UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dept. of Computer Engineering

Obtaining OpenweatherMap data through Thingsboard property in Raspberry Pi

Master's Thesis

Master's Degree in Computer and Networking Engineering

AUTHOR: Mo , Xiaojie

Tutor: Manzoni, Pietro

ACADEMIC YEAR: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Informática de Sistemas y Computadores

Obtención de datos OpenweatherMap a través de Thingsboard basándose en Raspberry

Obtaining OpenweatherMap data through Thingsboard property in Raspberry

Trabajo Fin de Máster

Máster Universitario en Ingeniería de Computadores y Redes

AUTORA: Mo, Xiaojie

TUTOR: Manzoni, Pietro

CURSO ACADÉMICO: 2021-2022

December 2022

TRIBUNAL:

Presidente: ______________________________

Vocal: __________________________________

Secretario(a): ____________________________

FECHA DE DEFENSA: ________________________

CALIFICACIÓN: _____________________________

Presidente	Vocal	Secretario(a)
Fdo.:	Fdo.:	Fdo.:

ABSTRACT

The measurement and forecasting of weather is being revolutionised by the Internet of Things (IoT). The Internet of Things (IoT) has made it possible for us to more precisely monitor temperature, humidity, wind speed, and other crucial meteorological characteristics. These sensors' data can be used to create complex algorithms that can deliver real-time weather forecasts and warn users of potentially dangerous situations. With the help of modern technology, we can safeguard our communities more effectively and decide on our safety with more knowledge. We still need to do more research on the cost and data reliability challenges.Weather systems' IoT sensors may collect inaccurate or partial data, which could lead to unreliable readings or forecasts. IoT equipment's complexity and need for specialists might make it expensive to install and maintain.

This thesis explores these two elements. We focus on the integration of IoT technology with a weather station based on the low-cost Raspberry Pi single-board computer. The research will centre on the use of the ThingsBoard platform to connect to the OpenWeatherMap API and the Raspberry Pi as the data collection system. In addition, the creation of a web-based front-end to show the data will be investigated. The implementation of this project will provide a comprehensive and cost-effective solution to monitor weather conditions, allowing for better decision-making in many areas of life, such as agriculture, energy management, and public safety. We discuss the advantages of using ThingsBoard, as well as the challenges of integrating such a complex system, and provide details of our proposed system architecture and development process. The results of this project will provide a solid foundation for building more advanced weather monitoring systems.

RESUMEN

Internet of Things (IoT) está revolucionando la medición y el pronóstico del tiempo. Internet of Things(IoT) nos ha permitido monitorear con mayor precisión la temperatura, la humedad, la velocidad del viento y otras características meteorológicas cruciales. Los datos de estos sensores se pueden utilizar para crear algoritmos complejos que pueden ofrecer pronósticos meteorológicos en tiempo real y advertir a los usuarios sobre situaciones potencialmente peligrosas. Con la ayuda de la tecnología moderna, podemos salvaguardar nuestras comunidades de manera más efectiva y decidir sobre nuestra seguridad con más conocimiento. Todavía necesitamos investigar más sobre los desafíos de costos y confiabilidad de los datos. Los sensores IoT de los sistemas meteorológicos pueden recopilar datos inexactos o parciales, lo que podría conducir a lecturas o pronósticos poco confiables. La complejidad de los equipos de IoT y la necesidad de especialistas pueden hacer que su instalación y mantenimiento sean costosos.

Esta tesis explora estos dos elementos. Nos enfocamos en la integración de la tecnología IoT con una estación meteorológica basada en la computadora de placa única Raspberry Pi de bajo costo. La investigación se centrará en el uso de la plataforma ThingsBoard para conectarse a la API OpenWeatherMap y Raspberry Pi como sistema de recopilación de datos. Además, se investigará la creación de un front-end basado en la web para mostrar los datos. La implementación de este proyecto proporcionará una solución integral y rentable para monitorear las condiciones climáticas, lo que permitirá una mejor toma de decisiones en muchas áreas de la vida, como la agricultura, la gestión de la energía y la seguridad pública. Discutimos las ventajas de usar ThingsBoard, así como los desafíos de integrar un sistema tan complejo, y brindamos detalles de la arquitectura del sistema y el proceso de desarrollo propuestos. Los resultados de este proyecto proporcionarán una base sólida para construir sistemas de monitoreo meteorológico más avanzados.

ACKNOWLEDGMENT

This work serves as a pivotal moment in both my professional and personal lives. I can say with pride that they have made me very happy. I had a variety of companions with me on this voyage, and I dedicate these words to them.

My life is going through a lot of change in 2020. I moved out of my hometown and my parents to live alone. This is a new life and a major challenge. I've developed over the last three years from a young teenager to a strong, independent adult.

I'm appreciative of myself for persevering and not giving up while working hard to achieve my goal. With optimism and positivity, face all the joy and sorrow in life.

I'm appreciative of my father's support. It's amazing how much spiritual energy you have. For me, you serve like a sturdy tree. You made everything better, and I will always adore you.

I'm grateful to my mother for helping me follow my aspirations because with you by my side, anything is possible. You shoulder a lot. I'll walk alongside you as you make your way forward.

I appreciate to Cheng, my beloved friend, for your constant concern for me, encouragement, and support. You compare me to the sun, but to me, you are the sun. Miss you.

I am appreciative of everyone who has ever supported me in my life. My life is full of hope as a result of you. I appreciate your generosity and zeal. I'm wishing you the greatest amount of luck possible.

I would like to express my sincere gratitude to my director Pietro for his guidance and support throughout my thesis studies. Your expertise and perspective were crucial to completing this task. I appreciate all of your support and encouragement during this process.

You inspire me and many others with your dedication to science and study. I appreciate the chance to collaborate with you on this project, and I am extremely proud with the results.

Life's pictorial scroll has been unrolled, and I will fill it with the most exquisite hues.

Mo, Xiaojie

Valencia, 2022

GENERAL INDEX

IMAGE INDEX

LIST OF ACRONYMS

5G	5th Generation wireless systems
API	Application Programming Interface
APPID	Application Identification
ARM	Advanced RISC Machines
COAP	The Constrained Application Protocol
CPU	Central Processing Unit
CRUD	Create/Read/Update/Delete
ECDSA	Elliptic Curve Digital Signature Algorithm
GPIO	General Purpose Input Output
GUI	Graphical User Interface
HDMI	High Definition Multimedia Interface
HTTP	Hyper Text Transfer Protocol
IoT	Internet of Things
JSON	JavaScript Object Notation
LAN	Local Area Network
LWM2M	Lightweight Machine to Machine
MCU	Micro Control Unit
MQTT	Message Queuing Telemetry Transport
OAUTH	Open Authorization
OS	Operating System
PC	Personal Computer
RAM	Random Access Memory
REST	Representational State Transfer
RSA	RSA algorithm
SQL	Structured Query Language

UI	User Interface
URL	Uniform Resource Locator
USB	Universal Serial Bus
UV	Ultraviolet
VNC	Virtual Network Console
XML	Extensible Markup Language

1. INTRODUCTION

Thanks to advances in computing and networking technology, traditional computing systems have been able to be integrated into everyday objects and interconnected, forming a global network infrastructure known as the Internet of Things.[1] IoT is a network of physical items that are connected to the internet and have the ability to gather data, communicate with one another, and take action based on that data. Its name literally translates to "connected network of everything." These things can be anything, from wearable devices to vehicles to machines used in manufacturing, for example. In point of fact, it has undergone significant development over the course of the past 20 years, ever since the concept of auto-identification was first proposed in the United States of America in 1999, and is now widely utilised in a variety of contexts. The Internet of Things is an extension and expansion of the core of the Internet that enables the exchange and communication of information between objects through wireless networks with minimal intervention from humans. Additionally, the Internet of Things is a network that is built from physical objects. The Internet of Things lowers the cost of human labour while also increasing the speed at which data can be transmitted and processed.

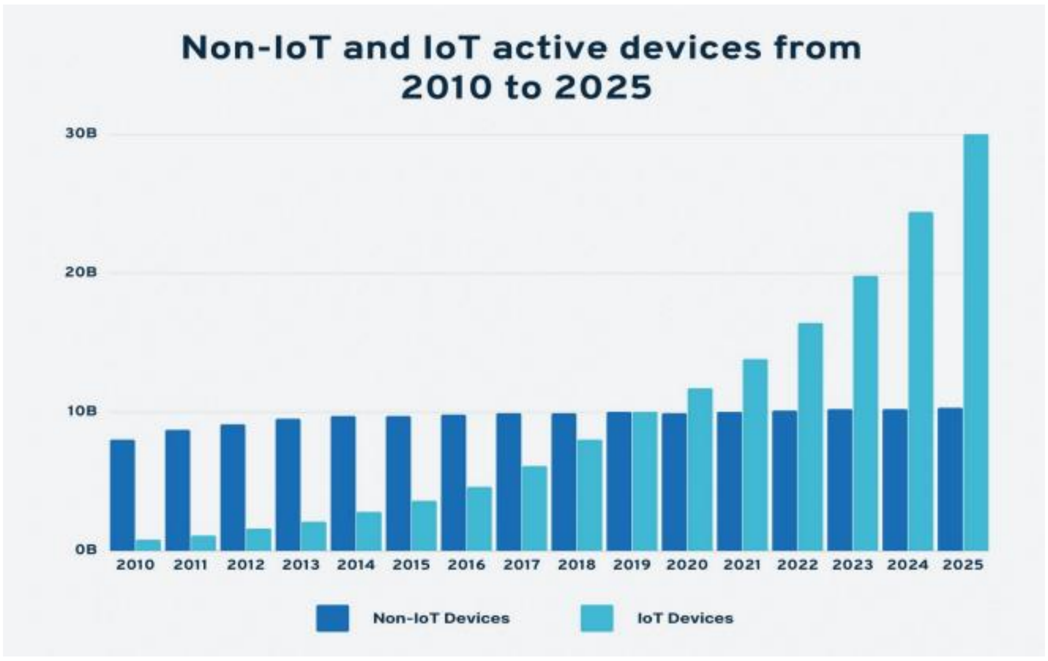


Figure 1-1: IoT devices on a Global level

The Internet of Things has the potential to completely transform how we live, work, and play in the world. It has the potential to make our lives easier and more efficient, as well as assist us in better comprehending and navigating the environment that we live in. IoT is a disruptive force that is redefining how humans and physical items communicate with one another and with each other in the digital world. It is anticipated that there will be 75 billion connected devices in use around the globe by the year 2025. IoT is still in its infancy, and there are a great number of obstacles that must be overcome before it can realise its full potential. Security, privacy, scalability, and interoperability are some of the problems that must be overcome.

This project is based on IoT, and its main goal is to build a weather station using the Raspberry Pi computer, the Thingsboard IoT platform, and the OpenweatherMap service. The weather station will employ sensors to collect data from weather stations across the world using the OpenweatherMap API. The sensors will detect temperature, humidity, and atmospheric pressure, among other weather variables. A dashboard made with the Thingsboard platform will show the data.

1.1 MOTIVATION

Building a weather station that may be utilised to gather information about the local weather conditions is the reason we are working on this project. We may utilise this information to better understand the regional weather patterns and to guide our choices regarding how to safeguard our homes and places of business from the effects of severe weather. Our weather station is built around a Raspberry Pi since it is an affordable, adaptable platform that is perfect for this kind of project. ThingsBoard offers us a robust set of capabilities for data gathering, analysis, and visualisation, thus we are adopting it as our platform for data visualisation and analysis. We are utilising OpenWeatherMap to give us up-to-date information on the local weather. And in order to share our data with the world, we are using the IoT protocol to connect our weather station to the internet.

This project will make use of the Thingsboard platform to receive data from OpenweatherMap. The data will be stored in a database on the Raspberry Pi 4. A Python script will be used to retrieve the data from OpenweatherMap and store it in the database. A web interface will be created using the Flask framework. This web interface will allow the user to view the data stored in the database. The user will be able to select a location for which they want to view the weather data.

1.2 OBJECT

In this project, we'll turn the Raspberry Pi into a weather station that uses the ThingsBoard open-source IoT platform to communicate data to the cloud. On a dashboard, the data will be gathered from the OpenWeatherMap service and shown in real-time. This project is a great way to get started with the Internet of Things and learn about weather data, cloud computing, and how to build dashboards using IoT.

1.3 STRUCTURE

The remaining essay has the following organisational structure:

- Chapter 1: **Introduction**: Aims, objectives and Motivation for this document;
- Chapter 2: **State-of-the-art**: Gives some background information on modern technology and discusses both the benefits and the drawbacks of using it;
- Chapter 3: **Problem analysis**: The investigation of feasible solutions, the decision-making process, and the justifications behind the many design options that were ultimately selected for the particular implementation;
- Chapter 4: **Proposed Solution**: Discuss the proposed solution and provide an explanation of the structure of the system as well as the technologies that were utilised;
- Chapter 5: **Implementation**: Demonstrate the comprehensive development process and project findings;

- Chapter 6: **Conclusions**: Examines the results and outlines potential avenues for further research.

2. STATE OF THE ART

Anang Suryana et al[2]. experimented with using weather station data to imitate a room weather station. Employing their study apparatus, which included a DHT22 sensor for monitoring temperature/humidity, a BMP180 sensor for measuring wind/wind speed, and a BH1750 sensor for measuring light intensity. Their research apparatus includes a DHT22 temperature/humidity sensor, a BMP180 wind/wind speed sensor, and a BH1750 light intensity sensor. They used a weather regulator, which included a heater and cooler, water jets, fans, and light bulbs for temperature, humidity, wind speed, and light intensity, to set the room to the same weather conditions in order to compare the weather data there with the weather data they obtained from OpenweatherMap. The regulator was managed by an Arduino Uno microcontroller.

Yasser Asrul Ahmad et al.[3] conducted experiments to compare the performance of a networked temperature system to an industrial system. The experiment found that the IoT temperature detection system using the DHT22 sensor had higher sensitivity and accuracy, allowing accurate measurements of up to 0.10 degrees Celsius. The project built an IoT temperature monitoring system using the DHT22 temperature sensor with a Raspberry Pi and compared it to an industrial-grade thermocouple with a Keithley 6517-TP industrial-grade temperature probe.

