VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Viet Hoang Tran

RASPBERRY PI – BASED IOT SOLUTIONS: INTERGRATING SENSORS, PYTHON AND CLOUD SERVICES FOR SMART NOTIFICATIONS AND CONTROL

School of Technology
2024

Style Definition: TOC 2

Commented [KS1]: Do not hyphenate words in the title

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Hoang Viet Tran
Title	Raspberry Pi-based IoT solutions: Integrating Sensors, Python and Cloud Services for Smart Notification and Control
Year	2024
Language	English
Pages	43 + 3 Appendices
Name of Supervisor	Smail Menani

This thesis was made to demonstrate the making of an Internet of Things application with Raspberry Pi as central and Python as the main programming language. The idea of this thesis is the author's own project, taking preference from knowledge he gained during his student period.

As a powerful and convenient as well as low cost, Raspberry Pi 3B was chosen as the main development kit for this application, alongside with certain sensors such as DHT11, motion sensors, LEDs. Python was chosen as the development language due to its compact, light-memory requirements and Linux compatibility. For cloud storing the data, AWS was the suitable cloud service due to its features and functioning cost. Node-Red was used as a service for visualizing and creating GUI for the application.

The result of this thesis is an IoT application which provide user with insights about their home or offices, such as temperature, humidity, status of lights and status of movement detected.

Keywords Raspberry Pi, AWS, Internet of Things, Node-Red

Formatted: Indent: Left: 1.5", First line: 0"

Commented [KS2]: The first letter: of words capital: Integrating Sensors , Python and Cloud Services....

CONTENTS

ABSTRACT

1 INTRODUCTION 5
2 TECHNOLOGY DESCRIPTION 6
2.1 Raspberry Pi 6
2.2 DHT11 6
2.3 PIR Motion Sensors 6
2.4 Python 6
2.5 SMTP 7
2.6 AWS IoT Core 7
2.7 Node-Red 7
3 SYSTEM ARCHITECTURE AND FUNCTIONALITY 8
3.1 System Architecture 8
3.2 Application Functionality 8
4 IMPLEMENTATION 10
4.1 Sensor's Data Retrieving 10
4.2 Cloud Service Data Storage 14
4.3 Data Visualization 18
4.4 Email Notification 23
4.5 Automated Actions 25
5 CONCLUSIONS 26
REFERENCES 27

LIST OF CODE SNIPPETS

Code Snippet 1 Data reading implementation 13
Code Snippet 2 Sending data to AWS IoT Core ..... 17~~16~~
Code Snippet 3 Email notification implementation ..... 24~~23~~
Code Snippet 4 Automated actions implementation ..... 25~~24~~

LIST OF FIGURES AND TABLES

Figure 1 Application architecture 8
Figure 2 Work-flow diagram 9
Figure 3 Sensors and LED connections to Raspberry Pi 11
Figure 4 Raspberry Pi pins map 12
Figure 5 Console output of collected data 14
Figure 6 AWS IoT Core device add ..... 15~~14~~
Figure 7 AWS IoT Core device certificates and keys ..... 16~~15~~
Figure 8 Data stored in AWS IoT Core ..... 18~~17~~
Figure 9 Node-Red system diagram ..... 19~~18~~
Figure 10 AWS subscribe node configuration ..... 20~~19~~
Figure 11 Sample JSON data ..... 21~~20~~
Figure 12 Code to separate data in function node ..... 21~~20~~
Figure 13 Gauge configuration ..... 22~~21~~
Figure 14 Chart configuration ..... 22~~21~~
Figure 15 Data visualization output ..... 23~~22~~

1 INTRODUCTION

Nowadays, an IoT application has become popular and can be seen widely integrated in daily life as well as industry. Thanks to the development of sensors, hardware as well as web technology, IoT applications now can provide the user with insights that can be accessible anytime, anywhere. Compared to a few decades ago, monitorization of environmental data or status of electronics devices could only be seen in high-end industrial factories or high security area. Now its popularity and commercialization has increased to a point where almost everywhere from schools to malls and regular household can install IoT applications.

This Raspberry Pi – based IoT solution is aimed to be a cheap and easy to implement application for monitoring indoor/outdoor environment data and lighting status of house-hold environment for user, in which based on those insights the user can have better actions on energy consumption as well as security.

This thesis contains five chapters. The first chapter provides an overview about the project documented in the thesis. All technical and theoretical information about technologies used in development process of this application are listed in the second chapter. The next chapter describes the architecture workflow and functionalities of the application. The fourth chapter documents the implementation of the application by the author. The fifth chapter contains conclusions, drawbacks, as well as future improvements for the application.

Commented [KS3]: Not a theoretical background but the description of used technology.
Change the title. Also, delete ALL subheadings since all subsections ate too short to e subsections (at least two text paragraphs).
You also need to add the citations to the sources, from which you have taken the information

2 TECHNOLOGY DESCRIPTION

2.1 Raspberry Pi

Raspberry Pi is a single-board computer developed by the Raspberry Pi Foundation, in association with Broadcom. Raspberry Pi, with its capability to run a Linux-based operating system like Raspbian, allows the device to perform complex processing activities. Moreover, Raspberry Pi also offers built-in GPIO connections, which make interacting with various types of sensors seamless and easy. /1/

Raspberry Pi also offers internet connection through a built in Ethernet port or Wi-fi, a very beneficial function for Internet of Things project.

