

Smart Cafe

Group 18: Zamik Shahid, Faisal Khan, and Usman Mirza

Service Computing Department, IAAS, University of Stuttgart
firstname.lastname@uni-stuttgart.de

1 System Introduction

1.1 Background Information

Popularity of Cafes has increased in recent years. Cafes have limited seating arrangements and people sit in them for longer duration. The customer visiting times for them are unusual and not like restaurants, as restaurants only get people during lunch and dinner time only.

1.2 Problem Statement

Traditionally, the lights in a cafe are turned on always, same is the case with ventilation or heating systems. It leads to wastage of energy resources. A person has to constantly count the number of free spaces available and update the inlet sign at the door accordingly, this takes a time. Also, the quantity of staff has to be increased to cater for these needs, as some has to count the customers, some has to turn on/off the lights, and some has to adjust the temperature. So the management of human and energy resources gets difficult and majority of the resources are wasted usually, which increases running cost. Likewise, the cafe manager has to manage all the persons and tasks. Furthermore, the manager or owner cannot get real time notifications of the things that are happening around in the cafe.

1.3 Project Goals

This project aims to embed the concept of smart cities and Internet of things into a cafe, this would be realized by inclusion of smart sensors and actuators in the building. It would automate major processes of the cafe without or limited involved of humans and help the manager in saving human and energy resources easily. Further, it would add some extra features that would increase the ambiance of the cafe and proper management by using real time status of the different sensors installed in the premises.

2 System Analysis

2.1 Functional System Requirements

User Story: Smart Cafe

As a: Cafe manager

I want to: know if the customer is present in the restaurant at the moment or not

So that: I do not appoint a person to keep the record of number of customers visit each day

As a: Cafe manager

I want to: Know the temperature of the cafe and notifications of the cooling system

So that: the smart cafe system can intelligently control cooling to save energy and maintain comfortable environment for our customers.

As a: Cafe manager

I want to: let the smart cafe system control the lights depending upon need

So that I: Can save energy and electricity bills

As a: Cafe manager

I want to: Know which customer needs assistance

So that: I hire fewer waiters to find out who's calling from which table

2.2 Non-Functional System Requirements

As a: Cafe manager

I want to: Satisfy my customers with the quality of service and ambiance even though I will be using smart devices and IOTs.

So that: I can set the mood and ambience of the place to comfort customers

2.3 Acceptable Criteria

Given: that the cafe is opened

- Electricity should be there
- Smart IOT system should be switched ON
- Internet should be working
- User has dashboard for monitoring

When: The Cafe manager request for the sensor's information

Then: Ensure that the user gets the latest information at that time

2.4 Mandatory Requirements

- If a customer enters into the cafe, the light must be turned ON during night time only
- If the temperature is high, the fan must be turned ON. Otherwise, it should remain OFF
- The cafe should have enough seating so that the customers can sit, the occupancy level should be correct

2.5 Desirable Requirements

- If a customer enters into the cafe, the notification that the light is turned ON can be delayed
- The waiter should reach the customer as soon as the button is pressed
- Customers get satisfied with the overall experience of the smart cafe