Dushyant Kumar Singh et al.[4] carried out research on low-cost IoT-based systems for weather monitoring at the local level. Additionally, for data receiving, calibration, and data posting to the cloud, Node MCU served as the primary control device. The wind speed sensor and wind direction sensor are utilized in the data acquisition phase as analog sensors to simulate the output signal, and the DHT11 sensor is used to detect temperature and humidity. The project sends data from a month's worth of recording to the Thingspeak server. The experiment suggests that this system has outstanding performance in terms of low cost and monitoring dependability, and can operate continuously for more than 10 hours without experiencing any mistakes.

Anshika Gupta et al. [5]carried out research to create a smart tiny weather station for more compact regions. The project, which was built on the ESP32 platform, employs sensors to gather information about temperature, humidity, UV index, and rainfall, which is then wirelessly sent to the firebase cloud database. An Android application designed exclusively for data visualization has been created by the project, and the data is refreshed every 5 seconds. The weather station can run for 11 hours without external power support thanks to the system's two 2200mAh solar panels.

Thomas Lee Scott et al.[6] describe using the ThingsBoard IoT platform to create IoT monitoring systems that upload sensor data to the cloud using the CoAP protocol to investigate how CoAP fits into the Internet ecosystem. The project connects sensors to Raspberry Pi devices as IoT nodes, gathers sensor data, and then periodically sends JSON data to the ThingsBoard cloud endpoint over CoAP. Real-time IoT dashboards are then created to display the sensor data and make it available to users.

Christine Dewi et al.[7] explore using the python programming language to broadcast real-time weather data gathered from OpenweatherMap to Twitter To help users who desire social weather forecasts. The OAuth and Tweepy libraries are used in this study to connect Python programming to Twitter.

2.1 PROPOSAL

Receiving data through a REST API that is external to a large weather website allows you to observe and collect weather data from more cities for future analysis, whereas a self-built analogue weather station can often only monitor weather changes in one area. Large weather data websites have more professional equipment with lower error rates to provide more accurate data. The specialised websites are able to offer a more comprehensive selection of services, which may include weather predictions, sun radiation, historical weather, and other similar topics. At the same time, low-powered, compact devices have a number of benefits, including the fact that they are less expensive, require less power, and may be used for monitoring on a broad scale. The entirety of the system is simple to operate and is completely automated, which enables monitoring to occur nonstop without the need for human intervention. Additionally, the creation of real-time weather dashboards via the ThingsBoard renders the data more suitable for use in a commercial setting.

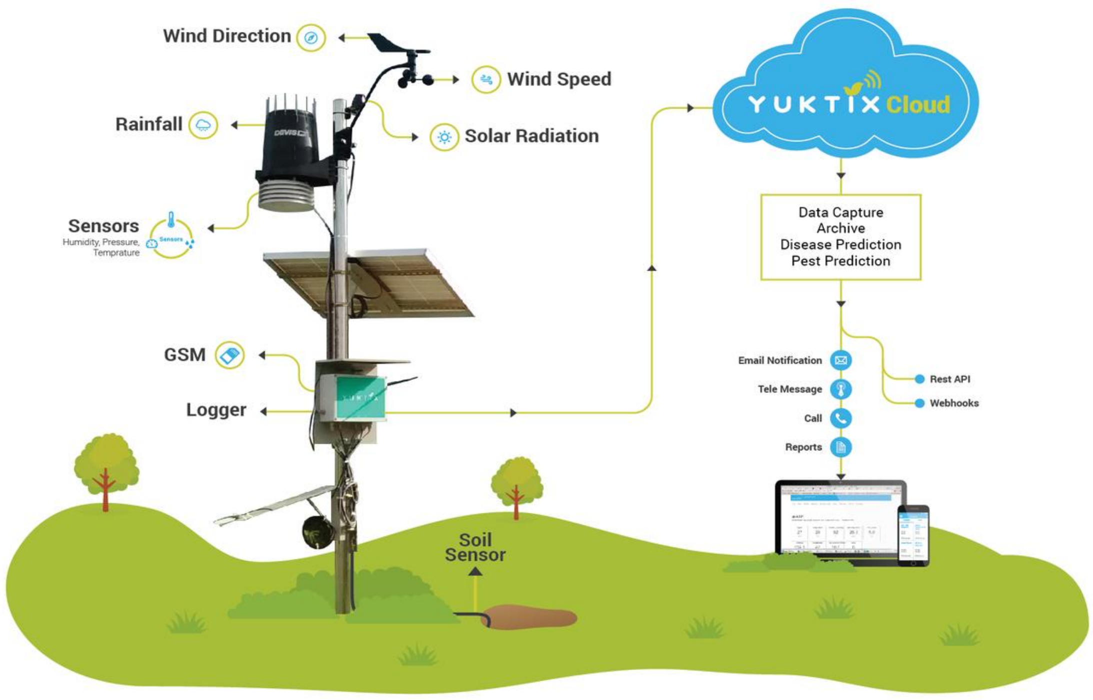


Figure 2-1: Weather station

3. PROBLEM ANALYSIS

The purpose of this thesis is to provide an answer to the question of how one can make use of a Raspberry Pi 4 to establish a connection to Thingsboard in order to acquire data from OpenweatherMap. The objective of the thesis is to create a system that is able to be used to receive data from OpenweatherMap, and then to use this data to control a device that is connected to the Raspberry Pi 4, and to visualise the data in ThingsBoard as a dashboard for viewing purposes. This will be accomplished by developing a system that can be used to receive data from OpenweatherMap. The following procedures need to be carried out in order to accomplish this goal:

1. What are the procedures involved in configuring a connection between the Raspberry Pi 4 and the Thingsboard?
2. How exactly does the Thingsboard interface operate, and what kind of information can be obtained from OpenweatherMap?
3. What kinds of trends and patterns are discernible in the information that was obtained from OpenweatherMap?

3.1 Identification and analysis of possible solutions

3.1.1 Demo Board

Demo boards are boards used for the development of embedded systems. They contain a number of hardware components, such as memory, central processors, input/output devices, data buses, and external resource interfaces. These components enable the user to interact with the product or service in a variety of different ways, such as through the use of a keyboard, mouse, or touch screen. The majority of the electronic gadgets that people use in their day-to-day lives are first created on a development board, where they are tested for functioning, validated for practicality, and then engineered to be portable through the use of integrated circuitry for consumer usage. In the context of this project, the demo board serves the role of the intelligent device in the perception layer of the Internet of Things architecture. In this section, I will discuss a few demo boards that are now in use.

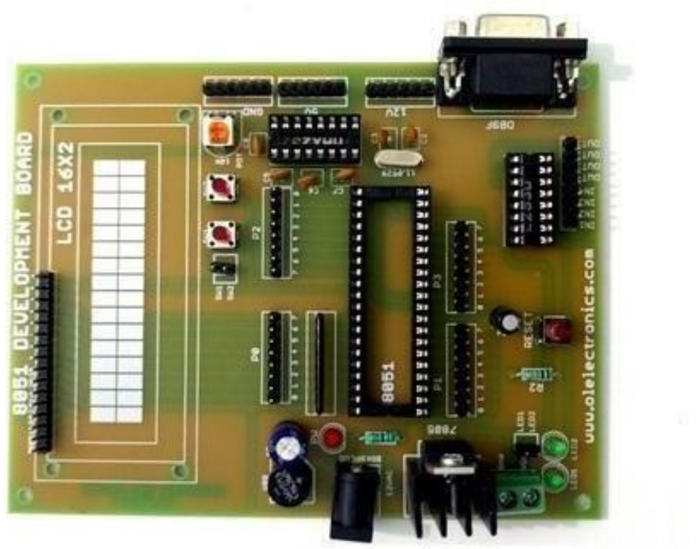


Figure 3-1: Demo board

● Arduino

Arduino[1] is a platform for creating electronic projects that is built on open-source software and user-friendly hardware. Because it is simple to use and does not need for the installation of any extra drivers, it has found widespread application in the field of electronics design. Its target audience is anybody who wants to create interactive projects.

The Arduino Uno, which is built on the ATmega328P and includes all of the necessary components for a microcontroller, is by far the most popular and widespread option. Arduino is able to detect its surroundings by receiving data from a number of different sensors. Additionally, Arduino is able to influence its surroundings by controlling a number of different lights, motors, and other actuators.

The Arduino programming language, which is based on Wiring, and the Arduino development environment are both used to write code for the microcontroller that is located on the board (based on Processing). Projects built using Arduino can function alone, or they can interface with software that is being run on a computer (e.g. Flash, Processing, MaxMSP).

When it comes to transmitting circuits or programmes, customers have access to a great deal of ease thanks to the unified architecture as well as conventional digital and analogue signal ports. The extensive peripheral connections make it possible to connect to a wide variety of modules and sensors, which is useful for a variety of applications like environmental monitoring stations, intelligent vehicles, and more. Its advantages include a low price and a plethora of general-purpose input/output (GPIO) pins, but it is not very good at meeting the more rigorous requirements placed on the CPU, such as those pertaining to image processing and complicated computation.

[1] https://www.arduino.cc/

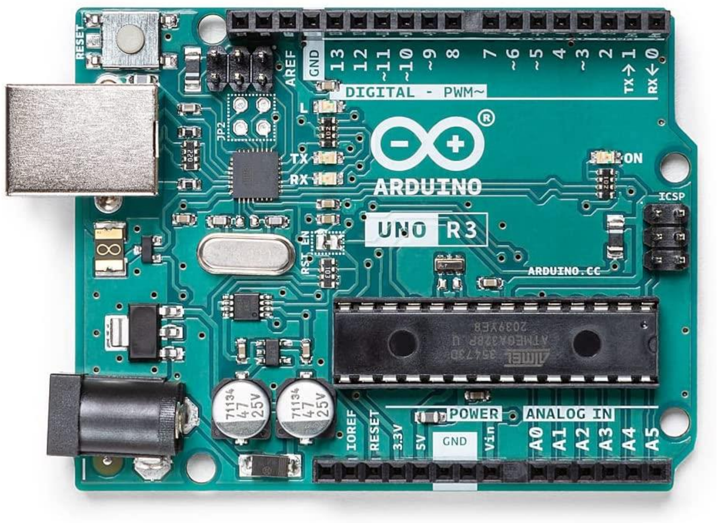


Figure 3-2: Arduino board

● ODROID C4

An open-source hardware business based in South Korea called Hardkernel Co., Ltd. developed a new generation of computing devices called ODROID. These new devices have the benefit of having technology that is both more powerful and more energy-efficient while also having a smaller footprint.

In addition to being able to run operating systems that are based on the ARM architecture, such as Android, Debian, and Ubuntu, the board is also capable of running Linux and other variants of the Android operating system. The CPU is an Amlogic S805, which is a quad-core version of the Cortex-A5 architecture and features Mali-450 MP2 graphics. The maximum frequency that the processor is capable of operating at is 1.5 GHz, while the maximum frequency that the graphics can operate at is 600 MHz. The circuit board has 1 gigabyte of random access memory (RAM), 8 gigabytes of eMMC storage, and a slot for a MicroSD card for extra storage. The motherboard features a Gigabit Ethernet port, which allows for faster data transfer speeds and more consistent performance. Additionally, the cooling on the motherboard is built-in, so it can function without having to be throttled due to overheating.[2] Applications for image encoding that need a greater resolution, including digital photo frames, the integration of environmental sensors into entertainment systems, and gaming suite combinations, are all areas in which the ODROID C4 works exceptionally well.

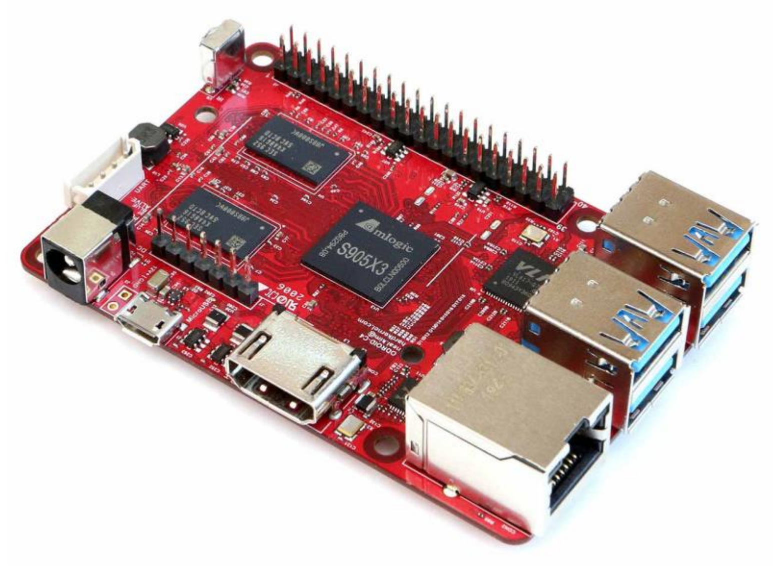


Figure 3-3: ODROID C4 board

● Raspberry Pi 4

The Raspberry Pi[3] is a Linux-based microcontroller computer that is about the size of a credit card and costs just $35. It features a powerful quad-core Cortex-A72 processor, up to 4GB of RAM, dual-display support at resolutions up to 4K, USB 3.0 connectivity, and improved thermal management, and the main operating system is Raspbian, a Debian-based Linux distribution with over 350,000 software packages in its repository. This forms an ecosystem that supports various programming languages, including Python, Java, C....

[2] https://www.hardkernel.com
[3] https://www.raspberrypi.org/

It offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+, while maintaining backwards compatibility and similar power consumption. It is now a complete desktop computer. It is the most powerful Pi board to date. It has three USB ports, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 5.0/BLE, faster Ethernet, and PoE capability through a separate PoE HAT, and it can power two independent 4K displays at the same time. Additionally, the connectivity and speed of Bluetooth and the dual-band wireless LAN have been improved. It provides an Ethernet connector, a USB port for a keyboard and mouse, an SD card expansion port, and an HDMI port for high-definition video output, which may be used with an external monitor or television.

With the capability of the Raspberry Pi, it is possible to realise a range of projects, such as weather monitoring systems, smart home devices, and industrial IoT systems. These projects may be realised by connecting an abundance of GPIOs to a variety of peripherals. Another significant benefit of using Raspberry Pi is its low power consumption, which enables it to run nonstop for seven days and seven nights, continuously monitoring data. The Raspberry Pi has a good eco-system, greater processing performance, and a low price, which have helped it attract a lot of supporters. Additionally, the Raspberry Pi's one-of-a-kind architecture performance contributes to the development and implementation of the Internet of Things. For the end user, the desktop performance of the Raspberry Pi 4 Model B is the same as that of an entry-level x86 PC.