2.2 DHT11

DHT11 is a lightweight and low-cost device used to measure humidity and temperature levels accurately. It can interact easily with microcontrollers like Raspberry Pi or Arduino due to its simple wiring and digital output format. Despite its low cost, the capability and performance of the sensor itself are reliable, making it popular among a wide range of projects, from weather monitoring to home automation. /2/

2.3 PIR Motion Sensors

PIR sensors allow system to sense motion, it is used to detect whether a human or an animal has moved in or out of the sensors range. The sensor is small and low cost, low operating power and durable. Motion sensors often referred to as PIR, “Passive Infrared”, “Pyroelectric”, or “IR motion” sensors. /3/

Commented [KS4]: Only human beings, what about animals

2.4 Python

Python is a high-level programming language created and released by Guido van Rossum in 1991. Python’s design philosophy emphasizes code readability with support of indentation. Python is compatible with various operating system and

development environment from Windows to Mac OS or Linux. Moreover, Python also has many built-in libraries in which support interacting with GPIO on Raspberry Pi. /4/

2.5 SMTP

The Simple Mail Transfer Protocol (SMTP) is an application used by mail servers to send, receive, and relay outgoing email between senders and receivers. As the technology behind email communication, SMTP is the protocol that allows sending send and receive emails. /5/

2.6 AWS IoT Core

AWS IoT Core is a managed cloud service that enables connected devices to securely interact with cloud applications and other devices. AWS IoT Core can support many devices and messages, and it can process and route those messages to AWS IoT endpoints and other devices. /6/

2.7 Node-Red

Node-RED is a flow-based, low-code development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs, and online services as part of the Internet of things. Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions. The elements of applications can be saved or shared for re-use. The runtime is built on Node.js. The flows created in Node-RED are stored using JSON. Since version 0.14, MQTT nodes can make properly configured TLS connections. /7/

3 SYSTEM ARCHITECTURE AND FUNCTIONALITY

3.1 System Architecture

In this IoT application, there are a total of four main components, on both software and hardware sides.

The hardware side includes Raspberry Pi and 2 types of sensors which are DHT11 and PIR motion sensors capable of providing temperature, humidity, and motion detection data. LED will act as a simulation for the user's actual lighting setup. All mentioned sensors and LED are connected directly to Raspberry Pi's GPIO pins.

The software side in this case also includes Raspberry as an environment, AWS IoT Core and Node Red. Raspberry Pi runs a Linux based operating system Raspbian OS with Python 3 installed as a main development language. AWS IoT Core acts as a cloud data storage and distributor, AWS will communicate with Raspberry Pi for receiving data. Node-Red interacts with AWS IoT Core to collect data provided by Raspberry Pi and visualize to a web browser interface.

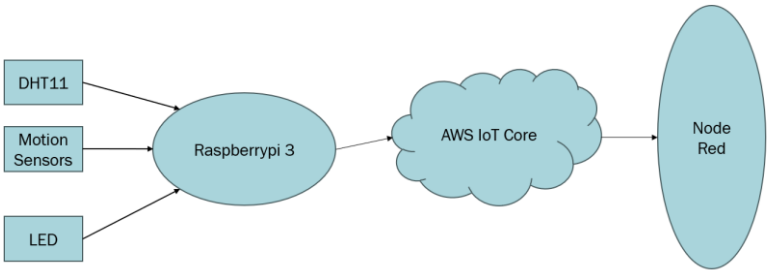

Figure 1. Application architecture

3.2 Application Functionality

The functionality side of the application involves three types of actions and five types of conditions.

The action sides include interacting with lights (switching on or off) and sending to user as notifications.

The condition sides check whether collected data such as temperature or humidity increase or decrease to a certain level. Motion can be detected in certain rooms. The main goal of these conditions is to act as an alert or support to the user about irregular behaviour in their household. Extreme high/low temperature could be from a potential radiator that supposed to be turned off, or motion is detected in certain room that should be empty. Automated lights switch also provide the user with convenience in environment, such as hallway or garage, and energy is saved when lights are turned off after set time.

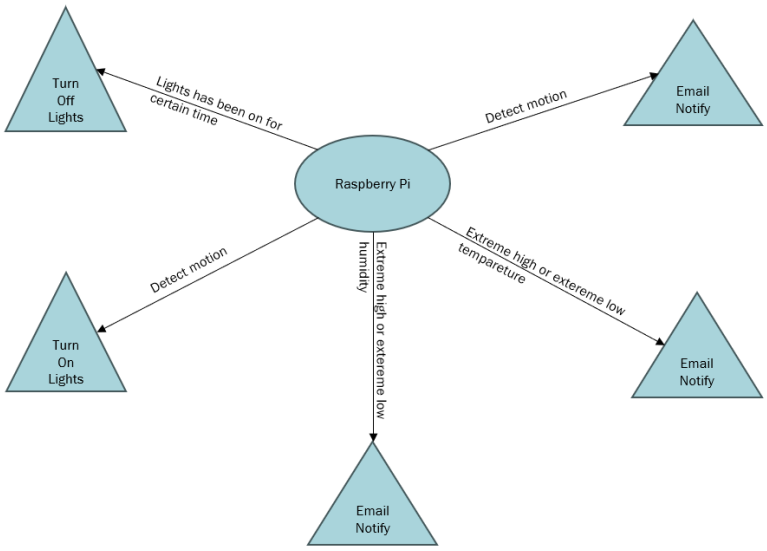

Figure 2. Work-flow diagram

4 IMPLEMENTATION

4.1 Sensor's Data Retrieving

DHT11, PIR motion sensor and LED are connected directly to the built-in GPIO pins of Raspberry Pi. The actual connections made are shown in Figure 3.

