**Workflow-Based Platform for a Home Automation System**

An Undergraduate Thesis by

**AMANSE**, Joe Christopher Paul Imperial

**BERMEJO**, Christian Yuson

Submitted to the

Department of Electronics, Computer and Communications Engineering

School of Science and Engineering

Loyola Schools

Ateneo de Manila University

Philippines

In Partial Fulfillment

of the Requirements for the Degree of

Bachelor of Science in Computer Engineering

May 2016

This is to certify that this undergraduate thesis, entitled

**Workflow-Based Platform for a Home Automation System**

was prepared and submitted by

Joe Christopher Paul I. Amanse, and Christian Y. Bermejo

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Dr. Rosula S.J. Reyes**

Panel Member

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Engr. Maria Leonora C. Guico**

Panel Member

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Engr. Samuel Matthew G. Dumlao**

Panel Member

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Mr. Carlos M. Oppus**

Thesis Adviser

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Engr. Juan Antonio G. Mariñas**

Thesis Adviser

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Dr. Rosula S.J. Reyes**

Chair, ECCE Department

Final Thesis Grade:

# About the Authors

|  |
| --- |
|  |
| **Joe Christopher Paul I. Amanse**  **Bachelor of Science**  **in Computer Engineering**  **christopheramanse@gmail.com** |
|  |
| **Christian Y. Bermejo**  **Bachelor of Science**  **in Computer Engineering**  cybermejo@gmail.com |

****



# Abstract

The thesis project is focused on the development of a workflow-based platform for a home automation system. The system provides a graphical user interface that users can access to create Workflows. Workflows are programs or rules that allow the automation of the connected home devices. The system is composed of a Main Hub with the graphical user interface, Smart Plugs, and Sensor Blocks.

The Main Hub acts as the coordinator of the whole system where it sends commands to the Smart Plugs, and receives triggered events from Sensor Blocks. The Sensor Blocks also receive the paired Main Hub’s information, and the events to be observed based on the Workflows created. The Main Hub and the Sensor Blocks were developed using Raspberry Pis loaded with lightweight server-client applications built upon the Node.js runtime environment. Additionally, the Sensor Blocks interface with Arduino microcontrollers to get data from their sensors. Technologies consisting of HTML, CSS, and JavaScript were used to build the graphical user interface. The interface allows users to connect or disconnect Smart Plugs and Sensor Blocks, and to create their own Workflows with different combinations of trigger events and actions. The Smart Plugs were developed using KanKun Smart Plugs, which were modified to allow control through HTTP requests. Together, all devices communicate within a local area network. However, the main hub can too be accessed remotely when the local area network is configured for remote connections.

Finally, the system was tested by connecting appliances to the Smart Plugs and actual sensors to the Sensor Blocks. The graphical user interface can be accessed using browsers in any device, and is optimized both for computers and mobile devices.

# Acknowledgements

We would like to express our heartfelt gratitude to the following, without whom none of this would be possible:

* Mr. Carlos Oppus and Engr. Juan Antonio Mariñas, our thesis advisers, for all their help, guidance, and encouragement throughout the academic year
* Dr. Rosula Reyes, Engr. Maria Leonora Guico, and Engr. Samuel Matthew Dumlao, our panelists, for their valuable comments, insights, and suggestion regarding the research topic
* Sir Daniel Lagazo, for introducing us to various thesis projects, and for initially guiding us as our thesis adviser
* Sir Sonny Toledo and Sir Mackie Luzano, for providing us materials for the thesis project
* Hitachi Room Thesis group, for their friendship and all their valuable comments
* ECCE 2016, for all the friendship, experiences, and obstacles we endured together
* Our friends and family, for their unconditional support
* Almighty God, for sustaining our strength and hope throughout the academic year

# Table of Contents

About the Authors i

Abstract ii

Acknowledgements iv

Table of Contents v

List of Figures viii

List of Tables ix

1. Introduction 1

1.1 Definition of Terms and Acronyms 1

1.1.1 Home Automation 1

1.1.2 Smart Plug 2

1.1.3 Sensor Block 2

1.1.4 Main Hub 2

1.1.5 Workflow 2

1.1.6 JavaScript Object Notation 2

1.1.7 Application Programming Interface 3

1.1.8 Representational State Transfer 3

1.2 Significance 4

1.3 Objectives 5

1.4 Related Literature 6

1.4.1 A ZigBee Based Home Automation System 6

1.4.2 Design Issues and Solutions in a Modern Home Automation System 6

1.4.3 Raspberry Pi as a Sensor Web Node for Home Automation 7

1.5 Scope and Limitations 8

2. System Components and Setup 9

2.1 Hardware 9

2.1.1 Raspberry Pi 2 Model B 9

2.1.2 gizDuino Version 5 w/ATmega328P 9

2.1.3 KanKun Smart Plug 9

2.1.4 Miscellaneous 10

2.1.4.1 Potentiometer 10

2.1.4.2 LM335 Temperature Sensor 10

2.1.4.3 Light Dependent Resistor 10

2.2 Software 11

2.2.1 KanKun Smart Plug Hack 11

2.2.2 Node.js 11

2.2.3 MongoDB 12

2.2.4 HTML/CSS/JavaScript 12

2.3 Setup 13

3. Implementation 14

3.1 Modification of the KanKun Smart Plug 14

3.2 Sensor Block 14

3.3 Main Hub 15

3.3.1 Hardware and Software Components 16

3.3.2 Graphical User Interface 16

3.3.3 Database 16

3.3.4 Smart Plugs 17

3.3.5 Sensor Blocks 17

3.3.6 Workflows 17

4. Test and Results 19

4.1 Final API 19

4.2 Final Graphical User Interface 20

4.3 Process Flow 22

4.4 Testing 25

5. Conclusion and Recommendations 26

5.1 Conclusion 26

5.2 Recommendations 26

References 28

Appendix A: Code 30

A.1. KanKun Smart Plug 30

A.1.1. kankut\_root/etc/config/network 30

A.1.2. kankut\_root/etc/config/wireless 30

A.1.3. kankut\_root/www/cgi-bin/name 30

A.1.4. kankut\_root/www/cgi-bin/relay.cgi 31

A.1.5. kankut\_root/www/cgi-bin/set\_name.cgi 31

A.1.6. kankun\_setup.sh 32

A.1.7. kankun\_setup\_webserver.sh 32

A.1.8. README.md 32

A.2. Sensor Block 33

A.2.1. arduino/sendTorPi/sendTorPi.ino 33

A.2.2. mongodb/data/db/README.md 33

A.2.3. routes/api.js 33

A.2.4. app.js 36

A.2.5. db.js 37

A.2.6. device-info.js 38

A.2.7. device-info.json 38

A.2.8. event-handler.js 38

A.2.9. hub-info.js 40

A.2.10. hub-info.json 40

A.2.11. nodemon.json 40

A.2.12. package.json 41

A.2.13. README.md 41

A.2.14. start-db.sh 41

A.3. Main Hub 42

A.3.1. bin/www 42

A.3.2. mongodb/data/db/README.md 43

A.3.3. public/css/homeautomation.css 43

A.3.4. public/css/index.css 48

A.3.5. public/js/sensor-finder.js 51

A.3.6. public/js/sensor-remover.js 52

A.3.7. public/js/smartplug-finder.js 52

A.3.8. public/js/smartplug-remover.js 53

A.3.9. public/js/toggle.js 54

A.3.10. public/js/workflow-man.js 55

A.3.11. routes/add-sensors.js 56

A.3.12. routes/add-smartplugs.js 57

A.3.13. routes/api.js 58

A.3.14. routes/index.js 60

A.3.15. routes/sensors.js 60

A.3.16. routes/smartplugs.js 61

A.3.17. routes/workflows.js 62

A.3.18. views/partials/navigation.html 65

A.3.19. views/partials/slidenav.html 65

A.3.20. views/add-sensors.html 66

A.3.21. views/add-smartplugs.html 66

A.3.22. views/error.html 67

A.3.23. views/index.html 67

A.3.24. views/layout.html 69

A.3.25. views/sensors.html 70

A.3.26. views/smartplugs.html 72

A.3.27. views/workflows.html 74

A.3.28. app.js 78

A.3.29. db.js 80

A.3.30. device-fetcher.js 80

A.3.31. package.json 82

A.3.32. start-db.sh 82

# List of Figures

Fig. 1. System diagram of the home automation system 13

Fig. 2. Search Smart Plugs interface 20

Fig. 3. Smart Plugs list interface which includes a toggle and remove button 21

Fig. 4. Workflow list interface which includes add, edit and remove buttons 21

Fig. 5. Add Smart Plug process flow 22

Fig. 6. Add Sensor Block process flow 23

Fig. 7. Add Workflow process flow 24

Fig. 8. Workflow operation process flow 25

# List of Tables

Table 1. Smart Plug API 19

Table 2. Sensor Block API 19

Table 3. Main Hub API 20

# Introduction

## Definition of Terms and Acronyms

The following were the index or key terms that were used throughout the study.

### Home Automation

The definition and capabilities of home automation have changed considerably over the years, with terms and concepts such as “home of the future”, “smart homes” and “The Internet of Things” being used to describe it. Three decades ago, when home automation was rarely thought of by most people, it was primarily known to X10 enthusiasts and was installed by professionals in luxury homes. Progressively, research and commercial versions have been built including Mozer’s adaptive house [1], Georgia Tech Aware Home [2], Orange [3], eHome [4], and MIT’s House\_n [5][6].

Today’s technology has brought home automation into the mainstream, with the advent of simpler and more affordable means for homeowners to install their own home automation systems. [7]

In their study, Gill defines home automation as the introduction of technology within the home to enhance the quality of life of its occupants through the provision of different services such as telehealth, multimedia entertainment, and energy conservation [8]. On the other hand, Lucero [9] defines it as the capability to automate and control multiple disparate systems. Operating on these definitions, home automation can then be thought of as a technological solution intended to provide seamless automation, regulation and control of electronic, electrical and other technology-based systems within the home [10] with the goal of providing an improved and more convenient way of life.

### Smart Plug

In this thesis, a Smart Plug is referred to a Wi-Fi connected socket adapter that can be controlled wirelessly. It is paired with the Main Hub for automation.

### Sensor Block

In this thesis, a Sensor Block is referred to a Wi-Fi connected device that has sensors. The value of the sensors are continuously observed by the Sensor Block. Additionally, it is paired with the Main Hub to allow coordination with the Smart Plugs and Workflows.

### Main Hub

The Main Hub is the core of the thesis project’s home automation system. It coordinates the connected Sensor Blocks and Smart Plugs. It also contains the database for all the paired Smart Plugs, Sensor Blocks, and the Workflows created by the user.

### Workflow

In this thesis, a Workflow is referred to the simple program generated by the user. The program has an input – a condition for a sensor value of a Sensor Block, and an output – a Smart Plug device and its command. These values are configurable through the Main Hub’s graphical user interface.

### JavaScript Object Notation

JSON (JavaScript Object Notation) is an open-standard structured format of representing data. It is easy for both machines and humans to understand. Additionally, it is independent of programming language, such that most programming languages have built-in features to generate and parse JSON-formatted data. [11] The standard data format will allow the system to be easily extensible for devices conforming to the same format.

### Application Programming Interface

An application programming interface (API), is a set of defined standardized routines or functions. It allows a programmer to simply call an API to run a complex routine without knowing how it is implemented. In a web service, this can be a set of HTTP requests that performs an action, such as a query, and returns its output, such as a query result. [12]

In this study, the API are defined by the researchers to create a standard for communication between devices. This also allows the system to be extensible, and allows access to other interfaces, such as mobile and desktop applications, in addition to the web interface.

### Representational State Transfer

REST (Representational State Transfer) is a software architectural style for distributed hypermedia systems. Hypermedia systems distributes text, audio, video and hyperlinks. The style provides architectural constraints that allows a system to emphasize its scalability of component interactions, generality of interfaces, and independent deployment of components. [13]

When applied to a web service, the system will be able to communicate independently of operating systems, programming languages, processors, and internal protocols. REST is also not bounded to any protocols wherein any application protocol can use REST’s architectural style. [13]

In this study, HTTP is used as the protocol for receiving and transmitting data, while using REST’s architectural style.

In terms of representation, JSON is used to represent the data for transmitting or receiving as it is the commonly used format for web services.

## Significance

During the past few years, there has been an increase in demand for home automation systems from consumers [14]. Home automation systems allow people to interact with their personal appliances either remotely through mobile devices and computers or through mediums such as voice and gestures. These systems also allow the automated operations, i.e., activation and deactivation of appliances. They also grant people the possibility of monitoring and securing their homes as well as saving more energy, now having an increased awareness and control in the power consumption of one’s appliances.

Different home automation systems use various types of network communication standards such as Bluetooth, Wireless LAN, and ZigBee. The study will use HTTP delivered through wired or wireless local area networks (LANs) as the means of communication between the devices. Most homes are already equipped with such systems and thus ensures quick integration of the output of the study with current home systems.

Some issues can be observed with current home automation systems. Many still require the interaction from the user, and current programmable home automation systems often only permit basic programming. Some systems only provide manual controls either through mobile phone or a web interface while others offer limited programs for automation such as simple usage of time as the only variable.

To improve the current state of home automation systems, the study will focus on creating a home automation system that is easily programmable by using visual workflows created by users from a user-friendly graphical user interface.

The study aims to develop a configurable home automation system, thus, the group focuses on the dynamics of workflows rather than developing lots of features for the system. The use of standard RESTful Web APIs, however, allows the system to be extensible for further development.

## Objectives

The primary objective of the study is to develop a workflow-based home automation system. Workflows are triggered by sensor events created by the user. In order to achieve this, the proponents aim to accomplish the following objectives:

1. Design and develop the components for the home automation system using Raspberry Pis, Arduino microcontrollers, and Smart Plugs while using a local area network for data transmission, and HTTP as its transmission protocol
2. Develop a Workflow-based platform for the system that would enable the automation of the connected home devices
3. Develop an interactive graphical user interface (GUI) for users that will allow them to create workflows for the home automation system so that non-technical users can configure their own system
4. Test the home automation system using actual appliances and sensors

## Related Literature

### A ZigBee Based Home Automation System

This paper by Khusvinder Gill, Shuang-Hua Yang, Fang Yao and Xin Lu reviews the existing state of home automation systems and identifies five areas which their proposition of a Zigbee-based system aims to solve - namely, the complexity and expense of the architectures adopted by existing systems, the intrusiveness of the system installations, the lack of interoperability between different home automation technologies and the lack of interoperability between systems developed by different manufacturers that utilize the same technology. Interface inflexibility and the inconsistent approaches adopted towards security and safety were also cited as problems [15]. The presented system architecture uses a combination of Wi-Fi technology and ZigBee communications protocol to monitor and control the different home devices either locally within the home environment or remotely through the Internet. Furthermore, a home gateway and a virtual home were also implemented to achieve the interoperability between the aforementioned networks through a unified interface and to coordinate the system and security respectively. The paper also highlights the feasibility and effectiveness of the proposed system through real world testing using some developed Zigbee devices (i.e, a light switch, radiator valve, safety sensor and Zigbee remote control). The proposed architecture will be used in this thesis as a model for developing our home programmable home automation system.

### Design Issues and Solutions in a Modern Home Automation System

This paper by Mattia Gamba, Alessandro Gonella and Claudio E. Palazzi [16] of the University of Padua attempts to answer the issues of scalability and customization, device communication and user interface design of a modern ZigBee-based home automation system. In this thesis, mobile devices such as smartphones and laptops served as the home automation controllers, which controlled and manipulated the XBee devices or appliances through a web interface. The thesis contains some more information which we will use in our own thesis to serve as models or foundations when preparing our system and user-programmed workflows. Namely, the presented Control Unit layout - a Linux PC with a USB-connected XBee module, the “Bottom Up” device-oriented approach to the user interface design, and the underlying philosophy of their Rule Engine System, which relied on a ZigBee binding mechanism called “end-device binding”.

