Project Report: Smart Cities and Internet of Things

Group No. 7: Daniel Ryan Degutis and Tung Dinh

Service Computing Department, IAAS, University of Stuttgart

Abstract. This document is the final report submitted as a requirement of the Smart City and Internet of Things lab, in which a smart office control is modeled.
This project leverages a Raspberry Pi, a Grove Pi sensor/actuator set and an Arduino Uno to implement the smart building control. It monitors, analyzes and consequently automatically controls the building.

Keywords: Smart Building · Internet of Things · Raspberry Pi. · Grove Pi · Arduino · PDDL · RabbitMQ · MongoDB

1 Introduction

With the emergence of IoT in every corner of the modern life, the communication between devices plays an important role in transforming a traditional building to a smart building, which can operate stably standalone and interact with human and environment. According to Green Building Council Australia (2013), a smart commercial building consumes 66% less electricity and produces 62% fewer greenhouse gas than a traditional one.

The heart of a smart building is data, which is harvested via sensors and end devices, then transferred to the solution by utilizing devices to satisfy specific constraints. The goal of this project is to design and implement a smart office using a Raspberry Pi manipulating a lighting system, air conditioner, and a door via a relays with the information from Grove Pi sensors. The model shall optimize energy efficiency and ensure the safety, privacy, comfort of its occupants.

This document is the final report of this project including the architecture and the requirement specifications, which can be viewed deeply into technologies and functions of the system.

2 Requirement Specifications

The system is programmed to perform specific 3 modes during a typical working day with the cooperation of sensors and actuators. The modes can be switched manually by admin only or will be automatically scheduled according to current time.

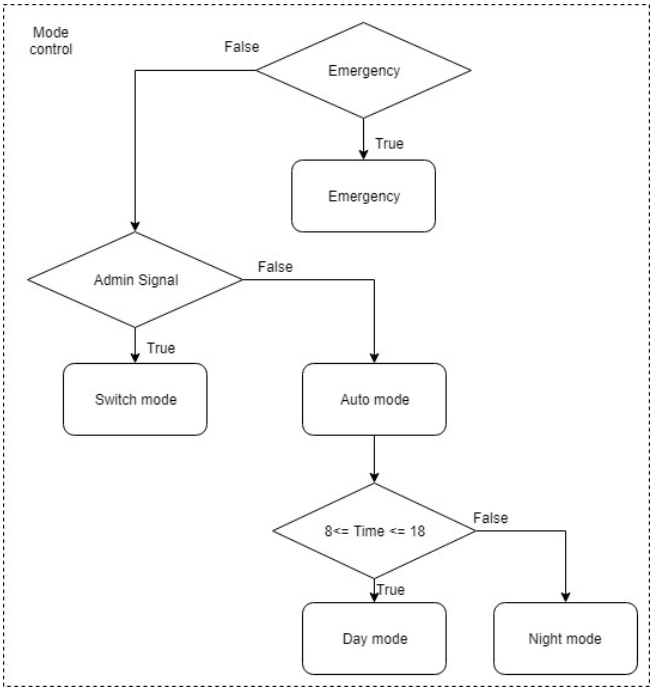

Fig. 1. The devices controller overview.

Working mode: from 8am to 18pm, lighting system, air conditioner and door are triggered to perform pre-defined tasks. Light should turn on when people inside and the room is too dark, door should open when one enter or leave the room, and air conditioner should keep the office at 23°C.

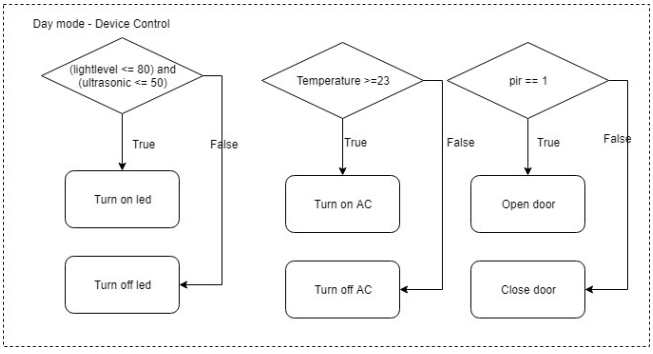

Fig. 2. Day mode.

Night mode: from 18pm to 8am, light and air conditioner are off, while door is open if there are still people inside. The door should not open for someone entering room.