Pins 9 and 29 are used as ground, pin 1 is for 3.3V power supply and pin XX is for 5V power supply. Data are sent through pin XX for DHT11 and pin XX for PIR motion sensors. The LED is connected in series with a 220 Ohm resistor. Details about

the pins map of Raspberry Pi are shown in Figure 4

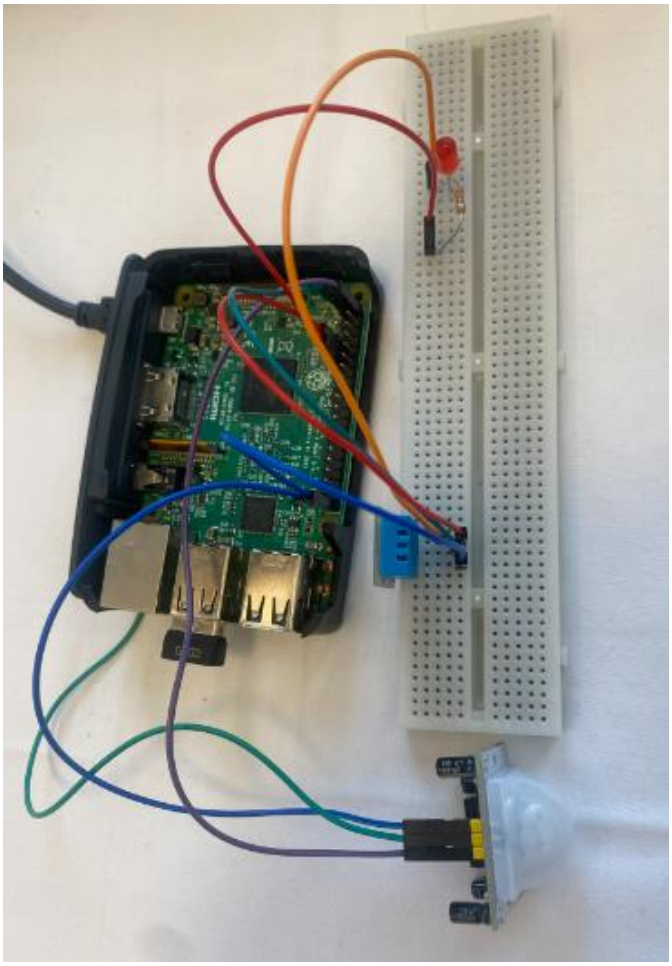

Figure 3. Sensors and LED connections to Raspberry Pi

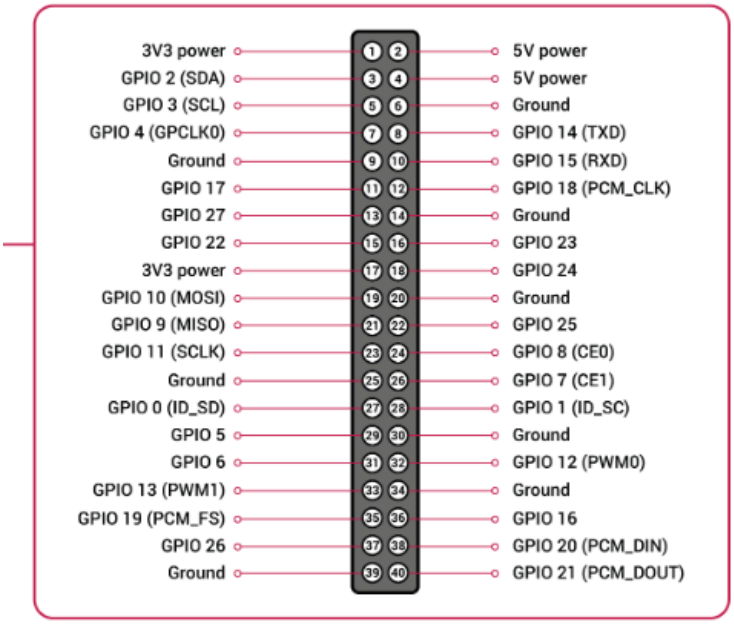

Figure 4. Raspberry Pi pins map

After hardware connections are made, data are collected to Raspberry Pi with a Python 3 script as in Code Snippet 1. An Adafruit DHT library is used to read DHT11 sensors data, the library is written and developed by Adafruit, and it can be included directly without extra installation steps.

