkarni

Service Computing Department, IAAS, University of Stuttgart
st169660@stud.uni-stuttgart.de
st169671@stud.uni-stuttgart.de
st169744@stud.uni-stuttgart.de

**Abstract.** This project description contains, to the point summary of how Smart City and Internet of Things can be implemented in various sectors including non-residential buildings like factories, industries, and research buildings, and so on. With this description, we have designed a model which would help in safety, security, and monitoring of a Nuclear Research Building. There is a detailed summary of System Analysis and System Architecture Design of the proposed model.

**Keywords:** Smart City · Internet of Things · Nuclear Research Building.

## 1 System Introduction

Gradually over the years growth of the Internet and smart devices like mobile phones and laptops has been major. These smart devices can be connected over the Internet constantly and the whole underlying idea of Smart City rests on the notion that these devices should talk to each other and be able to change state to serve us better[1]. Developing a Smart City where the devices communicate over Internet with each other can have a significant impact on energy efficiency, comfort, safety, security, and so on. Using Smart cities and Internet of things concept we have proposed a model that would efficiently work for a Nuclear Research Building in the aspect of safety, security, and energy efficiency. As we know that Nuclear radiation can have a dangerous effect on human beings if leaked (for e.g Chernobyl disaster), all the experiments in the Nuclear Research Building are classified i.e only scientists or government employees are only given access plus energy management is needed in such big buildings so that the energy is not wasted when no one is there. Keeping in mind these aspects of the non-resdential building i.e Nuclear Research Building, the model is proposed. The main goal we are trying to touch here in the model is the safety of the people inside the building if the radiation level detected by the sensor is above a threshold safe value by opening the emergency doors, turning on the buzzer and alarm system, turning on the entire lighting of the building giving a pathway to the exit. The other goals being access control to the research building i.e authorized entry to the classified personnel only other than them no one should be allowed into the building, and displaying the details of the staff inside the building the entire time so that if there is an emergency, the people stuck in the building can be known.

## 2 System Analysis

Employees of the nuclear research office are the primary actors for our system. An employee would be granted access if he/she is a classified employee meant to access the research area, else the access would be rejected. This refers to the include relationship with the case 'verify access credentials' and extend with 'Access error' respectively. The presence of the person will be detected and the lighting system would work smartly.

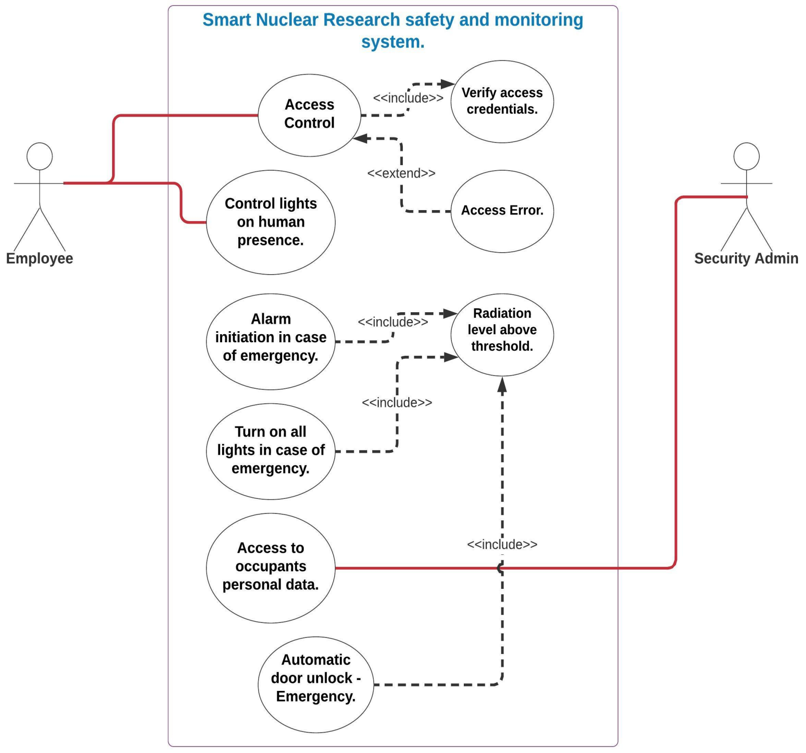
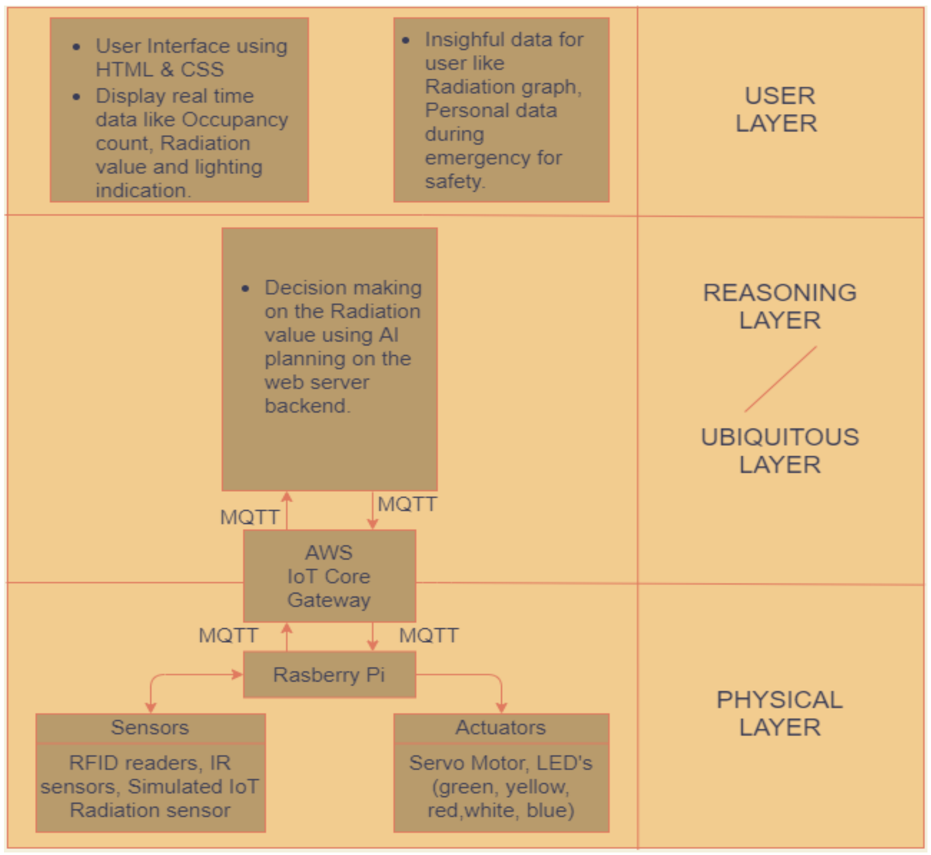

