Design Document

version 0.0.2

Chaojie Wang

Bumsik Kim

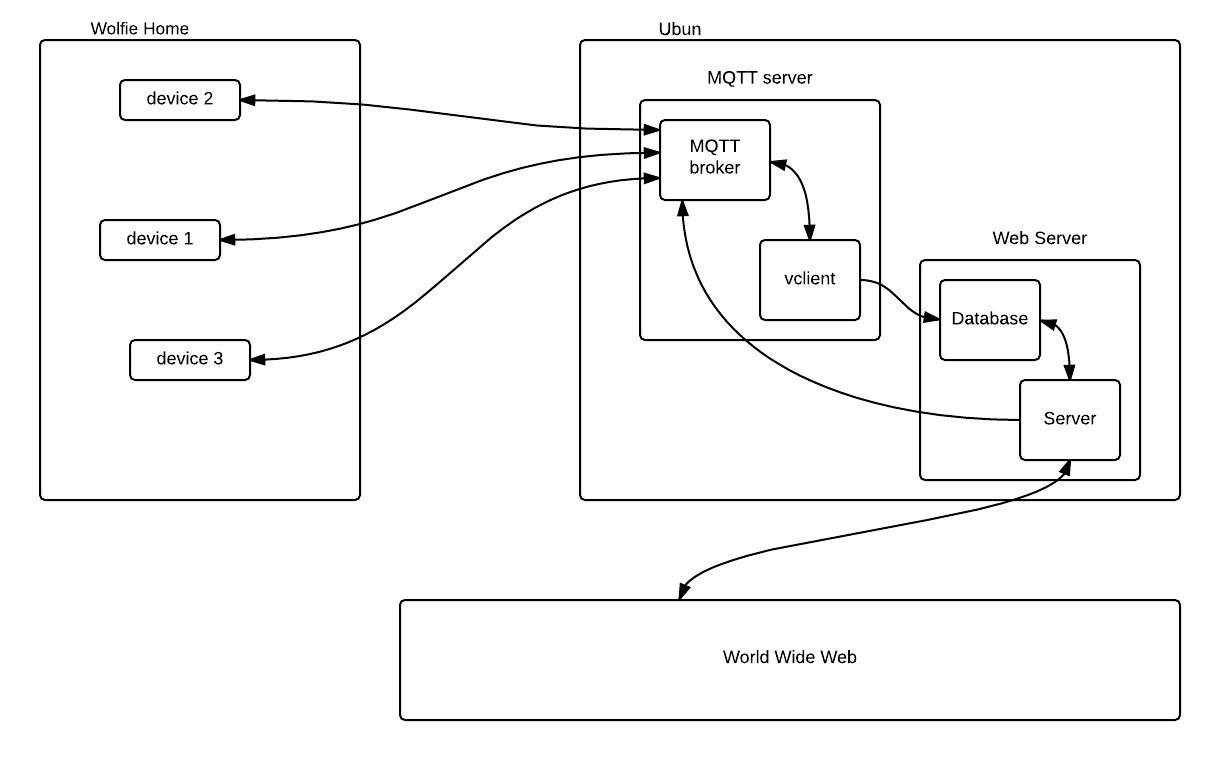
1. **Purpose**

This document is a technical design document. This document specifies components developers need to implement and how pieces work together. This should be used by developers as a reference for implementing Wolfie Home.

1. **Overview**

There are three components in this project: microcontrollers that are able to connect to the Internet, a MQTT server, and a webserver with a database.

Microcontrollers communicate among each other with the MQTT server. Inside MQTT server, there are two components: one is MQTT broker, and the other one is called vclient. The MQTT broker handles the inter-communications among microcontrollers and vclient. The vclient is the virtual client that communicates with database and the webserver. The webserver communicates with users and the MQTT server. All communications are done with JSON format. We will use [JSON schema](http://json-schema.org/documentation.html) to specify the content of the data. They are stored in schema/ folder.

The following topology illustrates the idea:

1. **Microcontrollers**

**3.1 types of devices**

In the table blow, “Type” is used for topics of message.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Component | Data type | controllable | Module type | Description |
| 1 | Mag | 0=closed,  1=open |  | Security module | Magnet reed sensor, used to detect if a door is open or not |
| 2 | PIR | 1=present,  0=absent |  | Security module | PIR motion detector. Once it detected it will not updated for maximum 30 sec |
| 3 | Lgt | Integer |  | Security module | Ambient light sensor |
| 4 | Hum | Integer |  | Environment sensing module | Humidity Sensor. |
| 5 | Tmp | Integer |  | Environment sensing module | Contactless infrared temperature sensor |
| 6 | Bar | Integer |  | Environment sensing module | Barometric Pressure sensor |
| 7 | Led | 0 = led off,  1 = led on | O | Speaker & Led moudle | Color LED on/off |
| 8 | Ledr | Integer  (max 255) | O | Speaker & led module | Color Led Red color  The module also sends data when this value updated by LED preset color |
| 9 | Ledg | Integer  (max 255) | O | Speaker & led module | Color led Green color  The module also sends data when this value updated by LED preset color |
| 10 | Ledb | Integer  (max 255) | O | Speaker & led module | Color led blue color  The module also sends data when this value updated by LED preset color |
| 11 | Ledp | # of preset color,  0 = the current color is none of preset | O | Speaker & led module | Number more than 0 indicates pre-defined color (e.g., 2 = RED, 3 = Violet....) |
| 12 | Spkr | 0 = off,  Else = preset of bells | O | Speaker & led module | Speaker. Number more than 0 indicate pre-defined music stored in the SD card (1.wav, 2.wav...) |
| 13 | Btyp | 0 = 1s lipo,  1 = coin cell,  2 = 2\*AA |  | Battery module | Kinds of battery |
| 14 | Bat | Integer  0 to 100 |  | Battery module | Percentage of remaining battery |
| 15 | PB1 | 1 = on,  0 = off |  | Wireless module |  |
| 16 | PB2 | 1 = on,  0 = off |  | Wireless module |  |
| 17 | PB3 | 1 = on,  0 = off |  | Wireless module |  |