Variables for each GPIO pins used are defined in the beginning of the script. Data reading functions run constantly in a while loop to ensure data will be provided all the time. Data will be printed out to a console for ease of development. The terminate input and GPIO clean-up of the program are also added. The console output data are shown in Figure 5.

```python
import Adafruit_DHT
import RPi.GPIO as GPIO
import time

# Pin Definitions
DHT_PIN = 21  # Pin 21 for DHT11 sensor
LED_PIN = 17  # GPIO 17 for LED
PIR_PIN = 26  # Pin 26 for PIR motion sensor

# Setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)
GPIO.setup(PIR_PIN, GPIO.IN)

# Function to read DHT11 sensor data
def read_dht11_data():
    humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11, DHT_PIN)
    if humidity is not None and temperature is not None:
        return {'temperature': temperature, 'humidity': humidity}
    else:
        return None

# Main loop
try:
    while True:
        # Read DHT11 sensor data
        dht_data = read_dht11_data()
        if dht_data:
            print("Temperature: {:.1f}°C, Humidity: {:.1f}%".format(dht_data['temperature'], dht_data['humidity']))

        # Check PIR motion sensor
        motion_status = GPIO.input(PIR_PIN)
        print("Motion detect status:", motion_status)

        # Read LED status
        led_status = GPIO.input(LED_PIN)
        print("LED status:", led_status)

        # Turn on LED if motion detected
        GPIO.output(LED_PIN, motion_status)

        # Delay for a while
        time.sleep(1)

except KeyboardInterrupt:
    print("Program terminated by user.")
finally:
    # Cleanup
    GPIO.cleanup()
```

Code Snippet 11. Data reading implementation

Formatted: English (United Kingdom)

Formatted: English (United Kingdom)

Field Code Changed

```
Temperature is 25.0 and Humidity is 20.0 Motion detect is 0 Light 1 is OFF
Published: '{"id": "Rpi_Device", "Temperature": 25.0, "Humidity": 20.0, "Motion": 0, "Light1"
: "OFF"}' to the topic: 'iotdata'
0
OFF
Temperature is 25.0 and Humidity is 20.0 Motion detect is 0 Light 1 is OFF
Published: '{"id": "Rpi_Device", "Temperature": 25.0, "Humidity": 20.0, "Motion": 0, "Light1"
: "OFF"}' to the topic: 'iotdata'
0
OFF
Temperature is 25.0 and Humidity is 20.0 Motion detect is 0 Light 1 is OFF
Published: '{"id": "Rpi_Device", "Temperature": 25.0, "Humidity": 20.0, "Motion": 0, "Light1"
: "OFF"}' to the topic: 'iotdata'
0
OFF
Temperature is 25.0 and Humidity is 20.0 Motion detect is 0 Light 1 is OFF
Published: '{"id": "Rpi_Device", "Temperature": 25.0, "Humidity": 20.0, "Motion": 0, "Light1"
: "OFF"}' to the topic: 'iotdata'
0
OFF
Temperature is 25.0 and Humidity is 20.0 Motion detect is 0 Light 1 is OFF
Published: '{"id": "Rpi_Device", "Temperature": 25.0, "Humidity": 20.0, "Motion": 0, "Light1"
: "OFF"}' to the topic: 'iotdata'
0
```

Figure 5. Console output of collected data

Commented [KS5]: This has to on the same page with the figure.

4.2 Cloud Service Data Storage

AWS IoT Core is a powerful platform that provides many tools and functionalities for developers to reach their goal in building their IoT applications. An AWS accounts with Basic Subscription are created and used in the implementation.

The first step, a device needs to be added in the AWS IoT Core, to be recognized as the main development device. In this case the device was named as rpiDevice. Certain definitions for the device, such as data type, policy, storage size will also need to be provided. These definitions will hold keys value to the accessibility of the data.

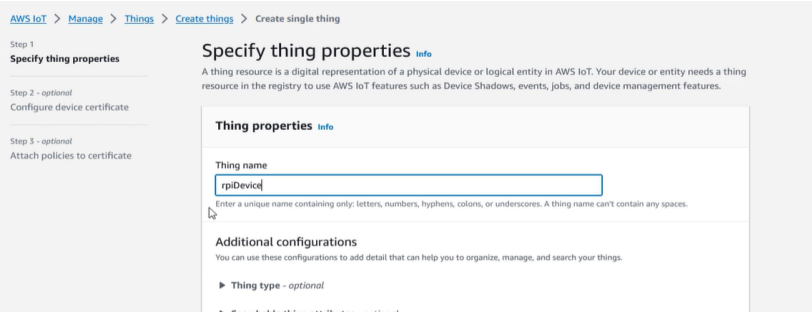

Figure 6. AWS IoT Core device add

After creating device on AWS IoT Core, certification files are generated. These certification and key files will act as access keys, allowing Raspberry Pi to gain access and communicate with AWS. There is a total of five files provided by AWS, including device certificate, public key, private key and two root CA certificates files. All files mentioned must be downloaded directly after AWS has provided them and placed in the same directory in Raspberry Pi as the project.