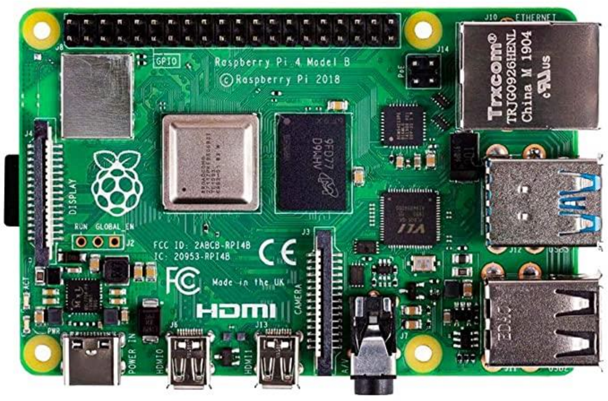


Figure 3-4:   Raspberry Pi 4 board

We chose the Raspberry Pi as the demo board for this project because of its low cost, low power consumption, strong performance in programming and processing hardware and software applications, and the abundance of reference materials available to help in the development of the project.

3.1.2 IoT Platforms

The Internet of Things is made up of these three primary parts:

Actuators and sensors make up the components of the sensory layer. Temperature sensors, humidity sensors, motion sensors, and so on are all examples of the types of sensors that fall into this category. These are the devices that collect data about the physical world and turn it into digital information that can be processed by computers.

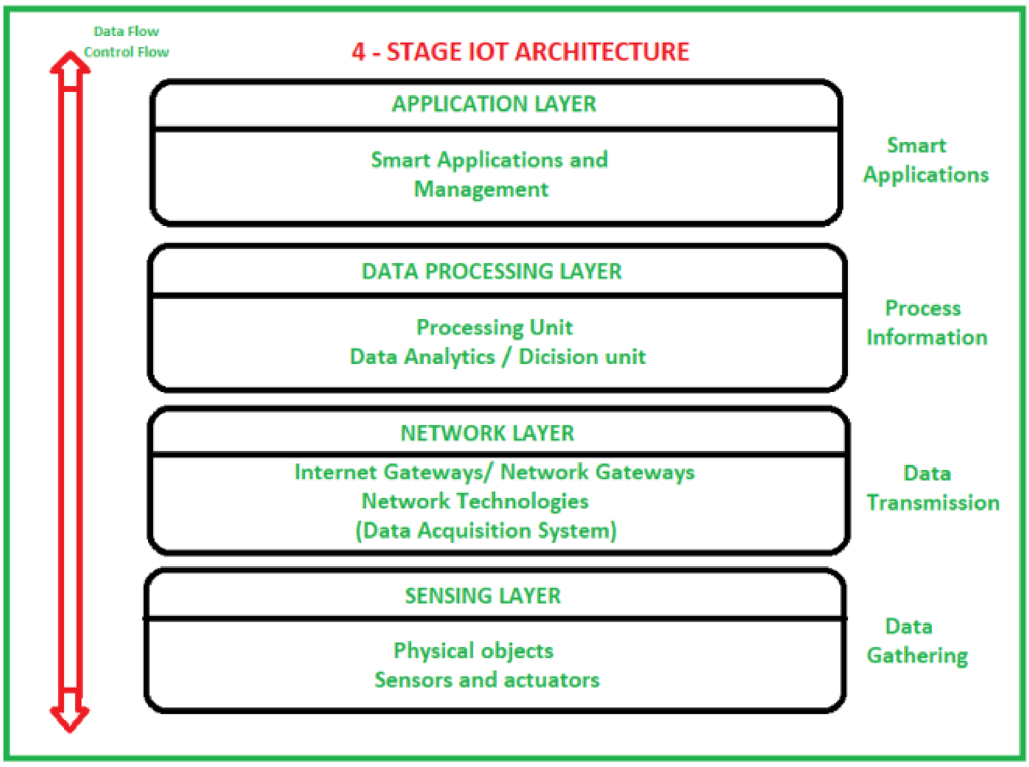


Figure 3-5: IoT Architecture

The internet and a network of physical items that are connected to one another make up what is known as the network layer. It establishes a connection between the physical things and the internet so that the information collected at the sensing layer may be sent along the network. Wi-Fi, Bluetooth, cellular data, ZigBee, and other similar technologies are examples of common types of network connections. The use of 5G technology in the future will also be of assistance to the development of IoT.

The application layer is the objective of the development of the Internet of Things. Its job is to process, store, and analyse the data that is collected from the sensing layer. It then combines this data with the requirements of industry to produce intelligent applications.

IoT platforms are an essential part of the larger IoT ecosystem, a network that connects devices and allows for social engagement and data exchange. It typically consists of a server that manages connection protocols and a software programme that users can use to interact with the devices. This configuration supports and connects all system components to perform tasks like managing software/hardware connection protocols and controlling the devices. The platform could also include a communication network to link the devices together and a database to store data from the devices. also used for data collection, analysis, and visualisation.

The ability to connect and remotely operate devices is the major advantage of adopting a IoT platform. This may be helpful for a number of purposes, including managing security systems, reducing energy consumption, and monitoring equipment performance. Some of the more sophisticated IoT platforms may be configured to start alerts, monitor or control equipment from a distance, etc. A IoT platform may also offer insights regarding people and device behaviour that can be utilised to enhance the effectiveness of procedures and operations. Some popular IoT platforms include the ones listed below.

- **FIWARE**

FIWARE[4] is a collection of open-source software tools that facilitate the creation of intelligent cloud applications by programmers in a straightforward manner. The FIWARE platform is made up of a collection of generic enablers that can be put to use in the process of developing applications for a broad variety of industries, including but not limited to smart cities, smart energy, eHealth, agriculture, and transportation, amongst others.

FIWARE offers developers a collection of tools that are intuitive to work with so that they may create cloud-based apps. The FIWARE platform is made up of a collection of generic enablers, which are pieces of software that may be repurposed in the process of developing applications for a variety of different industries. The RESTAPI may execute CRUD activities in contextual information. The fundamental system is Orion —context agent, which manages, conducts updates, and accesses contextual information. The Internet of Things, robots, and third-party systems can all be connected using these components; contextual data can be managed, published, and monetized; information can be processed, analysed, and visualised; and so on. The Context Broker enabler can be used to build applications that need to manage and query large amounts of data in real time, such as a smart city application that needs to track the location of city buses in real time.

The simplicity with which it enables developers to build apps that run in the cloud is the primary advantage offered by FIWARE. It offers a collection of reusable software components that may be incorporated into the production of applications for a variety of industries. This enables developers to rapidly design new apps without having to start from the ground up, which saves them both time and effort. The platform does not provide message encryption which negatively affects its security.[8]

[4] https://www.fiware.org/developers/catalogue/

Because FIWARE is an open source project, it is possible for anybody to make contributions to the continued growth and improvement of the platform. The FIWARE community is made up of developers from all over the world who are collaborating to advance the platform in some way.

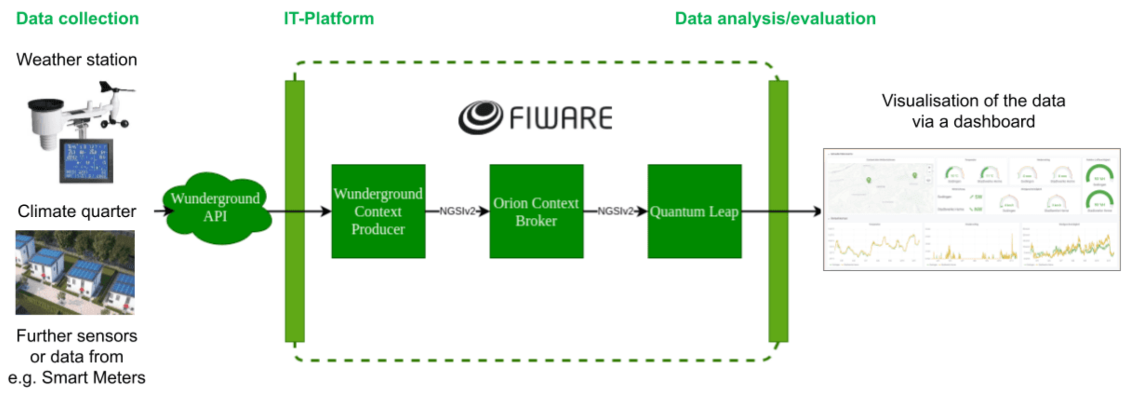


Figure 3-6: Fiware system Architecture of Smart sensor monitoring weather

● SiteWhere

SiteWhere[5] is an open-source Internet of Things application support platform that is built to industrial standards. It enables the large-scale collecting, storage, processing, and integration of device data. It offers features for device management, data management, and analytics for applications related to the Internet of Things.

A method known as "microservices" is used for the architecture of SiteWhere. The platform is made up of a collection of services that may be deployed independently of one another and communicate with one another via a clearly defined API. Because of the architecture, both new services and replacements for current ones can be implemented without disrupting the functionality of the platform as a whole. In order to handle the load and scale of large-scale Internet of Things projects, SiteWhere makes

[5] https://sitewhere.io/es/

use of a number of different technologies, including Apache Kafka and Docker. Microservices can be scaled independently at the level of each component thanks to the distributed design, which enables the system to be tailored to the manner in which the client actually employs it.

SiteWhere is functional because it is able to connect many types of devices to the platform via a range of protocols, including HTTP, MQTT, and CoAP. Once a device has been linked, it will be able to send data to SiteWhere, and that data will be recorded in a database that tracks time series. The capabilities of the platform for doing analytics can then be used to query and examine this data. The Communication Engine and the Tenant Engine, both of which are contained as fundamental elements, are what are responsible for connecting devices and other applications. It provides device specification, device groups, asset allocation, and a comprehensive management GUI.[9]

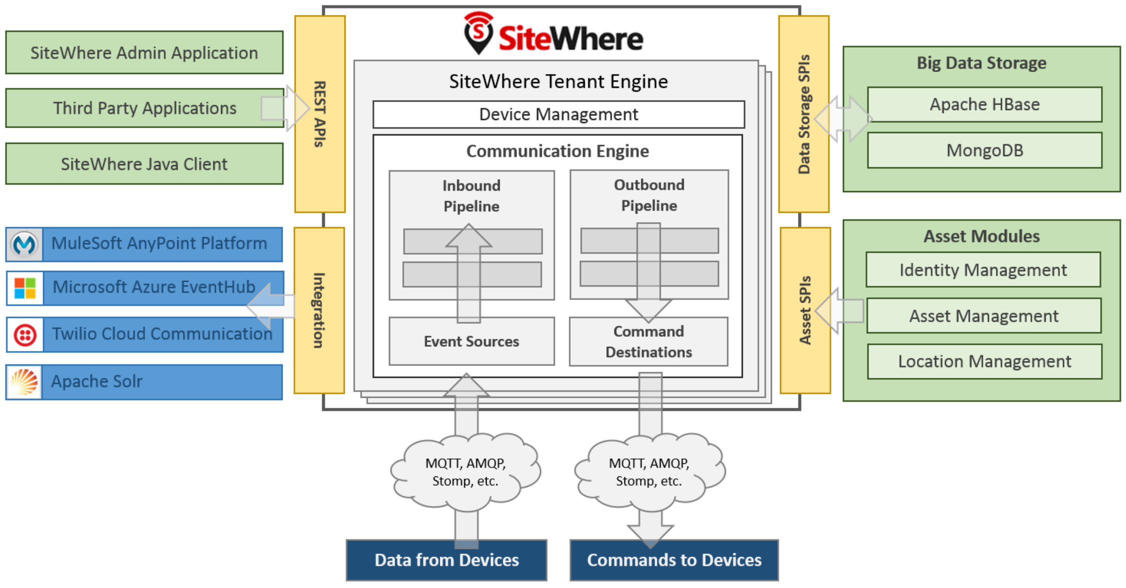


Figure 3-7: SiteWhere system Architecture

- **ThingsBoard**

Thingsboard[6] is an open-source Internet of Things platform that enables users to visualise data, monitor it, and manage connected devices. It is possible to deploy it either on your own premises or in the cloud, and it is designed to function with a wide variety of different devices and protocols. The modular design of Thingsboard, which is comprised of a variety of different services, is used in its construction. These services may be colocated on the same node or deployed separately on distinct nodes. Alternatively, they may be colocated on distinct nodes. ThingsBoard supports different standard IoT communication protocols (MQTT, CoAP, and HTTP) for device connectivity and supports both local and cloud deployments.[10] Scalability, high fault tolerance, and consistent performance are three of the many advantages offered by ThingsBoard, industry-standard encryption technologies that are industry-standard, such as RSA and ECDSA, as well as high security to ensure that users do not lose data while it is being transmitted. Analytics, individualised widgets, adaptable dashboards with real-time data visualisation, the ability to set off alarms on demand, and a great deal more are all included.

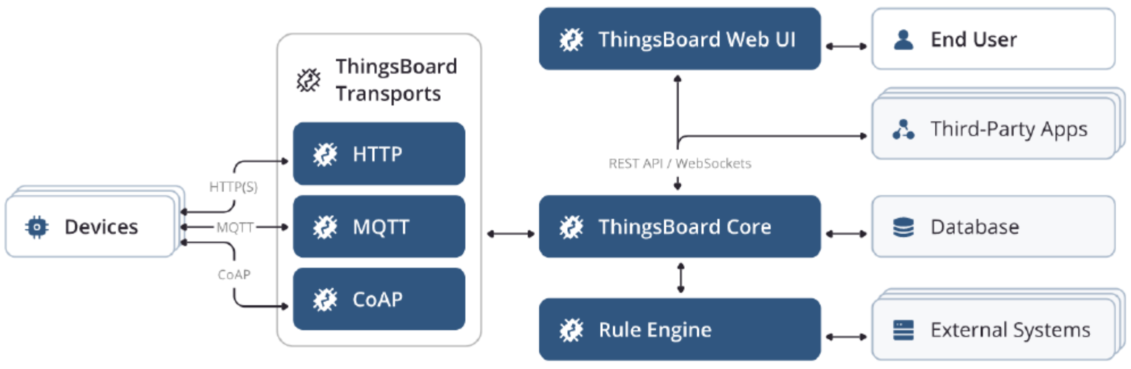


Figure 3-8: ThingsBoard system Architecture

[6] https://thingsboard.io/

ThingsBoard was selected as the platform for the Internet of Things for this experiment because it is an open-source server-side platform that has a mature design and a rich and complete set of examples on its website. Some of these examples include smart energy, smart agriculture, fleet tracking, environmental monitoring, and other similar applications. Additionally, it has a higher level of activity on GitHub, and the documentation and guidelines that are offered on the website are quite easy to understand and use. ThingsBoard allows installation on a variety of operating systems, including Windows, Linux, Raspberry Pi, and Docker, as well as supporting HTTP API, MQTT API, LwM2M API, IoT gateway, and other device communication methods, and it provides an interface for REST API. managing devices and assets, obtaining telemetry data, and performing very well in terms of data processing, data visualisation, and security are all part of this solution.