1. **MQTT Server**

Both the web server and the MQTTserver are running on a Linux computer.

**4.1 MQTT broker**

The MQTT broker is implemented according to MQTT protocol. There is an open source MQTT broker called Moquitto, which is used in this project.

**4.2 Communication among device modules**

for both obtaining data and control device modules, we use following topic format: (house)/(uid)/(module). The role of this topic is just addressing modules. If all fields are specified, the addressing is unique.

|  |  |
| --- | --- |
| field | description |
| house | house name, i.e. “jim\_house”. |
| uid | uid, unique identifier of a microcontroller. |
| module | one of module type in section 3.1, i.e. “wireless\_module”. |

The complexity of communication lies in the payload. The payload is JSON formatted. The schema of the JSON format is specified in payload.json file in the current directory.  
**4.3 Obtaining house information**

We need a topic for obtaining general information about the house. This help a new module makes sense of the existing system and for webserver know the house better. The topic format is Info/(house):

**4.4 vclient**

Moquitto also provides library for writing MQTT client. It is used for writing vclient. vclient is short for virtual client. vclient is responsible for obtaining all status data from device modules and store them into the database. See 5.1 for specific data that we need to save.

**5 Web Server**

**5.1 MySQL**

The database vendor is MySQL. Use account *chaojie*, and password *dummypass* to login. The MySQL has a database called *wolfie\_home*. *wolfie\_home* has two table: *users* and *house*.

There is a *users* table. It stores information about users and their house information.

|  |  |  |
| --- | --- | --- |
| COLUMN NAME | SQL TYPE | DESCRIPTION |
| uid | INT | Unique identifier of the user |
| username | VARCHAR(128) | Username |
| password | VARCHAR(128) | Password |
| email | VARCHAR(128) | Email |
| house\_info | JSON | House information. |

Each user has a *house* table. The table stores status of all device modules in a user’s house. Data stores in a *house* table are generated by vclient. house\_info is a JSON of house\_info.json schema.

|  |  |  |
| --- | --- | --- |
| COLUMN NAME | SQL TYPE | DESCRIPTION |
| house | VARCHAR(128) | House name, it is unique. |
| uid | VARCHAR(128) | Unique identifier of the device. |
| mod\_name | VARCHAR(128) | Module name. one of predefined module name. |
| mod\_content | JSON | Content of a device module. One of the device modules content. |
| Time | TIMESTAMP | Time the value was recorded. |

**5.2 Server**

When a user log in, the web server queries data from *wolfie\_home* table, and display to the user. It also accepts HTTP request from user to control devices module.

**5.2.1 Backends**

The job of backend is to communicating with vclient and the MYSQL database on behalf of users. It has following urls: api/login, api/logout, api/house, api/control, api/module

* login: verify login information of a user. If it is verified, a cookie is sent to identify user. The cookie is the username.
* logout: destroy the cookie.
* devices: query information about devices
* control: controlling

API specifications:

* login: AJAX with POST, with fields *username* and *password*. Each field is type of string. On success, status code is 301 Moved Permanently. On failure, status code is 400 Bad Request. For example, a user can send AJAX with POST method: username=jack, password:123456.
* logout: AJAX with POST, with no fields. On success, status code is 200 OK. On failure, status code is 400 Bad Request.
* house: AJAX with POST, with no fields. On success, status code is 200 OK, and a JSON data with schema house\_info.json is returned. On failure, status code is 400 Bad Request.
* modules: AJAX with POST. It has fields *command* and *module\_uids*. There are two *command*:

|  |  |
| --- | --- |
| command | quick description |
| get\_module\_recent | Get the most recent data about one or more modules |
| get\_module\_all | Get all data about one or more modules |

The *module\_uids* is a list of one or more module uids that separated by commas. On success, the status code is 200 OK, the data is JSON with modules\_data.json schema

* control: AJAX with POST. It has fields topic and payload, which are the same in section 4.2. On success, the status code is 200 OK, otherwise it is 400 BAD Request.

1. **Security**

Security is very important in this project. To be filled.