### Raspberry Pi as a Sensor Web Node for Home Automation

This paper by Vladimir Vujović and Mirjana Maksimović briefly discusses the definition of home automation systems, its types and how it is made ubiquitous and expansive through the use of Sensor Web elements. It discusses the high cost of ownership, inflexibility, poor manageability and difficulty achieving security as the main challenges of home automation. [17] To attempt to solve these, they proposed a flexible, secure and low cost Internet-based based home automation system by utilizing a Raspberry Pi as a Sensor Web node. [17] The system would be capable of controlling and automating appliances through an easy manageable web interface. Technologies such as Java, Apache Tomcat and JSON were used to build the system. Furthermore, a prototype fire-safety system was also implemented as a means of application of what is possible through the proposed system architecture. The paper contains a detailed discussion on the Raspberry Pi, including specifications, comparison tables between different models as well as applicable projects. The group looks into further detail the section exploring the implementation of a Sensor Web testbed, wherein a Raspberry Pi is used as the processing unit of sensor data as well as a RESTful service; this shall be used as a basis for our output system.

## Scope and Limitations

The project shall focus on the configurability and flexibility of interactions between the Sensor Blocks and Smart Plugs, while using a main hub as their coordinator. Thus, the project will be limited to the programmability feature rather than implementing lots of features such as interaction with a web service such as email. However, with the use of standard HTTP REST API, the final output is extensible without changing too much code.

For the simplicity of the project, the output of the home automation system are only the Smart Plugs, which turn on or off the appliances connected to them. On the other hand, the input of the system are analog sensors compatible with Arduino that are converted to digital input which is then sent to the Raspberry Pi through the USB Serial communication. To test the system, the group uses the analog pins of the Arduino to get a 10-bit data which is enough for the temperature and light sensors. Existing smart appliances, such as smart light bulbs, or smart thermostats will not be integrated with the system, since this require further development depending on the specification of each device. Although, the system can be extended to integrate existing devices.

To also lessen the complexity of the project, workflows are limited to a one-to-many event-output relationship, such that output or actions can only be triggered by one event, i.e., it is not possible for one action to be triggered by a combination of events.

Lastly, the project is a proof-of-concept such that it serves as a foundation for a framework or platform for developing a home automation system, rather than a final marketable product.

# System Components and Setup

## Hardware

### Raspberry Pi 2 Model B

The Raspberry Pi 2 Model B is a credit card sized computer. It has a 900MHz quad-core ARM Cortex-A7 CPU, and 1GB RAM. It also has four USB ports, GPIO pins, HDMI port, Ethernet port, a combined 3.5mm audio jack and composite video, CSI camera interace, DSI display interface, MicroSD card slot, and a VideoCore IV 3D graphics core. With its specifications, it can run a Linux OS. [18]

In this project’s case, the Raspberry Pi runs on Raspbian OS, and uses a Wi-Fi USB dongle so that it can connect to a local area network.

### gizDuino Version 5 w/ATmega328P

The gizDuino is an Arduino clone that is sold locally in the Philippines. Since the Arduino is open source, this gives manufacturers to use the hardware design of the Arduino, thus enabling them to create their own cheaper versions. The gizDuino is cheaper than the Arduino since it uses cheaper electronic materials. The gizDuino Version 5 w/ATmega328p comes with a case, and a USB connector. Internally, it has an ATmega328P integrated chip, which can be programmed using Arduino IDE. [19]

### KanKun Smart Plug

A KanKun Smart Plug is used as the output of the home automation system. Since the Smart Plug has only two outputs, on or off, the home automation system can only turn on or off devices. In its out-of-the-box state, the Smart Plug can be configured and controlled using its companion mobile app, which is available both for iOS and Android. It also has LEDs which indicates the state of the plug, and if it’s connected to a network. [20]

### Miscellaneous

#### Potentiometer

A potentiometer is simply a voltage divider circuit with a variable load. The variable load can be changed using its knob. Once connected to the power source, the output of the potentiometer can be varied from 0 to the voltage level of the power source. In the Arduino, the output can be placed in an analog pin. Since the Arduino has its own analog-to-digital converter, the analog input is now converted from 0 to 1023, where 1023 is 5V and 0 is 0V. [21]

#### LM335 Temperature Sensor

The LM335 integrated circuit is a temperature sensor which is directly calibrated to the Kelvin temperature scale. It has 1°C initial accuracy, and operates from 400 µA to 5 mA. It is low cost and easily calibrated. [22]

In the system, the temperature sensor is connected to the Arduino of a Sensor Block. This allows the Sensor Block to have data regarding the current temperature.

#### Light Dependent Resistor

The light dependent resistor is a variable resistor that is controlled using light. Its resistance decreases as the light intensity increases. [23]

When the light dependent resistor is used as the variable load of a voltage divider circuit, the output voltage can now be controlled using light. In the system’s case, the circuit is placed on the Arduino of a Sensor Block, which allows the Sensor Block to have data regarding the light.

## Software

### KanKun Smart Plug Hack

To enable the KanKun Smart Plug work with the home automation system, it needs to be controllable in the network via HTTP requests. Fortunately, Peter Jennings had published his instructions on how to hack the KanKun Smart Plug. The Smart Plug is accessible through SSH, and has a factory default password of “p9z34c”, without the quotes. In his findings, there is also a built-in web server that is turned on by default. Through the use of Bash scripts and the web server, Peter Jennings was able to make the KanKun Smart Plug controllable through HTTP requests. [24]

### Node.js

Node.js is a JavaScript runtime built on top of Google Chrome’s V8 JavaScript engine. It allows JavaScript code to run natively in the computer rather than just in browsers. Moreover, by default, it has an event-driven and non-blocking IO API, which will be perfect for the home automation system since the system is also event driven. [25]

Additionally, since the system will have a web graphical user interface, the researchers used Express, a web application framework for Node.js. It is a lightweight and flexible framework which is enough for the home automation system. [26]

Express also uses a couple of dependencies, which have their own open source licenses. The researchers also used other Node.js modules to make the code easier to read, which are managed by “npm”, a package manager for Node.js. The package manager enables fast installation of required modules of a project. The modules used are listed in the project’s “package.json” file.

### MongoDB

The home automation system needs a persistent storage for storing information for the connected Smart Plugs and Sensor Blocks, and the created Workflows. The researchers used the popular open source NoSQL database, MongoDB. Having a NoSQL database makes the data structure of the system to be dynamic, with no effort, as data is stored in standard JSON format. Thus, with NoSQL, there is no need to set up a schema for the database. [27]

The researchers used MongoDB because it is cross-platform, easy to install, and little to no setup required.

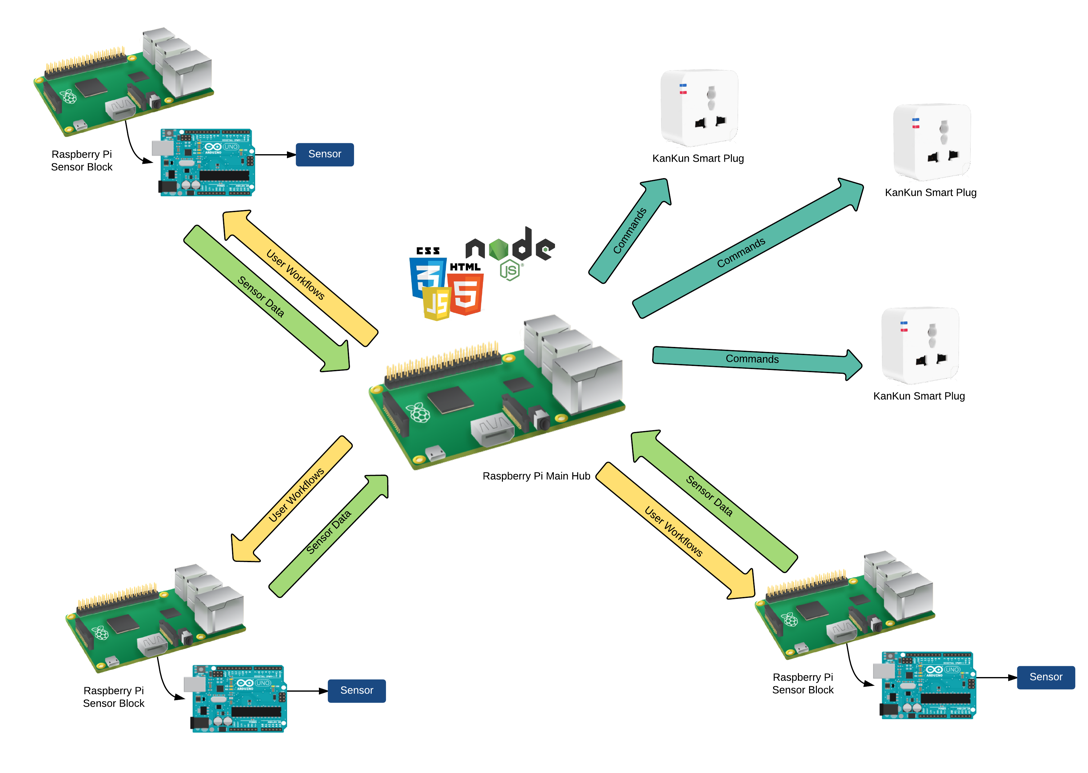
In the home automation system, the Main Hub and the Sensor Block have their own MongoDB database. The Main Hub uses its database to store the information of connected Smart Plugs and Sensor Blocks, and user-generated Workflows. On the other hand, the Sensor Block uses its database to store the events to be observed based on the user-generated Workflows. Each Sensor Block has its own database.

### HTML/CSS/JavaScript

To serve the graphical user interface to the local area network, the researchers used HTML, CSS and JavaScript, which should generate web pages. The researchers designed their own layout and site structure, however, in order for the web page to be well-designed aesthetically, the researchers used a CSS framework called Bootstrap Material Design [28]. This framework lets web elements use Google’s Material Design [29]. This also allows the web page to be optimized for mobile devices, thus the home automation system is usable in small screen devices.

## Setup

The Main Hub, Smart Plug, and Sensor Block are all connected to a single network so they can communicate between each other. The Main Hub and Sensor Blocks are Raspberry Pi 2 Model B device, but for faster development, a computer or laptop with a Linux or OS X operating system is used. The Sensor Block also has an Arduino connected via USB Serial port, where the sensors are connected. A system diagram shown in the figure below illustrates how the devices communicate between each other.



1. System diagram of the home automation system

# Implementation

## Modification of the KanKun Smart Plug

To make the KanKun Smart Plug compatible with our home automation system, the KanKun Smart Plug was modified based on the code of Peter Jennings, which is published in GitHub and distributed under the MIT License [24]. The codes are web-accessible bash scripts which turn on or off the device.

After modifications, the device is now controllable using simple HTTP requests to the device. The proponents then made the responses of the requests to JSON format for better data structure, which is also used throughout the system. The responses contain the device information such as its MAC address and state of the switch.

The Smart Plug has corresponding HTTP requests for turning it on, off, toggling and checking its status.

## Sensor Block

The Sensor Block was built using a Raspberry Pi 2 and an Arduino, which are connected to each other through their USB ports. The Arduino serves as the receiver of sensor data, which then converts it to a sequence of characters and sends it to the Raspberry Pi. The Raspberry Pi reads the data from the USB Serial port, and analyzes if a condition is triggered. If it is triggered, the Sensor Block contacts the Main Hub information and sends the appropriate HTTP request. During development, an OS X and Linux computer was used.

The Sensor Block also has a MongoDB database of the events it will observe. It receives the event from the Main Hub whenever a Workflow is created. Therefore, the Main Hub sends an HTTP request to the Sensor Block containing the event data. For example, if a created Workflow has a condition of a light value greater than a raw value of 150, then the Main Hub sends it to the target Sensor Block.

For the Sensor Block to receive HTTP requests from the Main Hub, the researchers also created a similar, but lightweight, web server to the Main hub’s web server. The web server contains only a REST API, such that the responses to requests made to the Sensor Block are JSON data only instead of HTML, CSS and JavaScript.

Every time the Sensor Block’s Raspberry Pi receives data from its Arduino, it checks if any of the conditions are triggered. Additionally, the Sensor Block creates simulated signals for each condition, such that a condition will only be triggered on its rising edge signal. In simpler terms, the Sensor Block will send a trigger event to the Main Hub if and only if the condition became true and its previous value is false. This will prevent bursts of requests to the main hub since the sensor data is continuous and it may satisfy a condition 10 times, more or less, in a second. Moreover, the Arduino is configured to send only the data in 100ms gaps.

## Main Hub

The Main Hub is the core of the system. It is the coordinator of the entire system, such that it enables communication between the Smart Plugs and Sensor Blocks. It is also where the data for all the connected devices in the network and created Workflows are stored. It has a web server that serves the graphical user interface for the users through a browser on any device connected to the same local area network.

### Hardware and Software Components

The Main Hub was built using a Raspberry Pi 2 Model B. However, Mac OS X and Linux computers were used in development and testing for a faster development workflow.

The main application was built using the Node.js runtime environment. To create the web server, the Express web framework for Node.js was used.

MongoDB was used as the database of the application. To integrate the MongoDB database with the main application in Node.js, the “mongodb” Node.js module, MongoDB’s official client for Node.js.

### Graphical User Interface

For the graphical user interface, the group used the HTML, CSS, and JavaScript web stack. It is integrated with the Node.js application through the Swig Node.js template engine. The Bootstrap Material Design CSS framework was used for aesthetic design. A web page is created for each use case of the home automation system.

### Database

The MongoDB database of the Main Hub contains the information of the connected Smart Plugs and Sensor Blocks, and the created Workflows. The database is stored locally in the Main Hub, and is always started before running the main application.

For the Smart Plugs and Sensor Blocks, the IP address and MAC address of each device is stored in the database. On the other hand, the Workflows are stored with the condition, the MAC address of a Sensor Block, and the MAC address of a Smart Plug.

### Smart Plugs

In the adding of Smart Plugs, the code for finding Smart Plugs in the local area network is implemented in the client’s browser. The Smart Plugs were searched using JavaScript – it loops all through IP addresses and checks if it is a Smart Plug. When a user adds a Smart Plug, data is sent to the Main Hub, where the Main Hub then saves its information in the database. On the other hand, if a user removes a Smart Plug, the client’s browser sends the Smart Plug information to the Main Hub, so the Main Hub will know which Smart Plug needs to be removed. In turn, the Main Hub removes the Smart Plug from its database.

In toggling of Smart Plugs, the code is also implemented in the client’s browser, since when the client accesses the page, they also get the information of each Smart Plug connected. The client’s browser sends the appropriate HTTP request to the Smart Plug directly. This allows less work for the Main Hub.

### Sensor Blocks

The implementation of the addition and removal of Sensor Blocks is similar to the implementation of adding and removing Smart Plugs. Similar to adding Smart Plugs, adding Sensor Blocks takes place in the client’s browser, while the removal of Sensor Block results in the client’s browser sending its information back to the Main Hub, where the Main Hub will remove it from the database.

### Workflows

Finally, implementation of the Workflows is the core of the thesis project. Workflows allow the automation of turning on or off of the Smart Plugs based on the event. The event, source Sensor Block, and the target Smart Plug with its command are all customizable by the user. A graphical user interface for creating the Workflows was created so that users don’t need to know programming. This makes the home automation system customizable to the user’s needs.

A single web page for creating and viewing Workflows was created. Since Workflows need to know the connected Smart Plugs and Sensor Blocks, they are first fetched from the database whenever its web page is accessed. The user can then create a Workflow based on the connected devices and customize their own values for the event.

During development, the conditions were simulated using an HTTP request to ensure that the code for handling Workflows worked. This HTTP request will also be used by the Sensor Blocks whenever it detects that an event was satisfied. The HTTP request contains the event information, such that when it is received by the Main Hub, it finds Workflows that matches the event received. For each Workflow found, the Main Hub sends the command to the target Smart Plug using HTTP request.

# Test and Results

## Final API

To better illustrate how the home automation system works internally, Tables 1, 2, and 3 shows the HTTP API of each device. Each HTTP request is called internally by a device depending on the current situation. The IP\_ADDRESS in the tables are to be replaced with the actual IP address of the device.