3 System Architecture Design

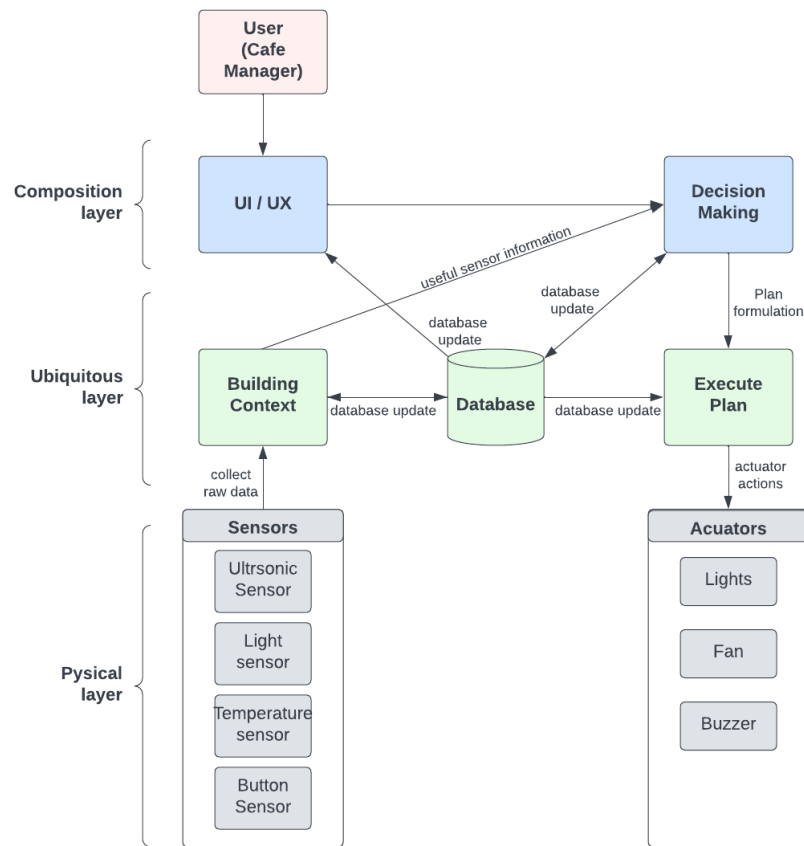


Fig. 1. System Architecture Design of Smart Cafe

4 System Implementation

In this section the system implementation of smart cafe will be explained. The project is divided in hardware and software part. The software part is further divided into back-end implementation and front-end implementation.

4.1 Hardware

The hardware consists of following components:

4.1.1 Raspberry Pi

In this project the Raspberry Pi 3 Model B+ has been used. It is running Raspbian For Robots operating system produced by Dexter Industries. Grove Pi+ board is used as a shield board which directly fits onto the Raspberry pi as given in the figure [fig 2] below.

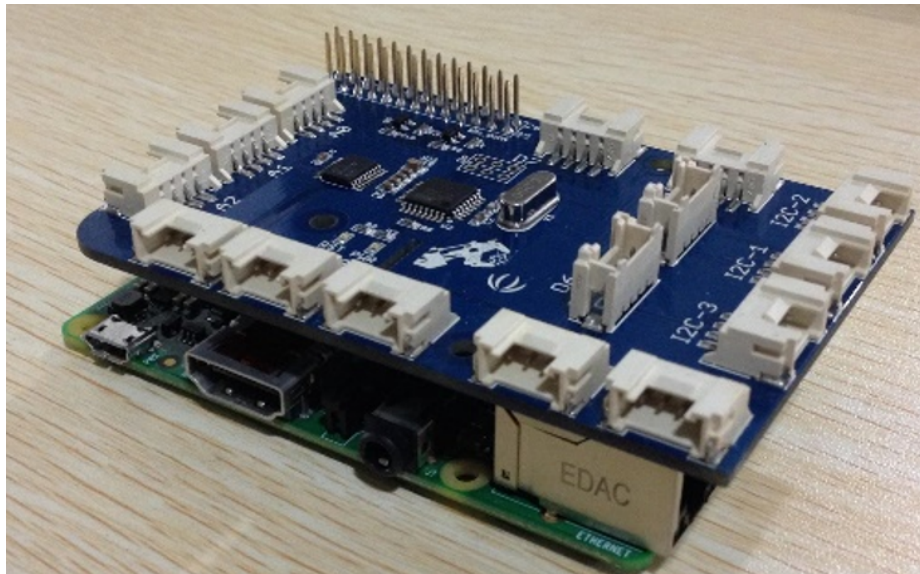


Fig. 2. Grove Pi+ board on Raspberry Pi

4.1.2 Grove Pi+ Board

The Grove Pi is an electronic auxiliary/shield board for the Raspberry Pi. Many different hardware modules such as sensors and actuators can be connected with it. It has its own library for integration with Raspberry Pi board. For getting the values from sensors and actuating the actuators it has built-in APIs.

4.1.3 Grove sensors

Four Grove sensors are used this project. These sensors are directly connected to Grove Pi+ board. The following table 1 contains the complete list of Grove sensors along with their respective pin numbers and types.

