

Bachelor's Thesis

Information and Communications Technology

2020

Trung Dang

# IOT FOR KITCHEN



BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2020 | 43

Trung Dang

## IOT FOR KITCHEN

The main purpose of this thesis was to create an IoT System capable of detecting the cooking food burnt on stove and immediately switching off the stove to make the kitchen safe. Three main sensors such as flame sensor, gas sensor and motion sensor were attached to Raspberry Pi which is a low cost, credit-card sized computer, were used to prevent potentially dangerous fires on a stove and make a kitchen safe. The system will turn off the light if there is an absence of people. Moreover, the DHT 11 sensor was used to detect the temperature and humidity in the kitchen. The developed system makes the kitchen more convenient and functional.

### KEYWORDS:

Home Assistant Raspberry Pi 3 Sensors Relay RPI Camera Buzzer

# CONTENTS

<b>CONTENTS</b>	<b>3</b>
<b>FIGURES</b>	<b>4</b>
<b>LIST OF ABBREVIATIONS</b>	<b>5</b>
<b>1 INTRODUCTION</b>	<b>6</b>
<b>2 IOT OVERVIEW</b>	<b>7</b>
2.1 The history of IoT	7
2.2 The definition of IoT and some concepts	10
<b>3 STUDY CASE FOR KITCHEN IOT</b>	<b>14</b>
3.1 Technical Requirements	14
3.2 System Architecture	14
3.2.1 Hardware Architecture	15
3.2.2 The General Home Assistant Architecture	17
3.2.3 User Interface of Home Assistant	19
3.2.4 The Architecture for Automation	21
3.3 Implementation	23
3.3.1 IP camera	23
3.3.2 Home Assistant System	25
3.4 Future Development	39
<b>4 CONCLUSION</b>	<b>41</b>
<b>REFERENCES</b>	<b>42</b>

## FIGURES

- Figure 1. History of IoT.
- Figure 2. Characteristics of IoT.
- Figure 3. Before and After Adding Fog Nodes.
- Figure 4. The Home Assistant Architecture.
- Figure 5. GPIO Pinout Diagram.
- Figure 6. Home Assistant Core (Home Control) Architecture.
- Figure 7. Interaction of Components.
- Figure 8. The Automation for Turning Off the Stove.
- Figure 9. The Automation for Switching Light.
- Figure 10. The Status of Motion Service.
- Figure 11. Scanning and Listing the Wi-Fi Access.
- Figure 12. Showing Connection.
- Figure 13. Creating a New Account.
- Figure 14. Starting SSH Service.
- Figure 15. Configuration File.
- Figure 16. The Overview of Home Assistant Front-end UI.
- Figure 17. The Automation List.
- Figure 18. Triggers for Smoke/Flame Alarm.
- Figure 19. The Conditions for Smoke/Flame Alarm.
- Figure 20. The Actions for Smoke/Flame Alarm.
- Figure 21. The Conditions for Flame/Temperature Alarm.
- Figure 22. The Trigger for Turning on The Light.
- Figure 23. The Condition and Action for Turning On the Light.
- Figure 24. The Trigger for Turning Off the Light.
- Figure 25. The Condition and Action for Turning Off the Light.
- Figure 26. Alexa Devices.

## LIST OF ABBREVIATIONS

6LoWPAN	IPv6 Low-power Wide Personal Area Networks
AI	Artificial Intelligence
BLE	Bluetooth Low Energy
HassOS	Home Assistant Operating System
IoT	Internet of Things
IPv4/6	Internet Protocol version 4/6
IP Camera	Internet Protocol Camera
LED	Light-emitting Diode
NFC	Near Field Communication
UI	User Interface
VCC	Voltage at the Common Collector

# 1 INTRODUCTION

The Internet of Things (IoT) is an umbrella term for networks, systems, and applications with connected physical objects (things) which can send and receive data, use Internet technology and can be uniquely addressed. These things include IoT devices and IoT-enabled physical assets (Internet of Things – essential IoT business guide). So far, there have been billions of Things (physical devices) connected to each other and to the Internet to gain huge benefits for every individual, business or any organization. All the information can be shared and controlled or analyzed through the Internet with some elements such as Cloud computing, networking connection.

IoT can be classified into Consumer IoT, Enterprise IoT and Industrial IoT. In this thesis topic, IoT for Kitchen is part of Consumer IoT.

This main purpose of this thesis project is to recognize a potential fire from a stove via a flame sensor and a gas sensor. When smoke is detected but there is no response from user in hassio interface – the front-end user interface of Home Assistant, the IoT platform, and no presence of people around kitchen detected from motion sensor, then the system can turn off the stove automatically. The prototype of the whole system includes a Raspberry Pi, a Flame sensor, a Gas sensor, a Motions sensor, a LED, a relay, a DHT11 sensor and a Home Assistant platform.

Firstly, the general knowledge of IoT should be learned before determining the IoT for Kitchen. Secondly, technical requirements must be listed to make sure what kind of components and software are necessary *and* essential to build the *whole* system. Next, the system architecture of the system is fulfilled in a detailed way to understand how the system works. Then implementation of the IoT system is conducted. Moreover, there is a plan for the future development based on the built IoT system. Finally, the conclusion is made to display final results.

## 2 IOT OVERVIEW

This chapter takes a look at the history of IoT and how it has been experienced in every period of time before discussing the definition of IoT and explaining how it works.

### 2.1 The history of IoT

The timeline of major IoT events [1] illustrated in Figure 1:

- In 1969, DARPA, the U.S Defense Advanced Research Projects agency developed ARPANET, predecessor of the Internet – considered as a part of IoT. It was used as a service and the first message was sent over the APRANET.
- In 1982, the members of the Carnegie-Mellon Computer Science Department connected successfully to a Coca-Cola vending machine to the Internet so people could check if cold sodas were available before going to purchase one. It was considered as one of the first IoT devices.
- In 1990, John Romkey switched on a toaster via the internet.
- In 1993, the first webcam at work was developed by engineers at University of Cambridge to take picture of a coffee machine.
- In 1995, the U.S. government completed the GPS, a big step for detecting location.
- In 1998, 128-bit IPv6 were created to provide unique identifiers for  $2^{128}$  devices.
- In 1999, the term IoT was first used by Kevin Ashton, the Head of MIT's Auto-ID labs.
- In 2000, LG introduced its first internet refrigerator.
- In 2009, the Internet-connected pacemaker, the first IoT medical device was released by St. Jude Medical Center.

- In 2010, Nest released a smart thermostat which can adjust a home's temperature automatically based on the habits of the user. It can be considered as one of the first steps to the "Smart Home" concept.
- In 2013, Google Glass was released.
- In 2014, Amazon released Echo, a smart speaker. It was the big step for home automation.
- From 2017-2020, IoT market has been more and more popular. It has been cheaper and easier to establish the system from simple one to complex one. There have been more IoT platforms for personal users or Industrial IoT.



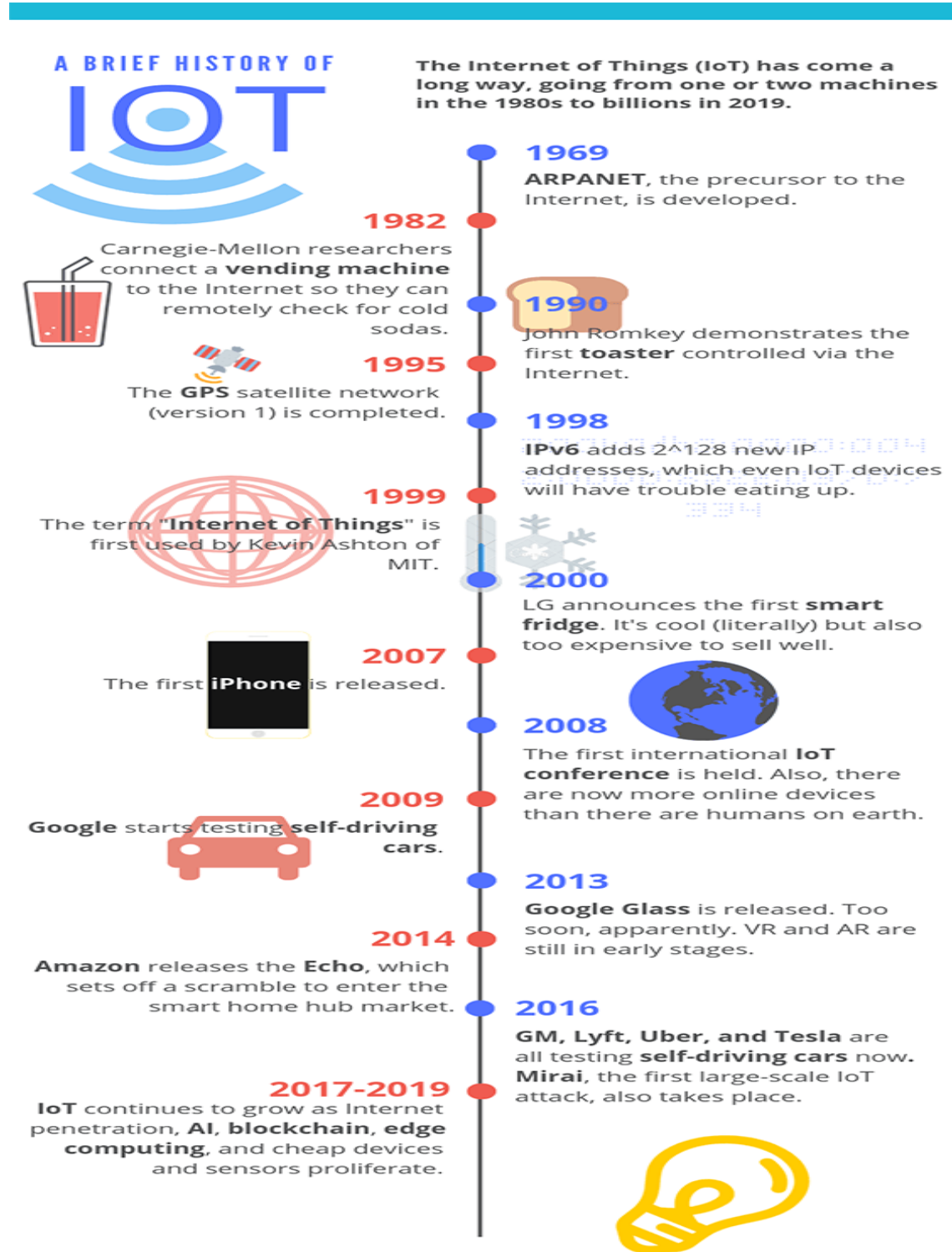


Figure 1. History of IoT.