4. PROPOSED SOLUTION

For the purpose of gathering information from OpenweatherMap, this project will make use of the Thingsboard platform. The Raspberry Pi 4 will serve as the storage device for the information in the form of a database. The information will be retrieved from OpenweatherMap with the use of a Python script, and then it will be saved in the database.

The Flask framework will be utilised in the development of a web interface. The user will have the ability to view the data that is saved in the database through the usage of this web interface and will have the ability to choose a place at which they wish to view the corresponding weather data.

4.1 PLAN OF THE WORK

The primary objective of the design of this project is to automatically detect weather data for a specific location, receive the data once every 15 seconds and save it in a database, perform all data processing through the ThingsBoard server, and display the dashboard in the form of an alignment chart to make it simple for the user to observe and evaluate the data. The Raspberry Pi 4 is the primary computing unit. Both the ThingsBoard server and the open-source database management system PostgreSQL have been installed on the Raspberry Pi 4 computer. The Raspberry Pi4 and the monitor can both be connected to the HDMI connector in order to display the operating system graphical user interface. You may browse the ThingsBoard website by logging in directly through the Raspberry Pi4, or you can use VNC Connect to control the Raspberry Pi remotely from another location.

4.2 SYSTEM ARCHITECTURE

The project architecture that was designed for this project is depicted in figure. The IoT node for this real-time weather system is a microcontroller called a Raspberry Pi. This Pi is equipped with Wi-Fi so that users can easily obtain instant weather data from the device. The Raspberry Pi setup includes the installation of the ThingsBoard server as well as PostgreSQL for data storage. The ThingsBoard uses a REST API to establish a connection to OpenweatherMap in order to retrieve the weather information for the cities whose names are stored on the server. The data is sent to ThingsBoard in the form of a JSON string, where it is then processed by the rules engine to extract the data that the user requires. The data is then stored and updated in the ThingsBoard server dashboard, where the user can monitor and record both real-time data and historical data.

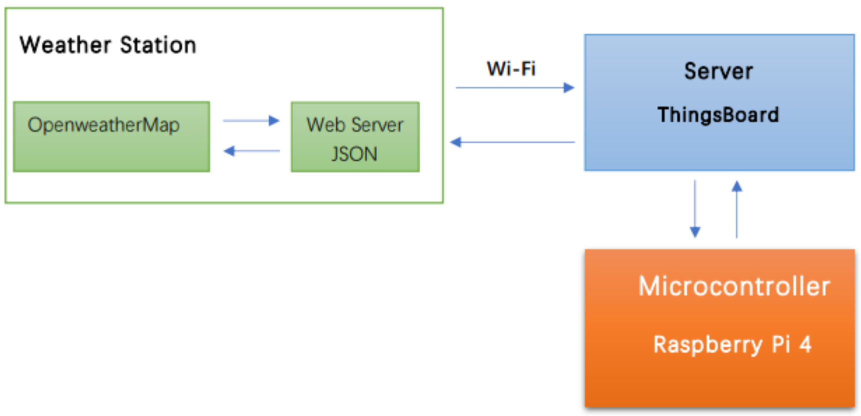


Figure 4-1: Project architecture

ThingsBoard makes advantage of the REST API to make calls to the Rule Chain in order to read the weather data provided by OpenweatherMap. The procedure for the receipt of data is depicted here in Figure 4.2: Make a Generator node to produce the request, obtain the API key, and configure the server attributes, which are analogous to latitude and longitude. The subsequent step is to put the data into telemetry and display it on the ThingsBoard dashboard. This will be accomplished by first putting the REST API call into execution, then obtaining the weather data, processing the data, filtering the data we require, and finally putting the data into telemetry.

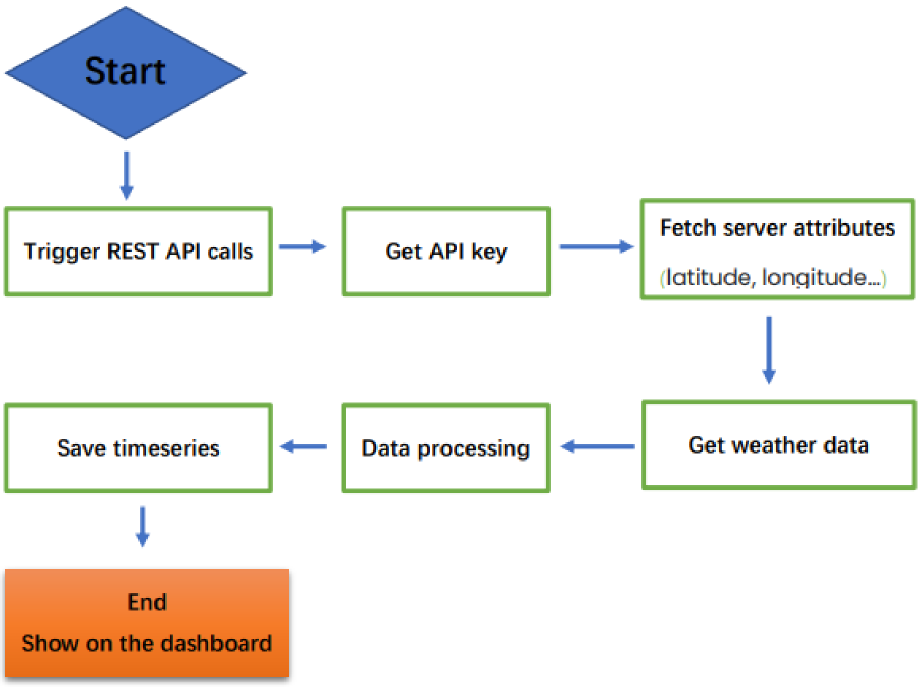


Figure 4-2: Procedure for the receipt of data

4.3 TECHNOLOGY USED

This project is an experiment that is based on the Internet of Things. It involves developing a real-time weather observation system that is based on the Raspberry Pi4 microcontroller, acquiring data about the weather by accessing the OpenweatherMap REST API interface, and visualising the data on the ThingsBoard platform. The Internet of Things, the intelligent open-source hardware Raspberry Pi4, the Internet of Things platform ThingsBoard, and the OpenweatherMap website are some of the technologies that are utilised.

4.3.1 Description of Raspberry Pi 4

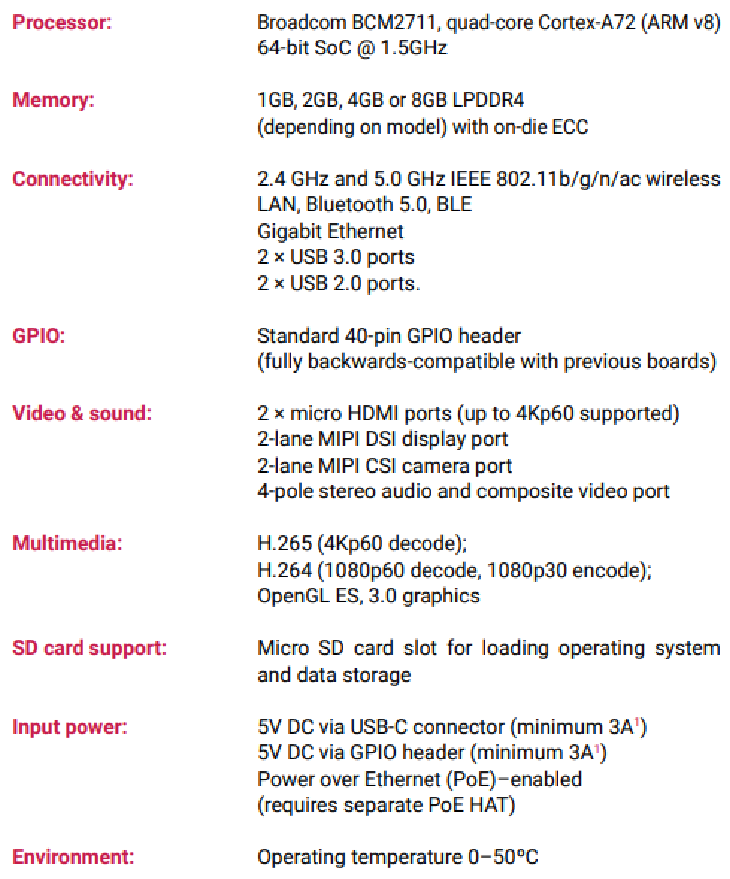
The past few years have witnessed a rapid development of open-source platforms and software, as well as the rapid emergence of various open-source hardware products, such as the Arduino microcontroller, Raspberry Pi, ESP8266, ODROID, and so on. This trend is expected to continue in the foreseeable future. Low prices, open infrastructure, and open-source software tools make it possible to quickly meet the requirements of the IoT and put it into practise.

This endeavour makes use of low-cost open-source hardware in the form of a Raspberry Pi4, the most recent computer board released by the Raspberry Pi Foundation. It features the most powerful board that has been made available to date and can function as a full desktop computer. The board has a fast quad-core ARM Cortex-A72 processor, Gigabit Ethernet, dual-band 802.11ac Wi-Fi, Bluetooth 5.0, two USB 3.0 ports, two USB 2.0 ports, and a 40-pin GPIO header. It is approximately the size of a credit card. Raspberry Pi OS is a variant of Linux that is based on the Debian operating system, and it is the official operating system for the Raspberry Pi4 computer.

The Raspberry Pi 4 comes equipped with a quad-core ARM Cortex-A72 processor that operates at a frequency of 1.5 GHz. When compared to the previous generation Raspberry Pi 3, which included a quad-core Cortex-A53 processor operating at 1.2GHz, this is a huge improvement. Because of its faster clock speed and more powerful processor, the Raspberry Pi 4 can now be used as a desktop computer to perform tasks such as web browsing, office application use, and even light video and image editing. It also has the most flexible open-source hardware on the market right now. This is because both its hardware and software have been improved.

The addition of dual-band 802.11ac Wi-Fi is one of the most significant enhancements that have been made to the Raspberry Pi 4 platform. Its network capability supports Ethernet ports and Wi-Fi for gigabit throughput while still being lightweight and durable at 46 g. This implies that the board can now connect to the 5GHz band in addition to the 2.4GHz band. A graphics processing unit (GPU) with a speed of 500 MHz and two micro HDMI ports are included in the device. Because of this, it can operate on televisions that have a resolution of up to 4K. The Raspberry Pi has established itself as a frontrunner in the competition to make widespread use of the IoT thanks to the special architectural qualities that it possesses. The system is compatible with a diverse selection of programming languages and provides access to more than 350,000 open-source library collections. Python, C, and Java are just a few examples of these languages. When it comes to the creation of apps for the internet of things, this makes possible a high level of convenience.

The Raspberry Pi computer is utilised in a wide number of IoT applications, some examples of which include real-time weather detection systems, smart furniture systems, soil moisture monitoring, and a great deal of other similar applications. In addition to this, it is capable of interacting with the wider world, and it has been implemented in a broad variety of digital maker projects. These projects range from music machines and parent detectors to weather stations and tweeting birdhouses with infrared cameras. In this aspect, the Raspberry Pi has a considerable advantage over hardware processors since hardware processors are unable to handle an exceptionally large volume of data. On the other hand, it can process an extraordinarily large volume of data.



Processor:	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memory:	1GB, 2GB, 4GB or 8GB LPDDR4 (depending on model) with on-die ECC
Connectivity:	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 2.0 ports.
GPIO:	Standard 40-pin GPIO header (fully backwards-compatible with previous boards)
Video & sound:	2 × micro HDMI ports (up to 4Kp60 supported) 2-lane MIPI DSI display port 2-lane MIPI CSI camera port 4-pole stereo audio and composite video port

Multimedia:	H.265 (4Kp60 decode);
	H.264 (1080p60 decode, 1080p30 encode);
	OpenGL ES, 3.0 graphics

SD card support:	Micro SD card slot for loading operating system and data storage
Input power:	5V DC via USB-C connector (minimum 3A[1]) 5V DC via GPIO header (minimum 3A[1]) Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment:	Operating temperature 0–50ºC

Figure 4-3: Raspberry Pi 4 Specification

4.3.2 Description of ThingsBoard

ThingsBoard is an open-source Internet of Things platform that is built on the Java programming language and enables rapid development, management, and scalability of Internet of Things projects. It is a server-side architecture that is used by a variety of Internet of Things applications. It enables device communication by utilising the industry-standard protocols for the Internet of Things (MQTT, CoAP, and HTTP). The capacity to store data permanently and lessen the risk of losing data are two benefits that come with scalability, high performance, and the capability to handle mistakes. Because the platform was developed using an extensible microservices architecture, it is able to expand both laterally and vertically by adding new services and more instances of current services, respectively.

ThingsBoard is available in both a open-source community and a paid professional edition. We are use the open-source community version of ThingsBoard for this project.

It is distinguished by the following important qualities:

- ✧ Device management
- ✧ Data collection
- ✧ Data visualization
- ✧ Rule engine
- ✧ Alarm management
- ✧ Integration with popular IoT protocols
- ✧ Multi-tenancy and role-based access control
- ✧ Flexible data model
- ✧ Support for a versatile data format - Compatible with the REST API, MQTT, CoAP, WebSocket, SQL, and NoSQL protocols

- Clustering
- Horizontal scalability
- High availability

The data that ThingsBoard gets is classified as telemetry data and is saved in a database. The database used to save the data must be a SQL database. PostgreSQL, which is installed on the Raspberry Pi along with the ThingsBoard IoT server platform, is the database management system that is recommended by Raspberry Pi's official website as the best option for storing data for future use.[11] We are able to see historical data on the server platform, as well as plot it as a trend graph, thanks to the assistance of the database. This makes it easier for us to conduct analysis and make decisions based on centralised data trends.ThingsBoard is a platform that is suited for managing Internet of Things devices, collecting data from those devices, and visualising that data. The rule engine of the platform makes it possible to construct rules that, when certain circumstances are satisfied, cause alarms to sound or for specific actions to be carried out.