Grove Sensors	Type	Grove board pins
Temperature sensor	Digital	D6
Light sensor	Digital	A0
Ultrasonic sensor	Digital	D8
Button	Digital	D7

Table 1. List of sensors

The following table 2 shows the placement of sensors in the smart cafe system

Grove Sensors	Placement
Temperature sensor	Inside Cafe room
Light sensor	Outside Cafe in open air
Ultrasonic sensor	Placed near the table
Button	On each table in the cafe

Table 2. List of sensors placements in smart cafe

The Temperature sensor is placed inside the cafe. It has 2 states either High or Low. High means its hot while Low means its cold. Light sensor is placed outside the cafe to detect day time and night which will eventually control the lights to save the energy. Ultrasonic sensor is used to detect if the cafe is empty or not. It is placed next to the table which detects if customer is there or not. Every table also contains a button which is used to call the waiter conveniently without calling.

4.1.4 Grove Actuators The Grove LED, Grove buzzer and grove relay are used to represent state of the actuators. The table 3 below shows the pin assignment on Grove board and what each actuator represents in this project.

The actuators will be presented using Grove LED, Relay and Buzzer. The following tables (table 4 and 5) represents their corresponding pin numbers on Grove Pi board and what does their states represents in reality

4.2 Software

Back-end Implementation

Actuators in smart cafe	Actuator represented by
Cafe empty or not	LED1
Cafe lightning	LED2
Air conditioning/Fan	Relay
Calling waiter	Buzzer

Table 3. List of sensors placements in smart cafe

Grove Actuators	Type	Grove board pins
LED1	Digital	D5
LED2	Digital	D3
Relay	Digital	D2
Buzzer	Digital	D4

Table 4. List of Actuators and their pins

The back-end is implemented on 2 machines that are Raspberry Pi and Azure virtual machine. The physical layer and ubiquitous layer are implemented on the Raspberry Pi. The composition layer is implemented on Azure virtual machine. The composition layer also includes user interface which is also implemented on Azure virtual machine and would be discussed in next chapter front-end implementation.

The figure 3 below shows complete block diagram of the project Smart Cafe.

4.2.1 Raspberry Pi

Raspberry Pi is running Python version 3.5. It contains 2 python scripts:

1. Sense and Publish
2. Subscribe and Actuate

Grove Actuators	On state	Off state
LED1	Cafe not empty	Cafe empty
LED2	Lights On	Lights Off
Relay	Fan On	Fan Off
Buzzer	Calling Waiter	Not Calling Waiter