## 2.2 The definition of IoT and some concepts

According to Nicolas Windpassinger in the book “Digitize or Die” [2], IoT is “a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies”.

In addition, there are more definitions of the IoT by some companies or organizations [2]:

- For I-SCOOP, The Internet of Things is the interconnected sphere of physical devices with the Internet and other networks through uniquely identifiable IP addresses, whereby data is gathered and communicated through embedded sensors, electronics and software.
- IBM’s definition refers to the growing range of Internet-connected devices that capture or generate an enormous amount of information every day. For consumers, these devices include mobile phones, sports wearables, home heating and air conditioning systems, and more. In an industrial setting, these devices and sensors can be found in manufacturing equipment, the supply chain, and in-vehicle components.
- According to Cisco, the Internet of Things (IoT) is the intelligent connectivity of smart devices, expected to drive massive gains in efficiency, business growth and quality of life. In other words, when objects can sense each other and communicate, it changes how and where and who makes decisions about our physical world.
- Gartner’s defines the Internet of Things (IoT) as the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment.

IoT can also be defined as having seven aspects and it is demonstrated in Figure 2:

- Things: sensors, actuators, devices or any physical objects connecting to the system can capture many forms of environment such as temperature, moisture, motion, location and so on and turn it into data to store it in cloud service. Alternatively things can make an action from control of data.

- Data is an essential part to control the system. It can be used to analyze and store for a specific purpose or as an input in the monitoring service.
- There must be a data flow going through the whole system from device to device so that is why communication protocols should be used to connect digital devices. The communication protocols can be IPv4, IPv6, 6LoWPAN, ZigBee, BLE, Z-Wave, NFC, etc.
- For Connectivity, there is a wide range of connectivity technology from wired one to wireless one or connectivity from device to device, devices to cloud...
- Ecosystem is a connection between a group of enterprises, government and consumers and all IoT components.
- For Intelligence and Action in IoT, big data is used to analyze for improving the system which can make more actions than the capacity of connected devices. Therefore, AI will be combined in IoT.
- The data created from sensors or devices will be stored and can be used for automotive features with some tools in the system.

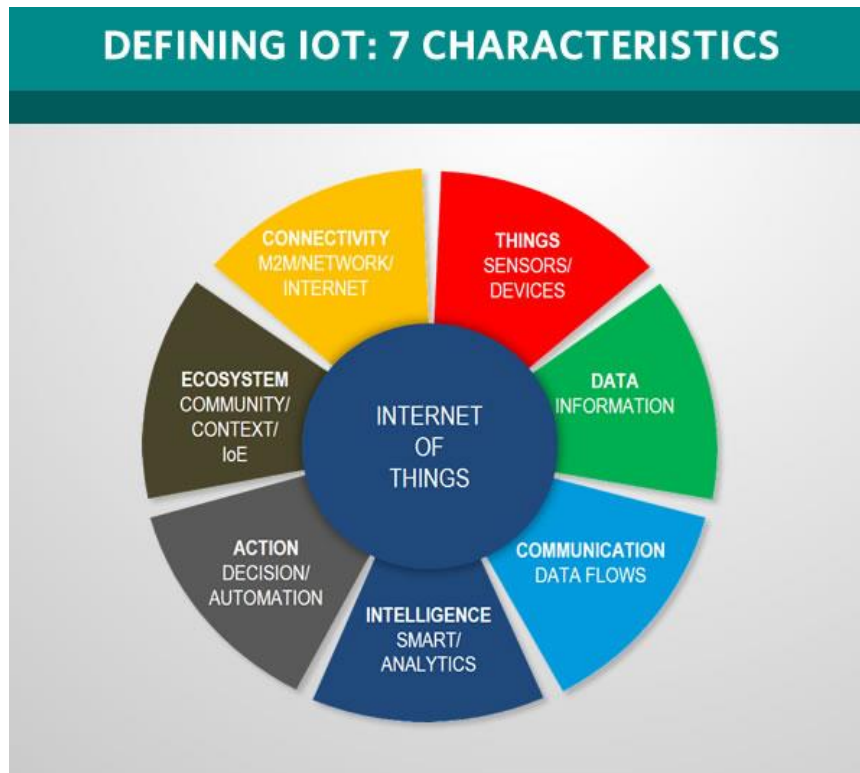


Figure 2. Characteristics of IoT [2].

### 2.3 The Cloud-Based and Fog-Based Architecture

As stated by IBM [3], Clouding Computing or Cloud is the use of computing resources - servers, databases management, data storage, networking, software applications, and special capabilities such as blockchain and artificial intelligence (AI) – over the internet, as opposed to owning and operating those resources yourself, on premises. Therefore, Cloud Computing has an important role in the IoT. Sensors or IoT- enabled devices create huge data, and cloud computing enables data flow to go through from sensors to actuators or be monitored in smart phone or web applications. The main point is the operation of cloud happens at the center while applications on the top and the network of smart things below it. There are some popular cloud platforms such as Amazon Web Services, GE Predix, Google Cloud IoT, Microsoft Azure IoT Suite, IBM Watson and Salesforce IoT Cloud.

Recently there has been a move to another architecture, Fog Computing [4] also known as Edge Computing. The sensors or IoT-connected devices will send data to a nearby edge computing device called a fog node for analysis or processing. , The purpose of fog computing to take data analysis nearest to the data source. That is why data created from the sensors can be analyzed very rapidly compared to cloud computing where its operation acts as a data center. For a fog node, there are many forms as long as a device has three capacities such as computing, storage, and connectivity. So that could be a switch, a router, or an industrial controller.

In fact, there should be a combination between cloud computing and fog computing. That depends on the data type or scale of IoT applications such as consumer IoT or industrial IoT. The combination is described in Figure 3.

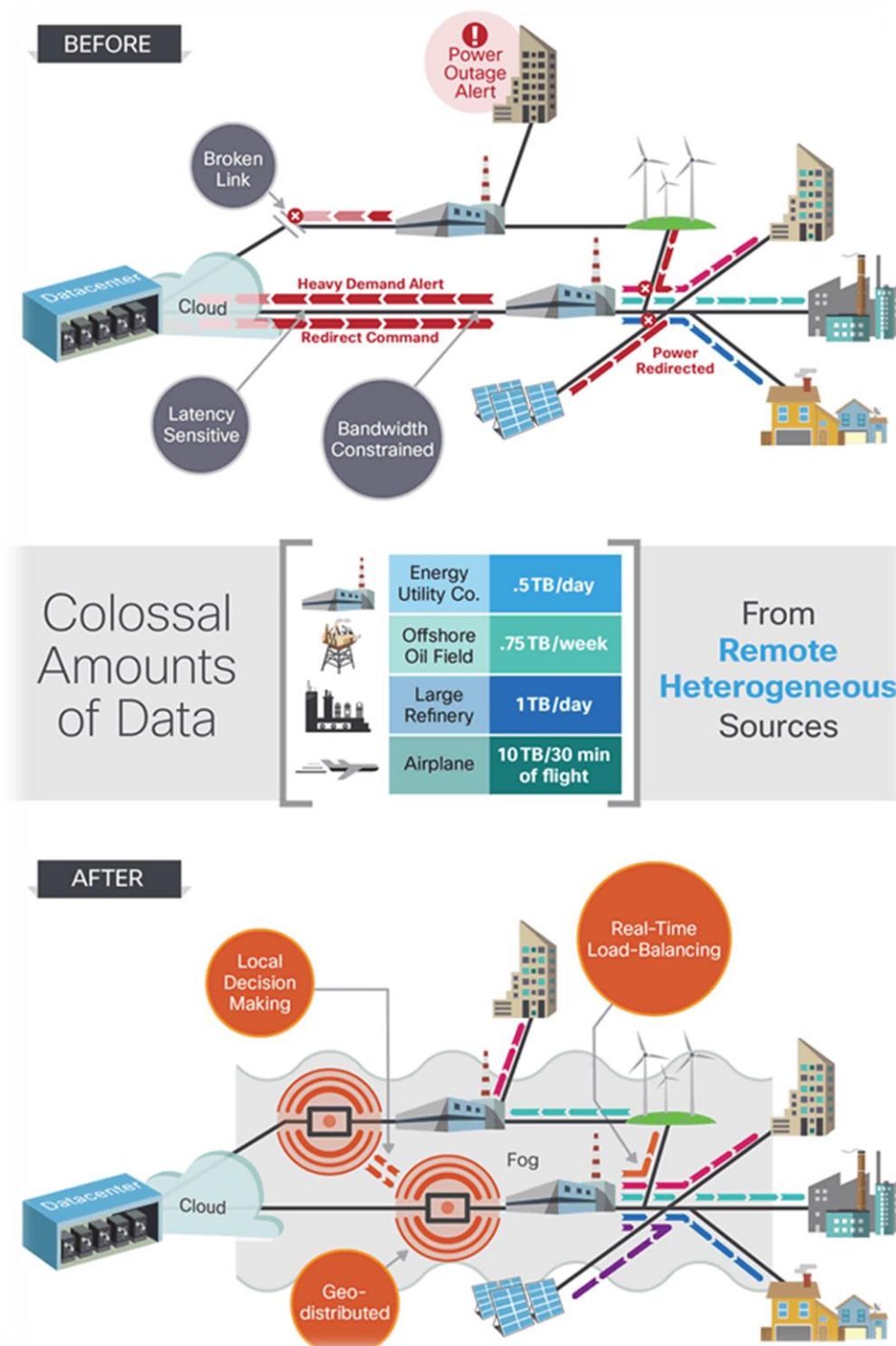


Figure 3: Before and After Adding Fog Nodes [4].