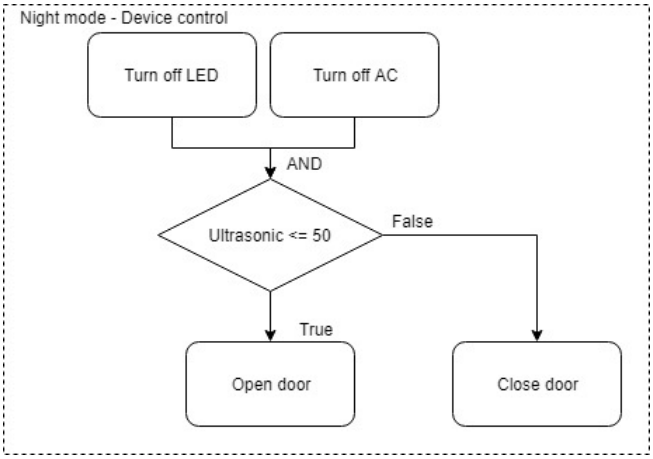

Fig. 3. Night mode.

Emergency mode: could be activated during the whole day via a button or web service, and can be deactivated only by admin of web service. Light and air conditioner are turned off while door is kept open and a buzzer rings. After the emergency case, the office reverts to its previous mode

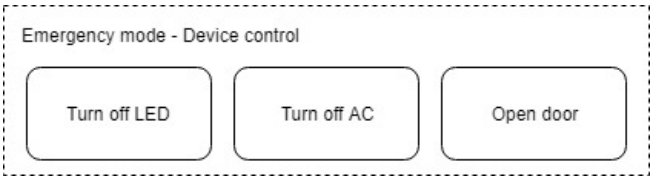

Fig. 4. Emergency mode.

3 Hardware specification

A Raspberry Pi serves as primary platform, performing both client and server side, with the environment data collected from Grove Pi sensor kit. In addition, an Arduino Uno R3 is connected via serial USB to the Raspberry Pi, fetching data from two ultrasonic sensors in replacement of pir sensor due to low functional performance , while the Raspberry Pi collects data for temperature, light intensity and distance, as well as receives commands for AC, LED and door.

4 Software specification

For precise information, including techs, packages and implementing manual, please refer to README.md on github.

5 System architecture

The system architecture has multiple components:

- Client and sensor/actuator hardware
- Decision making unit and pddl_actions_dispatcher
- RabbitMQ
- Frontend

The architecture overview shows how each component is connected (Fig **??**). Two ultrasonic sensors are addressed from the Arduino, which forwards their

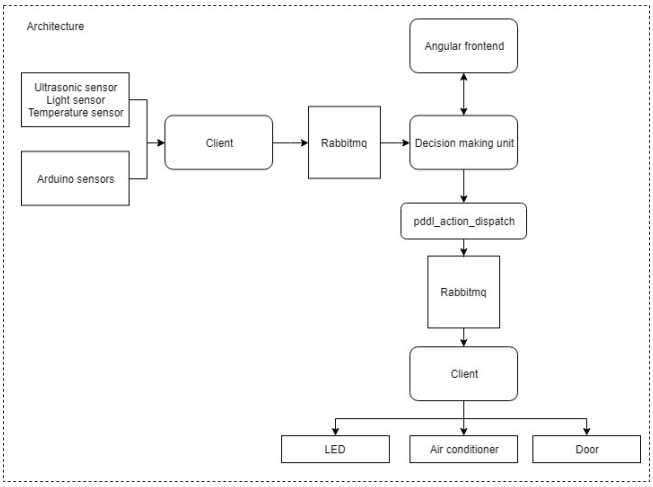

Fig. 5. The architecture overview.

readouts to the Raspberry Pi over a serial USB connection. All other sensors and actuators are hooked up to the Raspberry Pi over the Grove Pi.

The client sends all sensor values to the decision making unit via RabbitMQ.

The decision making unit, based on the current operating mode and the sensor values, decides which actions to take and the pddl_actions_dispatcher sends the appropriate commands via RabbitMQ to the client, and thus actuators.

The sensor values can be displayed in the frontend and one can override the operating mode from the frontend as well.

The decision making unit (Fig **??**) is further split up:

- server
- MongoDB
- Rest API
- pddl_planner and ff planner
- pddl_actions_dispatcher (this is the same component as in Fig **??**, and shows where the decision making unit connects to the rest of the system.