Smart Nuclear Research safety and monitoring system.
Employee
Security Admin
Access Control
Verify access credentials.
<<include>>
<<extend>>
Access Error.
Control lights on human presence.
Alarm initiation in case of emergency.
Radiation level above threshold.
<<include>>
<<include>>
Turn on all lights in case of emergency.
Access to occupants personal data.
<<include>>
Automatic door unlock - Emergency.

**Fig. 1.** System Analysis: Use case diagram for the system.

Alarm initiation, automatic door unlocks and turning on all lights in case of emergency would depend on the radiation threshold level and would only happen if the radiation is beyond a certain level. So, these use cases have a include relationship with the radiation level. Also, the personal data of the occupants of the office area would always be available for the security admin so that in case of an emergency the information could be directed to relevant people for getting the help. The security admin is a reactionary actor hence placed on the right-hand side of the system block.

## 3 System Architecture Design

The architecture of the system primarily consists of three layers. First one is the physical layer comprising of the sensors and actuators interfaced to the Rasberry Pi. The sensed data from the Raspbery Pi is sent using the MQTT protocol to the AWS IoT core which acts as a message broker. From the IoT core the data is subscribed by the Web Server which is hosted on the laptop.



• User Interface using HTML & CSS
• Display real time data like Occupancy count, Radiation value and lighting indication.
• Insighful data for user like Radiation graph, Personal data during emergency for safety.
USER LAYER
• Decision making on the Radiation value using AI planning on the web server backend.
REASONING LAYER
UBIQUITOUS LAYER
MQTT
MQTT
AWS IoT Core Gateway
MQTT
MQTT
Rasberry Pi
Sensors
RFID readers, IR sensors, Simulated IoT Radiation sensor
Actuators
Servo Motor, LED's (green, yellow, red,white, blue)
PHYSICAL LAYER

**Fig. 2.** Architecture design of the system.

In the Reasoning/Ubiquitous layer we have implemented the decision making logic using AI planning which resides on Web Server backend. This decision making logic recieves messages through subscribed topic on AWS IoT core. Depending on the outcome of the AI planning the relevant data is published back to the Rasberry Pi for the actuations.

The User layer consists of the Web Server frontend which is the Graphical User Interface(GUI). The GUI provides the insightful data for the user like graphs, personal data during the emergency and also the real time data like occupancy count, radiation value and the lighting indication. The frontend is built on HTML and CSS.