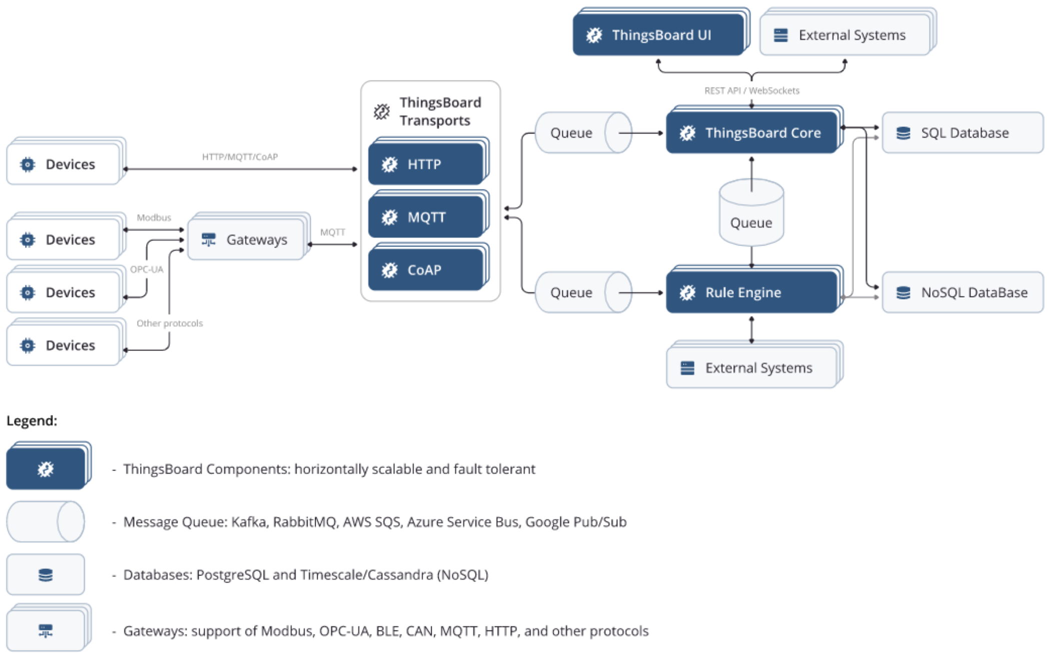

ThingsBoard UI
External Systems
REST API / WebSockets
ThingsBoard Transports
HTTP
MQTT
CoAP
Devices
HTTP/MQTT/CoAP
Devices
Modbus
Gateways
MQTT
OPC-UA
Devices
Other protocols
Devices
Queue
ThingsBoard Core
SQL Database
Queue
Queue
Rule Engine
NoSQL DataBase
External Systems
Legend:
- ThingsBoard Components: horizontally scalable and fault tolerant
- Message Queue: Kafka, RabbitMQ, AWS SQS, Azure Service Bus, Google Pub/Sub
- Databases: PostgreSQL and Timescale/Cassandra (NoSQL)
- Gateways: support of Modbus, OPC-UA, BLE, CAN, MQTT, HTTP, and other protocols

Figure 4-4: ThingsBoard data transfer structure

ThingsBoard is designed to run on a wide variety of hardware, and the company's website offers step-by-step installation guides for a variety of operating systems, including Ubuntu, Docker on Windows, Docker on Linux/Mac OS, Windows, Raspberry Pi 4, and more. These guides explain in detail how to set up the software on a computer.

4.3.3 Description of OpenweatherMap

OpenweatherMap is a free online weather data service that was established in 2005 by two Russian brothers named Evgeny and Alexander Tkachenko. The service is targeted for service and mobile application developers. It makes global weather data, such as current weather data, forecasts, and neighbourhood forecasts for any geographical location, available for over 200,000 locations around the world through an API interface from a variety of sources, such as weather stations, airports, and satellites, and then processes and packages the data into a variety of products, such as maps, text forecasts, and weather data layers. In addition, it provides historical weather data for any geographical location, which can be accessed through an API interface. Providing access to this data for developers so they can use it in their own apps. Weather data is sourced from the Global Weather Radio Service and over 40,000 weather stations.[7] OpenweatherMap is continually broadening the range of data it provides. This service now provides information on UV indices, pollen levels, and levels of air pollution. In addition to this, it is attempting to incorporate new data sources, such as satellite data.

Because a REST API serves as the interface for data interaction in OpenweatherMap, programmers are able to retrieve data from OpenWeatherMap's servers by making standard HTTP queries. The REST API is defined as an architectural tool designed on top of web services and focused on system resources.[12] The acronym "REST" refers to a type of software architecture known as Resource Representation State Transfer. This architecture style is characterised by a collection of architectural restrictions and principles. Create, read, update, and delete are the four operations that are supported by REST thanks to its implementation of the HTTP protocol and the use of uniform resource identifiers.

[7] https://openweathermap.org

There are six advantages of REST.

- ✧ **Client-Server separation:** Enabling the improvement of individual components while boosting the interface's usability.
- ✧ **Stateless:** Each request submitted by the client must contain all the information required by the server; this allows each request to be evaluated independently, making it easier to fix errors and conserving server resources.
- ✧ **Cachable:** If the information given by the server is designated as cached, the client can reuse it to send the request, thereby minimising the number of interactions and latency.
- ✧ **Layered System:** Components of the system are solely responsible for communication, hence reducing system complexity and enhancing scalability.
- ✧ **Uniform Interface:** Enhances the visibility of interactions and can improve individual components.
- ✧ **Code-On-Demand:** Highly scalable, rovide the client with the opportunity to use programming languages distinct from those used by the server.

Application Programming Interface is the full name of the API, which is a collection of subroutine definitions, protocols, and tools that are used in the process of constructing application software. In a nutshell, it is a collection of clearly defined methods of communication between the various software components. Authorization and access control are built into the API itself, ensuring that only those who are authorised to do so can gain access to particular data, thereby maintaining the data's confidentiality.

REST application programming interfaces are quite common in today's computer networks. These APIs typically communicate data over HTTP using JSON or XML. This makes interacting with REST APIs easier, as they are lightweight and easily readable, which makes them suitable for use in mobile applications due to their suitability in this context. The fact that it is completely free to use, that there is no limit to the number of API calls that can be made, and that there is no limit to the amount of data that can be obtained are three of its primary features. As a result of this, the API is perfect for developers who want to design applications connected to the weather but do not want to pay for access to the data. Last but not least, the OpenweatherMap API is dependable and is supported by a group of meteorologists. These meteorologists make certain that the data is correct and up to date, and they update the API on a consistent basis with new features and data sources.

For the purpose of this project, we will make use of the REST API interface that is made available by OpenweatherMap in order to collect current weather information and plot it on a ThingsBoard so that it may be viewed. Minute Forecast 1 Hour, Hourly Forecast 2 Days, Daily Forecast 7 Days, National Weather Alerts, Historical Weather 5 days, 60 calls/minute, 1,000,000 calls/month; and if you are a student or educator, you will get a higher level of service: 3,000 API calls per minute and 100,000,000 API calls per month, Historical Weather API, for 1 year of prior data. Minute Forecast 1 Hour, Hourly Forecast 2 Days, Daily Forecast 7 Days, National Weather Alerts, Historical Weather 5 days, 60 calls/minute.

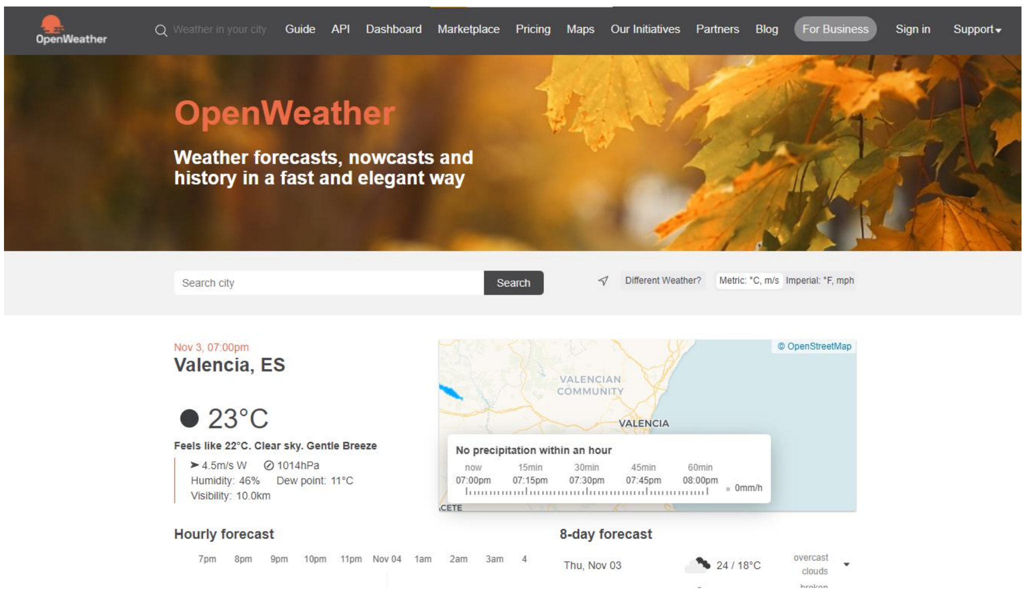


Figure 4-5: OpenweatherMap Official website

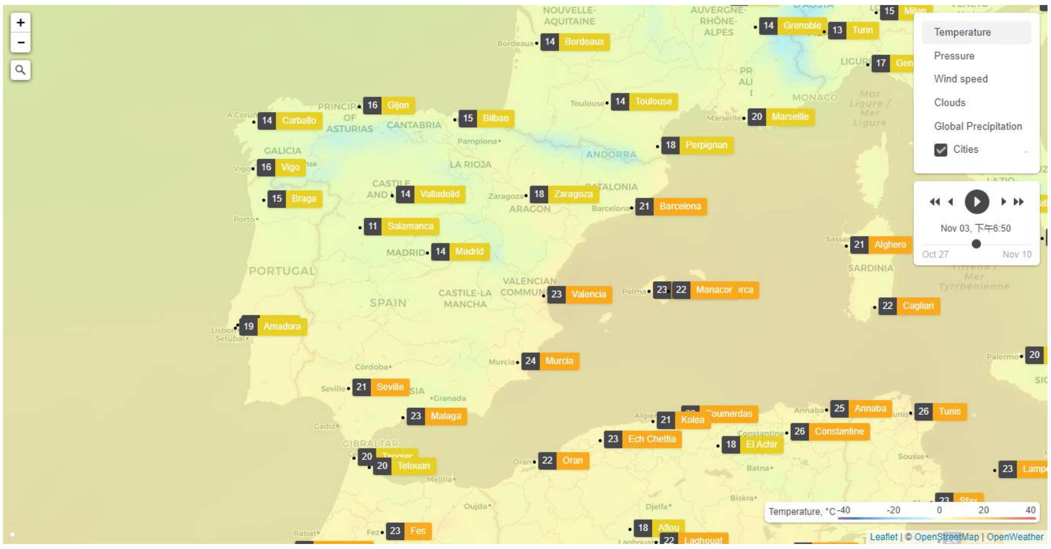


Figure 4-6: Interactive weathers maps

5. IMPLEMENTATION

In this undertaking, we will be constructing a simple machine by making use of a Raspberry Pi. The components of the hardware consist of an HDMI cable, a memory card, a board, and a casing for the Raspberry Pi4 development board kit. Additionally, there is a display with an HDMI input, a mouse, and a keyboard.

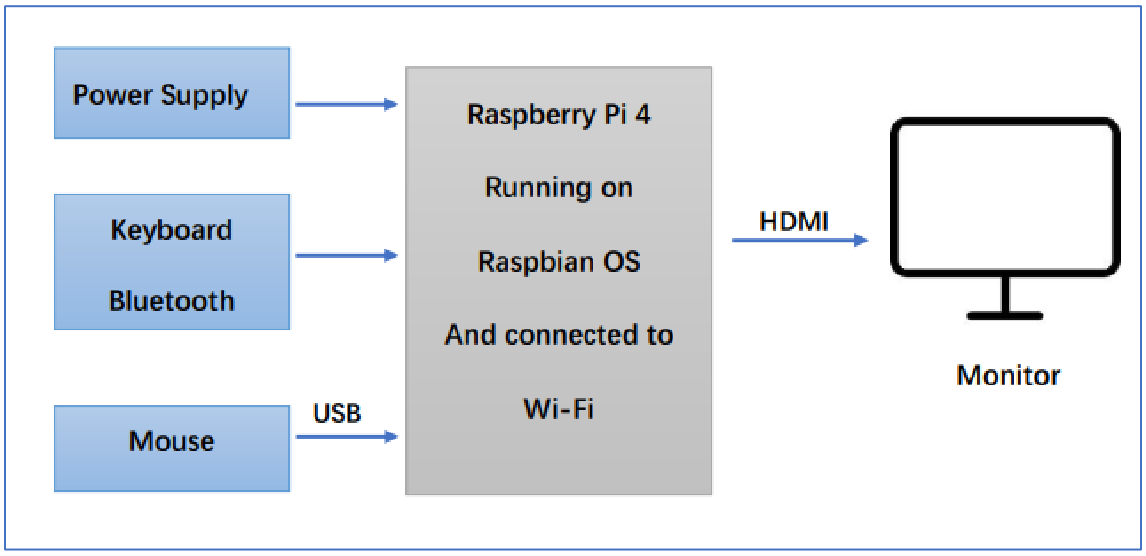


Figure 5-1: Project connection architecture

Installing the Raspberry Pi, then installing the ThingsBoard service on the Raspberry Pi, and then connecting the Raspberry Pi to the OpenweatherMap REST API through the ThingsBoard in order to collect data and make observations are the processes involved. These instructions may be found on the Raspberry website and the ThingsBoard website.

Thingsboard installation will be the first step. On the official website, you may find a reference to the Raspberry Pi installation process. It will not be introduced in this project.