## 3 STUDY CASE FOR KITCHEN IOT

### 3.1 Technical Requirements

The MoSCoW method is used to set the priority of the requirement. It stands for Must have, Should have, Could have and Won't have.

1. The system must have two Raspberry Pi.
2. The device must have Gas Sensor, Flame Sensor and Motion Sensor.
3. The device must support the control of external relay.
4. The system must have the RPI Camera.
5. The system must have DHT11 sensor.
6. The system could have a buzzer.
7. The system could have two LEDs.

### 3.2 System Architecture

The system mainly is based on two parts of hardware and software. The software is HassOS and the hardware includes Raspberry Pi, some sensors such as flame sensor, gas sensor MQ4, DHT11 sensor and motion sensor, RPI camera, buzzer, relay, and two LEDs. The Hass.IO architecture is illustrated in Figure 4. There are three main parts: Supervisor HassIO, Home Assistant core, and Host.

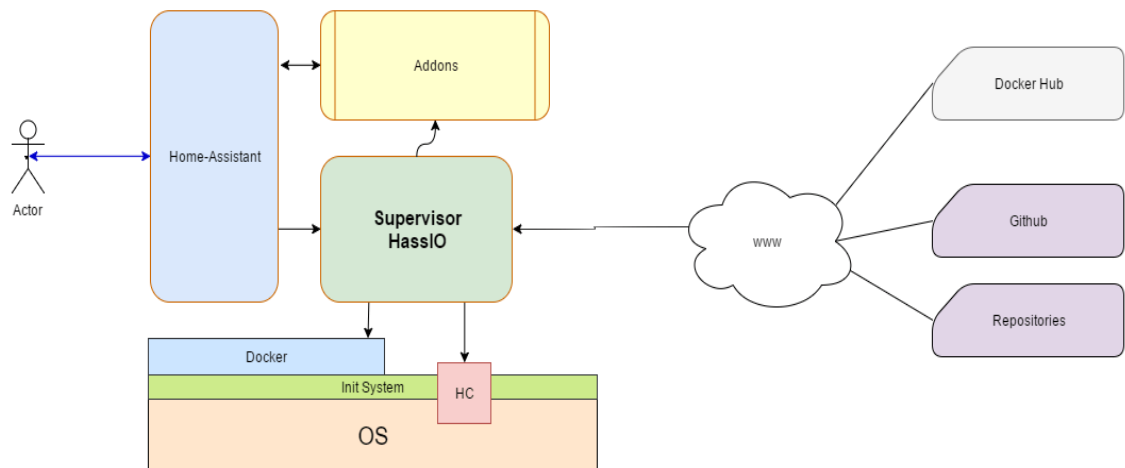


Figure 4. The Home Assistant Architecture [5].

### 3.2.1 Hardware Architecture

The idea for this project is to generate the system for keeping safe in the kitchen. The stove must be turned off if the system detects the danger from cooking without anyone around. It is possible to take LED to act like a stove, and a relay will control the LED in the same way controlling the stove. The main system is in a Raspberry Pi connected to all of the sensors, relay, two LEDs and a buzzer via GPIO ports. Pinout is shown in the figure 5 to know which ports are used to connected to devices.

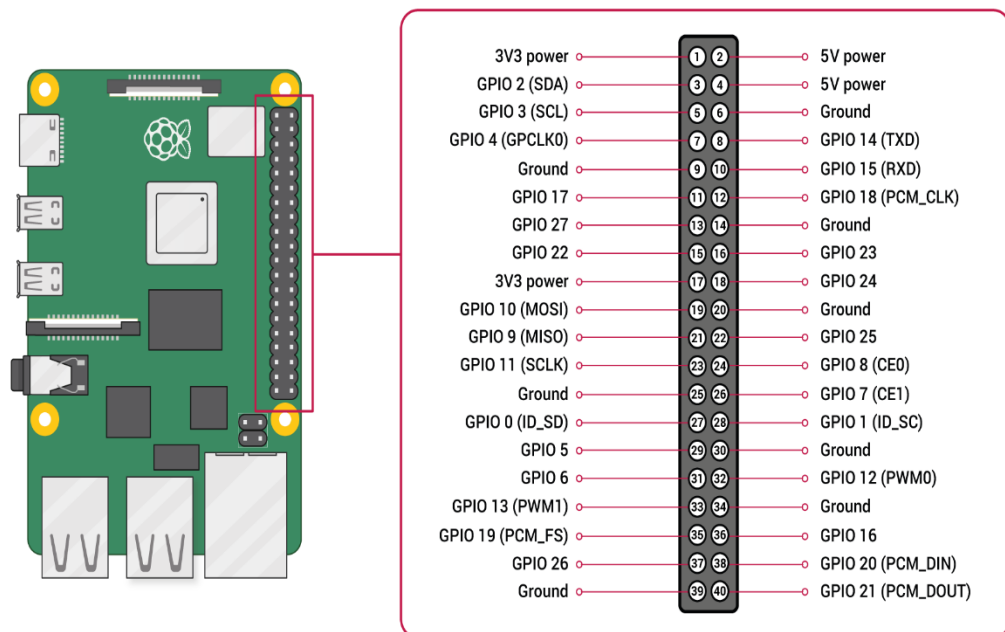


Figure 5. GPIO Pinout Diagram [6].

Specifically, it is described below:

- The gas sensor has four ports but only three ports including DO(digital port), GND (ground port) and VCC (power-supply). Digital port is connected to GPIO port 17, GND port is connected to GND port in Raspberry Pi, Vcc port is connected to 5V-power port which supplies 5 volts to the device.
- The flame sensor has similar ports as the gas sensor. There are three ports such as D0, GND, and VCC. DO port is connected to GPIO port 15, GND is connected to GND port and VCC port is connected to 5V port.
- The motion sensor is connected correspondingly. OUT port (output) goes with GPIO port 14, VCC goes with 5V port and GND goes with GND.
- In the DHT sensor, OUT is connected to GPIO port 4, VCC goes with the port 3.3V and GND goes with GND.
- Relay has three ports such as SIG (signal port), 5V(power-supply) and GND. SIG is connected to GPIO port 27, 5V port goes with 5V port and GND goes with GND. On the opposite side of ports of a relay module, there are two ports to



create a current. One port is connected to a red LED and another is connected to 3.3V port in Raspberry Pi. The 3,3V must be connected to the positive port of LED then resistor of 220 Ohms and finally the current ends in GND port.

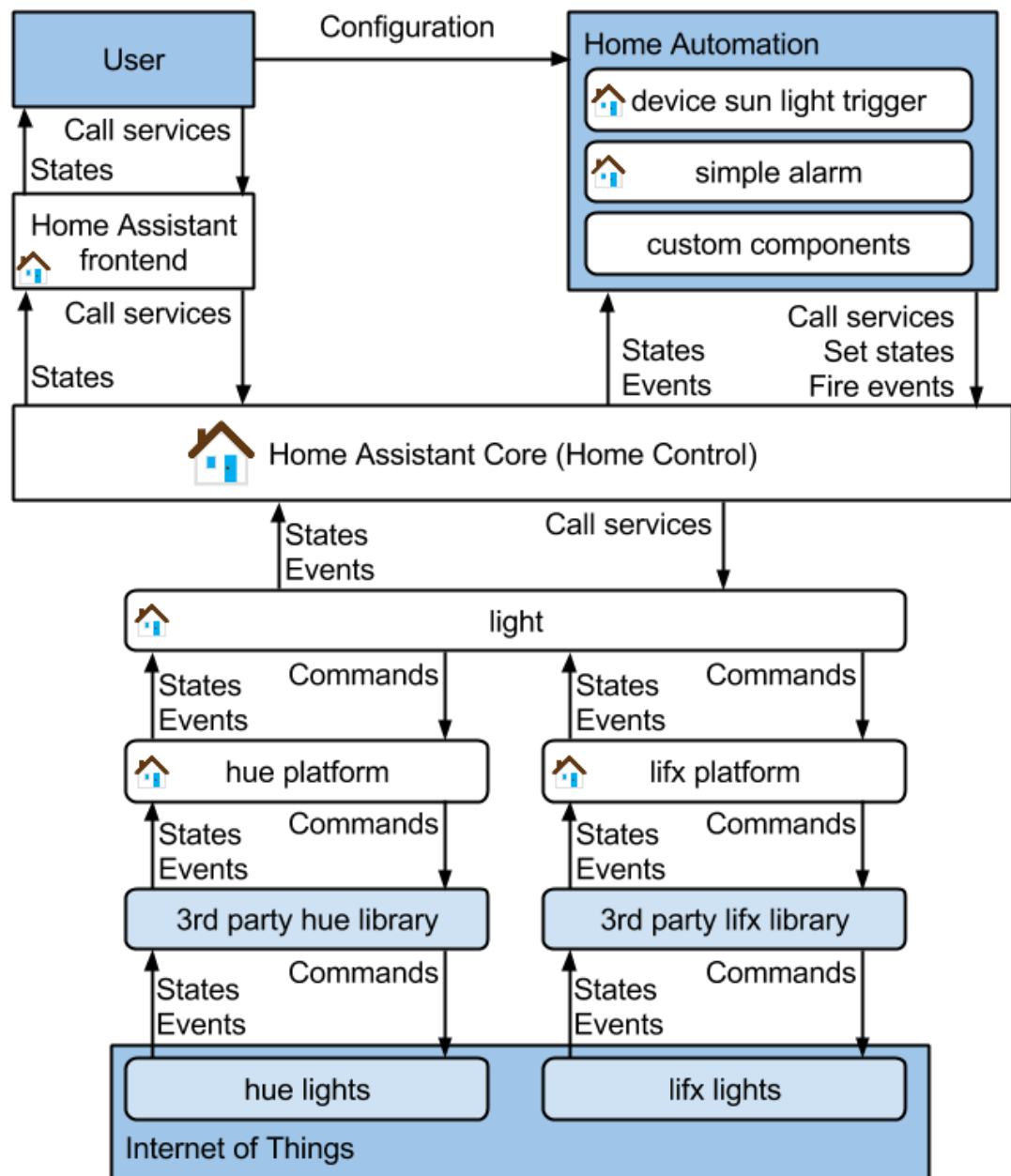
- The positive port of a green LED is connected to GPIO port 18 while a negative port is connected to a resistor of 220 Ohm then it goes to GND port.
- In buzzer, one port is connected to GPIO port 22 and another port is connected to GND.