1. Smart Plug API

|  |  |
| --- | --- |
| HTTP Request | Command |
| GET: http://IP\_ADDRESS/cgi-bin/relay.cgi?state | Get State |
| GET: http://IP\_ADDRESS/cgi-bin/relay.cgi?toggle | Toggle |
| GET: http://IP\_ADDRESS/cgi-bin/relay.cgi?on | Turn On |
| GET: http://IP\_ADDRESS/cgi-bin/relay.cgi?off | Turn Off |

1. Sensor Block API

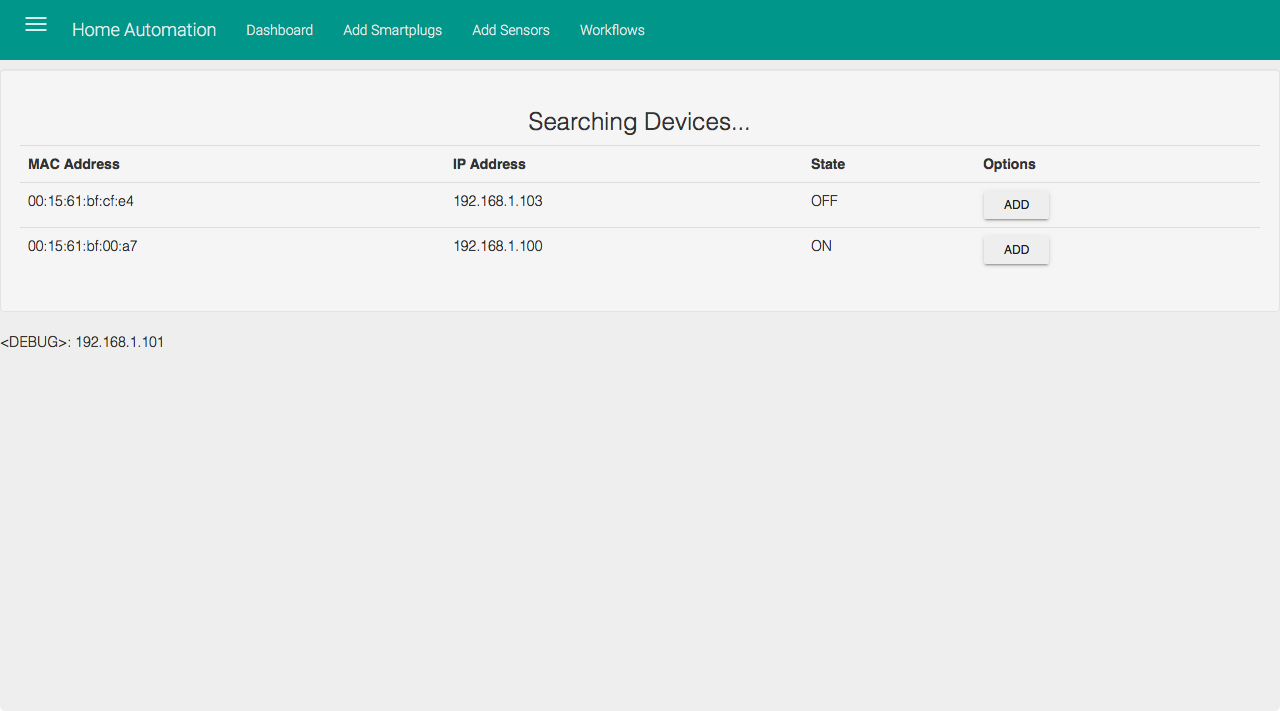
|  |  |
| --- | --- |
| HTTP Request | Command |
| GET: http://IP\_ADDRESS/api/hub/info | Get IP Address of hub |
| POST: http://IP\_ADDRESS/api/hub/info | Set IP Address of hub |
| GET: http://IP\_ADDRESS/api/device | Get MAC Address of device |
| GET: http://IP\_ADDRESS/api/events | Get events being observed |
| POST: http://IP\_ADDRESS/api/events | Add an event to be observed |
| POST: http://IP\_ADDRESS/api/events/debug/trigger | Manually send a value to debug |

1. Main Hub API

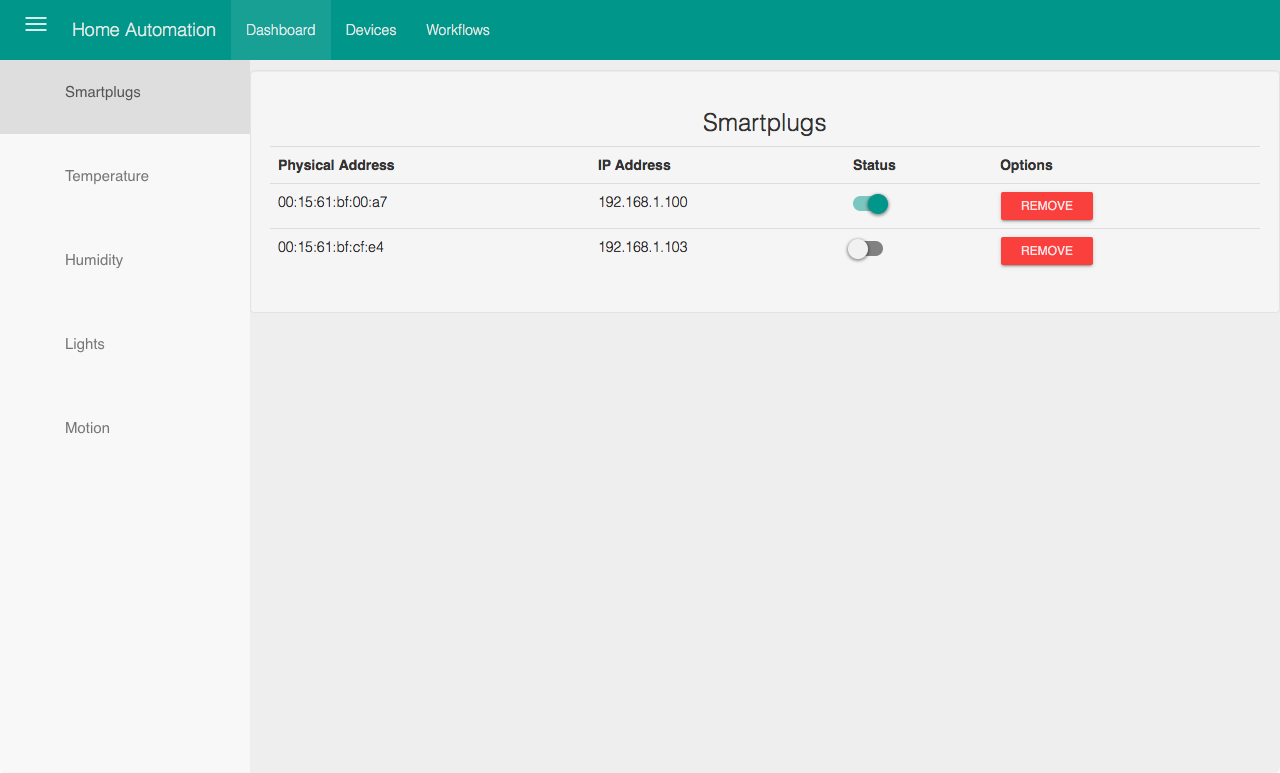
|  |  |
| --- | --- |
| HTTP Request | Command |
| POST: http://IP\_ADDRESS/api/event/trigger | Send a triggered event |

## Final Graphical User Interface

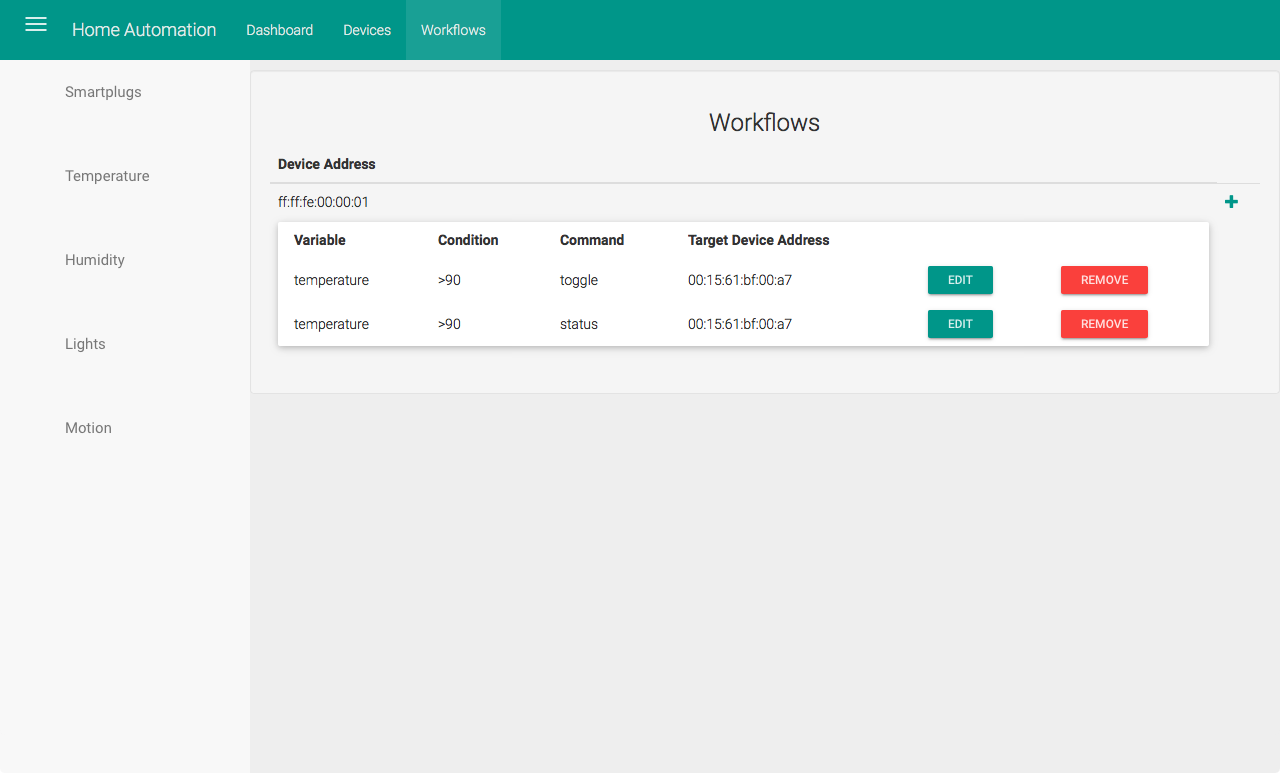
The following figures show the screenshots for the main parts of the graphical user interface of the Main Hub where they demonstrate the capabilities of each page. In this case, the web interface is accessed in Safari on an OS X operating system.



1. Search Smart Plugs interface



1. Smart Plugs list interface which includes a toggle and remove button



1. Workflow list interface which includes add, edit and remove buttons

## Process Flow

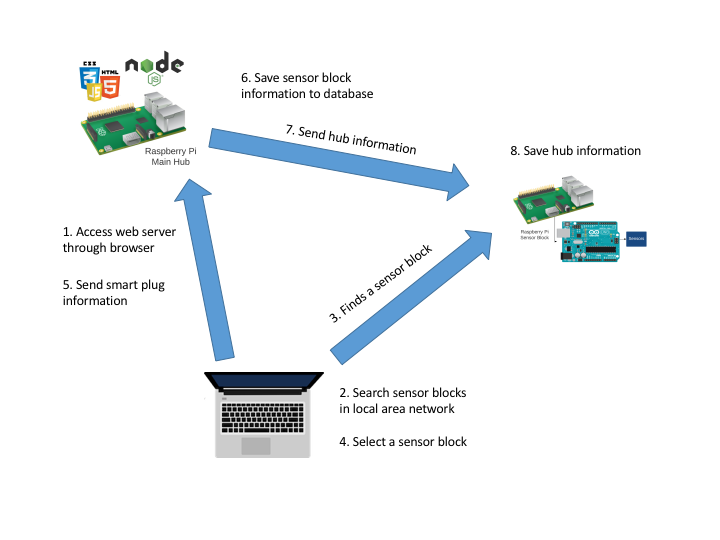
For each situation, transmission of data varies so that the system works efficiently as much as possible. The Main Hub and the Sensor Block have their own databases so that information can be used later without additional communication. On the other hand, the Smart Plug only receives a command and transmits its information. The process flows for each situation of the system is shown in Fig. 5, 6, 7 and 8. The order of the steps are ordered in the figures.

Fig. 5 shows the process flow for adding or pairing Smart Plugs to the Main Hub. The first data transmission occurs is when the user access the Main Hub’s web interface through a browser. The client browser then searches for Smart Plugs connected in the local area network, and displays them in a list. When the user selects a Smart Plug, the client browser sends the Smart Plug information back to the Main Hub, which, in turn, the Main Hub saves for later use.



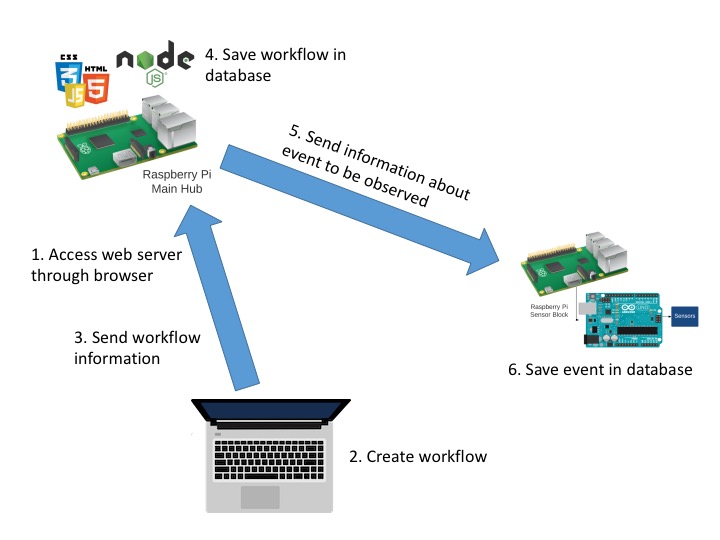
1. Add Smart Plug process flow

The process flow for pairing Sensor Blocks is almost similar to the process flow of pairing Smart Plugs, as shown in Fig. 6. The only difference for the pairing of Sensor Blocks is that it has an added step after the Main Hub saves the Sensor Block information – the Main Hub sends its own information, specifically its local IP address to the Sensor Block, which then the Sensor Block saves. This added step is required so that the Sensor Block can contact the Main Hub later when an event is triggered.



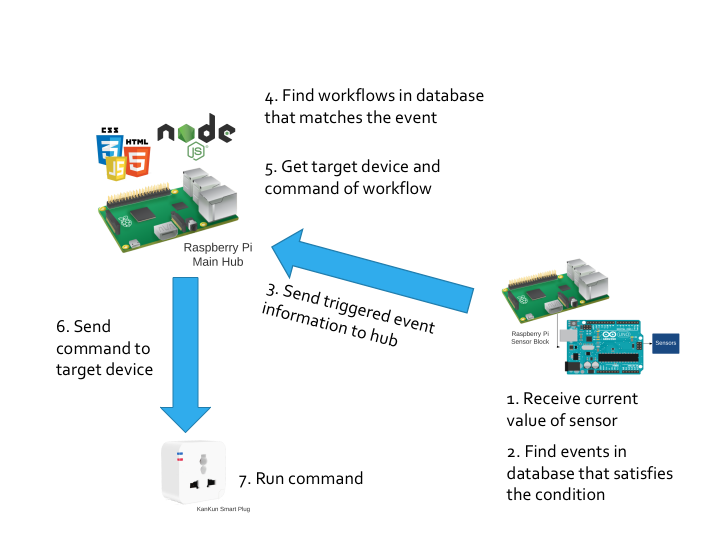
1. Add Sensor Block process flow

Fig. 7 illustrates the process flow for creating Workflows. Similar to the previous processes, the creation of Workflows starts when a user accesses the Main Hub’s web server. In the web interface, when the user creates a Workflow, it sends back the created Workflow information back to the Main Hub, which the Main Hub will save. After saving the Workflow to the Main Hub’s database, the information of the event to be observed will be sent to the Sensor Block that will observe such event. The Sensor Block will then save this event information to its own database so that every time it receives a value from the sensor, it will check if the condition for the event is satisfied.



1. Add Workflow process flow

Lastly, Fig. 8 illustrates how data is transmitted during Workflow processing, which occurs continuously whenever the system is up and running. This time, the data source starts with the Sensor Block. When the Sensor Block receives a sensor value, it finds the events in the database and checks whether it has satisfied the condition for that event. For instance, if the value satisfies the condition, the event is sent to the Main Hub. Then, the Main Hub will find all the Workflows in its own database that matches the event received. For each event found, the Main Hub will get the Smart Plug IP address, and the command from the Workflow. Finally, the Main Hub will send the command to the Smart Plug. This demonstrates that the system is fully automated once Workflows are created.