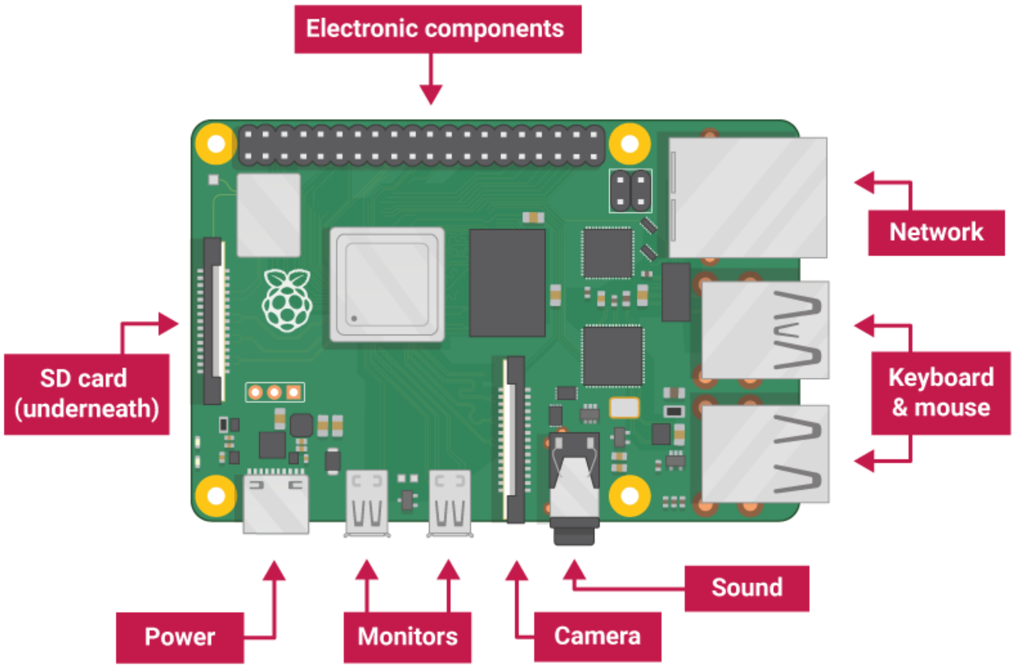


Figure 5-2: Raspberry Pi 4 hardware introduction

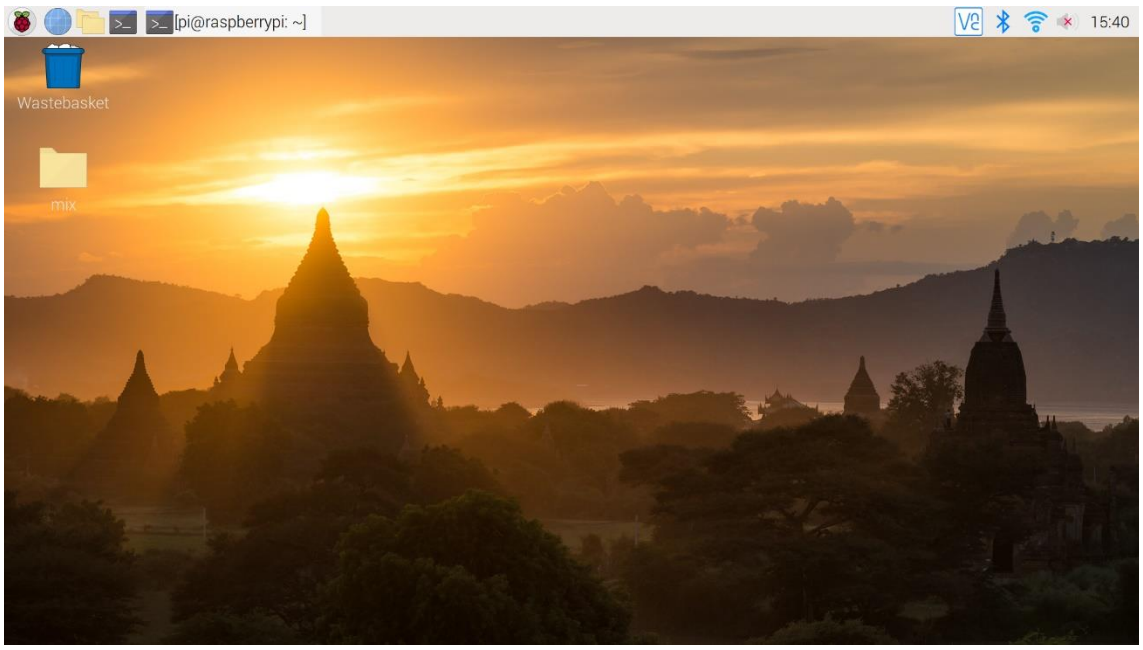


Figure 5-3: Raspberry Pi desktop

5.1 Installing ThingsBoard on Raspberry Pi 4

The installation of ThingsBoard will take place once we have completed the configuration of our Raspberry Pi 4 system.

Because the ThingsBoard service is built on Java 11, the Java 11 development kit OpenJDK must be installed:

```
sudo apt update
sudo apt install openjdk-11-jdk
```

With the help of the following command, the operating system will be set up to make use of OpenJDK11 by default.

```
sudo update-alternatives --config java
```

To ensure that the installation went smoothly.

```
java -version
```

Expected command output.

```
openjdk version "1.8.0_xxx"
OpenJDK Runtime Environment (...)
OpenJDK 64-Bit Server VM (build ...)
```

It is a good idea to include a command output photo to check the installed java version.

Before we begin installing the ThingsBoard service, we must first download the installation package from the website:

```
wget
https://github.com/thingsboard/thingsboard/releases/download/
v3.3.4/thingsboard-3.3.4.deb
```

Install the ThingsBoard server:

```
sudo dpkg -i thingsboard-3.3.4.deb
```

In order to save the data, we will need to first install PostgreSQL and configure the

ThingsBoard database:

```
# install **wget** if not already installed:
sudo apt install -y wget

# import the repository signing key:
wget --quiet -O -
https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-
key add -

# add repository contents to your system:
RELEASE=$(lsb_release -cs)
echo "deb http://apt.postgresql.org/pub/repos/apt/
${RELEASE}"-pgdg main | sudo tee
/etc/apt/sources.list.d/pgdg.list

# install and launch the postgresql service:
sudo apt update
sudo apt -y install postgresql-12
sudo service postgresql start
```

Create a new account or set a password for the primary user after PostgreSQL has been installed. Note: the PostgreSQL password is necessary, otherwise we will not be able to finish configuring the ThingsBoard file:

```
sudo su - postgres
psql
\password
\q
```

Once set up "Ctrl+D" to return to the main user console, connect to the database and create the ThingsBoard Database:

```
psql -U postgres -d postgres -h 127.0.0.1 -W
CREATE DATABASE ThingsBoard;
\q
```

In order to connect to the Postgres database, it is necessary to change the ThingsBoard configuration file:

```
sudo nano /etc/ThingsBoard/conf/ThingsBoard.conf
```

In the configuration file, add the following line and make sure that "P PUT_YOUR_POSTGRESQL_PASSWORD_HERE " is replaced with the actual password for the postgres user:

```
# DB Configuration
export DATABASE_ENTITIES_TYPE=sql
export DATABASE_TS_TYPE=sql
export
SPRING_JPA_DATABASE_PLATFORM=org.hibernate.dialect.PostgreSQL
Dialect
export SPRING_DRIVER_CLASS_NAME=org.postgresql.Driver
export
SPRING_DATASOURCE_URL=jdbc:postgresql://localhost:5432/things
board
export SPRING_DATASOURCE_USERNAME=postgres
export
SPRING_DATASOURCE_PASSWORD=PUT_YOUR_POSTGRESQL_PASSWORD_HERE
export SPRING_DATASOURCE_MAXIMUM_POOL_SIZE=5
# Specify partitioning size for timestamp key-value storage.
Allowed values: DAYS, MONTHS, YEARS, INDEFINITE.
export SQL_POSTGRES_TS_KV_PARTITIONING=MONTHS
```

ThingsBoard uses the queue service for API calls between microservices and can use the next queue service; however, by default, we select In Memory, and there are no additional configuration steps that are required to set it up:

Memory update for slow machines (1GB of RAM)

Repeating the previous step, edit the ThingsBoard configuration file and add the following line to the file:

```
sudo nano /etc/thingsboard/conf/thingsboard.conf
# Update ThingsBoard memory usage and restrict it to 256MB in
/etc/thingsboard/conf/thingsboard.conf
export JAVA_OPTS="$JAVA_OPTS -Xms256M -Xmx256M"
```

After installing the ThingsBoard service, you must then execute the following script and make any necessary updates to the database configuration:

```
# --loadDemo option will load demo data: users, devices,
assets, rules, widgets.
```

```
sudo /usr/share/thingsboard/bin/install/install.sh --loadDemo
```

It is now possible to begin using the ThingsBoard service:

```
sudo service thingsboard start
```

When the service is running, the web user interface can be accessed over the following connection:

```
http://localhost:8080/
```

Check to see that localhost is listed as the host number or whatever it is.

We should be able to log in using the following user because the -loadDemo option was supplied when the installation script was run:

System Administrator: sysadmin@ThingsBoard.org/ sysadmin

Tenant Administrator: tenant@ThingsBoard.org/ tenant

Customer User: customer@ThingsBoard.org/ customer

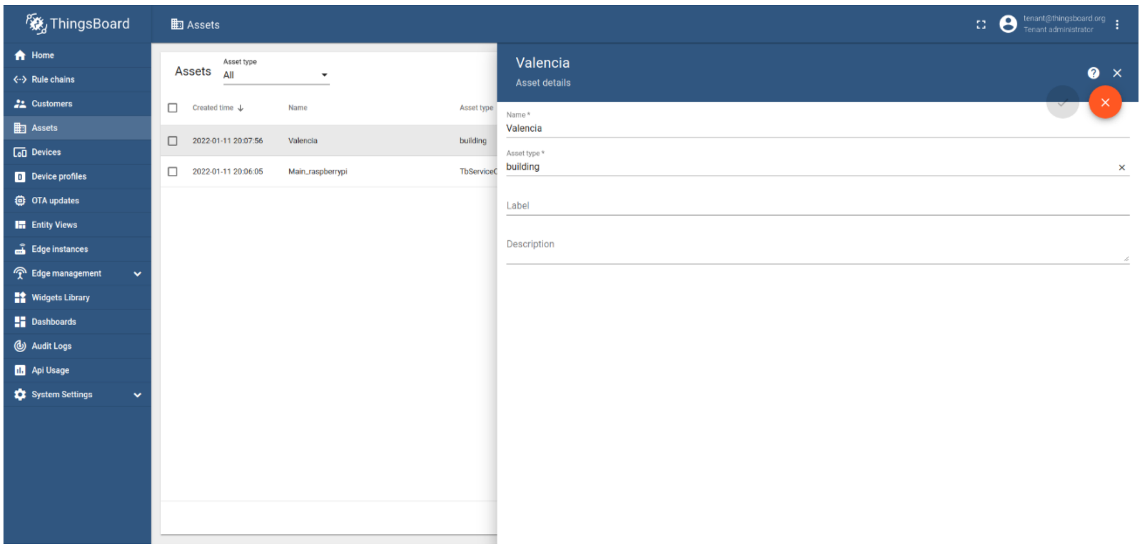
It may take up to 240 seconds for the Web UI to begin functioning, after which the user can access the account profile page to modify their password.

5.2 Receiving OpenweatherMap data via Thingsboard

Following the successful completion of the Raspberry Pi4 installation of the ThingsBoard server, a call is made to the OpenweatherMap REST API in order to read weather data. This demonstration monitors real-time weather conditions and displays them on the ThingsBoard dashboard with Valencia serving as the coordinate point.

● Configuring fundamental attributes

We need to add an Asset entity to the ThingsBoard and give it the name Valencia. The type should be set to Building.



Assets — Asset type: All

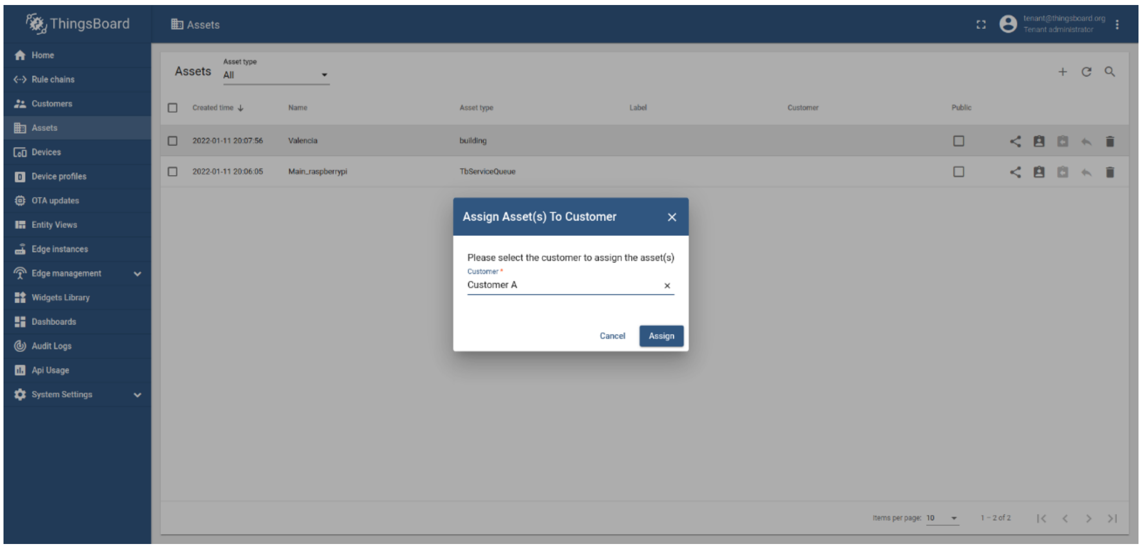
Created time ↓	Name	Asset type
2022-01-11 20:07:56	Valencia	building
2022-01-11 20:06:05	Main_raspberrypi	TbServiceC

Valencia
Asset details

Name *: Valencia
Asset type *: building
Label:
Description:

Figure 5-4: Add an asset

➢ **Assign the Asset to the customer**: Go to Assets->Assign to customer->Customer A->Assign.