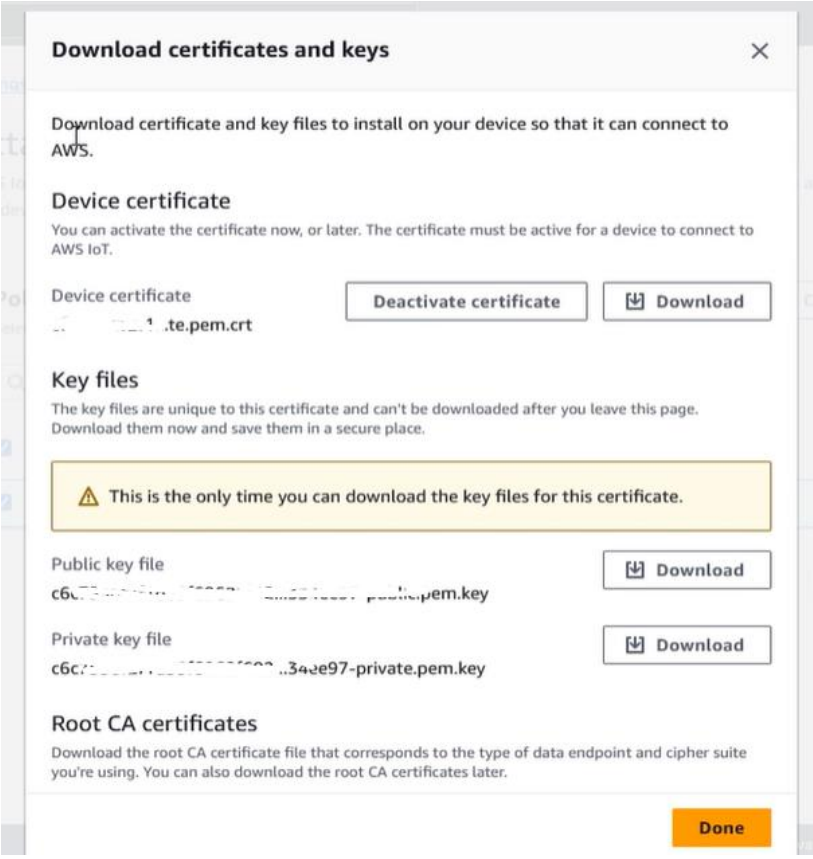

Figure 7. AWS IoT Core device certificates and keys

The Python script that provides communication to AWS is shown in Code Snippet 2. AWS IoT configurations are defined with an endpoint link to the services, path to certification and key files. Host resolver, client, and event loop group for establishing connections are created. MQTT connections are built with mutual certification using the provided keys, certificate.

Data collected in the previous steps are formatted into a JSON message. The JSON message will be published to the MQTT topic created using the MQTT connection provided by AWS.

The running process will run indefinitely with a 1000ms delay in between. The connection to AWS IoT Core will be disconnected when the loop is terminated.

```
from awscrt import io, mqtt, auth, http
from awsiot import mqtt_connection_builder
import time as time
import json
import Adafruit_DHT as dht
# Define ENDPOINT, CLIENT_ID, PATH_TO_CERT, PATH_TO_KEY, PATH_TO_ROOT, MESSAGE, TOPIC, and RANGE
ENDPOINT = "########.iot.us-east-1.amazonaws.com"
CLIENT_ID = "rpiDevice"
PATH_TO_CERT = "certificate.pem.crt" #certificate file path
PATH_TO_KEY = "private.pem.key"#private key file path
PATH_TO_ROOT = "AmazonRootCA1.pem" #root ca certificate file path
TOPIC = "iotdata"#iot topic

# Spin up resources
event_loop_group = io.EventLoopGroup(1)
host_resolver = io.DefaultHostResolver(event_loop_group)
client_bootstrap = io.ClientBootstrap(event_loop_group, host_resolver)
mqtt_connection = mqtt_connection_builder.mtls_from_path(
            endpoint=ENDPOINT,
            cert_filepath=PATH_TO_CERT,
            pri_key_filepath=PATH_TO_KEY,
            client_bootstrap=client_bootstrap,
            ca_filepath=PATH_TO_ROOT,
            client_id=CLIENT_ID,
            clean_session=False,
            keep_alive_secs=6
            )
print("Connecting to {} with client ID '{}'...".format(
        ENDPOINT, CLIENT_ID))
# Make the connect() call
connect_future = mqtt_connection.connect()
# Future.result() waits until a result is available
connect_future.result()
print("Connected!")
# Publish message to server desired number of times.
print('Begin Publish')
while(True):
    h,t=dht.read_retry(11,21) #sensor data pin is connceted to 21 (GPIO Numbering)
    print("Temperature is {} and Humidity is {}".format(t,h))
    message = {"id" : "Rpi_Device","Temperature":t,"Humidity":h}
    mqtt_connection.publish(topic=TOPIC, payload=json.dumps(message), qos=mqtt.QoS.AT_LEAST_ONCE)
    print("Published: '" + json.dumps(message) + "' to the topic: " + "'iotdata'")
    time.sleep(1)
print('Publish End')
disconnect_future = mqtt_connection.disconnect()
disconnect_future.result()
```

Code Snippet 2. Sending data to AWS IoT Core

When a connection is established and data is successfully delivered, the JSON format data can be seen from the AWS IoT Core page, as shown in Figure 8.