Table 5. List of Actuators and their states

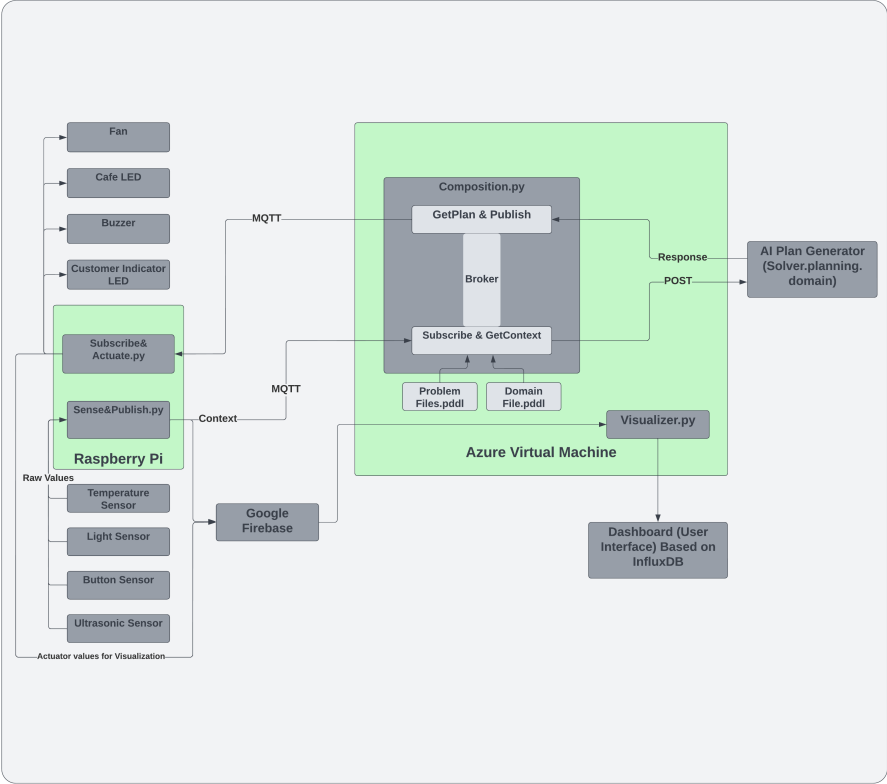


Fig. 3. Smart cafe block diagram

Sense and publish

This python script reads the raw data of sensors on physical layer using the Grove Pi board. After receiving, the raw data is processed to take more contextual information which is then published to the broker (Azure virtual machine). The raw values from the physical layer are converted into abstract values (contextual information) in this script of the ubiquitous layer.

For example, in this project, if the value of the temperature exceeds 30-degree Celsius, it is considered that the cafe is Hot otherwise the cafe is Cold. Another example is that the ultrasonic sensor which is placed on the table facing the chair on which customer sits. If the value of the sensor is below 25cm, it means that the customer is sitting on the chair.

The table 6 below shows the final states after the sensors data of sensors is processed for contextual information in ubiquitous layer.

Grove Sensors	States/Context
Temperature sensor	Hot/Cold
Light sensor	Day/Night
Ultrasonic sensor	Chair is Empty/ Not empty
Button	Waiter called/ No waiter called

Table 6. sensor states

Subscribe and Actuate

This python script has subscribed for the topics which contains plan to actuate desired actuators to achieve goal. The table 7 below shows the topics and its messages.

Topics	Messages
Plan/daynight	turn ON lights at night/ turn OFF lights at night/turn ON lights in day/ turn OFF lights in day
Plan/hotcold	Turn ON fan/ turn OFF fan
Plan/waiter	Turn ON bell/ turn OFF bell

Table 7. MQTT Topics

For instance, a new plan updates in the topic “Plan/hotcold” that asks to turn ON the fan, the script will get the new message and turn ON the fan.

4.2.2 Azure Virtual machine

This machine contains Composition.py script. This script gets the contextual information of sensors from the Raspberry Pi through MQTT protocol, executes the AI planning and then publish back the plan. Apart from this script, the virtual machine also contains Domain file and Problem files which are needed by AI planner to come up with the plan to achieve desired goal.

The table 8 below shows the topics and their messages in the form of context that are subscribed by Composition.py script.

Topics	Messages
data/daynight	Night and cafe not empty/ night and empty/ day and not empty/ day and empty
data/hotcold	Hot/Cold
data/waiter	Button pressed/ Button not pressed

Table 8. MQTT Topics

Every message contains its own problem file. When the topic is updated, its specific problem file is sent to AI planner along with the domain file. The table 9 below shows the list of problem files that are mapped to the messages.

Problem files (.pddl)	Messages
Hot	Hot
Cold	Cold
NightWithPerson	Night and cafe not empty
NightWithoutPerson	Night and Empty
DayWithPerson	Day and not Empty
DayWithoutPerson	Day and Empty
Pressed	Button pressed
NotPressed	Button not pressed

Table 9. Problem files

We are using Fast Forward AI planner. The domain and problem file are posted to the address <http://solver.planning.domains/solve> in json format. The AI planner applies the algorithm and returns the plan in response. This plan is then published on its respective topic.

4.2.3 Indirect MQTT communication

The MQTT communication is performed between Raspberry Pi and Azure virtual machine. The library used for MQTT communication is “Paho-MQTT”. Both the machines have a publisher and a subscriber. The Azure virtual machine is acting as a broker, and the Raspberry Pi as the client. The communication is indirect as the Raspberry Pi publish the sensors contextual information to the broker and Azure virtual machine subscribed to the topic “Data” gets the information, performs AI planning and publish back the plan. The Raspberry Pi subscribed to the topic “Plan” receives it from the broker and control the actuators to achieve goal for that specific plan(s).

4.2.4 AI planning

The purpose of AI planner is to give a plan that would achieve desired goals which leads to increase in efficiency and save more energy thus reducing the electricity bills and making the environment more friendly.