## 4 Implementation

**4.1) Hardware Implementation:**

In our system, Raspberry pi serves three main purposes. First, is collection of the data from the sensors. The second one is the transmission and reception of the data over the MQTT and the third is the actuation. We have interfaced two RFID readers, two IR sensors, five LEDs, one buzzer, one servo to the Raspberry Pi and simulated a Radiation sensor on Raspberry Pi. The RFID reader will read the unique identification number and person's information such as the name of the person, blood group, and designation from an RFID tag. IR sensors are used to detect the presence of the person in two different rooms. Five LEDs serves different purposes. Green led indicates low radiation, yellow led indicates medium radiation and red led indicates a high level of radiation. Yellow led also indicates access by an unauthorised person. White led is used as a light bulb in a main room and blue led is used as a light bulb in the storage room. Along with red led, buzzer indicates a high and medium level of radiation.

Access control operation is performed on Raspberry pi using RFID readers. Raspberry pi sends data such as timestamp, UID, person information, current radiation level, number of the occupants, the status of the light bulb in both the room to the server using MQTT protocol. Raspberry pi also receives the actuations to be performed on the LEDs, door, and buzzer during the high and mid radiation level from the server and actuates LEDs and door accordingly. We have simulated behaviour of the Radiation sensor using python because a physical radiation sensor is very costly to purchase. The output of the radiation sensor is in the form of pulses, and the unit of the radiation is CPS(count per seconds). We are generating radiation pulses randomly in three-levels, and levels are low level, medium level, and high level. Each level is assigned a probability of occurrence. Low level has the highest probability, the medium level has the lowest probability, and the high level comes with medium probability. The reason for assigning the highest probability to the low-level radiation is that system must stay in the normal condition for most of the time. Whenever a higher level of radiation hits, the system will go in the emergency condition for a short time.

We have used one RFID reader to allow the entry in the building and another RFID reader to allow the exit from the building. Both RFID readers are communicating with the Raspberry pi using SPI protocol( serial peripheral interface). RFID reader deployed at the entry of the building only allow an authorised person to enter the building by opening the door using a servo motor. When an unauthorised person tries to access the room, then the system will not allow that person to enter and yellow led will blink for some time indicating unauthorised access. The number of occupants present inside the building has been determined using both RFID readers and IR sensor. A light in the main room turns on when at least one person is present in the building. A light in the storage room turns on for a fixed amount of time assuming that this fixed amount of time is sufficient for a person to take or keep a required thing, resulting in the energy-saving and smart lighting to some extent. Lights in both the rooms turn off when occupancy count is zero.

We have used green, yellow, and red led indicator to indicate the radiation level in the building. The green indicator will turn on whenever there is a low radiation present in the building. Yellow led will turn on when mid radiation level hits. Red led will turn on, and the buzzer will start buzzing when higher-level radiation hits. Another feature we have added in the system is that during the emergency condition (High-level Radiation), lights in the first and second room will turn on, the servo motor will open the door so that people present inside the room can move out fast. We have implemented AI planning for the actuation needed during the high-level radiation condition and mid-level radiation condition. Raspberry pi sends radiation data to the AWS server using MQTT, and in server-side, we are comparing radiation data with the threshold value. If radiation data is above the threshold level of high radiation, then PDDL scripts will give us the plan for actuation. Red led ON, door OPEN, buzzer ON are the actuation given by the AI planning. Similarly, when radiation level is above the threshold of mid-level radiation, then our plan is to turn on the yellow led.

### 4.2) Dataflow in the System:

We have the data flow between the two devices in our system. The sensing and the actuation part are done using the Raspberry Pi and the core decision making logic resides on a web server which we have hosted on the laptop. There is a communication between these two devices as the sensed data from the Raspberry Pi is sent over to the server, the decision-making logic takes certain decisions on the sensed input and the resulting action data is sent back to the Pi for actuation. Here's a detailed description of the dataflow modules implemented in our system.

#### 4.2.1) Indirect Communication:

The communication between the two devices is indirect and is done using the Message Queue Telemetry Transport (MQTT) protocol. The sensed data from the Pi is published over a topic and the web server subscribes to that topic to get the messages. Similarly, the web server publishes the data arising out of the decision-making logic on another topic and the Pi subscribes to the topic, receives the messages and carries out the necessary actuations. The MQTT requires a message broker which receives the messages that are published on a certain topic and directs the messages to the subscriber which subscribes to that topic. In our system we have used the Amazon Web Services (AWS) IoT Core as the message broker. We have selected this to have our messages come through a cloud so that the system works even if the devices are not under one local network. Usage of a Cloud based platform enables the connectivity over the internet, making the system more versatile.