1. Workflow operation process flow

## Testing

The final product was tested using two Sensor Blocks, two Smart Plugs and a Main Hub, which are all connected to the same local area network. The first Sensor Block had a temperature sensor, and the second had a light sensor. The graphical user interface was testing using a Safari browser in an OS X operating system. Two workflows were created and an event that depends on the temperature and light, respectively. The workflows should toggle their corresponding Smart Plug.

A lamp and a charger were connected to each Smart Plug. The value for the temperature sensor was changed using a soldering iron, while the light sensor was changed using a flashlight. The two workflows successfully toggled both devices according to the conditions defined by their Workflow.

# Conclusion and Recommendations

## Conclusion

By using Raspberry Pis and Arduino microcontrollers, a home automation system was developed while using HTTP as the main protocol for communication between devices. In addition to the system, a graphical user interface was created for interacting with the system. The interface enables customizability of the system without code, thus users who don’t know how to code can customize the system to their needs.

Integrating the home automation system with current smart devices requires specifications of each device will take more time thus it was not prioritized in this study. However, since the home automation system uses customizable and extensible components, integration with other devices, especially Wi-Fi enabled devices, is possible. This proves that having a standard for communication between smart devices is also needed for seamless integration between theses devices. With a standard communication and data structure, newly manufactured devices would be integrated to the home automation system without changing anything in the home automation system as long as the new device conforms to the standard.

## Recommendations

For succeeding studies in the same field, the future researchers may replace the Sensor Block with smaller and more portable versions of the Raspberry Pi and Arduino, such as the Raspberry Pi Zero and Arduino Micro, respectively. The researchers may also test the system with different sensors and integrate multiple sensors in a Sensor Block.

Additionally, the future researchers may consider creating a graphical user interface for setting up and configuring the Smart Plugs, Sensor Blocks and Main Hub for convenient installation of software on existing hardware.

Finally, future researchers may also formally push a standard for communication between home automation devices so that devices can be integrated within a system seamlessly.

# References

1. Mozer, M.C. Lessons from an Adaptive House. In D. Cook and R. Das (eds.) Smart Environments: Technologies, Protocols, and Applications (pp. 273- 294). J. Wiley & Sons, Hoboken, NJ, 2005.
2. Kietz, J., Patel, S., Jones, B., Price, E., Mynatt, E., Abowd, A. The Georgia Tech Aware Home. Ext. Abstracts CHI 2008, 3675-3680.
3. Harper, R. (ed.) Inside the Smart Home. Springer, London, 2003.
4. Koskela, T., Väänänen-Vaninio-Mattila, K. Evolution towards smart home environments: empirical evaluation of three user interfaces. PUC 8, 3-4(2004), 234-240.
5. Intille, S. Designing a home of the future. IEEE Pervasive Computing 1, 2(2002), 80-86.
6. A. Brush, B. Lee, R. Mahajan, S. Agarwal, S. Saroiu and C. Dixon, Home Automation in the Wild: Challenges and Opportunities, 1st ed. Washington: Microsoft Research, 2012, p. 1.
7. "Home Automation | SmartHomeUSA.com", Smarthomeusa.com, 2016. [Online]. Available: http://www.smarthomeusa.com/home-automation/. [Accessed: 07- Apr- 2016].
8. K. Gill, S. Yang, F. Yao and X. Lum, “A ZigBee-based home automation system,” Consumer Electronics, IEEE Transactions on, vol. 55, no. 2, pp. 422-430, May 2009.
9. S. Lucero and K. Burden, "Home Automation and Control", ABI Research: Connected Home, 2010.
10. "What is a Home Automation System? - Definition from Techopedia", Techopedia.com, 2016. [Online]. Available: https://www.techopedia.com/definition/29999/home-automation-system. [Accessed: 07- Apr- 2016].
11. “Introducing JSON,” JSON [Online]. Available: http://www.json.org. [Accessed: 12-Apr-2016].
12. “API Definition from PC Magazine Encyclopedia”, PCMag [Online]. Available: http://www.pcmag.com/encyclopedia/term/37856/api. [Accessed: 02-May-2016].
13. O. Filho, and M. Ferreira, “Architectural Styles and the Design of Network-based Software Architectures,” in IADIS International Conference WWW/Internet 2009, p. 171.
14. icontrol­­® Networks, “2015 State of the Smart Home Report”, June 2015. [Online]. Available: http://www.icontrol.com/wp-content/uploads/2015/06/Smart\_Home\_Report\_2015.pdf. [Accessed: 27-Oct-2015].
15. K. Gill, S. Yang, F. Yao and X. Lum, “A ZigBee-based home automation system,” Consumer Electronics, IEEE Transactions on, vol. 55, no. 2, pp. 422-430, May 2009.
16. M. Gamba, A. Gonella, and C.E. Palazzi, “Design issues and solutions in a modern home automation system,” Computing, Networking and Communications (ICNC), 2015 International Conference on, pp. 1111-1115, February 2015.
17. V. Vujović and M. Maksimović, “Raspberry Pi as a Sensor Web node for home automation,” Computers & Electrical Engineering, vol. 44, pp. 153-171, 2015.
18. “Raspberry Pi 2 Model B,” Raspberry Pi Foundation [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/. [Accessed: 02-May-2016].
19. “gizDuino Version 5 w/ATmega328P,” e-Gizmo [Online]. Available: http://www.e-gizmo.com/KIT/gizduinov.html. [Accessed: 02-May-2016].
20. “Kankun Smart Wifi Plug Socket Remote for iPhone Andoid Smartphone - US Plug,” Amazon.com [Online]. Available: http://www.amazon.com/Kankun-Socket-Remote-iPhone-Smartphone/dp/B00N8N5NNK. [Accessed: 02-May-2016].
21. “Reading a Potentiometer (analog input),” Arduino [Online]. Available: https://www.arduino.cc/en/tutorial/potentiometer. [Accessed: 02-May-2016].
22. “LM335”, Texas Instruments [Onlone]. Available: http://www.ti.com/product/LM335. [Accessed: 02-May-2016].
23. “Light Dependent Resistor, Photoresistor, or Photocell,” Radio-Electronics.com [Online]. Available: http://www.radio-electronics.com/info/data/resistor/ldr/light\_dependent\_resistor.php. [Accessed: 02-May-2016].
24. P. Jennings, “LuaLoader/examples/Kankun WiFi Plug at master · GeoNomad/LuaLoader”, GitHub. [Online]. Available: https://github.com/GeoNomad/LuaLoader/tree/master/examples/Kankun%20WiFi%20Plug. [Accessed: 13-Apr-2016].
25. “Node.js, ” Node.js [Online]. Available: https://nodejs.org/en/. [Accessed: 02-May-2016].
26. “Express - Node.js web application framework,” Express [Online]. Available: http://expressjs.com. [Accessed: 02-May-2016].
27. “MongoDB for GIANT Ideas,” MongoDB [Online]. Available: https://mongodb.org. [Accessed: 02-May-2016].
28. “Material Design for Bootstrap,” Material Design for Bootstrap [Online]. Available: http://fezvrasta.github.io/bootstrap-material-design/. [Accessed: 02-May-2016].
29. “Introduction - Material Design,” Google Design Guidelines [Online]. Available: https://www.google.com/design/spec/material-design/introduction.html. [Accessed: 02-May-2016].
30. Code
31. KanKun Smart Plug
    1. kankut\_root/etc/config/network

config interface 'loopback'

option ifname 'lo'

option proto 'static'

option ipaddr '127.0.0.1'

option netmask '255.0.0.0'

config globals 'globals'

option ula\_prefix 'fd05:dd32:ff6f::/48'

config interface 'lan'

option ifname 'eth0'

option type 'bridge'

option proto 'static'

option ipaddr '192.168.10.253'

option netmask '255.255.255.0'

option ip6assign '60'

config interface 'wwan'

option ifname 'wlan0'

option proto 'dhcp'

* 1. kankut\_root/etc/config/wireless

config wifi-device 'radio0'

option type 'mac80211'

option channel '11'

option hwmode '11ng'

option path 'platform/ar933x\_wmac'

option htmode 'HT20'

list ht\_capab 'SHORT-GI-20'

list ht\_capab 'SHORT-GI-40'

list ht\_capab 'RX-STBC1'

list ht\_capab 'DSSS\_CCK-40'

option disabled '0'

option country 'CN'

config wifi-iface 'wwan'

option device 'radio0'

option network 'wwan'

option ssid 'ECCE Students'

option mode 'sta'

option encryption 'psk-mixed'

option key 'eccestudents'

* 1. kankut\_root/www/cgi-bin/name

default\_name

* 1. kankut\_root/www/cgi-bin/relay.cgi

#!/bin/sh

# HTTP Header

echo "Content-Type: application/json"

echo "Access-Control-Allow-Origin: \*"

echo "Cache-Control: no-cache, must-revalidate"

echo "Expires: Sat, 26 Jul 1997 05:00:00 GMT"

echo "Status: 200 OK"

echo # Required for separating header

RELAY\_CTRL=/sys/class/leds/tp-link:blue:relay/brightness

MAC\_ADDRESS\_FILE=/sys/devices/virtual/net/br-lan/address

NAME\_FILE=/www/cgi-bin/name

NAME=`cat $NAME\_FILE`

RESULT=OK

case $QUERY\_STRING in

state)

case `cat $RELAY\_CTRL` in

0) RESULT=OFF

;;

1) RESULT=ON

;;

esac

;;

on)

# Turn on

echo 1 > $RELAY\_CTRL

;;

off)

# Turn off

echo 0 > $RELAY\_CTRL

;;

toggle)

# Toggle

case `cat $RELAY\_CTRL` in

0) echo 1 > $RELAY\_CTRL

;;

1) echo 0 > $RELAY\_CTRL

;;

esac

;;

\*)

# Default

RESULT="Unknown command: $QUERY\_STRING"

;;

esac

echo -n "{\"name\":\"$NAME\","

echo -n "\"mac\_address\":\"`cat $MAC\_ADDRESS\_FILE`\","

echo -n "\"state\":`cat $RELAY\_CTRL`,"

echo "\"result\":\"$RESULT\"}"

* 1. kankut\_root/www/cgi-bin/set\_name.cgi

#!/bin/sh

# HTTP Header

echo "Content-Type: application/json"

echo "Access-Control-Allow-Origin: \*"

echo "Cache-Control: no-cache, must-revalidate"

echo "Expires: Sat, 26 Jul 1997 05:00:00 GMT"

echo "Status: 200 OK"

echo # Required for separating header

NAME\_FILE=/www/cgi-bin/name

NAME=`cat $NAME\_FILE`

echo $QUERY\_STRING > $NAME\_FILE

echo -n "{\"name\":\"$QUERY\_STRING\","

echo "\"previous\_name\":\"$NAME\"}"

* 1. kankun\_setup.sh

#!/bin/sh

if [ $# -eq 0 ]

then

echo "Error: No IP Address argument supplied."

exit 1

fi

echo "Copying files to KanKun Smart Plug..."

echo "-> The password is p9z34c."

scp -r kankun\_root/\* root@$1:/

echo "Finished copying files.\n"

echo "Rebooting..."

ssh root@$1 'reboot' # comment this out when debugging

echo "Reboot started. Wait for KanKun to connect to the specified network."

* 1. kankun\_setup\_webserver.sh

#!/bin/sh

if [ $# -eq 0 ]

then

echo "Error: No IP Address argument supplied."

exit 1

fi

echo "Copying web server files..."

echo "-> The password is p9z34c."

scp -r kankun\_root/www root@$1:/

echo "Finished copying web server files.\n"

* 1. README.md

# KanKun Smart Plug Web Server

A web server for hacking the KanKun Smart Plug

# To-do

- [x] JSON format for structured data

- [x] Script for easy install of repo to KanKun Smart Plug

# Usage

## Login Credentials

- username: root

- password: p9z34c

## Setting up KanKun Smart Plug

1. Edit `network` and `wireless` files located in `etc/config/` to the desired configuration.

- `wireless` contains settings for connecting the smart plug to a wireless network

- `network` contains settings for IP address of wireless network

2. Run `./kanun\_setup.sh` and follow the instructions.

- You have to know the IP address of your KanKun Smart Plug first

- You can find them using `nmap` or by looking at the client list of your router.

- If you're looking at the client list of your router, KanKun's Physical/MAC Address may start with `00:15:61` since this is its vendor ID.

1. Sensor Block
   1. arduino/sendTorPi/sendTorPi.ino

void setup() {

// initialize serial communication at 9600 bits per second

Serial.begin(9600);

}

void loop() {

// read the input on analog pin 0

int sensorValue = analogRead(A0);

// print out the value you read

Serial.println(sensorValue/4);

delay(100);

}

* 1. mongodb/data/db/README.md

# MongoDB Database path

This directory is required for the MongoDB database of the rpi-server.

* 1. routes/api.js

var express = require('express');

var router = express.Router();

var db = require('../db');

var eventHandler = require('../event-handler');

var hubInfo = require('../hub-info');

var deviceInfo = require('../device-info')

// GET: /api

router.get('/', (req, res) => {

var host = req.get('Host');

var protocol = req.protocol;

var baseURL = `${protocol}://${host}/`;

var response = {

status\_url: baseURL + 'api/status',

events\_url: baseURL + 'api/events',

add\_event\_url: baseURL + 'api/events/add'

};

res.setHeader('Content-Type', 'application/json');

res.send(JSON.stringify(response, null, 4));

});

router.get('/status', (req, res) => {

res.setHeader('Content-Type', 'application/json');

var response = {

mac\_address: "ff:ff:ff:00:00:01",

variables\_available: [

"temperature",

"switch"

]

};

res.send(JSON.stringify(response));

});

// GET: /api/device

router.get('/device', (req, res) => {

res.setHeader('Content-Type', 'application/json');

deviceInfo.getDeviceInfo((info) => {

res.send(JSON.stringify(info));

});

});

// GET: /api/events

router.get('/events', (req, res) => {

res.setHeader('Content-Type', 'application/json');

var eventsCollection = db.get().collection('events');

eventsCollection.find().toArray((err, docs) => {

res.send(JSON.stringify(docs));

});

});

// POST: /api/events

router.post('/events', (req, res) => {

res.setHeader('Content-Type', 'application/json');

var values = {

condition: req.body.condition,

variable: req.body.variable

};

if (values.condition != null && values.variable != null) {

var eventsCollection = db.get().collection('events');

eventsCollection.find(values).toArray((err, docs) => {

if (docs.length == 0) {

eventsCollection.insertOne(values, (err, result) => {

if (err) {

res.send(JSON.stringify({ result: "Error: Failed to add new event" }));

} else {

res.send(JSON.stringify({ result: "Success: Added new event" }));

}

});

} else {

res.send(JSON.stringify({ result: "Success: Event already exists" }));

}

});

} else {

res.send(JSON.stringify({ result: "Error: 'condition' and 'variable' should be set" }));

}

});

// POST: /api/events/debug/trigger

router.post('/events/debug/trigger', (req, res) => {

res.setHeader('Content-Type', 'application/json');

var values = {

value: req.body.value,

variable: req.body.variable

};

if (values.value == null || values.variable == null) {

res.send(JSON.stringify({ result: "Error: 'value' and 'variable' should be set" }));

return

}

var eventsTriggered = [];

hubInfo.getHubInfo((info) => {

var targetURL = `http://${info.ip\_address}/api/event/trigger`

eventHandler.handleCurrentEvent(values, (event) => {

eventsTriggered.push(event);

}, () => {

res.send(JSON.stringify({

result: "Success: Current event handled",

events\_triggered: eventsTriggered,

targetURL: targetURL

}, null, 4));

});

});

});

// GET: /api/hub/info

router.get('/hub/info', (req, res) => {

res.setHeader('Content-Type', 'application/json');

hubInfo.getHubInfo((info) => {

res.send(JSON.stringify(info));

});

});

// POST: /api/hub/info

router.post('/hub/info', (req, res) => {

res.setHeader('Content-Type', 'application/json');

// Save only when ip\_address is available

var input = req.body

var response = {}

response.input = input

if (input.ip\_address) {

hubInfo.saveHubInfo(input, (err) => {

if (err == null) {

response.result = "Successfully saved"