AI planner focused on the following tasks in this project smart cafe: 1. Changing the state of the room lights when someone enters in the morning. 2. Keeping the temperature of the room in a desired range. 3. Checking if the cafe is empty or not 4. Calling a waiter if someone wants to order

Artificial intelligence planning is implemented to determine the required actions to change the state of the room smart cafe. The domain and problem files are in PDDL.

Case 1. The temperature Inside the room is too hot

This problem file is called by the AI planner when the following conditions are met:

The temperature inside the cafe is above 30 degrees Celsius

The actions that follow:

Actions	Precondition	Goal
Turn on Fan	High temperature, fan off	Lower the Temperature, Fan On

Table 10.

Case 2. The temperature Inside the room is too Cold

This problem file is called by the AI planner when the following conditions are met:

The temperature inside the cafe is below 30 degrees Celsius

The actions that follow:

Actions	Precondition	Goal
Turn off Fan	Low temperature, fan on, fan off	Increase the Temperature, Fan Off On

Table 11.

Case 3. Customer wants to call the waiter This problem file is called by the AI planner when the following conditions are met:

The button on the table is pressed by the customer

The actions that follow:

Actions	Precondition	Goal
Turn on bell	Button pressed, Bell off	Button not pressed, Turn on the bell On

Table 12.

Case 4. The Cafe is open at day time This problem file is called by the AI planner when the following conditions are met:

The cafe is not empty there is day outside. Lights are On

The actions that follow:

Actions	Precondition	Goal
Turn off lights	Day, Lights On	Day time, Turn off all the lights

Table 13.

Case 5. The Cafe is open at the night This problem file is called by the AI planner when the following conditions are met:

The cafe is not empty there is night outside. Lights are off

The actions that follow:

Actions	Precondition	Goal
Turn on the lights	Cafe not empty, Lights are off	Cafe not empty, Turn on all the lights

Table 14.

Case 6. The Cafe is closed at night

This problem file is called by the AI planner when the following conditions are met:

The cafe is empty Lights are On

The actions that follow:

Actions	Precondition	Goal
Turn off lights	cafe is empty, Lights On	Turn off all the lights the lights

Table 15.

Front-end implementation

The front-end is implemented using Influx DB dashboard. In the Raspberry Pi, sensors raw data is filtered out in more meaningful context contains information such as current temperature, light status, button state, number of people in the cafe. This sensor contextual information along with the status of actuators is then uploaded on Google Firebase which is a Real time database. There is a python script on Azure virtual machine which maps data on Firebase to Influx DB dashboard for virtualization.

The reason for using Google Firebase is because the Influx DB do not support python version 3.5 which is used by Raspberry Pi. We are bound to use version 3.5.3 on raspberry Pi as it is the maximum version supported by the Grove Pi board library.

The dashboard can be accessed by Address: <http://markar.faisalwork.net:8086/signin>. The login page given in figure 4 will appear.

The user will need to login to have access to the measurements. The login credentials are as follow:

Username: `iot`
Password: `iot12345`

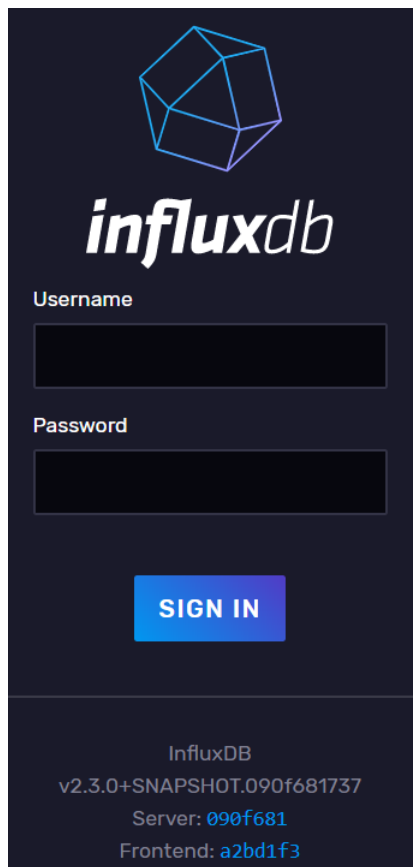
The image shows the InfluxDB login page. At the top, there is a blue wireframe cube icon. Below it, the word "influxdb" is written in a white, lowercase, sans-serif font. Underneath the logo, the word "Username" is displayed in a small, white, uppercase font, followed by a white rectangular input field. Below this, the word "Password" is displayed in a small, white, uppercase font, followed by another white rectangular input field. A blue rectangular button with the text "SIGN IN" in white, uppercase letters is positioned below the password field. At the bottom of the page, the text "InfluxDB" is shown in a small, white, uppercase font, followed by "v2.3.0+SNAPSHOT.090f681737" in a smaller, white, uppercase font. Below this, the text "Server: 090f681" is shown in a small, white, uppercase font, and finally, "Frontend: a2bd1f3" is shown in a small, white, uppercase font.