**4.3) Web Server and the Decision-making logic:**

We have hosted a web server that subscribes to the MQTT topic on which the Raspberry Pi publishes the sensor data. The server's backend is written in Python using the Flask framework. The Flask is a framework suitable for web development built with a light core and provides an easy routing of the html pages. The backend logic consists of getting the MQTT messages, then processing them using decision-making logic, publishing back the necessary data for actuation and rendering the data on to the appropriate html pages for visualization and data insights.

We have used Paho MQTT client in python to subscribe to and publish the MQTT messages. When the messages are received to the MQTT client, we store the messages in appropriate data structures and perform the necessary processing on them. The most important data for our decision-making logic is the radiation value that is received from the Raspberry Pi. The decision-making logic is our AI plan and is implemented using the PDDL domain and the problem files. There are two problem files in our case- one is for the mid-level of radiation value and the other is for the case where the radiation value is high. Depending on the level of radiation that we receive, we pass in the correct domain and the problem files as input to the planner and get the plan generated. Then, we parse the file that has the plan generated and depending on the plan publish the data back to the Raspberry Pi and the User Interface for the actuation. There are different levels of actuation for the mid-level and the high-level of the radiation value. The high-level is the most critical one and has a total actuation like opening the door of the building, turning on the lights in the rooms, ringing the alarm and showing the emergency status in red coloured font on the user interface which the admin monitors. We also have maintained a list of people to display on the UI in case of emergency. This is because the application requires to know the people exposed to a high level of radiation in order to track them to ensure their safety.

**4.3.1) AI Planning Algorithm:**

The input for AI planning in our system is the radiation value. Depending on the radiation value the system sends the correct input and triggers a python script Planner.py which generates the plan by giving a post request to the server (the online AI planner in this case). The inputs to this python script are the problem.pddl and the domain.pddl files for our conditions in the system. Once the plan is generated, we parse the plan file and get the necessary action items. Depending on the action items the data is published back to the Raspberry Pi and the actuation takes place thereby updating the status onto the User Interface as well.

**4.4) The Graphical User Interface:**

The graphical user interface is the client side of the web server. When we run the server.py script the server starts and the User Interface comes up on the address localhost:5000 in the browser i.e. the communication takes place on the port number 5000 of our local machine. On the main page of the UI we display the Room Occupancy count, the status of the lights in both the rooms, the radiation value received from Pi and the status of the door, alarm and lights in case of an emergency. In the emergency, the status appearing on the UI would be displayed in a red font in the emergency tab.

There are two buttons on the User Interface which provide more details on the current system state. The first button called Get Graph which is in the radiation value tab navigates to a new page that displays two charts. One is the live radiation level graph where we show 10 values at a time. That is helpful if someone wants to see the current trend of the radiation value. The other graph next to it shows the average radiation value per day. This is an insightful data because it shows the radiation pattern over some period and if this value is on the higher side then the building can have some additional safety measures in place. So, this graph would allow to figure out the trend providing some meaningful insight into the data and act accordingly if needed. For the demonstration, we have shown the average value of ten radiation values as an average value for one day.

The other button called Personal Details is on the emergency tab on the UI. Clicking that will navigate to a new tab showing the list of people currently present in the building. This is designed to provide the list of persons inside the building during the time of the emergency, so that they can be provided with good medical care if exposed to high radiation levels in the office. This can be accessed during the non-emergency times as well. So, clicking this button would provide a list of people present in the building. The data on the main UI page comes from the web server's backend implemented in Python. We have used the JavaScript's AJAX (Asynchronous JavaScript and XML) call to update the data on the web page in every 500ms. The function written in JavaScript which takes the data from the server backend and displays it on the web page UI is called after every 500ms so that we get the live data on the UI. This function is the AJAX call to the server's backend. The data into the server's backend is obtained by subscribing to the topic on which the Raspberry Pi publishes the data. After processing using the AI planning algorithm, the data is rendered onto the UI with the AJAX requests to the web server.

The User Interface styling is done with the CSS files. The features like hovering effect on the buttons and the display of the current date and the time have been added to make the IoT dashboard more attractive for the user.

**Links to Source code:**

https://github.com/sarthakbapat/Smart-Nuclear-Safety-and-Monitoring

## 5 Discussion and Conclusions

The safety aspect is paramount in the Nuclear Research Building and with our current system design we have addressed the issue using the Internet of Things. The system we have built acts as smartly after detecting high level of radiation and does appropriate actuations for the safety of the employees. We have achieved energy efficiency using smart lighting upto a certain extent.

## 6 Future Scope

Further extension of this project can be to maintain a proper Database to record all the system activities and use image processing to have access control in the building.

## References

1. Paul, C., Ganesh, A., Sunitha, C.: An overview of iot based smart homes. In: 2018 2nd International Conference on Inventive Systems and Control (ICISC). pp. 43–46. IEEE (2018).
2. Smart Cities and IOT Practical videos and slides.