Created time	Name	Asset type	Label	Customer	Public
2022-01-11 20:07:56	Valencia	building			☐
2022-01-11 20:06:05	Main_raspberrypi	TbServiceQueue			☐

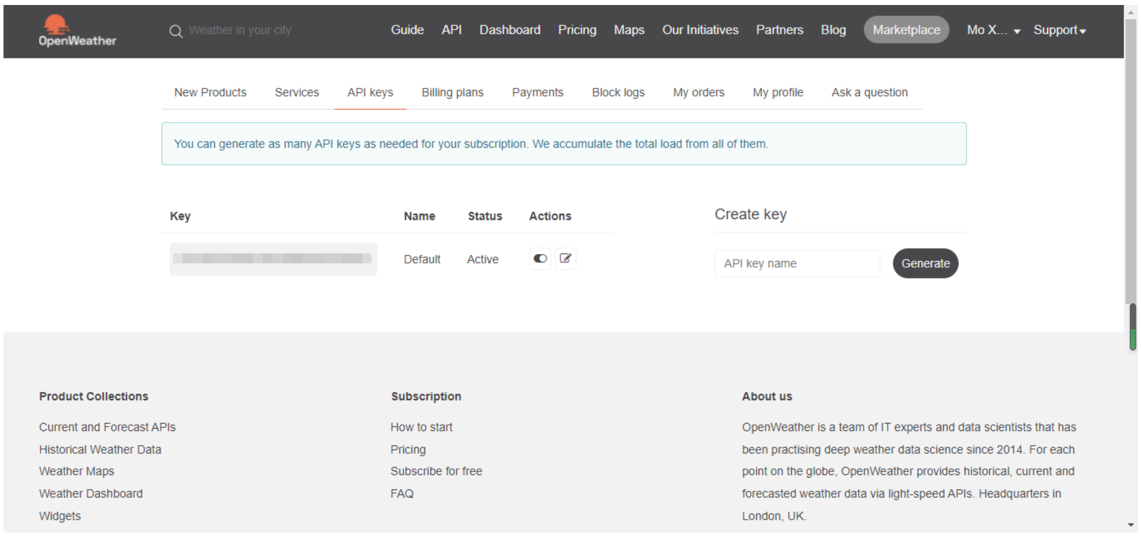
Assign Asset(s) To Customer

Please select the customer to assign the asset(s)

Customer *: Customer A

Figure 5-5: Assign the Asset to the customer

➢ **Register on the OpenweatherMap website**: Once registration is finished, obtain the API key and enable it. This is to be put to the customer server-side settings in order for the customer to receive data:



You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.

Key	Name	Status	Actions
[redacted]	Default	Active	

Product Collections
Current and Forecast APIs
Historical Weather Data
Weather Maps
Weather Dashboard
Widgets

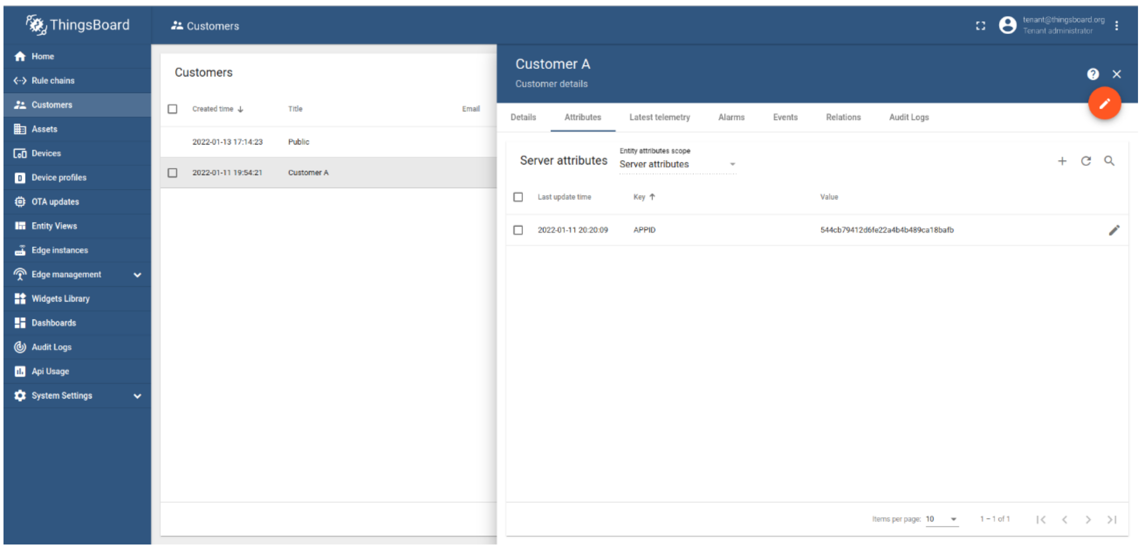
Subscription
How to start
Pricing
Subscribe for free
FAQ

About us
OpenWeather is a team of IT experts and data scientists that has been practising deep weather data science since 2014. For each point on the globe, OpenWeather provides historical, current and forecasted weather data via light-speed APIs. Headquarters in London, UK.

Figure 5-6: Register on the OpenweatherMap website

- **Create customer attributes**: Assigned customer->Attributes ->Add

  Carry out the REST API call with the following parameters included in the URL: the API key, the longitude, the latitude, and the measurement units. Add the API key to the server-side attributes that are associated with the Assigned customer, and add the remainder of the parameters to the Asset server-side attributes.

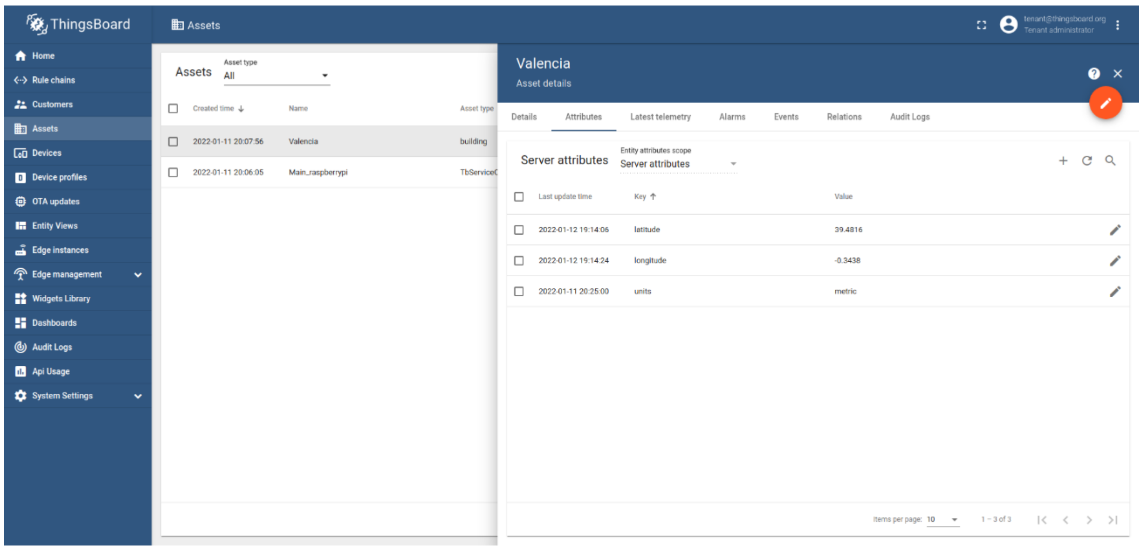

Customers — Customer A — Customer details — Server attributes — Entity attributes scope: Server attributes

Created time	Title	Email
2022-01-13 17:14:23	Public	
2022-01-11 19:54:21	Customer A	

Last update time	Key	Value
2022-01-11 20:20:09	APPID	544cb79412d6fe22a4b4b489ca18bafb

Figure 5-7: Create customer attributes

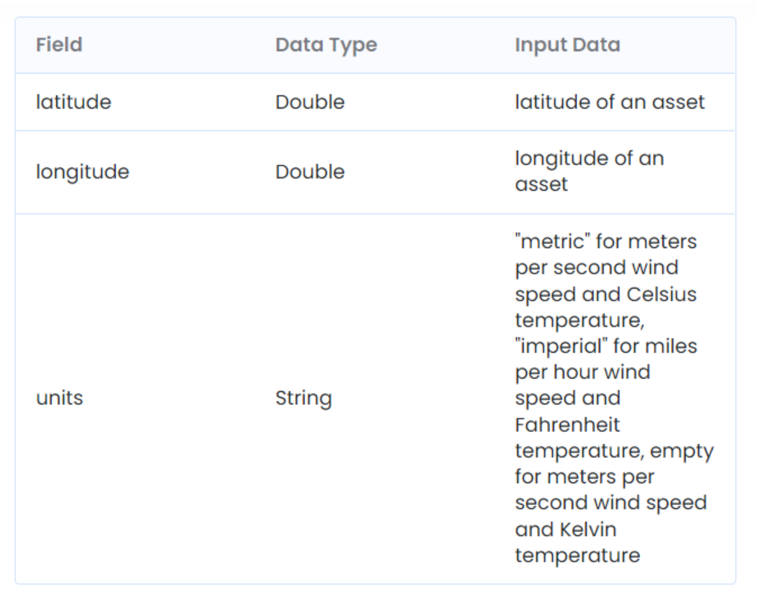
- **Asset attributes**: Building A->Attributes->Add, use Valencia's coordinate system and metric measurement units for this project.


Assets — Asset type: All — Valencia — Asset details — Server attributes — Entity attributes scope: Server attributes

Created time	Name	Asset type
2022-01-11 20:07:56	Valencia	building
2022-01-11 20:06:05	Main_raspberrypi	TbServiceC

Last update time	Key	Value
2022-01-12 19:14:06	latitude	39.4816
2022-01-12 19:14:24	longitude	-0.3438
2022-01-11 20:25:00	units	metric

Figure 5-8: Asset attributes



Field	Data Type	Input Data
latitude	Double	latitude of an asset
longitude	Double	longitude of an asset
units	String	"metric" for meters per second wind speed and Celsius temperature, "imperial" for miles per hour wind speed and Fahrenheit temperature, empty for meters per second wind speed and Kelvin temperature

Figure 5-9: Asset attributes details

● Developing a rule chain

Message flow includes a rule chain whose aim is to send API calls to OpenweatherMap at regular intervals of 15 seconds and to deliver the Asset the pertinent temperature and humidity data. The function of each node is broken down in greater detail in the following section, and the figure demonstrates how the rule chain is supposed to look and work.

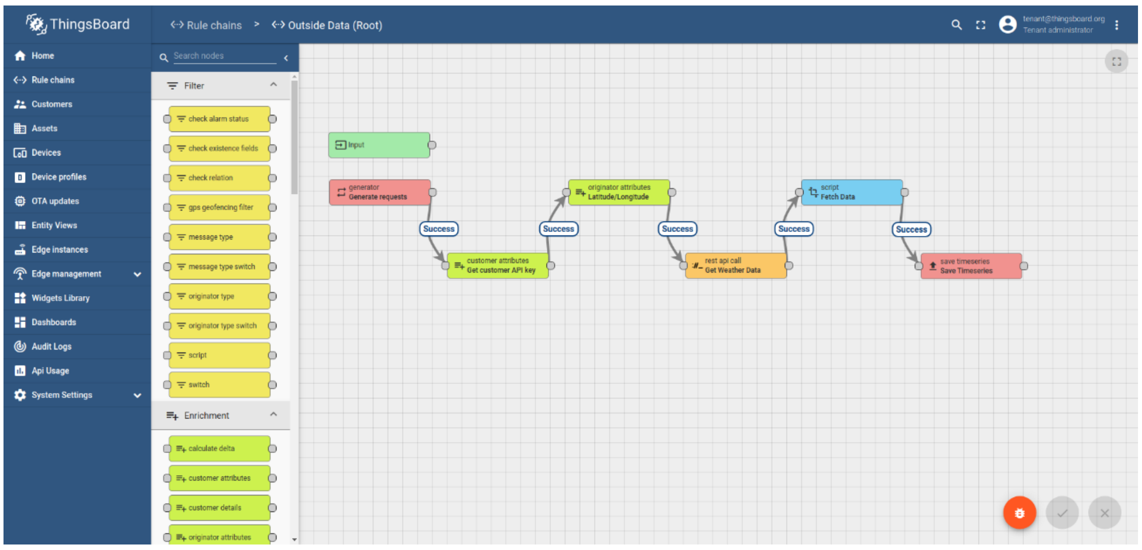
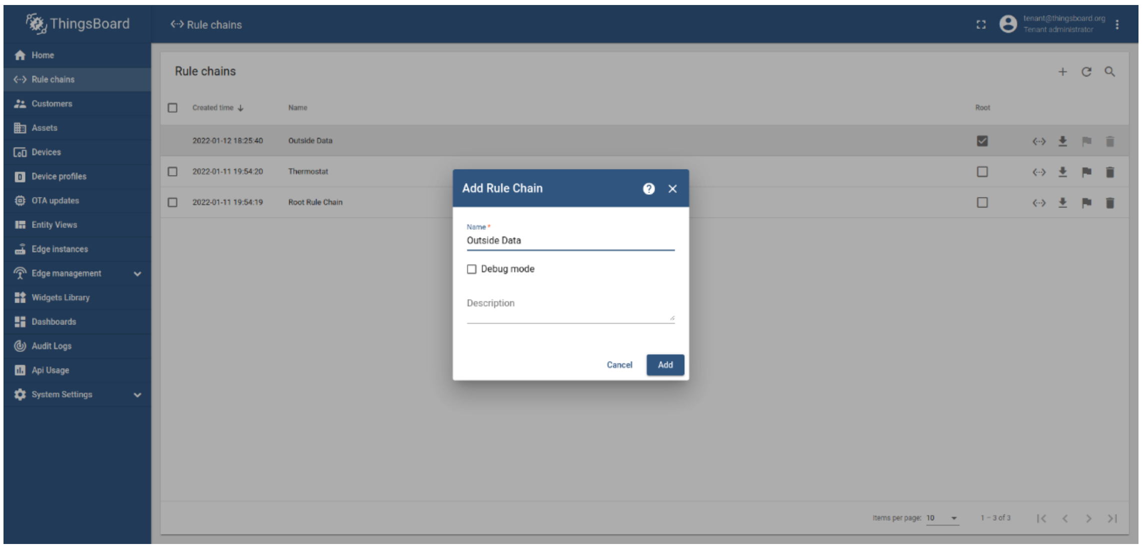


Figure 5-10: Rule chain

➢ **Create a new Rule Chain (Outside Data)**: Rule Chains->Add new Rule Chain

Configure the Name as “Outside Data”.