} else {

response.result = "Failed to save"

}

res.send(JSON.stringify(response));

});

} else {

response.result = "ip\_address is required"

res.send(JSON.stringify(response));

}

});

module.exports = router;

* 1. app.js

var serialport = require("serialport");

var express = require('express');

var bodyParser = require('body-parser');

var debug = require('debug')('rpi-server:app');

var app = express();

var SerialPort = serialport.SerialPort;

var eventHandler = require('./event-handler');

// MongoDB

var db = require('./db');

db.connect('mongodb://localhost:27017/rpi', function (err) {

if (err) {

debug('Unable to connect to MongoDB.');

process.exit(1);

} else {

debug('Connected to MongoDB.');

}

});

// Arduino

var ser = new SerialPort("/dev/ttyUSB0", {

baudrate: 9600,

parser: serialport.parsers.readline("\n")

});

var previousData = 0;

ser.on("open", function () {

debug('open');

ser.on('data', function(data) {

if (previousData != data) {

debug(data);

previousData = data;

}

var event = {

value: data,

variable: 'temperature'

};

eventHandler.handleCurrentEvent(event, (doc) => {}, () => {});

});

});

ser.on('error', (err) => {

debug(err);

});

// Web server

// -- routes

var api = require('./routes/api');

// -- setup

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: false }));

app.use((req, res, next) => {

res.setHeader('Access-Control-Allow-Origin', '\*');

res.setHeader('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept');

next();

});

// -- add routes

app.use('/api', api);

const port = 80;

app.listen(port, function () {

debug(`RPi listening on port ${port}!`);

});

* 1. db.js

var MongoClient = require('mongodb').MongoClient;

var state = {

db: null

};

module.exports.connect = function (url, done) {

if (state.db) {

return done();

}

MongoClient.connect(url, function (err, db) {

if (err) {

return done(err);

}

state.db = db;

done()

});

}

module.exports.get = function () {

return state.db;

}

module.exports.close = function (done) {

if (state.db) {

state.db.close(function (err, result) {

state.db = null;

state.mode = null;

done(err);

});

}

}

* 1. device-info.js

var fs = require('fs');

function getDeviceInfo(callback) {

fs.readFile(`${\_\_dirname}/device-info.json`, 'utf8', (err, data) => {

var hubInfo = JSON.parse(data);

callback(hubInfo)

});

}

function saveDeviceInfo(info, callback) {

fs.writeFile(`${\_\_dirname}/device-info.json`, JSON.stringify(info, null, 4), callback);

}

module.exports = {

getDeviceInfo: getDeviceInfo,

saveDeviceInfo: saveDeviceInfo

};

* 1. device-info.json

{

"mac\_address": "ff:ff:fe:00:00:01"

}

* 1. event-handler.js

var request = require('request');

var debug = require('debug')('rpi-server:event-handler');

var db = require('./db');

var hubInfo = require('./hub-info')

var deviceInfo = require('./device-info')

var currentValues = [];

function handleCurrentEvent(event, callback, doneCallback) {

if (event.value == null || event.variable == null) {

return

}

// Get hub info

hubInfo.getHubInfo((info) => {

deviceInfo.getDeviceInfo((device) => {

if (info == null) { return }

var targetURL = `http://${info.ip\_address}/api/event/trigger`

var currentValue = parseInt(event.value);

var values = {

variable: event.variable

};

var eventsCollection = db.get().collection('events');

eventsCollection.find(values).toArray((err, docs) => {

docs.forEach((doc) => {

var condition = doc.condition;

var key = condition + "\_" + doc.variable; // Key for saving/accessing previous value

var operator = condition.charAt(0) || '=';

var targetValue = parseInt(condition.substr(1) || '0');

var conditionSatisfied = false

switch (operator) {

case '<':

if (currentValue < targetValue) {

conditionSatisfied = true;

}

break;

case '=':

if (currentValue == targetValue) {

conditionSatisfied = true;

}

break;

case '>':

if (currentValue > targetValue) {

conditionSatisfied = true;

}

break;

default:

break;

}

if (conditionSatisfied) {

var previousValue = currentValues[key] || false;

currentValues[key] = true;

// If previous value is false, trigger, else ignore since it was already triggerred

// Essentially, trigger only on positive-edge (when false -> true)

if (!previousValue) {

debug('Trigger Event:');

debug({ condition: condition, variable: doc.variable });

// Send trigger event to hub

var postData = {

condition: condition,

variable: doc.variable,

mac\_address: device.mac\_address || '00:00:01:00:00:01'

};

debug(`Target URL: ${targetURL}`);

debug(`POST Data: ${JSON.stringify(postData, null, 4)}`);

request.post({ url: targetURL, form: postData }, (err, httpResponse, body) => {

if (err) {

debug(err);

return

}

debug("POST Success!");

debug(body);

});

callback(doc);

}

} else {

currentValues[key] = false;

}

});

doneCallback();

});

});

});

}

module.exports = {

handleCurrentEvent: handleCurrentEvent

}

* 1. hub-info.js

var fs = require('fs');

function getHubInfo(callback) {

fs.readFile(`${\_\_dirname}/hub-info.json`, 'utf8', (err, data) => {

var hubInfo = JSON.parse(data);

callback(hubInfo)

});

}

function saveHubInfo(info, callback) {

fs.writeFile(`${\_\_dirname}/hub-info.json`, JSON.stringify(info, null, 4), callback);

}

module.exports = {

getHubInfo: getHubInfo,

saveHubInfo: saveHubInfo

};

* 1. hub-info.json

{

"ip\_address": "localhost:3000"

}

* 1. nodemon.json

{

"verbose": true,

"ignore": ["hub-info.json", "mongodb/"]

}

* 1. package.json

{

"name": "rpi-arduino-sensor-hub",

"version": "1.0.0",

"description": "files to interface nodeJS, rPi and Arduino",

"main": "app.js",

"dependencies": {

"body-parser": "^1.15.0",

"express": "^4.13.4",

"mongodb": "^2.1.8",

"request": "^2.69.0",

"serialport": "^2.0.6"

},

"devDependencies": {},

"scripts": {

"nodemon-debug": "sudo DEBUG=rpi-server:\* nodemon app.js",

"test": ""

},

"repository": {

"type": "git",

"url": "git+ssh://git@gitlab.com/programmable-home-automation/rpi-arduino-sensor-hub.git"

},

"author": "Christian Bermejo",

"license": "ISC",

"bugs": {

"url": "https://gitlab.com/programmable-home-automation/rpi-arduino-sensor-hub/issues"

},

"homepage": "https://gitlab.com/programmable-home-automation/rpi-arduino-sensor-hub#README"

}

* 1. README.md

# RPi Arduino Sensor Block

## Usage

1. Run `npm install` to install dependencies

- Run `./start-db.sh` to start mongo database

- Make sure to have `nodemon` installed and run `npm run nodemon-port-80` to start Sensor Block

* 1. start-db.sh

mongod --dbpath mongodb/data/db

1. Main Hub
   1. bin/www

#!/usr/bin/env node

/\*\*

\* Module dependencies.

\*/

var app = require('../app');

var debug = require('debug')('hub-web-server:server');

var http = require('http');

/\*\*

\* Get port from environment and store in Express.

\*/

var port = normalizePort(process.env.PORT || '3000');

app.set('port', port);

/\*\*

\* Create HTTP server.

\*/

var server = http.createServer(app);

/\*\*

\* Socket.io

\*/

var io = app.io;

io.attach(server);

/\*\*

\* Listen on provided port, on all network interfaces.

\*/

server.listen(port);

server.on('error', onError);

server.on('listening', onListening);

/\*\*

\* Normalize a port into a number, string, or false.

\*/

function normalizePort(val) {

var port = parseInt(val, 10);

if (isNaN(port)) {

// named pipe

return val;

}

if (port >= 0) {

// port number

return port;

}

return false;

}

/\*\*

\* Event listener for HTTP server "error" event.

\*/

function onError(error) {

if (error.syscall !== 'listen') {

throw error;

}

var bind = typeof port === 'string'

? 'Pipe ' + port

: 'Port ' + port;

// handle specific listen errors with friendly messages

switch (error.code) {

case 'EACCES':

console.error(bind + ' requires elevated privileges');

process.exit(1);

break;

case 'EADDRINUSE':

console.error(bind + ' is already in use');

process.exit(1);

break;

default:

throw error;

}

}

/\*\*

\* Event listener for HTTP server "listening" event.

\*/

function onListening() {

var addr = server.address();

var bind = typeof addr === 'string'

? 'pipe ' + addr

: 'port ' + addr.port;

debug('Listening on ' + bind);

}

* 1. mongodb/data/db/README.md

# MongoDB Database path

This directory is required for the MongoDB database of the hub-web-server.

* 1. public/css/homeautomation.css

/\*

body CSS taken from Bootstrap Navbar template example

card CSS taken from http://bootsnipp.com/snippets/featured/material-cards

scrollable menu CSS taken from http://stackoverflow.com/questions/19227496/scrollable-menu-with-bootstrap-3-menu-expanding-its-container-when-it-should-n

slidebar CSS taken from http://www.bootply.com/EOgJSIzPGS#

Snippets License (MIT license)

Copyright (c) 2013 Bootsnipp.com

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated

documentation files (the "Software"), to deal in the Software without restriction, including without limitation

the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and

to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,

INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR

PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE

FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,

ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

\*/

body {

min-height: 1000px;

padding-top: 70px;

/\* slidebar \*/

overflow-x: hidden;

}

/\* slidebar \*/

.slidebar-form {

padding: 7px 15px 13px 15px;

}

.slidebar-nav {

left: 250px;

list-style: none;

height: 100%;

margin: 0;

margin-left: -250px;

overflow-y: auto;

padding: 0;

position: fixed;

top: 50px;

width: 0;

z-index: 1000;

-webkit-transition: 0.35s ease;

-moz-transition: 0.35s ease;

-o-transition: 0.35s ease;

transition: 0.35s ease;

}

#wrapper.toggled .slidebar-nav {

width: 250px;

}

.slidebar-nav .navbar-collapse {

padding: 0;

max-height: none;

}

.slidebar-nav ul {

float: none;

width: 100%;

}

.slidebar-nav ul:not {

display: block;

}

.slidebar-nav li {

float: none;

display: block;

}

.slidebar-nav li a {

padding-top: 12px;

padding-bottom: 12px;

}

.slidebar-nav .open .dropdown-menu {

position: static;

float: none;

width: auto;

margin: 0;

padding: 5px 0;

background-color: transparent;

border: 0;

-webkit-box-shadow: none;

box-shadow: none;

}

.slidebar-nav .open .dropdown-menu > li > a {

padding: 5px 15px 5px 25px;

}

.slidebar-nav .navbar-brand {

width: 100%;

}

.slidebar-nav .open > a > b.caret {

border-top: none;

border-bottom: 4px solid;

}

.slidebar-nav .navbar-nav {

margin: 0;

}

#wrapper {

padding-left: 0;

-webkit-transition: 0.35s ease;

-moz-transition: 0.35s ease;

-o-transition: 0.35s ease;

transition: 0.35s ease;

}

#wrapper.toggled {

padding-left: 250px;

}

#page-wrapper6 {

width: 100%;

position: absolute;

padding: 15px;

top: 50px;

}

#wrapper.toggled #page-wrapper6 {

position: absolute;

margin-right: -250px;

}

@media(min-width:768px) {

#wrapper {

padding-left: 250px;

}

#wrapper.toggled {

padding-left: 0;

}

#page-wrapper6 {

padding: 20px;

position: relative;

}

#wrapper.toggled #page-wrapper6 {

position: relative;

margin-right: 0;

}

.slidebar-nav {

width: 250px;;

}

#wrapper.toggled .slidebar-nav {

width: 0;

}

}

.slidebar-toggle {

position: relative;

float: left;

padding: 9px 10px;

margin-top: 8px;

margin-left: 15px;

margin-bottom: 8px;

background-color: transparent;

background-image: none;

border: 0;

border-radius: 4px;

}

.slidebar-toggle:focus {

outline: 0;

background-color: transparent;

}

.slidebar-toggle .icon-bar {

display: block;

width: 22px;

height: 2px;

border-radius: 1px;

background-color: inherit;

border: 1px solid;

}

.slidebar-toggle .icon-bar + .icon-bar {

margin-top: 4px;

}

.navbar-default .slidebar-toggle {

border-color: transparent;

border: 0;

}

.navbar-default .slidebar-toggle:hover,

.navbar-default .slidebar-toggle:focus {

background-color: transparent;

}

.navbar-default .slidebar-toggle .icon-bar {

background-color: inherit;

border: 1px solid;

}

/\* scrollable menu \*/

.scrollable-menu {

height: auto;

max-height: 200px;

overflow-x: hidden;

}

/\* card \*/

.mat-card {

box-shadow: 0 2px 5px 0 rgba(0, 0, 0, 0.16), 0 2px 10px 0 rgba(0, 0, 0, 0.12);

}

.mat-card {

margin-top: 10px;

box-sizing: border-box;

border-radius: 2px;

background-clip: padding-box;

background-color: white;

width: 100%;

}

.mat-card span.mat-card-title {

color: #fff;

font-size: 24px;

font-weight: 300;

text-transform: uppercase;

}

.mat-card .mat-card-image {

position: relative;

overflow: hidden;

}

.mat-card .mat-card-image img {

border-radius: 2px 2px 0 0;

background-clip: padding-box;

position: relative;

z-index: -1;

}

.mat-card .mat-card-image span.mat-card-title {

position: absolute;

bottom: 0;

left: 0;

padding: 16px;

}

.mat-card .mat-card-content {

padding: 16px;

border-radius: 0 0 2px 2px;

background-clip: padding-box;

box-sizing: border-box;

}

.mat-card .mat-card-content p {

margin: 0;

color: inherit;

}

.mat-card .mat-card-content span.mat-card-title {

line-height: 48px;

}

.mat-card .mat-card-action {

border-top: 1px solid rgba(160, 160, 160, 0.2);

padding: 16px;

}

.mat-card .mat-card-action a {

color: #ffab40;

margin-right: 16px;

transition: color 0.3s ease;

text-transform: uppercase;

}

.mat-card .mat-card-action a:hover {

color: #ffd8a6;

text-decoration: none;

}

/\*Added custom multilevel list for workflows

http://www.bootply.com/DglnYJTSKA

\*/

.list-group.panel > .list-group-item {

border-bottom-right-radius: 4px;

border-bottom-left-radius: 4px

}

.list-group-submenu {

margin-left:20px;

}

/\*

http://stackoverflow.com/questions/18075794/bootstrap-table-without-stripe-borders/24830641

\*/

.table-borderless tbody tr td, .table-borderless tbody tr th, .table-borderless thead tr th {

border: none;

border-spacing: 0;

border-collapse: collapse;

padding: 8px;

padding-left: 8px;

}

/\*

used for modal pop-up scrollable-menu

\*/

.drop-menu {

max-height: 400px;

overflow: hidden;

overflow-y: auto;

}

* 1. public/css/index.css

\* {

box-sizing: border-box;

}

.btn {

margin: 0px 1px

}

.header-panel {

background-color: #009587;

height: 144px;

position: relative;

z-index: 3;

}

.header-panel div {

position: relative;

height: 100%;

}

.header-panel h1 {

color: #FFF;

/\*font-size: 20px;\*/

/\*font-weight: 400;\*/

position: absolute;

bottom: 10px;

padding-left: 35px;

}

.menu {

overflow: auto;

padding: 0;

}

.menu, .menu \* {

-webkit-user-select: none;

-moz-user-select: none;

user-select: none;

}

.menu ul {

padding: 0;

margin: 7px 0;

}

.menu ul li {

list-style: none;

padding: 20px 0 20px 50px;

font-size: 15px;

font-weight: normal;

cursor: pointer;

}

.menu ul li.active {

background-color: #dedede;

position: relative;

}

.navbar-default .navbar-nav > .active > a.sidebar\_item,

.navbar-default .navbar-nav > .active > a.sidebar\_item:hover,

.navbar-default .navbar-nav > .active > a.sidebar\_item:focus {

background-color: transparent;

}

.menu ul li a {

color: rgb(51, 51, 51);

text-decoration: none;

}

.pages {

position: absolute;

top: 0;

right: 0;

z-index: 4;

padding: 0;

overflow: auto;