In another Raspberry Pi, it is booted with Raspbian Operating System (Linux Distribution). RPI camera is connected to a camera port in Raspberry Pi. Because the latest version of Hass Core fails to support *rpi\_camera* platform so RPI camera cannot work well if it is connected directly to Raspberry Pi which is running HassOS. That is why it is necessary to use RPI camera in Raspberry Pi running Raspbian OS, it works as an IP camera.

### 3.2.2 The General Home Assistant Architecture

Home Assistant core takes responsibility for the Home Control. It can integrate with many devices to make automation. Configuration is performed in */config/configuration.yaml*. Users can interact with the system via a front-end website. The *Supervisor* offers an API to manage the host and running the Docker containers and Host is used as HassOS. In Home Assistant, there are also two important parts such as components and entities.

In the documents of Developer Docs [7], each component takes responsibility for a specific domain within Home Assistant. Components can listen for or trigger events, offer services, and maintain states. It is written in Python. There are two kinds of components: That interacts with an IoT domain and another responds to the event that happened within Home Assistant. Built-in components can be supported with a specific platform providing various functions, it is also called integration which is provided by Home Assistant Community. There are 1565 integrations.



 = part of the Home Assistant code base.

Figure 6. Home Assistant Core (Home Control) Architecture [7].

For example, sensor and camera component. In a sensor component, there is a platform: *dht* which supports two types of it including DHT11 and DHT22. It is similar for a camera component, there are two typical platforms: *mjpeg* for an IP camera and *rpi\_camera* for Raspberry Pi Camera.

The Entity is created by component and friendly name into a unique ID entity. It is essential to control and get data from devices. Thank to entity, the user can check the state and call the service from devices.

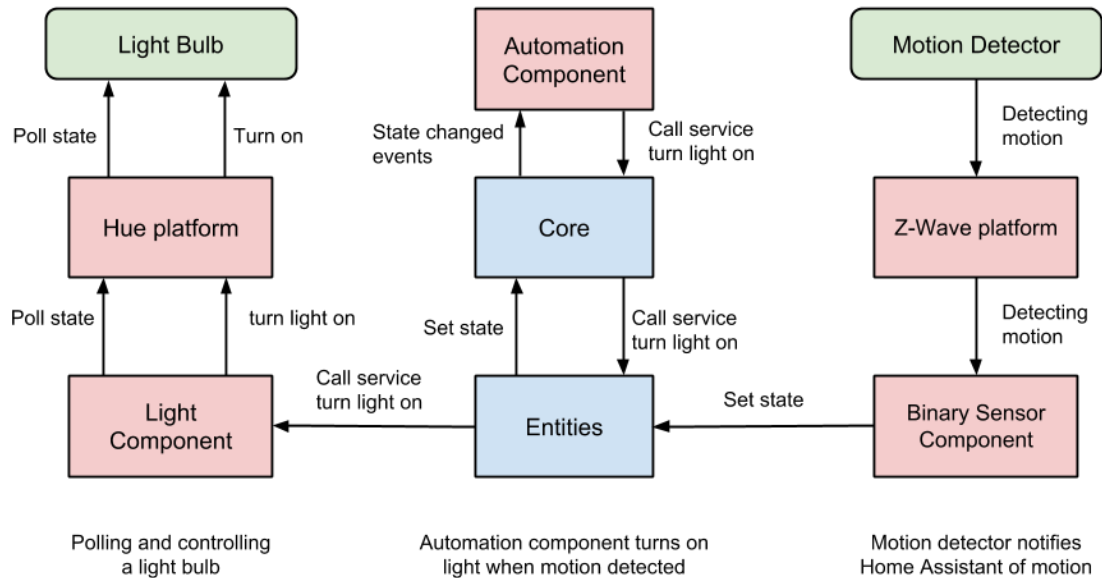


Figure 7. Interaction of Components [8].

In Figure 7, the whole picture describes clearly how it works. Binary sensor component for motion detector is used to create an entity ID *binary\_sensor.motion\_detector* to return state: *on*(active) or *off*(not active). The user can also make an automation component from the entity binary sensor to turn on the light if the state from a motion detector is on. The unique entity ID for the automation should be *automation.control\_light*. Furthermore, the entity for the light performed in a similar way is used with support from a service call. The service for turning on the light is *light.turn\_on* going with an entity ID: *light.living\_room*. The result is to turn on the light in the living room. Therefore, when the system gets the state on from the entity with an ID: *binary\_sensor.motion\_detector*, it will call service to turn on the light.

### 3.2.3 User Interface of Home Assistant

There are two ways to interact with the Hass system: CLI (Command Line User Interface) and web interface.

In Hass, the user usually does not interact with the system by using directly I/O hardware but by supporting from web service with port 8123 and SSH (Secure Shell) protocol. The web interface is designed mainly to monitor and control the system. The Overview tab is designed by the user to make virtual rooms in the house to monitor all devices along with the default weather entity. Generally, there are three main panels for controlling system: *Developer Tools*, *Supervisor* and *Configuration*. The user often takes advantage of the *STATES* tab and *SERVICES* tab in the *Developer Tools* panel to check if devices work. Supervisor panel is used to switch the host, take a snapshot to restore the current system state, manage the *Add-ons*. *Configuration* panel is possible for the user to configure the components and Home Assistant.

Specifically, the chosen entity ID will return the current state of that entity which can be from a device or automation. The state depends on the entity, that can be *On/Off* or numeric state in the *STATE* tab while the *SERVICES* tab will call the service for entities. For example, the *camera.snapshot* service going with *camera.kitchen\_camera* entity ID will need data from Service Data such as filename: */config/www/picture.jpg* to save a photo. When the user clicks the *Call Service* button, the Hass will take a photo from the camera and save it with the name *picture.jpg*.

For CLI User Interface, SSH protocol is used to connect Hass to Laptop. A laptop running Linux Operating System can use Terminal with a command line:

```
$sudo ssh root@hassio.local
```

In the Raspberry Pi with Raspbian OS, the motion package is installed in the Terminal to run the RPI camera as an IP camera. User can watch streaming video in the web browser with the address: *http://:Raspberry-IP-address:8081/mjpeg*

### 3.2.4 The Architecture for Automation

The idea is to put the system in the kitchen. When a user leaves a working stove without being aware of it, the system will detect the smoke or fire with the gas sensor and the flame sensor. If the smoke or flame is detected on the stove, the system will turn off the stove and take a photo of the stove. Furthermore, user can leave a stove working without using it, the DHT11 sensor will measure the temperature. If the temperature is over 50 degrees Celsius, the system can turn off the stove. A buzzer will make an alarm. Besides, the system also turns on the light when the user is present in the kitchen and it will turn off the light if someone leaves after one minute to save electricity.

The main automation is to get data from the flame sensor and the gas sensor to check if there is fire or smoke on the stove then the system will turn off the stove to prevent the potential fire or damage. The relay connected to the red LED works as a switch to the stove in this system.

As long as the stove is working, that means the red LED is on. The system will check data from the motion sensor, flame sensor, and gas sensor. If data from motion is on (presence) and from the flame sensor or the gas sensor is off (there is a fire or smoke), the relay will be off to turn off the red LED and the IP camera will take a photo and save it. For light automation, the green LED works as a kitchen light. The motion is on, the green LED will be on and it will be off if the motion is off after one minute. Diagrams in Figure 6 and 7 will show how it works.

In Home Assistant, there are three main parts to make an Automation: *Trigger*, *Condition* and *Action*.

- *Triggers* [9] are used to start the processing of an automation. When the trigger turns true, Home Assistant will run an automation with condition and call an action. There are 13 types of *Triggers*: *Device*, *Event*, *Geolocation*, *Zone*, *State*, *Home Assistant*, *MQTT*, *Numeric state*, *Sun*, *Template*, *Time*, *Time Pattern* and *Webhook*. *Entity* is chosen to allow user to modify state of it.
- *Conditions* [10] are not compulsory but when an automation goes with a condition, the system will not execute an automation if all conditions are not satisfied. There are 9 types of conditions: *Or*, *And*, *Device*, *State*, *Numeric state*, *Sun*, *Template*, *Time* and *Zone*. *Or* condition is when one or more conditions are true while *And* condition must be true for all of conditions.