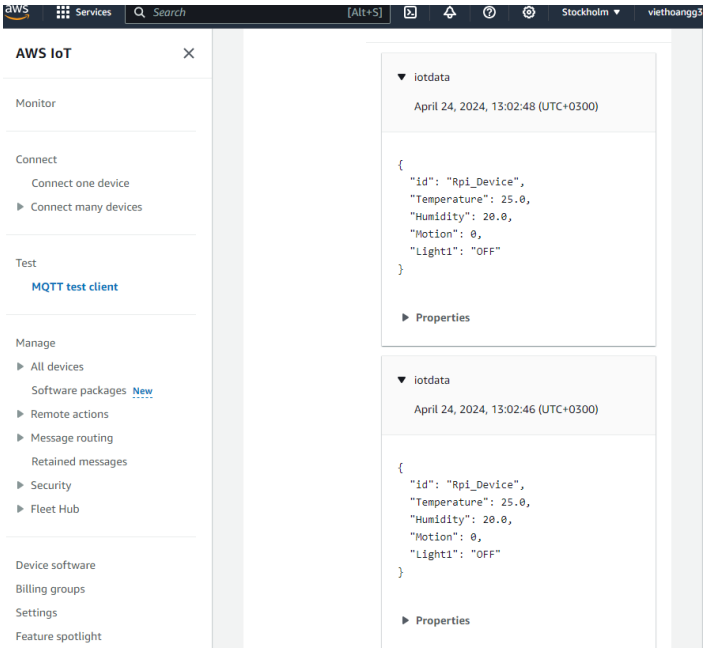

Figure 8. Data stored in AWS IoT Core

4.3 Data Visualization

Node-Red is a visual programming tool for IoT applications, it offers a user-friendly interface to connect devices. Node-Red allows various protocols and cloud services to communicate. A FlowFuse account with basic subscription is required to use Node-Red.

Each nodes contain a different feature, and all connected nodes will have messages flow through. The node either generates a new message or processes an incoming message.

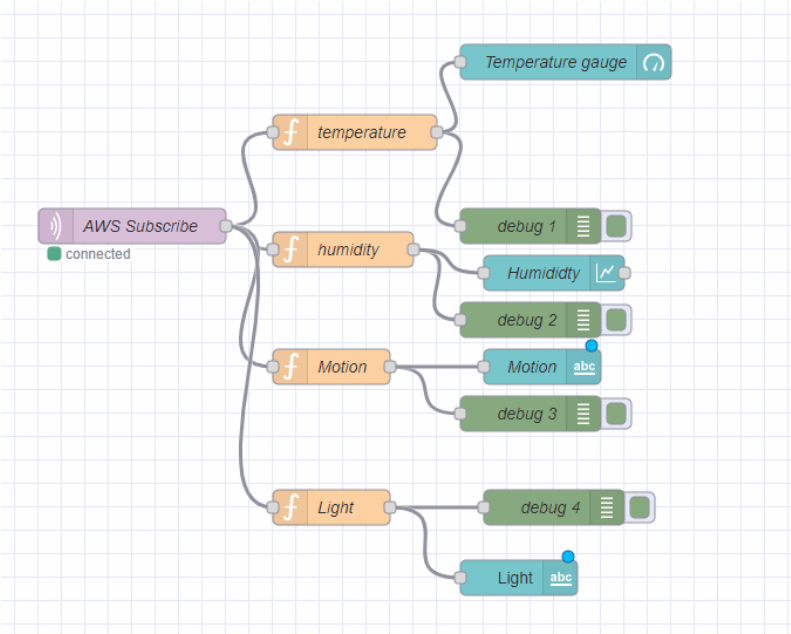

Figure 9. Node-Red system diagram

As shown in Figure 9, there are three types of nodes used in the system. The distributor node is for distributing communication with AWS, the function node for processing data and the widget node for visualizing data.

Commented [KS6]: Choose either but not both

In the configuration of AWS subscribe node, an endpoint link to AWS IoT code needs to be provided, as shown in Figure 10. Key files and certificates provided by AWS IoT Core are also needed to be uploaded to this configuration. This AWS subscribe node will use the endpoint link and those keys to gain access to sensor data that was previously stored in AWS.

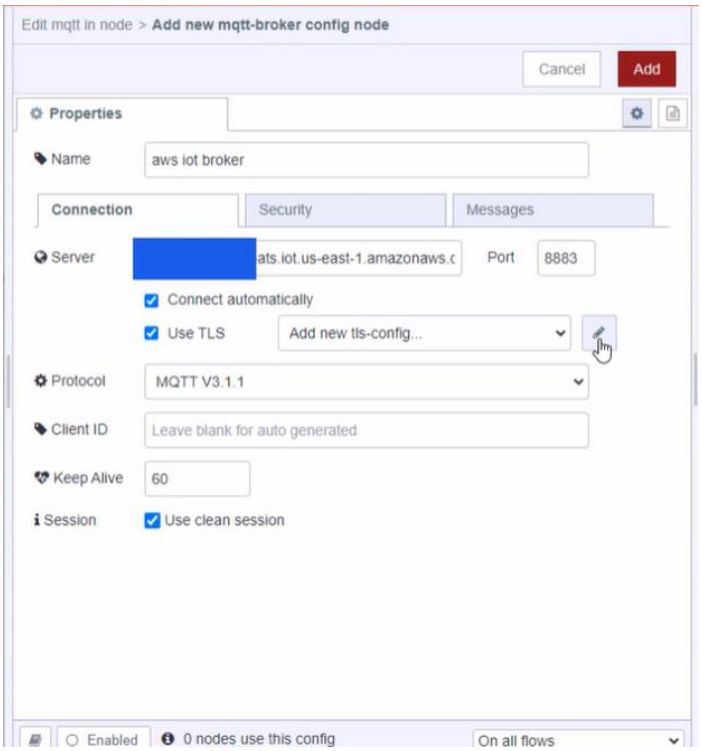

Figure 10. AWS subscribe node configuration

After establishing successful connections, the JSON format data will flow into the AWS subscribe node, as shown in Figure 11. To extract each value out, a JavaScript code needs to be included in each function node, shown in Figure 12. This code isolates data from the incoming payload and sets it as a new payload for further processing and visualization.

```
{
  "id": "Rpi_Device",
  "Temperature": 28,
  "Humidity": 75,
  "Light Status": "ON",
  "Motion Status": 0
}
```

Figure 11. Sample JSON data

```
var x = msg.payload.Humidity;
msg.payload = x;
return msg;
```

Figure 12. Code to separate data in function node

After the function nodes receive the data, it will continue to flow to the widget nodes for data visualization. The widget nodes allow the user to select from various kind of widgets from text message to gauge and chart.