}

.pages > div {

padding: 0 5px;

padding-top: 64px;

}

.pages .header {

color: rgb(82, 101, 162);

}

.page {

transform: translateY(1080px);

transition: transform 0 linear;

display: none;

opacity: 0;

font-size: 16px;

}

.page.active {

transform: translateY(0px);

transition: all 0.3s ease-out;

display: block;

opacity: 1;

}

.opensource {

color: rgba(0, 0, 0, 0.62);

position: fixed;

margin-top: 50px;

margin-left: 50px;

z-index: 100;

}

#source-modal h4 {

color: black;

}

#paypal .btn {

padding: 5px 30px 6px 30px;

}

#paypal input {

background: transparent;

border: 0;

}

.cbwrapper div {

display: none;

}

.cbwrapper div:nth-child(2) {

display: block;

}

#carbonads, #fakecb {

border: 1px solid #d5d5d5;

font-size: 11px;

line-height: 15px;

overflow: hidden;

width: 340px;

padding: 20px;

margin: auto;

height: 142px;

border-radius: 2px;

}

#carbonads .carbon-img {

float: left;

display: block;

}

#carbonads .carbon-text, #carbonads .carbon-poweredby {

float: left;

width: 150px;

padding: 0 10px 10px 10px;

}

#carbonads .carbon-text:hover, #carbonads .carbon-poweredby:hover {

text-decoration: none;

}

#carbonads .carbon-poweredby {

color: #9D9D9D;

}

#shadow .sample {

width: 100px;

height: 100px;

margin: 16px;

padding: 16px;

display: inline-block;

}

#shadow-sample2 {

display: inline-block;

width: 100px;

height: 100px;

margin: 16px;

padding: 16px;

cursor: pointer;

-webkit-user-select: none;

-moz-user-select: none;

user-select: none;

}

#shadow-sample3 {

display: inline-block;

width: 100px;

height: 100px;

margin: 16px;

padding: 16px;

border-radius: 100px;

cursor: pointer;

-webkit-user-select: none;

-moz-user-select: none;

user-select: none;

}

#input .inputs {

width: 80%;

}

* 1. public/js/sensor-finder.js

var foundDevices = {};

var socket = io();

function findDevices() {

var address = $('#hub\_ip\_address').text();

console.log("IP Address:");

console.log(address);

var components = address.split('.');

for (i = 1; i < 255; i++) {

var targetAddress = [components[0], components[1], components[2], i].join('.');

fetchAddress(targetAddress);

}

function fetchAddress(address) {

var url = 'http://' + address + '/api/device';

// TODO: Use XMLHttpRequest (firefox fails to fetch for some reason)

$.ajax({

dataType: 'json',

url: url,

timeout: 20000,

success: function (data) {

data.ip\_address = address;

handleData(data);

}

});

}

function handleData(data) {

foundDevices[data.ip\_address] = data;

console.log('Received!');

console.log(JSON.stringify(data));

var $row = $('<tr>');

var $macColumn = $('<td>').text(data.mac\_address);

var $ipColumn = $('<td>').text(data.ip\_address);

var $addColumn = $('<td>');

var $button = $('<button class="btn btn-raised btn-default btn-sm">').text('Add');

$button.click(function () {

socket.emit('add sensor', data);

$button.text('Added!');

});

$addColumn.append($button);

$row.append($macColumn);

$row.append($ipColumn);

$row.append($addColumn);

$('#device\_list').append($row);

}

}

findDevices();

* 1. public/js/sensor-remover.js

var socket = io();

// Bind elements with .remove-device even after DOM creation

$(document).on('click', '.remove-device', function () {

removeDevice(this.id);

// <tr> element

var rowElement = this.parentNode.parentNode;

rowElement.remove();

})

function removeDevice(id) {

console.log('Removing ' + id);

socket.emit('remove sensor', id);

}

* 1. public/js/smartplug-finder.js

var foundDevices = {};

var socket = io();

function findDevices() {

var address = $('#hub\_ip\_address').text();

console.log("IP Address:");

console.log(address);

var components = address.split('.');

for (i = 1; i < 255; i++) {

var targetAddress = [components[0], components[1], components[2], i].join('.');

fetchAddress(targetAddress);

}

}

function fetchAddress(address) {

var url = 'http://' + address + '/cgi-bin/relay.cgi?state';

// TODO: Use XMLHttpRequest (firefox fails to fetch for some reason)

$.ajax({

dataType: 'json',

url: url,

timeout: 20000,

success: function (data) {

data.ip\_address = address;

handleData(data);

}

});

}

function handleData(data) {

foundDevices[data.ip\_address] = data;

console.log('Received!');

console.log(JSON.stringify(data));

var $row = $('<tr>');

var $macColumn = $('<td>').text(data.mac\_address);

var $ipColumn = $('<td>').text(data.ip\_address);

var stateText = data.state == 1 ? 'ON' : 'OFF';

var $stateColumn = $('<td>').text(stateText);

var $addColumn = $('<td>');

var $button = $('<button class="btn btn-raised btn-default btn-sm">').text('Add');

$button.click(function () {

delete data.state;

delete data.result;

delete data.name;

socket.emit('add smartplug', data);

$button.text('Added!');

});

$addColumn.append($button);

$row.append($macColumn);

$row.append($ipColumn);

$row.append($stateColumn);

$row.append($addColumn);

$('#device\_list').append($row);

}

findDevices();

* 1. public/js/smartplug-remover.js

var socket = io();

// Bind elements with .remove-device even after DOM creation

$(document).on('click', '.remove-device', function () {

removeDevice(this.id);

// <tr> element

var rowElement = this.parentNode.parentNode;

rowElement.remove();

})

function removeDevice(id) {

console.log('Removing ' + id);

socket.emit('remove smartplug', id);

}

* 1. public/js/toggle.js

// Update states of toggles based on device state

$(".toggle-switch").each(function (index) {

updateState(this);

});

$(".toggle-switch").click(function() {

check(this)

});

function sendRelay(ipAddress, cmd) {

var request = new XMLHttpRequest();

var url = "http://" + ipAddress + "/cgi-bin/relay.cgi?" + cmd

request.open('GET', url, true);

var oldValue = !$("[id='" + ipAddress + "']")[0].checked

// Timeout

request.timeout = 4500;

// Download

request.onreadystatechange = function() {

// window.alert(ipAddress + ", status: " + request.status);

if (request.readyState == 4) {

if (request.status == 200) {

// Status OK

var toggle = false

var responseObject = JSON.parse(request.responseText);

var state = responseObject.state

// window.alert("<DEBUG> Success! IP: " + ipAddress + " State: " + state);

if (state == 1) {

toggle = true;

}

$("[id='" + ipAddress + "']")[0].checked = toggle;

} else {

// Failed

window.alert("Failed to toggle " + ipAddress);

$("[id='" + ipAddress + "']")[0].checked = oldValue;

}

}

}

// Send request

request.send();

}

function check(checkbox) {

var command = 'off'

if (checkbox.checked) {

command = 'on'

}

sendRelay(checkbox.id, command);

}

function updateState(checkbox) {

sendRelay(checkbox.id, 'state');

}

* 1. public/js/workflow-man.js

var socket = io();

// used when scraping appropriate sensor MAC address when adding workflows

var mac\_address = "";

var editWorkflowID = "";

$(document).on('click', '.to-edit-workflow', function () {

toEditWorkflow(editWorkflowID);

})

$(document).on('click', '.edit-workflow', function () {

finishEditWorkflow(editWorkflowID);

})

$(document).on('click', '.add-workflow', function () {

addWorkflow(this.id);

})

$(document).on('click', '.remove-workflow', function () {

removeWorkflow(this.id);

// <tr> element

var rowElement = this.parentNode.parentNode;

rowElement.remove();

})

function removeWorkflow(id) {

console.log('Removing ' + id);

socket.emit('remove workflow', id);

}

function addWorkflow(id) {

// TODO: Better way to scrape data

var condition = document.getElementsByTagName("input")[1].value + document.getElementsByTagName("input")[2].value;

var newWorkflow = {

from\_id: mac\_address,

to\_id: document.getElementsByTagName("input")[4].value,

command: document.getElementsByTagName("input")[3].value,

condition: condition,

variable: document.getElementsByTagName("input")[0].value

};

console.log('Adding new Workflow' + id);

socket.emit('add workflow', newWorkflow);

}

function toEditWorkflow(id) {

console.log('Editing workflow with ObjectId: ' + id);

socket.emit('to edit workflow', id);

}

function finishEditWorkflow(id) {

var condition = document.getElementsByTagName("input")[6].value + document.getElementsByTagName("input")[7].value;

var newWorkflow = {

\_id: id,

from\_id: mac\_address,

to\_id: document.getElementsByTagName("input")[9].value,

command: document.getElementsByTagName("input")[8].value,

condition: condition,

variable: document.getElementsByTagName("input")[5].value

};

socket.emit('edit workflow', newWorkflow);

}

// Scrape appropriate sensor MAC address when adding a new workflow

function getMacAddress(i) {

mac\_address = i;

}

function getWorkflowToEdit(i) {

editWorkflowID = i;

}

* 1. routes/add-sensors.js

var debug = require('debug')('hub-web-server:server');

var express = require('express');

var db = require('../db');

var dns = require('dns');

var os = require('os');

var http = require('http');

var request = require('request');

var router = express.Router();

module.exports = function (io, host\_address) {

/\* GET users listing. \*/

router.get('/', function(req, res, next) {

dns.lookup(os.hostname(), function (err, address, fam) {

if (err) {

return;

}

var context = {

active: 'Devices',

title: 'Devices',

hub\_ip\_address: address

};

res.render('add-sensors', context);

});

});

io.on('connection', function (socket) {

debug('<SENSORS> connected');

socket.on('add sensor', function (data) {

debug('Received Add Sensor');

data.mac\_address = data.mac\_address.toLowerCase()

data.\_id = data.mac\_address

debug(data);

addSensor(data);

});

socket.on('disconnect', function () {

debug('<SENSORS> disconnected');

});

});

return router;

};

function addSensor(data) {

db.get().collection('sensors').updateOne({ "\_id": data.\_id },

{

$set: data

},

{

upsert: true

},

function (err, result) {

if (err) {

debug("Insert failed:");

debug(err);

} else {

debug("Successfully added/updated sensor");

// Get own IP Address

dns.lookup(os.hostname(), function (err, address, fam) {

if (err) {

return;

}

// Set hub ip address using URL

var targetURL = `http://${data.ip\_address}/api/hub/info`;

var postData = {

ip\_address: address

};

debug(`TargetURL: ${targetURL}`);

debug(`POST Data: ${JSON.stringify(postData)}`);

request.post({ url: targetURL, form: postData }, (err, httpResponse, body) => {

if (err) {

return

}

debug('Saved hub IP Address to sensor block');

debug(body);

});

});

}

});

}

* 1. routes/add-smartplugs.js

var debug = require('debug')('hub-web-server:server');

var express = require('express');

var db = require('../db');

var dns = require('dns');

var os = require('os');

var http = require('http');

var router = express.Router();

module.exports = function (io, host\_address) {

/\* GET users listing. \*/

router.get('/', function(req, res, next) {

dns.lookup(os.hostname(), function (err, address, fam) {

if (err) {

return;

}

var context = {

active: 'Devices',

title: 'Devices',

hub\_ip\_address: address

};

res.render('add-smartplugs', context);

});

});

io.on('connection', function (socket) {

debug('<DEVICES> connected');

socket.on('add smartplug', function (data) {

debug('Received Add Device');

data.mac\_address = data.mac\_address.toLowerCase()

data.\_id = data.mac\_address

debug(data);

addSmartplug(data);

});

socket.on('disconnect', function () {

debug('<DEVICES> disconnected');

});

});

return router;

};

function addSmartplug(data) {

db.get().collection('smartplugs').updateOne({ "\_id": data.\_id },

{

$set: data

},

{

upsert: true

},

function (err, result) {

if (err) {

debug("Insert failed:");

debug(err);

} else {

debug("Successfully added/updated smartplug");

}

});

}

* 1. routes/api.js

var express = require('express');

var debug = require('debug')('hub-web-server:api');

var http = require('http');

var router = express.Router();

var db = require('../db');

/\* GET home page. \*/

router.get('/', function(req, res, next) {

res.setHeader('Content-Type', 'application/json');

var response = {

status: "ok"

};

res.send(JSON.stringify(response));

});

router.get('/smartplugs', function (req, res, next) {

res.setHeader('Content-Type', 'application/json');

var smartplugCollection = db.get().collection('smartplugs');

smartplugCollection.find().toArray(function(err, docs) {

res.send(JSON.stringify(docs))

});

});

router.get('/workflows', function (req, res, next) {

res.setHeader('Content-Type', 'application/json');

findWorkflows({}, function (docs) {

res.send(JSON.stringify(docs))

});

});

router.post('/event/trigger', function(req, res, next) {

res.setHeader('Content-Type', 'application/json');

var values = {};

values.from\_id = req.body.mac\_address.toLowerCase();

values.condition = req.body.condition;

values.variable = req.body.variable;

findWorkflows(values, function (docs) {

runWorkflows(docs);

res.send(JSON.stringify(docs));

});

});

function findWorkflows(values, callback) {

var workflowCollection = db.get().collection('workflows');

workflowCollection.find(values).toArray(function (err, docs) {

callback(docs);

});

}

function runWorkflows(workflows) {

var count = workflows.length;

for (var i = 0; i < count; i++) {

handleWorkflow(workflows[i]);

}

}

function handleWorkflow(workflow) {

var smartplugCollection = db.get().collection('smartplugs');

smartplugCollection.find({"\_id": workflow.to\_id}).toArray(function (err, docs) {

if (docs.length > 0) {

debug("Found target device!");

var device = docs[0];

handleDevice(docs[0], workflow);

}

});

}

function handleDevice(device, workflow) {

var ip\_address = device.ip\_address;

var command = workflow.command;

var url = "http://" + ip\_address + "/cgi-bin/relay.cgi?" + command;

debug("URL: " + url);

var options = {

host: ip\_address,

path: '/cgi-bin/relay.cgi?' + command,

agent: false

}

var request = http.get(options);

request.on('error', function (err) {

debug(err);

});

request.setTimeout(20000, function () {

request.abort();

});

}

module.exports = router;

* 1. routes/index.js

var express = require('express');

var router = express.Router();

/\* GET home page. \*/

router.get('/', function(req, res, next) {

var context = {

active: 'Home',

title: 'Dashboard',

};

res.render('index', context);

});

module.exports = router;

* 1. routes/sensors.js

var debug = require('debug')('hub-web-server:server');

var express = require('express');

var router = express.Router();

var db = require('../db');

module.exports = function (io) {

/\* GET Smartplugs list. \*/

router.get('/', function(req, res, next) {

var sensorCollection = db.get().collection('sensors');

sensorCollection.find().toArray(function (err, docs) {

var context = {

active: 'Home',

sidebar\_active: 'Sensors',

title: 'Sensors',

list: docs

};

res.render('sensors', context);

});

});

io.on('connection', function (socket) {

debug('<SENSORS> connected');

socket.on('remove sensor', function (id) {

removeSensor(id);

});

socket.on('disconnect', function () {

debug('<SENSORS> disconnected');

});

});

return router;

}

function removeSensor(id) {

var sensorCollection = db.get().collection('sensors');

sensorCollection.deleteOne({

"\_id": id

}, function (err, results) {

if (err) {

debug("Failed to delete " + id);

} else {

debug("Successfully deleted Sensor with id: " + id);

}

});

}

* 1. routes/smartplugs.js

var debug = require('debug')('hub-web-server:server');

var express = require('express');

var router = express.Router();

var db = require('../db');

module.exports = function (io) {

/\* GET Smartplugs list. \*/

router.get('/', function(req, res, next) {

var smartplugCollection = db.get().collection('smartplugs');

smartplugCollection.find().toArray(function (err, docs) {

var context = {

active: 'Home',

sidebar\_active: 'Smartplugs',

title: 'Smartplugs',

list: docs

};

res.render('smartplugs', context);

});

});

io.on('connection', function (socket) {

debug('<SMARTPLUGS> connected');

socket.on('remove smartplug', function (id) {

removeSmartplug(id);

});

socket.on('disconnect', function () {

debug('<SMARTPLUGS> disconnected');