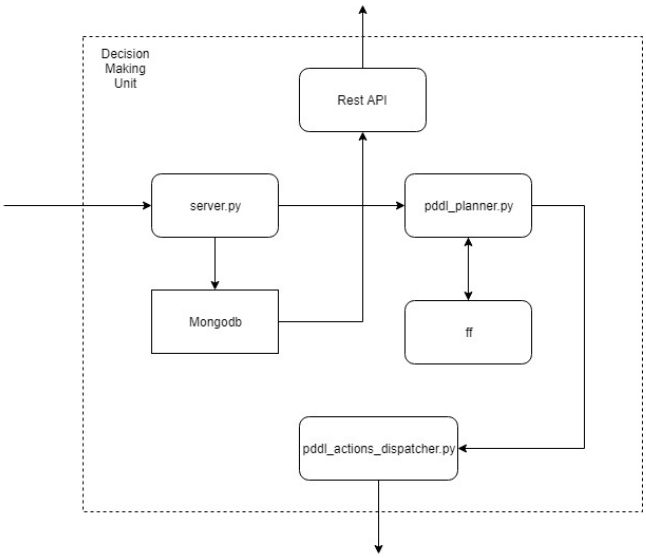

Fig. 6. The decision making unit architecture overview.

The system architecture was originally intended to have separate clients for each sensor and actuator, to simulate that they could be running on different edge devices (Fig. **??**). However, due to difficulties with multiple threads accessing the Grove Pi at the same time, this idea was discarded.

6 Modification

1. Architecture of client and queue: because of the limitation in performance of Raspberry Pi 3B+, the initial idea was consequently simplified. The initial model can work on Pi, it can however last for at most 1 minute, in which, every sensor and actuator are single clients, and all communicate via Rabbitmg.

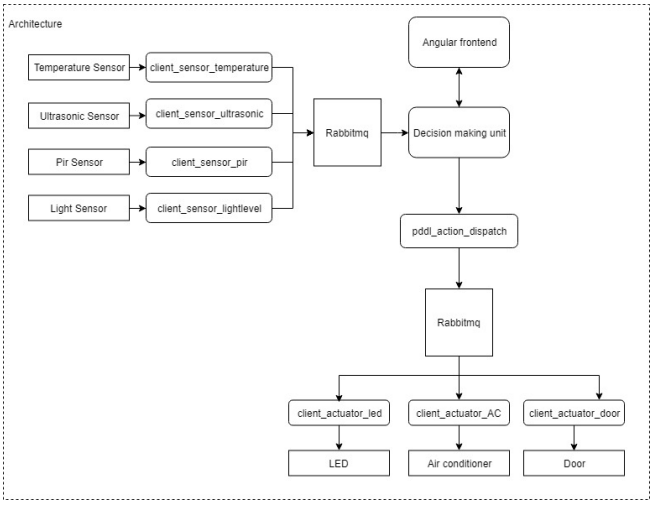

Fig. 7. The initial architecture overview.

2. Three modes, instead of four: they are working, night and emergency modes; with current relays, only 2 types signal can be output, which are ON and OFF, the idea of power/intensity adjustment is hence eliminated.
3. Forecast temperature: in reality, it is highly recommended to use, because of energy saving. However, with hardware restraints in relays , the use of forecast temperature to adjust the power of air conditioner is hence unnecessary. It is now kept to run the whole working period, to provide the pre-defined temperature of 23°C and should not be turned on and off repeatedly, since it will require 2 hours in average to recover to the desired temperature.

7 Future improvement

- Forecast temperature: could be extracted from weatherbit.io, store in database 3 hours in advance in order to prepare the working environment, run by HVAC and save energy accordingly.
- HVAC: could be exploited for smart office model in Germany, which requires heating and cooling features, in this small context nevertheless, only air conditioner is sufficient.
- Hierarchy planner: PDDL hinders the potential of the system in mode and state alternation; the FD besides is not functioning properly with unexpected error without clue.
- People counting: could be taken into account in bigger model by counting number of people connected to office wifi.
- Relay shield should be replaced by separate relays for the ease of use and working stability

8 Conclusions

The project overall runs well without any major problems, it covers the basic functions of a smart office. With the application of recommended toolkit and tech in slides, it satisfies the requirements of this assignment. As a reason of simple and small model, this project still has scaling limitation.