- *Actions* [11] are what is executed after triggers and conditions are run. There are 7 action type including *Call service*, *Condition*, *Delay*, *Fire Event*, *Activate Scene*, *Device* and *Wait*. *Call service* is often used for an action.

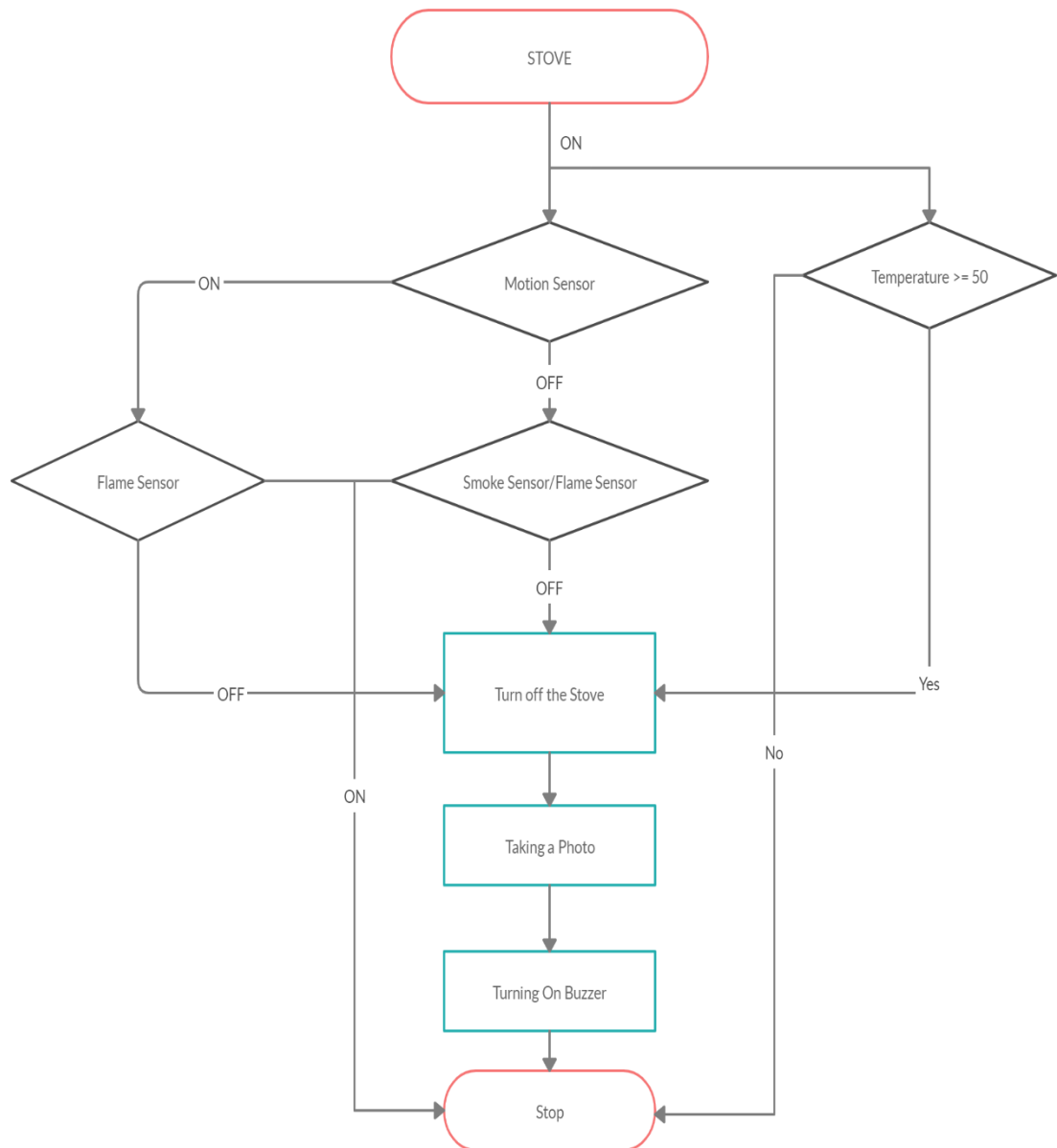


Figure 8. The Automation for Turning Off the Stove.

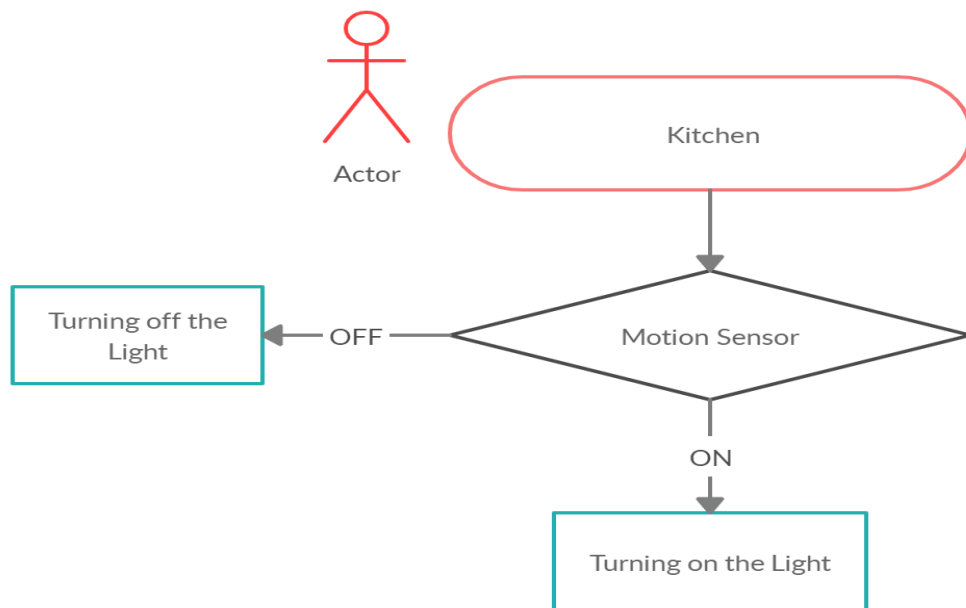


Figure 9. The Automation for Switching Light.

### 3.3 Implementation

#### 3.3.1 IP camera

In the Terminal of Raspbian OS, Motion package [12] should be installed with a few steps:

```
$sudo apt-get update
```

```
$sudo apt-get upgrade -y
```

```
$sudo apt-get install motion -y
```

Motion package is configured in configuration file of motion with a command below:

```
$sudo nano /etc/motion/motion.conf
```

The following lines in motion.conf must be adjusted:

```
# Start in daemon (background) mode and release terminal (default: off)
```

```
daemon on
```

# well as movies. Valid values: 0 (default = no rotation), 90, 180 and 270.

rotate 180

# Restrict stream connections to localhost only (default: on)

stream\_localhost off

Configuration file is saved with CTRL+O then CTRL+X. Then another file is opened to activate daemon (a background process) to let run the motion service.

```
$ sudo nano /etc/default/motion
```

```
# set to 'yes' to enable the motion daemon
```

```
start_motion_daemon=yes
```

It is saved and exited with CTRL+O then CTRL+X. Then service will be run and checked with two command lines:

```
$sudo systemctl start motion
```

```
$sudo systemctl status motion
```

```
pi@raspberrypi:~ $ sudo systemctl start motion
pi@raspberrypi:~ $ sudo systemctl status motion
● motion.service - LSB: Start Motion detection
   Loaded: loaded (/etc/init.d/motion; generated)
   Active: active (running) since Sat 2020-03-28 16:41:44 GMT; 1 weeks 2 days ago
     Docs: man:systemd-sysv-generator(8)
  Process: 310 ExecStart=/etc/init.d/motion start (code=exited, status=0/SUCCESS)
    Tasks: 3 (limit: 2077)
   Memory: 536.1M
    CGroup: /system.slice/motion.service
            └─518 /usr/bin/motion

Warning: Journal has been rotated since unit was started. Log output is incomplete.
lines 1-11/11 (END)
```

Figure 10. The Status of Motion Service.

Then testing should be made in web browser in laptop with IP address:

```
http://Raspberry-IP-address:8081/mjpeg
```



### 3.3.2 Home Assistant System

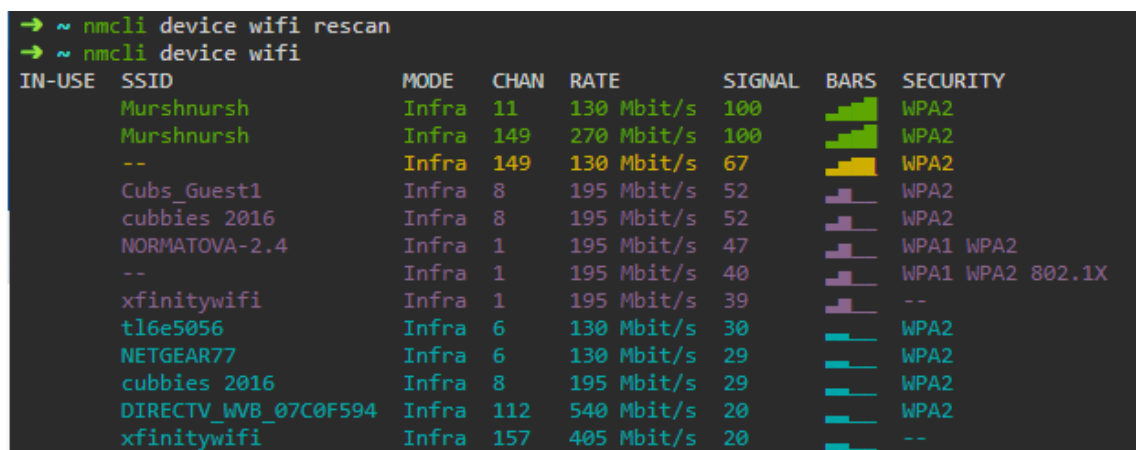
#### Installation

First, HassOS image is booted in the Raspberry Pi with support from Home Assistant Installation [13]. For setting up Wi-Fi, it could be a challenge to connect Wi-Fi manually according to the guidance on the website. It seems simple with some steps but it is difficult to check the process. Therefore, that could be a very clear way to set up the Wi-Fi connection by interacting directly with HassOS in the Raspberry Pi. The system booting takes about 25 minutes for the first time then there must be login with username: root then typing word *login* again is used to access ash shell environment. The *nmcli* command standing for Network Manager Command Line [14] is used to detect and connect Wi-Fi in the following steps.

```
~ nmcli device wifi rescan
~ nmcli device wifi
~nmcli device wifi connect "YOUR_SSID" password
"YOUR_WIFI_PASSWORD"
```