});

});

return router;

}

function removeSmartplug(id) {

var smartplugCollection = db.get().collection('smartplugs');

smartplugCollection.deleteOne({

"\_id": id

}, function (err, results) {

if (err) {

debug("Failed to delete " + id);

} else {

debug("Successfully deleted Smartplug with id: " + id);

}

});

}

* 1. routes/workflows.js

var debug = require('debug')('hub-web-server:server');

var express = require('express');

var request = require('request');

var router = express.Router();

var db = require('../db');

var mongodb = require('mongodb');

var editedWorkflow = [];

module.exports = function(io) {

/\* GET smartplugs listing. \*/

router.get('/', function(req, res, next) {

// Get collections

var smartplugCollection = db.get().collection('smartplugs');

var sensorsCollection = db.get().collection('sensors');

var workflowCollection = db.get().collection('workflows');

// Fetch docs

smartplugCollection.find().toArray(function (err, smartplugs) {

sensorsCollection.find().toArray(function(err, sensors) {

workflowCollection.find().toArray(function(err, workflows) {

var deviceWorkflows = {};

// Initialize all workflows for each sensor

sensors.forEach((sensor) => {

var mac\_address = sensor.mac\_address

if (deviceWorkflows[mac\_address] == null) {

deviceWorkflows[mac\_address] = {};

}

var currentDeviceWorkflow = deviceWorkflows[mac\_address];

// Initialize workflow if needed - redundancy

if (currentDeviceWorkflow.workflows == null) {

currentDeviceWorkflow.workflows = [];

}

});

// Get device workflows

workflows.forEach((workflow) => {

// Get currentDeviceWorkflow, key is mac\_address

if (deviceWorkflows[workflow.from\_id] == null) {

deviceWorkflows[workflow.from\_id] = {};

}

var currentDeviceWorkflow = deviceWorkflows[workflow.from\_id];

// Initialize workflow if needed - redundancy

if (currentDeviceWorkflow.workflows == null) {

currentDeviceWorkflow.workflows = [];

}

// Save workflow

var transformedWorkflow = {

\_id: workflow.\_id,

to\_id: workflow.to\_id,

condition: workflow.condition,

variable: workflow.variable,

command: workflow.command

};

currentDeviceWorkflow.workflows.push(transformedWorkflow);

// Save currentDeviceWorkflow

deviceWorkflows[workflow.from\_id] = currentDeviceWorkflow;

});

// Debug

console.log("Device Workflows:");

console.log(JSON.stringify(deviceWorkflows, null, 4));

// Pass collections and transformed workflows

var context = {

active: 'Workflows',

title: 'Workflows',

smartplugs: smartplugs,

sensors: sensors,

workflows: workflows,

deviceWorkflows: deviceWorkflows,

};

res.render('workflows', context);

});

});

});

});

io.on('connection', function (socket) {

debug('<WORKFLOWS> connected');

socket.on('remove workflow', function (id) {

removeWorkflow(id);

});

socket.on('add workflow', function(id) {

addWorkflow(id);

});

socket.on('edit workflow', function (id) {

console.log("received edit workflow workflowjs");

editWorkflow(id);

});

socket.on('to edit workflow', function(id) {

db.get().collection('workflows').find({

"\_id" : mongodb.ObjectId(id)

}).toArray(function (err, results) {

if (err) {

debug("Failed to locate workflow with ObjectId: " + id);

} else {

socket.emit('to edit workflow', results);

console.log(results);

debug("Successfully located workflow with ObjectId: " + id);

}

});

});

socket.on('disconnect', function () {

debug('<WORKFLOWS> disconnected');

});

});

return router;

}

function removeWorkflow(id) {

db.get().collection('workflows').deleteOne({

"\_id": mongodb.ObjectId(id)

}, function (err, results) {

if (err) {

debug("Failed to delete " + id);

} else {

debug("Successfully deleted Workflow with ObjectId: " + id);

}

});

}

function addWorkflow(workflow) {

// Insert workflow

db.get().collection('workflows').insert(workflow, function (err, results) {

if (err) {

debug("Failed to add " + workflow);

} else {

debug("Successfully added Workflow: " + JSON.stringify(workflow));

saveEventToSensor(workflow);

}

});

}

function editWorkflow(data) {

// Update workflow

db.get().collection('workflows').updateOne({ "\_id": mongodb.ObjectId(data.\_id)},

{

$set: {

"from\_id": data.from\_id,

"to\_id": data.to\_id,

"command": data.command,

"condition": data.condition,

"variable": data.variable

}

}, function (err, result) {

if (err) {

debug("Insert failed:");

debug(err);

} else {

debug("Successfully updated workflow");

saveEventToSensor(data);

}

});

}

function saveEventToSensor(workflow) {

// Fetch sensor data to get IP address

db.get().collection('sensors').find({ \_id: workflow.from\_id.toLowerCase() }).toArray((err, docs) => {

if (docs.length > 0) {

var sensor = docs[0];

debug(`Found sensor: ${JSON.stringify(sensor)}`);

// Add event to sensor

var sensorURL = `http://${sensor.ip\_address}/api/events`;

var postData = {

condition: workflow.condition,

variable: workflow.variable

}

debug(`SensorURL: ${sensorURL}`);

debug(`POST Data: ${JSON.stringify(postData)}`);

request.post({ url: sensorURL, form: postData }, (err, httpResponse, body) => {

if (err) {

debug('Failed to save event to sensor block!');

return

}

debug('Saved event to sensor block');

debug(body);

});

} else {

debug('No sensor found!');

}

});

}

* 1. views/partials/navigation.html

<nav class="navbar navbar-default navbar-fixed-top">

<div class="navbar-header">

<button href="#menu-toggle" class="slidebar-toggle" id="menu-toggle">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-responsive-collapse">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="/">Home Automation</a>

</div>

<div class="navbar-collapse collapse navbar-responsive-collapse">

<ul class="nav navbar-nav">

<!-- TODO: Fix hyperlink and active on js pages' -->

<li {% if active == 'Home' %}class="active"{% endif %}><a href="/">Dashboard</a></li>

<li {% if active == 'Add Smartplugs' %}class="active"{% endif %}><a href="/add-smartplugs">Add Smartplugs</a></li>

<li {% if active == 'Add Sensors' %}class="active"{% endif %}><a href="/add-sensors">Add Sensors</a></li>

<li {% if active == 'Workflows' %}class="active"{% endif %}><a href="/workflows">Workflows</a></li>

</ul>

</div>

</nav>

* 1. views/partials/slidenav.html

<!-- Sidebar -->

<div class="slidebar-nav">

<nav class="navbar-default menu" role="navigation">

<ul class="nav navbar-nav">

<li class="withripple {% if sidebar\_active == 'Smartplugs' %}active{% endif %}" ><a class="sidebar\_item" href="smartplugs">Smartplugs</a><div class="ripple-container"></div></li>

<li class="withripple {% if sidebar\_active == 'Sensors' %}active{% endif %}" ><a class="sidebar\_item" href="sensors">Sensors<div class="ripple-container"></a></div></li>

</ul>

</nav><!--/.navbar -->

</div>

* 1. views/add-sensors.html

{% extends 'layout.html' %}

{% block title %}{{ title }}{% endblock %}

{% block content %}

{% include './partials/navigation.html' %}

<div class="well">

<h3 class="header"><center>Search Sensors</center></h3>

<table id="device\_list" class="table table-hover">

<tr>

<th>MAC Address</th>

<th>IP Address</th>

<th>Options</th>

</tr>

</table>

</div>

<div class="debug-element">

<span>&lt;DEBUG&gt;: </span><span id="hub\_ip\_address">{{ hub\_ip\_address }}</span>

</div>

<script src="/socket.io/socket.io.js"></script>

<script src="/js/sensor-finder.js"></script>

{% endblock %}

* 1. views/add-smartplugs.html

{% extends 'layout.html' %}

{% block title %}{{ title }}{% endblock %}

{% block content %}

{% include './partials/navigation.html' %}

<div class="well">

<h3 class="header"><center>Search Smartplugs</center></h3>

<table id="device\_list" class="table table-hover">

<tr>

<th>MAC Address</th>

<th>IP Address</th>

<th>State</th>

<th>Options</th>

</tr>

</table>

</div>

<div class="debug-element">

<span>&lt;DEBUG&gt;: </span><span id="hub\_ip\_address">{{ hub\_ip\_address }}</span>

</div>

<script src="/socket.io/socket.io.js"></script>

<script src="/js/smartplug-finder.js"></script>

{% endblock %}

* 1. views/error.html

{% extends 'layout.html' %}

{% block title %}{{ title }}{% endblock %}

{% block content %}

<h1>{{ message }}</h1>

<h2>{{ error.status }}</h2>

<pre>

{{ error.stack }}

</pre>

{% endblock %}

* 1. views/index.html

{% extends 'layout.html' %}

{% block title %}{{ title }}{% endblock %}

{% block content %}

{% include './partials/navigation.html' %}

<div id="wrapper">

{% include './partials/slidenav.html' %}

<main id="page-wrapper6 pages">

<div class="well page" id="getting1" style="display: none;">

<h3 class="header"><center>Temperature</center></h3>

<center>

Temp here!

</center>

</div>

<div class="well page" id="getting2" style="display: none;">

<h3 class="header"><center>Humidity</center></h3>

<center>

Humid here!

</center>

</div>

<div class="well page" id="getting3" style="display: none;">

<h3 class="header"><center>Lights</center></h3>

<center>

Lights here!

</center>

</div>

<div class="well page" id="getting4" style="display: none;">

<h3 class="header"><center>Motion</center></h3>

<center>

Motion here!

</center>

</div>

<div class="well page" id="toggle-button" style="display: none;">

<h1 class="header">Toggle Button</h1>

<div class="togglebutton">

<label>

Wi-Fi

<input type="checkbox" checked=""><span class="toggle"></span>

</label>

</div>

<div class="togglebutton">

<label>

Bluetooth

<input type="checkbox"><span class="toggle"></span>

</label>

</div>

</div>

</main>

</div>

<!-- /#wrapper -->

<!-- Placed at the end of the document so the pages load faster -->

<script>

window.page = window.location.hash || "#about";

$(document).ready(function () {

if (window.page != "#about") {

$(".menu").find("li[data-target=" + window.page + "]").trigger("click");

}

});

$(window).on("resize", function () {

$("html, body").height($(window).height());

$(".main, .menu").height($(window).height() - $(".header-panel").outerHeight());

$(".pages").height($(window).height());

}).trigger("resize");

$(".menu li").click(function () {

// Menu

if (!$(this).data("target")) return;

if ($(this).is(".active")) return;

$(".menu li").not($(this)).removeClass("active");

$(".page").not(page).removeClass("active").hide();

window.page = $(this).data("target");

var page = $(window.page);

window.location.hash = window.page;

$(this).addClass("active");

page.show();

var totop = setInterval(function () {

$(".pages").animate({scrollTop: 0}, 0);

}, 1);

setTimeout(function () {

page.addClass("active");

setTimeout(function () {

clearInterval(totop);

}, 1000);

}, 100);

});

function cleanSource(html) {

var lines = html.split(/\n/);

lines.shift();

lines.splice(-1, 1);

var indentSize = lines[0].length - lines[0].trim().length,

re = new RegExp(" {" + indentSize + "}");

lines = lines.map(function (line) {

if (line.match(re)) {

line = line.substring(indentSize);

}

return line;

});

lines = lines.join("\n");

return lines;

}

$("#opensource").click(function () {

$.get(window.location.href, function (data) {

var html = $(data).find(window.page).html();

html = cleanSource(html);

$("#source-modal pre").text(html);

$("#source-modal").modal();

});

});

</script>

<script src="/js/toggle.js"></script>

<!-- Material Design Bootstrap js -->

<script src="/js/material.js"></script>

<script src="/js/ripples.min.js"></script>

<!-- Initiate MD Bootstrap -->

<script>

jQuery(document).ready(function() {

jQuery.material.init();

});

</script>

<script>

$("#menu-toggle").click(function(e){

e.preventDefault();$("#wrapper").toggleClass("toggled");

});

</script>

{% endblock %}

* 1. views/layout.html

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>{% block title %}My Site{% endblock %}</title>

<!-- Mobile support -->

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Material Design -->

<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">

<link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Roboto:300,400,500,700" type="text/css">

<!-- Bootstrap -->

<link href="/css/bootstrap.css" rel="stylesheet" media="screen">

<!-- Bootstrap Material Design -->

<link rel="stylesheet" type="text/css" href="/css/bootstrap-material-design.css">

<link rel="stylesheet" type="text/css" href="/css/ripples.css">

<link rel="stylesheet" type="text/css" href="/css/homeautomation.css">

<link rel="stylesheet" type="text/css" href="/css/index.css">

<link rel="stylesheet" type="text/css" href="/css/jquery.dropdown.css">

<link rel="stylesheet" type="text/css" href="/css/snackbar.min.css">

<link rel="stylesheet" type="text/css" href="/css/material.css">

<!-- jQuery js -->

<script src="/js/jquery.min.js"></script>

<!-- Bootstrap js -->

<script src="/js/bootstrap.min.js"></script>

<!-- Snackbar.js -->

<script src="/js/snackbar.min.js"></script>

{% block head %} {% endblock %}

</head>

<body>

{% block content %}{% endblock %} {% block socket %} {% endblock %}

<!-- Material Design Bootstrap js -->

<script src="/js/material.js"></script>

<script src="/js/ripples.min.js"></script>

<!-- Initiate MD Bootstrap -->

<script src="/js/jquery.dropdown.js"></script>

<script>

$.material.init();

jQuery(document).ready(function() {

// jQuery.material.init();

$(".select").dropdown({"optionClass": "withripple", "autoinit" : ".select"});

// $(".select").dropdown({"optionClass": "withripple"});

});

$("#menu-toggle").click(function(e){

e.preventDefault();$("#wrapper").toggleClass("toggled");

});

</script>

</body>

</html>

* 1. views/sensors.html

{% extends 'layout.html' %}

{% block title %}

{{ title }}

{% endblock %}

{% block content %}

{% include './partials/navigation.html'%}

<div id="wrapper">

{% include './partials/slidenav.html' %}

<div class="well" id="about" style="display: block;">

<h3 class="header"><center>Sensors</center></h3>

<center>

<table class="table table-hover">

<tr>

<th>Physical Address</th>

<th>IP Address</th>

<th>Options</th>

</tr>

{% for item in list %}

<tr id="">

<td class="mac">{{ item.mac\_address }}</td>

<td class="ip">{{ item.ip\_address }}</td>

<td>

<button id="{{ item.mac\_address }}" class="remove-device btn btn-raised btn-danger btn-sm">Remove</button>

</td>

</tr>

{% endfor %}

</table>

</center>

</div>

</div>

<!-- Placed at the end of the document so the pages load faster -->

<script>

window.page = window.location.hash || "#temp";

$(window).on("resize", function () {

$("html, body").height($(window).height());

$(".main, .menu").height($(window).height() - $(".header-panel").outerHeight());

$(".pages").height($(window).height());

}).trigger("resize");

function cleanSource(html) {

var lines = html.split(/\n/);

lines.shift();

lines.splice(-1, 1);

var indentSize = lines[0].length - lines[0].trim().length,

re = new RegExp(" {" + indentSize + "}");

lines = lines.map(function (line) {

if (line.match(re)) {

line = line.substring(indentSize);

}

return line;

});

lines = lines.join("\n");

return lines;

}

$("#opensource").click(function () {

$.get(window.location.href, function (data) {

var html = $(data).find(window.page).html();

html = cleanSource(html);

$("#source-modal pre").text(html);

$("#source-modal").modal();

});

});

</script>

<script src="/socket.io/socket.io.js"></script>

<script src="/js/toggle.js"></script>

<script src="/js/sensor-remover.js"></script>

<!-- Material Design Bootstrap js -->

<script src="/js/material.js"></script>

<script src="/js/ripples.min.js"></script>

<!-- Initiate MD Bootstrap -->

<script>

jQuery(document).ready(function() {

jQuery.material.init();

});

</script>

<script>

$("#menu-toggle").click(function(e) {

e.preventDefault();$("#wrapper").toggleClass("toggled");

});