Fig. 4. Dashboard Login page

After login the user will see the last value measured, the measurement history based on a chosen date and the FF Planner output. The figure 5 shows the dashboard user interface of smart cafe.

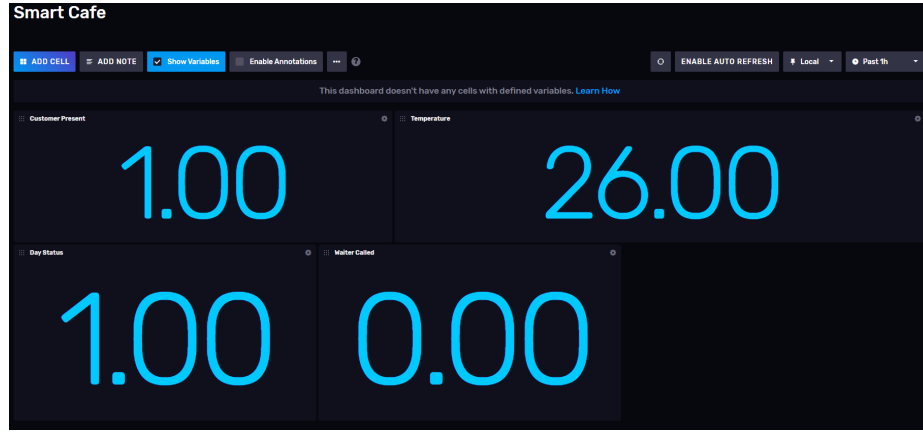


Fig. 5. Dashboard GUI

5 Discussion and Conclusions

The project Smart cafe represents a small prototype of actual smart cafe which can have multiple sensors, actuators and devices that communicate indirectly. This project specifically demonstrates all the requirements of IOT such as system distribution, which includes; raspberry Pi, Azure Virtual machine, Google Firebase and Influx DB dashboard. Indirect communication is implemented using MQTT protocol between them. AI planning is performed using domain file and problem file. Fast forward AI planner computed the plan which contains sequence of action read by the Raspberry Pi and actuators are controlled. Visualization is also performed on Influx DB dashboard.

Some difficulties were faced due to limited resources, for example, only one device was available, that could have easily increased by adding more devices into MQTT indirect communication. The project could be extended using one device as controller and multiples nodes in each cafe collecting data and using MQTT for the communication among the devices. Also the limited number of sensors was an issue, but an extension of this project would be increasing the number and kind of sensors to have a better control of the environment. For example, a motion sensor to decide if there are people in the room, that alone is not the best approach because the person could not be moving and sitting on a chair when he's inside the cafe. For this project it was assumed that the sensor is in front of the table The best approach would be using motion sensor in combination with a pressure sensor on the chair and then it would be more accurate to identify people in the room.

Even if the project had limited resources, we could show that based on the

proposed scenarios the AI planning was able to find solutions to our problem files which saves energy and increase efficiency in the smart cafe by controlling lights and temperature.

6 Access and Code

Access to the front-end:

<http://markar.faisalwork.net:8086/signin>

Username: iot

Password: iot12345

Access to all the back-end codes:

<https://github.com/usman747/IOTproject>

7 References

PDDL Online Editor Plan Generator, url = <http://editor.planning.domains/>,

The Eclipse Paho MQTT, url = <https://www.eclipse.org/paho/>,

GrovePi, url = <https://www.dexterindustries.com/GrovePi>,

Fast Forward-FF, url = <https://fai.cs.uni-saarland.de/hoffmann/ff.html>,