When the system is connected to WiFi successfully, the output will be:

*"Device 'wlan0' successfully activated with...."*



```
→ ~ nmcli device wifi rescan
→ ~ nmcli device wifi
```

IN-USE	SSID	MODE	CHAN	RATE	SIGNAL	BARS	SECURITY
	Murshnursh	Infra	11	130 Mbit/s	100	██████████	WPA2
	Murshnursh	Infra	149	270 Mbit/s	100	██████████	WPA2
	--	Infra	149	130 Mbit/s	67	██████████	WPA2
	Cubs_Guest1	Infra	8	195 Mbit/s	52	██████████	WPA2
	cubbies 2016	Infra	8	195 Mbit/s	52	██████████	WPA2
	NORMATOVA-2.4	Infra	1	195 Mbit/s	47	██████████	WPA1 WPA2
	--	Infra	1	195 Mbit/s	40	██████████	WPA1 WPA2 802.1X
	xfinitywifi	Infra	1	195 Mbit/s	39	██████████	--
	tl6e5056	Infra	6	130 Mbit/s	30	██████████	WPA2
	NETGEAR77	Infra	6	130 Mbit/s	29	██████████	WPA2
	cubbies 2016	Infra	8	195 Mbit/s	29	██████████	WPA2
	DIRECTV_WVB_07C0F594	Infra	112	540 Mbit/s	20	██████████	WPA2
	xfinitywifi	Infra	157	405 Mbit/s	20	██████████	--

Figure 11. Scanning and Listing the Wi-Fi Access.

To make sure that the connection is set up and works well, checking should be made with a command line as well as checking an IP address.

```
~ nmcli con show
~ ip addr show
```

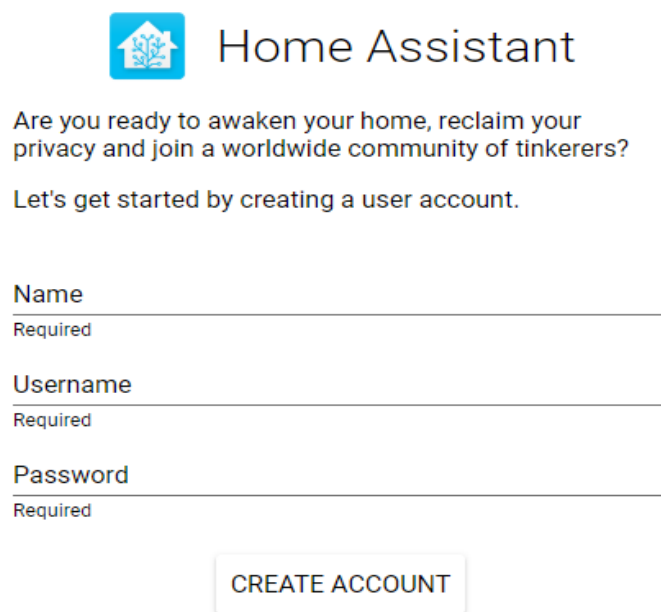
```
Device 'wlan0' successfully activated with '11ec21a1-964c-49c4-8b56-d2963205e4f7'.
→ ~ nmcli con show
NAME                UUID                                TYPE    DEVICE
HassOS default      f62bf7c2-e565-49ff-bbfc-a4cf791e6add ethernet eth0
Murshnursh          11ec21a1-964c-49c4-8b56-d2963205e4f7 wifi     wlan0
```


Figure 12. Showing Connection.

When determining the IP address of the Raspberry Pi, the user fills the IP address or *hassio.local* (the host of browser supports DNS service) with a default port 8123 in the web browser to interact with the Hass user interface.

<http://hassio.local:8123>

Hass UI allows the user to create a new account by filling the blank in the fields: *Name*, *Username*, *Password*.



 Home Assistant

Are you ready to awaken your home, reclaim your privacy and join a worldwide community of tinkerers?

Let's get started by creating a user account.

Name  
Required

Username  
Required

Password  
Required

CREATE ACCOUNT

Figure 13. Creating a New Account.

After that, *Terminal & SSH* should be activate by clicking *Start* in Add-on of the Supervisor panel. That activation makes a connection between Hass Core and laptop.

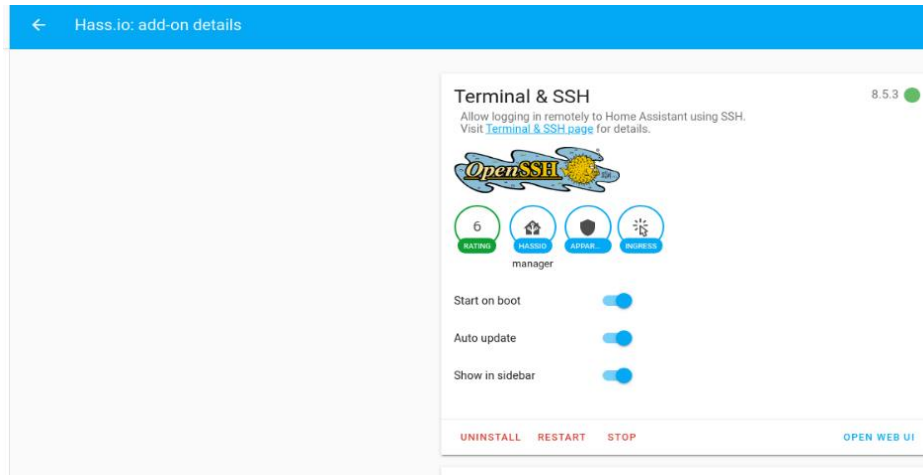


Figure 14. Starting SSH Service.

## Configuration

This step is considered as the most important one to connect all of the devices and services to Home Assistant Core. The system should be connected via SSH protocol to access the file system. The configuration file is located in `/config/configuration.yaml` and it is shown in Figure 15. Four main components are set up including *sensor*, *binary\_sensor*, *switch* and *camera*. There must not any tab while modifying the *yaml* file.

- *sensor* component [15]: *dht* platform is selected with sensor type: DHT11 and it is connected to GPIO port 4 in the Raspberry Pi. The system monitors the temperature and humidity via the DHT11 sensor.
- *binary\_sensor* component [16]: the main point is to get data from sensors and switch two states to the system: 0 or 1, True or False, On or Off... *rpi\_gpio* platform is set for using GPIO ports in Raspberry and port 17 for smoke, port 14 for motion detection, and port 15 for flame detection. When the system detects the smoke or flame, its value is *Off* while motion detection value is *On* if the system detects motion.
- *switch* component [17]: it is used to control the state of switches. The *rpi\_gpio* platform is also used to connect to the devices such as a light, a relay, and a buzzer. Therefore, the system can control them as an output by providing services such as *switch.turn\_on*, *switch.turn\_off* and *switch.toggle*. The port 27 is used for stove (the relay in the real device), port 18 for the kitchen light and port 22 for the buzzer.

- *camera* component [18]: *mjpeg* platform allows the system to integrate an IP camera for streaming video into Home Assistant by adding the full address to *mjpeg\_url*. The name for this item is *kitchen camera*.

In the front-end UI, there should be testing the state and service of entities. For example, *binary\_sensor.smoke* returns state: on, that means the system detects no smoke. Both gas sensor and flame sensor have a special case that state will return on if it detects no smoke/flame while it's off if there is smoke/flame over the stove. For the service, the system will provide some available services for entities such as *switch.turn\_on*. It means the system will support the service that turns on the switch component then the suitable entity will be chosen such as *switch.kitchen\_light*. By clicking *CALL SERVICE* button, the system will turn on the light. Then the user can monitor the overview of UI in Figure 16.

```

GNU nano 4.6 configuration.yaml Modified
# Configure a default setup of Home Assistant (frontend, api, etc)
default_config:

# Uncomment this if you are using SSL/TLS, running in Docker container, etc.
# http:
#   base_url: example.duckdns.org:8123

# Text to speech
homeassistant:
  whitelist_external_dirs:
    - /home/
tts:
  - platform: google_translate

group: !include groups.yaml
automation: !include automations.yaml
script: !include scripts.yaml
python_script:

sensor:
  platform: dht
  sensor: DHT11
  pin: 4
  monitored_conditions:
    - temperature
    - humidity

binary_sensor:
  - platform: rpi_gpio
    ports:
      17: smoke
      14: motion detection
      15: flame detection
switch:
  - platform: rpi_gpio
    ports:
      27: stove
      18: kitchen light
      22: buzzer
ios:
camera:
  - platform: mjpeg
    mjpeg_url: http://IP-address:8081/mjpeg
    name: kitchen camera

^G Get Help      ^O Write Out    ^W Where Is    ^K Cut Text     ^J Justify     ^C Cur Pos
^X Exit          ^R Read File   ^V Replace     ^U Paste Text  ^I To Spell   ^_ Go To Line

```

Figure 15. Configuration File.