</script>

{% endblock %}

* 1. views/smartplugs.html

{% extends 'layout.html' %}

{% block title %}{{ title }}{% endblock %}

{% block content %}

{% include './partials/navigation.html' %}

<div id="wrapper">

{% include './partials/slidenav.html' %}

<div class="well" id="about" style="display: block;">

<h3 class="header"><center>Smartplugs</center></h3>

<center>

<table class="table table-hover">

<tr>

<th>Physical Address</th>

<th>IP Address</th>

<th>Status</th>

<th>Options</th>

</tr>

{% for item in list %}

<tr id="">

<td class="mac">{{ item.mac\_address }}</td>

<td class="ip">{{ item.ip\_address }}</td>

<td>

<div class="togglebutton">

<label>

<input type="checkbox" id="{{ item.ip\_address }}" class="toggle-switch">

</label>

</div>

</td>

<td>

<button id="{{ item.mac\_address }}" class="remove-device btn btn-raised btn-danger btn-sm">Remove</button>

</td>

</tr>

{% endfor %}

</table>

</center>

</div>

</div>

<!-- Placed at the end of the document so the pages load faster -->

<script>

window.page = window.location.hash || "#temp";

$(window).on("resize", function () {

$("html, body").height($(window).height());

$(".main, .menu").height($(window).height() - $(".header-panel").outerHeight());

$(".pages").height($(window).height());

}).trigger("resize");

function cleanSource(html) {

var lines = html.split(/\n/);

lines.shift();

lines.splice(-1, 1);

var indentSize = lines[0].length - lines[0].trim().length,

re = new RegExp(" {" + indentSize + "}");

lines = lines.map(function (line) {

if (line.match(re)) {

line = line.substring(indentSize);

}

return line;

});

lines = lines.join("\n");

return lines;

}

$("#opensource").click(function () {

$.get(window.location.href, function (data) {

var html = $(data).find(window.page).html();

html = cleanSource(html);

$("#source-modal pre").text(html);

$("#source-modal").modal();

});

});

</script>

<script src="/socket.io/socket.io.js"></script>

<script src="/js/toggle.js"></script>

<script src="/js/smartplug-remover.js"></script>

<!-- Material Design Bootstrap js -->

<script src="/js/material.js"></script>

<script src="/js/ripples.min.js"></script>

<!-- Initiate MD Bootstrap -->

<script>

jQuery(document).ready(function() {

jQuery.material.init();

});

</script>

<script>

$("#menu-toggle").click(function(e){

e.preventDefault();$("#wrapper").toggleClass("toggled");

});

</script>

{% endblock %}

* 1. views/workflows.html

{% extends 'layout.html' %}

{% block title %}{{ title }}{% endblock %}

{% block content %}

{% include './partials/navigation.html' %}

<div id="wrapper">

{% include './partials/slidenav.html' %}

<div class="well" id="about" style="display: block;">

<h3 class="header"><center>Workflows </center></h3>

<!-- <center> -->

<table class="table">

<thead>

<tr>

<th>Device Address</th>

</tr>

</thead>

<tbody>

{% for mac\_address, value in deviceWorkflows %}

<tr>

<td>

{{ mac\_address }}

<table class="table-borderless mat-card">

<thead>

<tr>

<th>Variable</th>

<th>Condition</th>

<th>Command</th>

<th>Target Device Address</th>

</tr>

</thead>

<tbody>

{% for workflow in value.workflows %}

<tr id="">

<td>{{ workflow.variable }}</td>

<td>{{ workflow.condition }}</td>

<td>{{ workflow.command }}</td>

<td>{{ workflow.to\_id }}</td>

<td>

<a href="#edit-dialog" data-toggle="modal" onclick="getWorkflowToEdit('{{ workflow.\_id.toString() }}'); getMacAddress('{{mac\_address}}');"><button id="{{ workflow.\_id.toString() }}" class="to-edit-workflow btn btn-raised btn-primary btn-sm">Edit</button></a>

</td>

<td>

<button id="{{ workflow.\_id.toString() }}" data-toggle=snackbar data-content="Removed workflow!" class="remove-workflow btn btn-raised btn-danger btn-sm">Remove</button>

</td>

</tr>

{% endfor %}

</tbody>

</table>

</td>

<!-- Add onclick data scrape mac\_address -->

<td>

<div style="text-align:left"><a href="#add-dialog" data-toggle="modal" onclick="getMacAddress('{{mac\_address}}')"><span class="glyphicon glyphicon-plus"></span></a></div>

</td>

</tr>

{% endfor %}

</tbody>

</table>

</center>

</div>

</div>

<!-- Modals -->

<div id="add-dialog" class="modal fade" tabindex="-1" style="display: none; margin:40px auto; width=600px;">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">×</button>

<h4 class="modal-title"><b>Add New Workflow</b></h4>

</div>

<div class="modal-body">

<h4>Variable</h4>

<!-- <select id="variable" class="form-control select drop-menu"> -->

<select id="variable" class="form-control select drop-menu">

<option value="temperature" class="selected">temperature</option>

<option value="light">light</option>

</select>

<h4>Condition</h4>

<table>

<tr>

<td style="padding: 4px;">

<select id="condition-sign" class="form-control select drop-menu">

<option value=">" class="selected">></option>

<option value="<"><</option>

<option value="=">=</option>

</select>

</td>

<td style="padding: 4px;">

<div class="form-group label-floating">

<label class="control-label" for="focusedInput1">Value here</label>

<input class="form-control" id="condition" type="number" min="0" step="1">

</div>

</td>

</tr>

</table>

<h4>Command</h4>

<select id="command" class="form-control select drop-menu">

<option value="toggle" class="selected">toggle</option>

<option value="on">on</option>

<option value="off">off<option>

</select>

<h4>Target Device Address</h4>

<select id="target" class="form-control select drop-menu">

{% for smartplug in smartplugs %}

<option value="{{ smartplug.mac\_address }}" class="selected">{{ smartplug.mac\_address }}</option>

{% endfor %}

</select>

<div class="modal-footer">

<button type="button" class="btn btn-primary" data-dismiss="modal">Cancel<div class="ripple-container"><div class="ripple ripple-on ripple-out" style="left: 43.65625px; top: 12px; background-color: rgb(0, 150, 136); transform: scale(10.875);"></div></div></button>

<button type="button" data-toggle=snackbar data-content="Added a new workflow!" class="add-workflow btn btn-primary" data-dismiss="modal">Add<div class="ripple-container"><div class="ripple ripple-on ripple-out" style="left: 43.65625px; top: 12px; background-color: rgb(0, 150, 136); transform: scale(10.875);"></div></div></button>

</div>

</div>

</div>

</div>

</div>

<div id="edit-dialog" class="modal fade" tabindex=""-1 style="display: none; margin:40px auto; width=600px;">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">×</button>

<h4 class="modal-title"><b>Edit Workflow</b></h4>

</div>

<div class="modal-body">

<h4>Variable</h4>

<!-- <select id="variable" class="form-control select drop-menu"> -->

<select id="variable" class="form-control select drop-menu">

<option value="temperature" class="selected">temperature</option>

<option value="light">light</option>

</select>

<h4>Condition</h4>

<table>

<tr>

<td style="padding: 4px;">

<select id="condition-sign" class="form-control select drop-menu">

<option value=">" class="selected">></option>

<option value="<"><</option>

<option value="=">=</option>

</select>

</td>

<td style="padding: 4px;">

<div class="form-group label-floating">

<label class="control-label" for="focusedInput1">Value here</label>

<input class="form-control" id="condition" type="number" min="0" step="1">

</div>

</td>

</tr>

</table>

<h4>Command</h4>

<select id="command" class="form-control select drop-menu">

<option value="toggle" class="selected">toggle</option>

<option value="on">on</option>

<option value="off">off<option>

</select>

<h4>Target Device Address</h4>

<select id="target" class="form-control select drop-menu">

{% for smartplug in smartplugs %}

<option value="{{ smartplug.mac\_address }}" class="selected">{{ smartplug.mac\_address }}</option>

{% endfor %}

</select>

<div class="modal-footer">

<button type="button" class="btn btn-primary" data-dismiss="modal">Cancel<div class="ripple-container"><div class="ripple ripple-on ripple-out" style="left: 43.65625px; top: 12px; background-color: rgb(0, 150, 136); transform: scale(10.875);"></div></div></button>

<button type="button" data-toggle=snackbar data-content="Edited workflow!" class="edit-workflow btn btn-primary" data-dismiss="modal">Edit<div class="ripple-container"><div class="ripple ripple-on ripple-out" style="left: 43.65625px; top: 12px; background-color: rgb(0, 150, 136); transform: scale(10.875);"></div></div></button>

</div>

</div>

</div>

</div>

</div>

<!-- Placed at the end of the document so the pages load faster -->

<script>

window.page = window.location.hash || "#temp";

$(window).on("resize", function () {

$("html, body").height($(window).height());

$(".main, .menu").height($(window).height() - $(".header-panel").outerHeight());

$(".pages").height($(window).height());

}).trigger("resize");

function cleanSource(html) {

var lines = html.split(/\n/);

lines.shift();

lines.splice(-1, 1);

var indentSize = lines[0].length - lines[0].trim().length,

re = new RegExp(" {" + indentSize + "}");

lines = lines.map(function (line) {

if (line.match(re)) {

line = line.substring(indentSize);

}

return line;

});

lines = lines.join("\n");

return lines;

}

$("#opensource").click(function () {

$.get(window.location.href, function (data) {

var html = $(data).find(window.page).html();

html = cleanSource(html);

$("#source-modal pre").text(html);

$("#source-modal").modal();

});

});

</script>

<script src="/socket.io/socket.io.js"></script>

<!-- <script src="/js/toggle.js"></script> -->

<script src="/js/workflow-man.js"></script>

<!-- Material Design Bootstrap js -->

<script src="/js/material.js"></script>

<script src="/js/ripples.min.js"></script>

<!-- Initiate MD Bootstrap -->

<script src="/js/jquery.dropdown.js"></script>

<!-- Client Socket.io -->

<script src="/socket.io/socket.io.js"></script>

<script>

$.material.init();

jQuery(document).ready(function() {

$(".select").dropdown({"optionClass": "withripple", "autoinit" : ".select"});

});

$("#menu-toggle").click(function(e){

e.preventDefault();$("#wrapper").toggleClass("toggled");

});

</script>

<script>

var socket = io();

socket.on('connect', function(data) {

console.log("Hello from client");

});

socket.on('to edit workflow', function(data) {

console.log("CLIENT SIDE received");

$(document).ready(function() {

// Ugly split >35 into ">" and ["", "35"]

var conditionSign = data[0].condition.split("",1);

var conditionInput = data[0].condition.split(conditionSign);

// Ugly change each value accordingly

$("#edit-dialog #variable").val(data[0].variable).change();

$("#edit-dialog #condition-sign").val(conditionSign).change();

$("#edit-dialog #condition").val(conditionInput[1]).change();

$("#edit-dialog #command").val(data[0].command).change();

$("#edit-dialog #target").val(data[0].to\_id).change();

});

});

</script>

{% endblock %}

* 1. app.js

var express = require('express');

var path = require('path');

var favicon = require('serve-favicon');

var logger = require('morgan');

var cookieParser = require('cookie-parser');

var bodyParser = require('body-parser');

var swig = require('swig');

var socket\_io = require('socket.io');

var app = express();

// MongoDB

var db = require('./db')

db.connect('mongodb://localhost:30000/hub-web-server', function (err) {

if (err) {

console.log('Unable to connect to MongoDB.');

process.exit(1);

} else {

console.log('Connected to MongoDB.');

}

});

// Socket.io

var io = socket\_io();

app.io = io;

var routes = require('./routes/index');

var api = require('./routes/api');

var smartplugs = require('./routes/smartplugs')(io);

var sensors = require('./routes/sensors')(io);

var addSmartplugs = require('./routes/add-smartplugs')(io);

var addSensors = require('./routes/add-sensors')(io);

var workflows = require('./routes/workflows')(io);

// view engine setup

app.engine('html', swig.renderFile);

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'html');

// Caching templates

var cache = true;

if (app.get('env') === 'development') {

cache = false;

}

app.set('view cache', cache);

swig.setDefaults({

cache: false

});

// uncomment after placing your favicon in /public

//app.use(favicon(path.join(\_\_dirname, 'public', 'favicon.ico')));

app.use(logger('dev'));

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: false }));

app.use(cookieParser());

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use('/', routes);

app.use('/api', api)

app.use('/smartplugs', smartplugs);

app.use('/sensors', sensors);

app.use('/add-smartplugs', addSmartplugs);

app.use('/add-sensors', addSensors);

app.use('/workflows', workflows)

// Access-Control-Allow-Origin

app.use((req, res, next) => {

res.setHeader('Access-Control-Allow-Origin', '\*');

res.setHeader('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept');

next();

});

// catch 404 and forward to error handler

app.use(function(req, res, next) {

var err = new Error('Not Found');

err.status = 404;

next(err);

});

// error handlers

// development error handler

// will print stacktrace

if (app.get('env') === 'development') {

app.use(function(err, req, res, next) {

res.status(err.status || 500);

res.render('error', {

message: err.message,

error: err

});

});

}

// production error handler

// no stacktraces leaked to user

app.use(function(err, req, res, next) {

res.status(err.status || 500);

res.render('error', {

message: err.message,

error: {}

});

});

module.exports = app;

* 1. db.js

var MongoClient = require('mongodb').MongoClient;

var state = {

db: null

};

module.exports.connect = function (url, done) {

if (state.db) {

return done();

}

MongoClient.connect(url, function (err, db) {

if (err) {

return done(err);

}

state.db = db;

done()

});

}

module.exports.get = function () {

return state.db;

}

module.exports.close = function (done) {

if (state.db) {

state.db.close(function (err, result) {

state.db = null;

state.mode = null;

done(err);

});

}

}

* 1. device-fetcher.js

var dns = require('dns');

var os = require('os');

var http = require('http');

function getDevices(handlers) {

dns.lookup(os.hostname(), function (err, address, fam) {

if (err) {

return;

}

var finishedAddressesCount = 0;

var results = [];

var fetchFinishedHandler = handlers.onFinish;

var foundDeviceHandler = handlers.onDeviceFound;

var fetchHandlers = {

forEachRequestEnd: function (success, result) {

finishedAddressesCount++;

if (success) {

results.push(result);

// Found device handler

if (foundDeviceHandler) {

foundDeviceHandler(result);

}

}

if (finishedAddressesCount >= 254) {

fetchFinishedHandler(results);

}

}

}

startFetch(address, fetchHandlers);

});

}

function startFetch(address, handlers) {

var components = address.split('.');

for (i = 1; i < 255; i++) {

var targetAddress = [components[0], components[1], components[2], i].join('.');

handleTargetAddress(targetAddress, handlers.forEachRequestEnd);

}

}

function handleTargetAddress(address, callback) {

var done = function (success, result) {

callback(success, result);

}

var options = {

host: address,

path: '/cgi-bin/relay.cgi?state',

agent: false

}

var request = http.get(options, function (res) {

var body = '';

res.on("data", function(chunk) {

body += chunk;

});

res.on('end', function () {

if (res.statusCode == '200') {

var result = JSON.parse(body);

result.ip\_address = address;

done(true, result);

} else {

done(false);

}

});

});

request.on('error', function (err) {

done(false);

});

request.setTimeout(20000, function () {

request.abort();

});

}

module.exports = {

getDevices: getDevices

}

* 1. package.json

{

"name": "hub-web-server",

"version": "0.0.0",

"private": true,

"scripts": {

"start": "node ./bin/www",

"debug": "DEBUG=hub-web-server:\* node ./bin/www",

"nodemon-debug": "DEBUG=hub-web-server:\* nodemon ./bin/www",

"nodemon-port-80": "sudo DEBUG=hub-web-server:\* PORT=80 nodemon ./bin/www"

},

"dependencies": {

"body-parser": "~1.13.2",

"cookie-parser": "~1.3.5",

"debug": "~2.2.0",

"express": "~4.13.1",

"jade": "~1.11.0",

"mongodb": "^2.1.7",

"morgan": "~1.6.1",

"request": "^2.69.0",

"serve-favicon": "~2.3.0",

"socket.io": "^1.4.5",

"socket.io-client": "^1.4.5",

"swig": "^1.4.2"

}

}

* 1. start-db.sh

mongod --dbpath mongodb/data/db --port 30000