The gauge widget is used for visualizing temperature data; the chart widget is for humidity data and the text widget for displaying status of light and motion detection. The widgets nodes allow the user to have the full customization of the output. Figures 13 and 14 show the configuration of those widget nodes.

After deployment, a web browser generated by Node-Red server will include a user interface output for all the sensors data and light status of user. These data will be streamed from the Raspberry Pi to AWS IoT Core and visualize in Node-Red in real time. Figure 15 shows the output on web browser of the application.

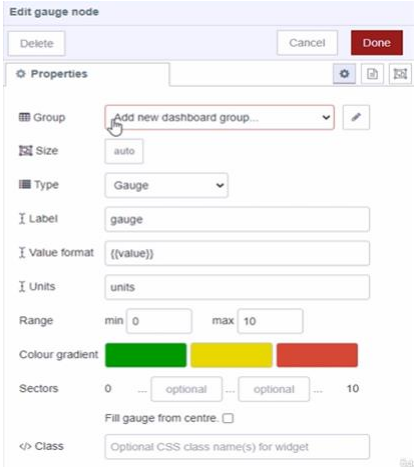

Figure 13. Gauge configuration

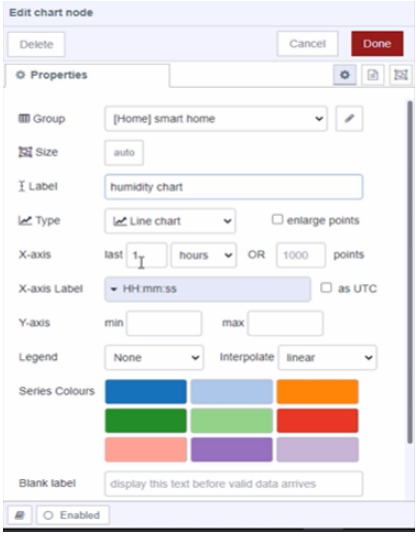

Figure 14. Chart configuration

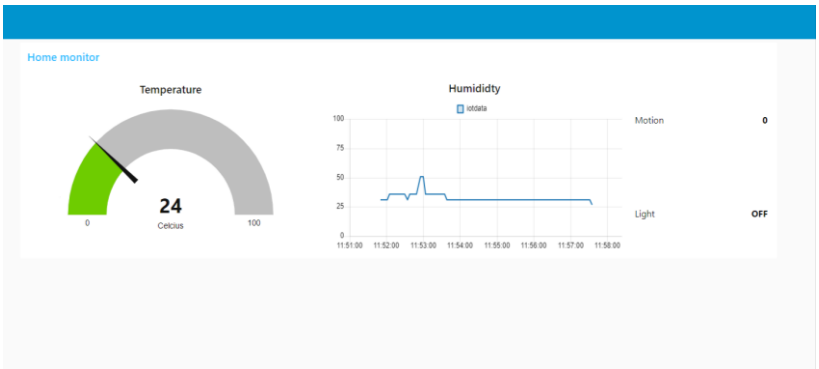

Figure 15. Data visualization output

4.4 Email Notification

Email notification will be carried out using Simple Mail Transfer Protocol (SMTP). Python 3 includes a simple and easy- to -use library for establishing SMPT connection.

For Python preform the action, email credentials need to be provided in the configuration. Python will gain access from those, and an email will be sent to the user's email, also provided in configuration.

SMTP server configuration includes a server link and a port number, in which are smtp.gmail.com and 587, see Code Snippet 3 for more details.

Conditions include a motion detection, extreme recorded temperature, or humidity. The goal of these actions is to act as an alert system to users when they are not in their household. temperature too high or too low could be from a potential malfunction of devices or extreme weather condition, and based on those alerts, the user can take proper actions accordingly. Motion detection notification aims to provide the user with an alert on un-invited guess in certain sensitive area of the household. A sample of email notifications is shown in Figure 16.

```python
import smtplib
import Adafruit_DHT as dht
import RPi.GPIO as GPIO
import time

# GPIO setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(21, GPIO.IN)

# Email configuration
sender_email = "your@gmail.com"
sender_password = "your_app_password"
receiver_email = "to_email_address"
message_template = "Subject: {}\n\n{}"

# SMTP server configuration
smtp_server = 'smtp.gmail.com'
smtp_port = 587

def send_email(subject, body):
    with smtplib.SMTP(smtp_server, smtp_port) as server:
        server.starttls()
        server.login(sender_email, sender_password)
        message = message_template.format(subject, body)
        server.sendmail(sender_email, receiver_email, message)

try:
    while True:
        # Read temperature and humidity
        humidity, temperature = dht.read_retry(dht.DHT11, 21)
        print("Temperature: {:.2f}°C, Humidity: {:.2f}%".format(temperature, humidity))

        # Check motion detection
        motion_detected = GPIO.input(21)

        # Send email if conditions are met
        if motion_detected:
            send_email("Motion Detected", "Motion has been detected.")
        if temperature > 30 or temperature < 10:
            send_email("Temperature Alert", "Temperature is outside the normal range: {:.2f}°C".format(temperature))
        if humidity > 30 or humidity < 10:
            send_email("Humidity Alert", "Humidity is outside the normal range: {:.2f}%".format(humidity))

        time.sleep(2)

except KeyboardInterrupt:
    print("Program terminated by user.")

finally:
    GPIO.cleanup()
```