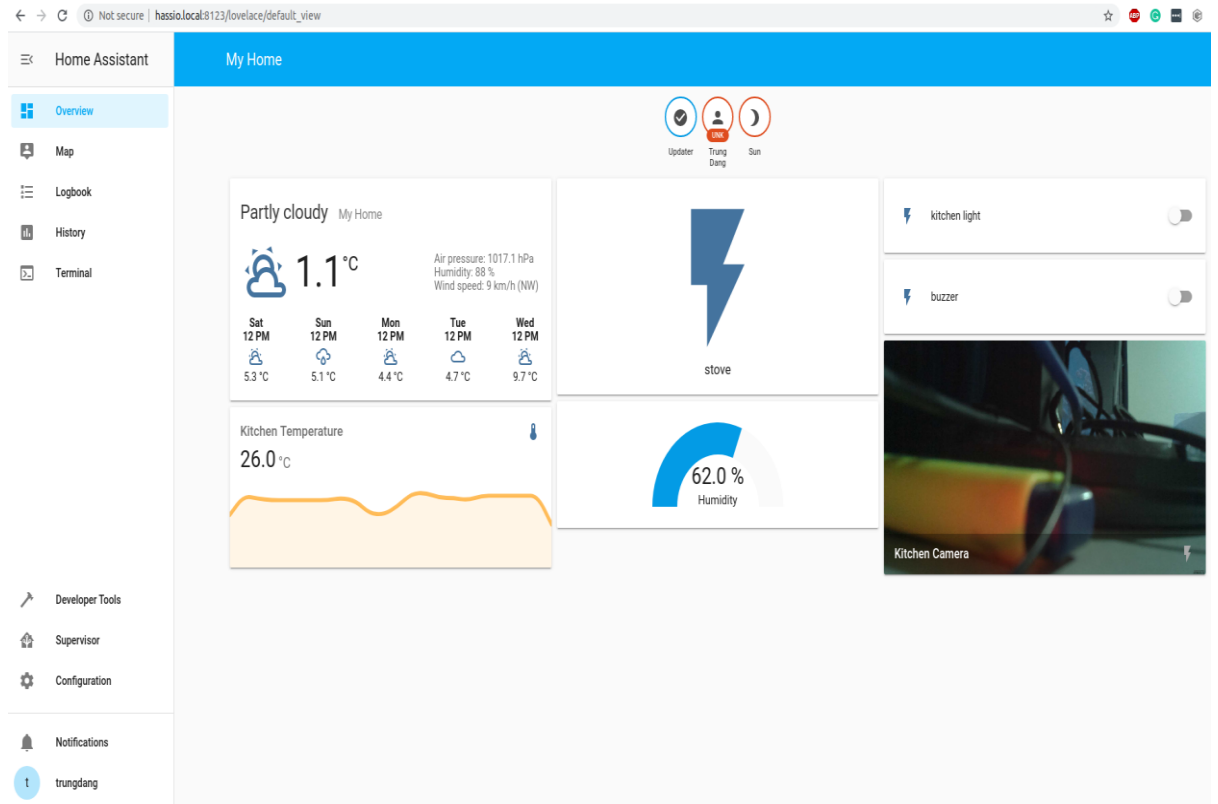


Figure 16. The Overview of Home Assistant Front-end UI.

## Automation

There are four automations for the system listed in Figure 17.

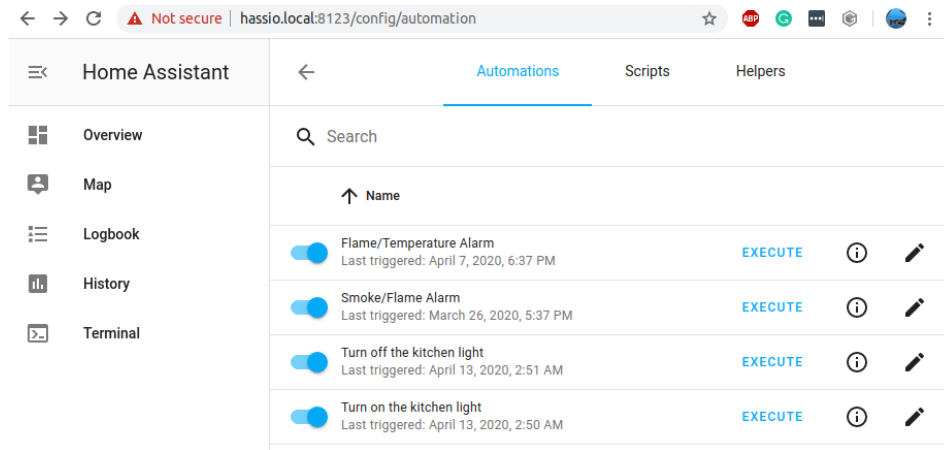


Figure 17. The Automation List.

### Smoke/Flame Alarm

The purpose to set turn off the stove if smoke/flame is detected when cooking.

The `switch.stove` entity with the state: On will start the process and its conditions are listed in Figure 18.

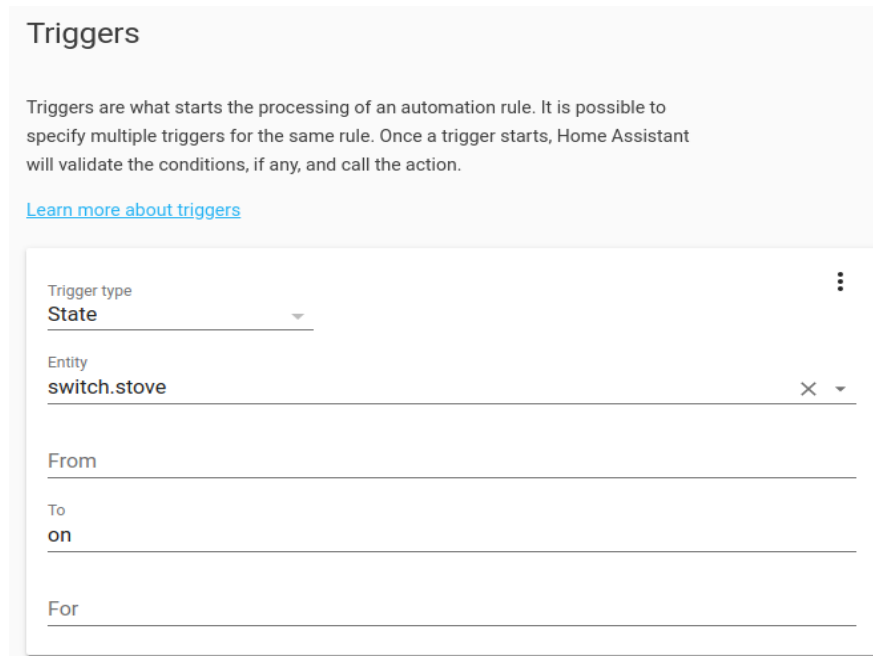


Figure 18. Triggers for Smoke/Flame Alarm.

## Conditions

Conditions are optional and will prevent further execution unless all conditions are satisfied.

[Learn more about conditions](#)

Condition type ⋮  
**And**

Condition type ⋮  
**State**

Entity **binary\_sensor.motion\_detection** ✕

State **off**

Condition type ⋮  
**Or**

Condition type ⋮  
**State**

Entity **binary\_sensor.smoke** ✕

State **off**

Condition type ⋮  
**State**

Entity **binary\_sensor.flame\_detection** ✕

State **off**

Figure 19. The Conditions for Smoke/Flame Alarm.

If motion detects no one in the kitchen and there is fire or flame detected, the system will turn off the stove and take a photo. The photo must be stored in `/config/www` with a filename created by current time and date so it would be easy to determine when it happens. Finally, the system will call the service for turning on the buzzer so it will pay attention to the user to check the kitchen.



**Actions**

The actions are what Home Assistant will do when the automation is triggered.

[Learn more about actions](#)

Action type: **Call service**

Service: **switch.turn\_off**

Name(s) of entities to turn off: **switch.stove**

Service data: **1**

Action type: **Call service**

Service: **camera.snapshot**

Name(s) of entities to create snapshots from: **camera.kitchen\_camera**

Service data:

```

1 entity_id: camera.kitchen_camera
2 filename: '/config/www/camera_{{ now().strftime("%Y%m%d-%H%M%S") }}.jpg'
3
```

Action type: **Call service**

Service: **switch.turn\_on**

Name(s) of entities to turn on: **switch.buzzer**

Service data:

Figure 20. The Actions for Smoke/Flame Alarm.

### Flame/Temperature Alarm

It is similar with the Smoke/Flame Alarm but it is different in the *Conditions*. Temperature is over 50 degrees Celsius and the flame state is off, the system will turn off the stove. No matter if the user is still in the kitchen or somewhere in the house, the fire can happen from the working stove without cooking. The system must be strict with any potential damage.

## Conditions

Conditions are optional and will prevent further execution unless all conditions are satisfied.

[Learn more about conditions](#)

Condition type ⋮

**Or** ▼

---

Condition type ⋮

**State** ▼

Entity **binary\_sensor.smoke** ✕ ▼

---

State **off**

---

Condition type ⋮

**Numeric state** ▼

Entity **sensor.dht\_sensor\_temperature** ✕ ▼

---

Above **50**

---

Below

---

Value template (optional)

---

Figure 21. The Conditions for Flame/Temperature Alarm.

## Turn on the Kitchen Light and Turn off the Kitchen Light

Switching the kitchen light automatically is based on the presence of user. The motion entity is used for a trigger. When the motion turns On state, the light kitchen is on. It is described below in the Figure 22 and 23.

To turn off the light, in the Triggers, the state of motion entity is off and for 1 minute. That means after 1 minute, system detects no one then the system will make an action. The process is illustrated in Figure 24 and 25.

### Triggers

Triggers are what starts the processing of an automation rule. It is possible to specify multiple triggers for the same rule. Once a trigger starts, Home Assistant will validate the conditions, if any, and call the action.

[Learn more about triggers](#)

Trigger type  
**State** ▼

Entity  
**binary\_sensor.motion\_detection** × ▼

From  
**off**

To  
**on**

For

Figure 22. The Trigger for Turning on The Light.

## Conditions

Conditions are optional and will prevent further execution unless all conditions are satisfied.

[Learn more about conditions](#)

Condition type  
**State** ▼

Entity  
**switch.kitchen\_light** × ▼

State  
**off**

**ADD CONDITION**

## Actions

The actions are what Home Assistant will do when the automation is triggered.

[Learn more about actions](#)

Action type  
**Call service** ▼

Service  
**switch.turn\_on** × ▼

Name(s) of entities to turn on  
**switch.kitchen\_light** × ▼

Service data

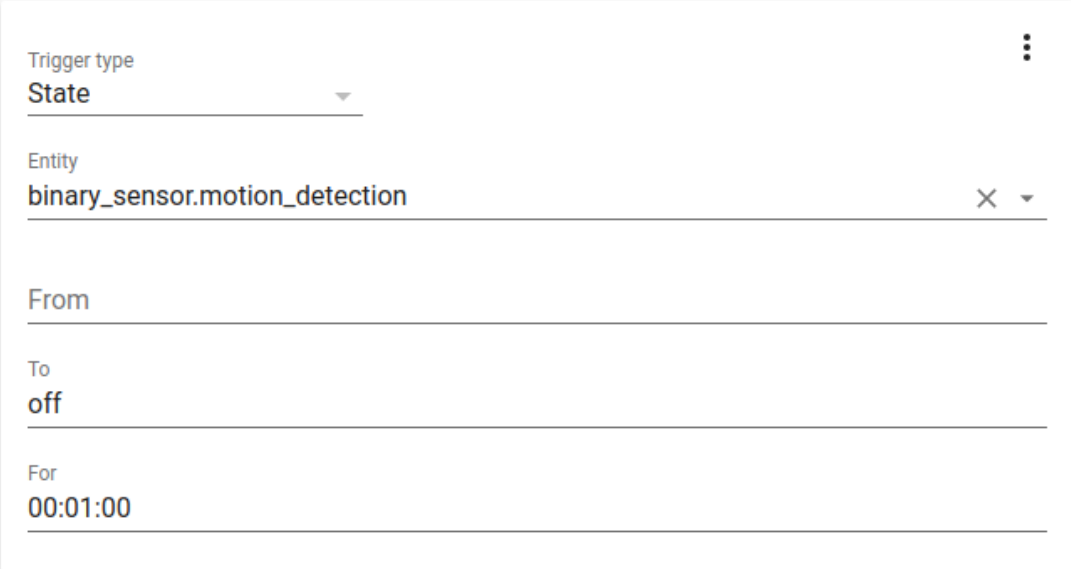
```
1 entity_id: switch.kitchen_light
2
```

Figure 23. The Condition and Action for Turning On the Light.

## Triggers

Triggers are what starts the processing of an automation rule. It is possible to specify multiple triggers for the same rule. Once a trigger starts, Home Assistant will validate the conditions, if any, and call the action.

[Learn more about triggers](#)



The screenshot shows a configuration form for a trigger. It includes the following fields:

- Trigger type:** A dropdown menu with the value **State** selected.
- Entity:** A text input field containing **binary\_sensor.motion\_detection**, with a clear button (X) and a dropdown arrow.
- From:** An empty text input field.
- To:** A text input field containing **off**.
- For:** A text input field containing **00:01:00**.

Figure 24. The Trigger for Turning Off the Light.

## Conditions

Conditions are optional and will prevent further execution unless all conditions are satisfied.

[Learn more about conditions](#)

Condition type ⋮  
**State** ▾

Entity  
**switch.kitchen\_light** X ▾

State  
**on**

[ADD CONDITION](#)

## Actions

The actions are what Home Assistant will do when the automation is triggered.

[Learn more about actions](#)

Action type ⋮  
**Call service** ▾

Service  
**switch.turn\_off** X ▾

Name(s) of entities to turn off.  
**switch.kitchen\_light** X ▾

Service data

```
1 | entity_id: switch.kitchen_light
2 |
```

Figure 25. The Condition and Action for Turning Off the Light.

### 3.4 Future Development

So far, the system has been stable and fast to keep the kitchen safe. However, there should be an improvement for the system. Home Assistant has huge benefits to develop or deploy the whole system such as 1574 integrations provided by Home Assistant community. Therefore, there will be the list of future development described below:

- Connecting to Alexa: it will be an enormous advantage to interact with a virtual assistant by oral commands which can control all of the devices in the house. It would be much more convenient for users.
- Using Thermal Camera instead of gas and flame sensor for the stove: Gas and flame sensors work well so far but the thermal camera can be more potential for safety in the kitchen. It can detect the temperature based on the range of colors with the high accuracy. Moreover, that system can also detect boiling over from the pot while cooking. There will be conducting some long research.
- Creating many rooms in a house in the overview panel: designing many rooms where all devices are connected to the system. The users can control all of the devices with an application in the cellphone.
- Using IFTTT component: it is used to send some photos of kitchen via Gmail when something happens in the kitchen such as smoke or fire.
- Building Face Recognition from Amazon Web Services (AWS): Strange people are recognized for the first time before entering the house and photo of them will be sent to user via Gmail or Alexa will inform it.



Figure 26. Alexa Devices [19].



## 4 CONCLUSION

IoT applications for house are becoming more and more popular due to its benefits and inexpensive cost. Home Assistant is one of the IoT platforms giving advantages and satisfaction to home consumers because of monitoring and automation capabilities.

The Kitchen IoT system developed in this thesis has utilized both hardware and software. Its purpose was to monitor kitchen and keep the stove safe from potential damage and this purpose was eventually achieved. The Home Assistant system provides good and friendly components to create the system for a kitchen. The automation in the system can react so quickly to smoke/fire and temperature via sensors to turn off the stove and the system can switch the light automatically based on the presence of the user. Furthermore, the user can monitor every device or service in the system easily with the front-end UI. The system should also be updated with a new version to become more stable and receive support from the Home Assistant Community.

## REFERENCES

- [1] IoT Tech Trends - History of IoT: A Timeline of Development. Available at <https://www.iottechrends.com/history-of-iot/>, Accessed February 1, 2020
- [2] I-SCOOP - The Internet of Things (IoT) – Essential IoT Business Guide. Available at <https://www.i-scoop.eu/internet-of-things-guide/>, Accessed February 3, 2020
- [3] IBM - Cloud Computing. Available at <https://www.ibm.com/cloud/learn/cloud-computing>, Accessed February 10, 2020
- [4] I-SCOOP - Fog Computing: Fog and Cloud Along the Cloud-to-Thing continuum. Available at <https://www.i-scoop.eu/internet-of-things-guide/fog-computing-cloud-internet-things/>, Accessed February 10, 2020
- [5] Home Assistant – Home Assistant Supervisor. Available at <https://developers.home-assistant.io/docs/supervisor/>, Accessed December 10, 2019
- [6] Raspberry Pi – GPIO. Available at <https://www.raspberrypi.org/documentation/usage/gpio/>, Accessed December 15, 2019
- [7] I-SCOOP - Architecture. Available at [https://developers.home-assistant.io/docs/architecture\\_index/](https://developers.home-assistant.io/docs/architecture_index/), Accessed February 15, 2020
- [8] Home Assistant – Components Architecture. [https://developers.home-assistant.io/docs/architecture\\_components](https://developers.home-assistant.io/docs/architecture_components), Accessed February 20, 2020
- [8] Home Assistant - Automation Trigger. Available at <https://www.home-assistant.io/docs/automation/trigger/>, Accessed February 20, 2020
- [10] Home Assistant – Automation Conditions. Available at <https://www.home-assistant.io/docs/scripts/conditions/>, Accessed February 21, 2020
- [11] Home Assistant – Automation Actions. Available at <https://www.home-assistant.io/docs/automation/action/>, Accessed February 25, 2020
- [12] Instructables Circuits - How to Make Raspberry Pi Webcam Server and Stream Live Video . Available at <https://www.instructables.com/id/How-to-Make-Raspberry-Pi-Webcam-Server-and-Stream-/>, Accessed March 1, 2020
- [13] Home Assistant - Installing Home Assistant . Available at <https://www.home-assistant.io/hassio/installation/>, Accessed August 1, 2019
- [14] Home Assistant – Guide: Connecting Pi with Home Assistant OS to Wi-Fi. Available at <https://community.home-assistant.io/t/guide-connecting-pi-with-home-assistant-os-to-wifi-or-other-networking-changes/98768>, Accessed January 15, 2020
- [15] Home Assistant – DHT Sensor . Available at <https://www.home-assistant.io/integrations/dht/>, Accessed September 2, 2019

- [16] Home Assistant – Binary Sensor . Available at [https://www.home-assistant.io/integrations/binary\\_sensor/](https://www.home-assistant.io/integrations/binary_sensor/), Accessed March 5, 2020
- [17] Home Assistant – Raspberry Pi GPIO . Available at [https://www.home-assistant.io/integrations/rpi\\_gpio/](https://www.home-assistant.io/integrations/rpi_gpio/), Accessed September 12, 2020
- [18] Home Assistant - MJPEG IP Camera . Available at <https://www.home-assistant.io/integrations/mjpeg/>, Accessed March 12, 2020
- [19] Wikipedia – Amazon Alexa. Available at [https://en.wikipedia.org/wiki/Amazon\\_Alexa](https://en.wikipedia.org/wiki/Amazon_Alexa), Accessed February 14, 2020