Code Snippet. 3 Email notification implementation

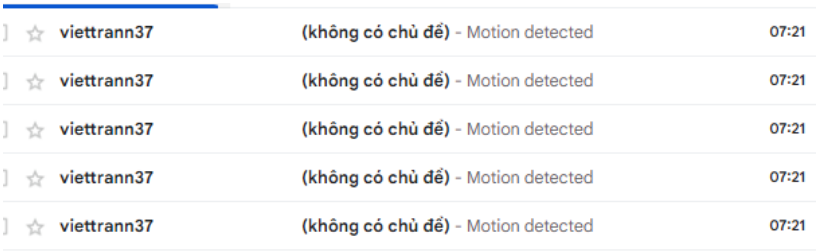

Figure 16. Email notification

4.5 Automated Actions

Certain automated actions will be performed based on the collected data from sensors, including turning light on and off. The PIR motion detector is monitored continuously, if motion is detected, an LED will be turned on accordingly. This feature aims to provide a hands-free experience in a bathroom or a garage, where the lighting system will automatically turn on when the user arrives.

The next feature will also be part of the automation, the LED that was driven on in the previous action or turned on manually by the user, will have a timer, and if the LED has been on for 2 hours, it will be turned off automatically. This feature aims to reduce the user's energy consumption.

Details for implementation are shown in Code Snippet 4.

```python
import RPi.GPIO as GPIO
import time

# Pin Definitions
LED_PIN = 17  # GPIO 17 for LED
PIR_PIN = 26  # Pin 26 for PIR motion sensor

# Setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)
GPIO.setup(PIR_PIN, GPIO.IN)

# Function to turn off LED
def turn_off_led():
    GPIO.output(LED_PIN, GPIO.LOW)

# Function to turn on LED
def turn_on_led():
    GPIO.output(LED_PIN, GPIO.HIGH)

# Main loop
try:
    led_on_time = None
    while True:
        # Check motion sensor
        if GPIO.input(PIR_PIN):
            print("Motion detected!")
            turn_on_led()
            led_on_time = time.time()

        # Check if LED has been on for 2 hours
        if led_on_time is not None and time.time() - led_on_time >= 2 * 60 * 60:  # 2 hours in seconds
            print("Turning off LED after 2 hours of continuous operation.")
            turn_off_led()
            led_on_time = None

        time.sleep(1)

except KeyboardInterrupt:
    print("Program terminated by user.")
finally:
    # Cleanup
    GPIO.cleanup()
```

Code Snippet 44. Automated actions implementation

Formatted: English (United Kingdom)

Formatted: English (United Kingdom)

Field Code Changed

5 CONCLUSIONS

In conclusion, this application provides the user with basic information about the functioning status of their household so that proper actions and measures can be done. The solution is IoT-based so the user can have access to data from far distance if an internet connection is made.

Commented [KS7]: This is not a very good description of the application.

Energy consumption is also reduced also if the user has access to device status data (lights on or off, too high, or too low temperature). The implementation and development of the application in this thesis can be considered successful. All functionalities are realistic and have been deployed successfully.

As for improvement, a log in services needs to be implemented for security access to data. Also, as another improvement factor, energy consumption data should be calculated and provided for the user, as well.

REFERENCES

/1/ Wikipedia. Raspberry Pi. Retrieved 2024-05-06.

https://en.wikipedia.org/wiki/Raspberry_Pi/2/ Adafruit. DHT11 Retrieved 2024-05-06.

https://www.adafruit.com/product/386

/3/ Wikipedia. Motion detector. Retrieved 2024-05-06.

https://en.wikipedia.org/wiki/Motion_detector

/4/ Wikipedia. Python (programming language). Retrieved 2024-05-06.

https://en.wikipedia.org/wiki/Python_(programming_language)

/5/ Twilio Sendgrid.SMTP). Retrieved 2024-05-06.

https://sendgrid.com/en-us/blog/what-is-an-smtp-server#:~:text=Curious%20about%20the%20SMTP%20meaning,to%20send%20and%20receive%20emails.

/6/ AWS. How IoT works. Retrieved 2024-05-06.

https://docs.aws.amazon.com/iot/latest/developerguide/aws-iot-how-it-works.html#:~:text=AWS%20IoT%20Core%20is%20a,IoT%20endpoints%20and%20other%20devices.

/7/ Wikipedia. Node-Red (definition). Retrieved 2024-05-06.

https://en.wikipedia.org/wiki/Node-RED

Commented [KS8]: This link is not available.

Formatted: Default Paragraph Font

Formatted: References