Rule chains

Created time	Name	Root
2022-01-12 18:25:40	Outside Data	✓
2022-01-11 19:54:20	Thermostat	☐
2022-01-11 19:54:19	Root Rule Chain	☐

Add Rule Chain

Name *: Outside Data

☐ Debug mode

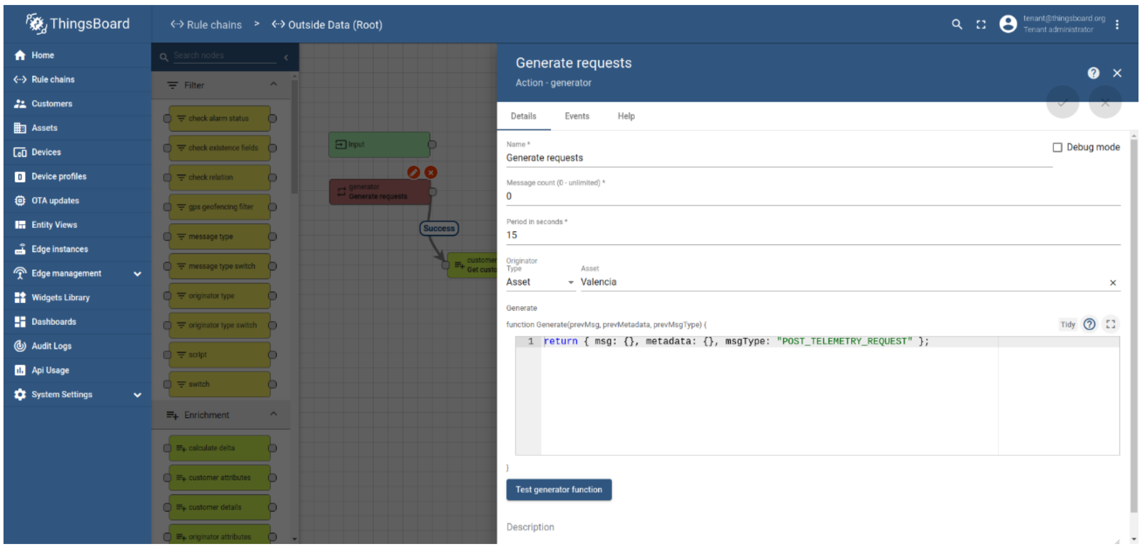
Description:

Cancel Add

Figure 5-11: Create a new Rule Chain

Click the "Edit" button once the new rule chain has been successfully generated, and then proceed to configure the rule chain. Within this rule chain, there are a total of six nodes that need to be created. These nodes are referred to as the Generator node, the Customer attributes enrichment node, the Originator attributes enrichment node, the External REST API call node, the Script transformation node, and the Save time-series node.

- **Generator node:** produces an empty message in order to initiate a call to a REST API, populating the fields with the following information.



Generate requests
Action - generator

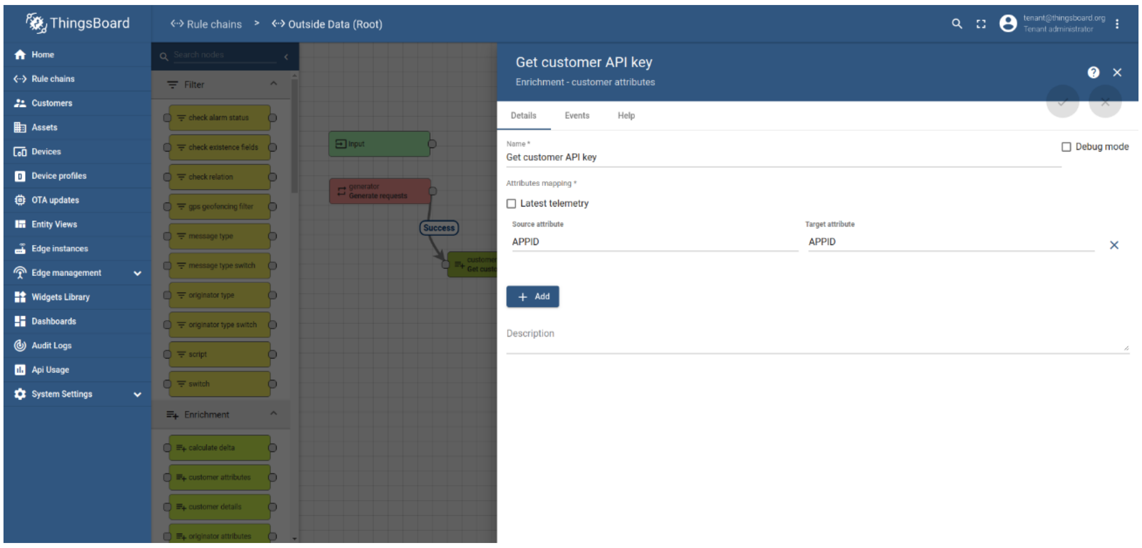
Name *: Generate requests
☐ Debug mode
Message count (0 - unlimited) *: 0
Period in seconds *: 15
Originator Type: Asset
Asset: Valencia
Generate

```
function Generate(prevMsg, prevMetadata, prevMsgType) {
return { msg: {}, metadata: {}, msgType: "POST_TELEMETRY_REQUEST" };
}
```

Description

Figure 5-12: Generator node

- **Customer attributes enrichment node:** It is the responsibility of this node to include the APPID of the customer characteristics within the metadata of the message and to establish a connection between it and the Generator node using a relation type of Success.



Get customer API key
Enrichment - customer attributes

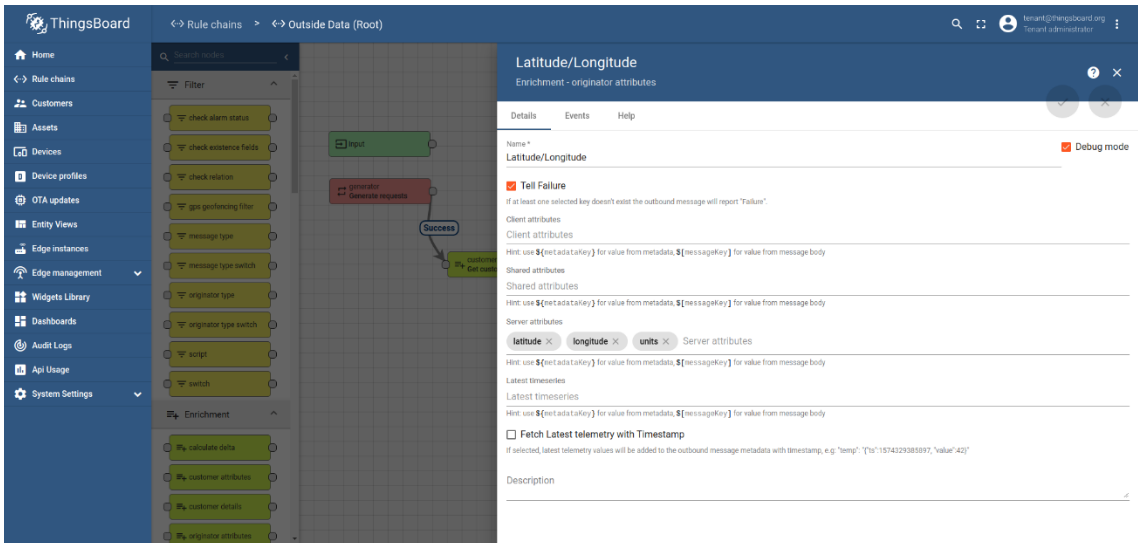
Name *: Get customer API key
☐ Debug mode
Attributes mapping *
☐ Latest telemetry

Source attribute	Target attribute
APPID	APPID

Description

Figure 5-13: Customer attributes enrichment node

➢ **Originator attributes enrichment node:** Connect this node to the Customer attributes node using a relation type of Success. This node will add the server attributes latitude, longitude, and units of the originator that were set in the Generator node to the metadata.



Latitude/Longitude
Enrichment - originator attributes

Name *
Latitude/Longitude

☑ Debug mode

☑ Tell Failure
If at least one selected key doesn't exist the outbound message will report "Failure".

Client attributes
Hint: use ${metadataKey} for value from metadata, $[messageKey] for value from message body

Shared attributes
Hint: use ${metadataKey} for value from metadata, $[messageKey] for value from message body

Server attributes
latitude, longitude, units
Hint: use ${metadataKey} for value from metadata, $[messageKey] for value from message body

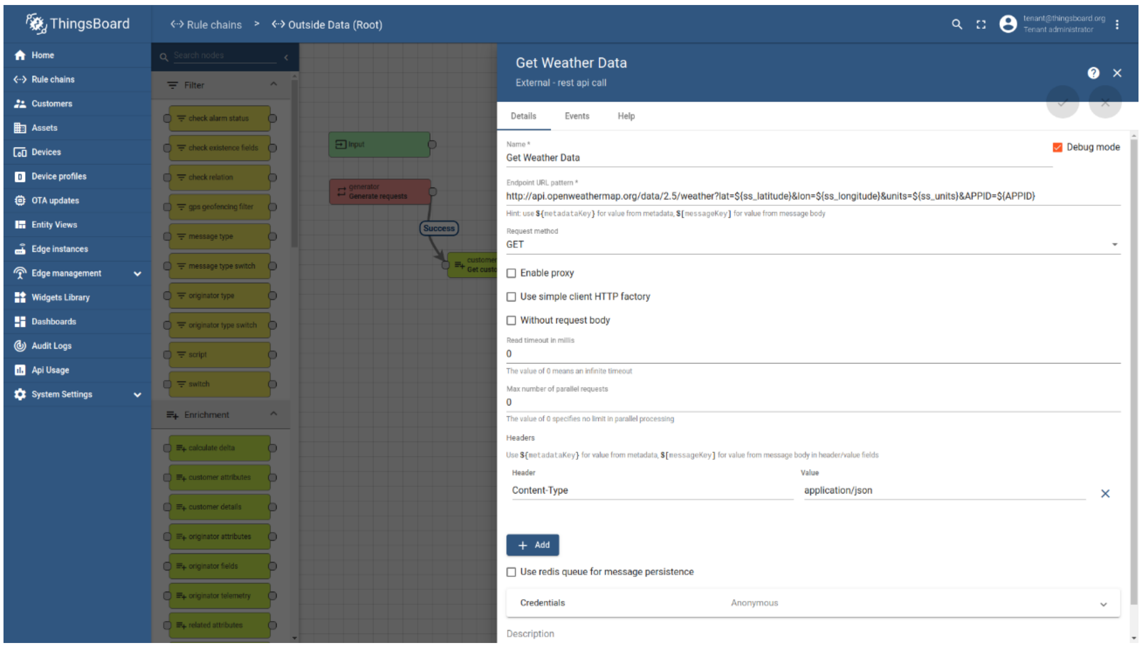
Latest timeseries
Hint: use ${metadataKey} for value from metadata, $[messageKey] for value from message body

☐ Fetch Latest telemetry with Timestamp
If selected, latest telemetry values will be added to the outbound message metadata with timestamp, e.g: "temp": "{"ts":1574329385897, "value":42}"

Description

Figure 5-14: Originator attributes enrichment node

➢ **External REST API call node:** This node will conduct a REST API call to OpenweatherMap's ss latitude, ss longitude, ss units, and ss APPID are set in the Originator attributes enrichment node with the server attributes from the metadata; connect it to the Originator attributes enrichment node with a relation type Success.



Get Weather Data
External - rest api call

Name *
Get Weather Data

[✓] Debug mode

Endpoint URL pattern *
http://api.openweathermap.org/data/2.5/weather?lat=${ss_latitude}&lon=${ss_longitude}&units=${ss_units}&APPID=${APPID}
Hint: use ${metadataKey} for value from metadata, $[messageKey] for value from message body

Request method
GET

[ ] Enable proxy
[ ] Use simple client HTTP factory
[ ] Without request body

Read timeout in millis
0
The value of 0 means an infinite timeout

Max number of parallel requests
0
The value of 0 specifies no limit in parallel processing

Headers
Use ${metadataKey} for value from metadata, $[messageKey] for value from message body in header/value fields

Header	Value
Content-Type	application/json

+ Add

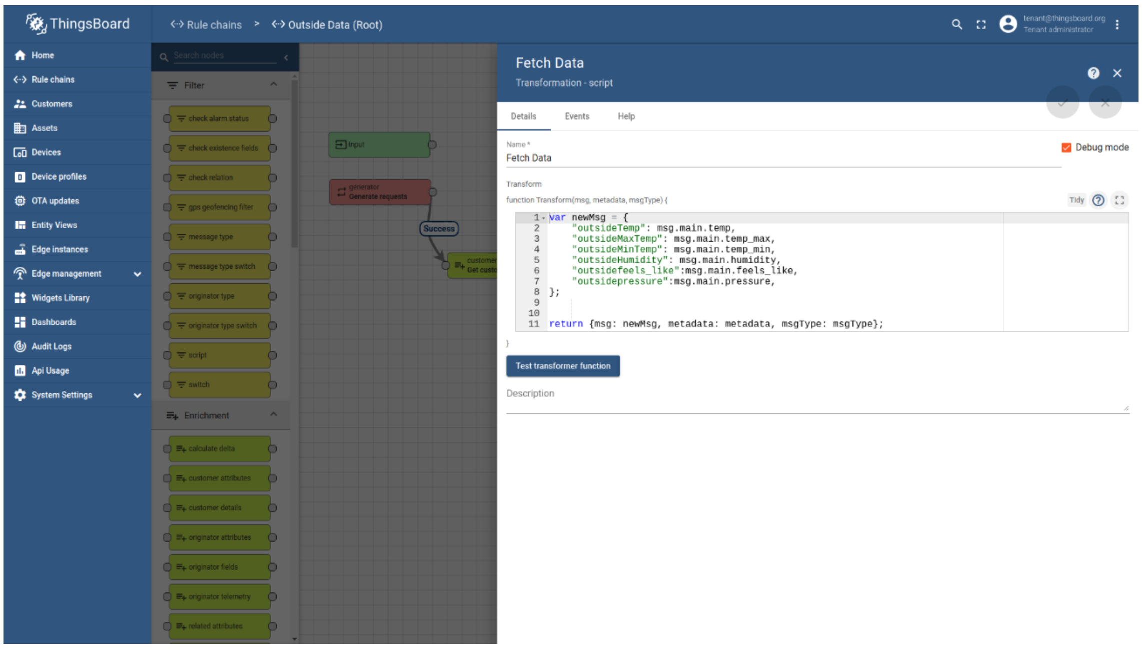
[ ] Use redis queue for message persistence

Credentials Anonymous

Description

Figure 5-15: External REST API call node

➢ **Script transformation node:** Connect this node to the External REST API call node using a relation type Success. This node will place the data that we require in the message, such as the current temperature, maximum temperature, minimum temperature, humidity, and wind speed, etc.



Fetch Data

Transformation - script

Name *
Fetch Data

Transform

function Transform(msg, metadata, msgType) {

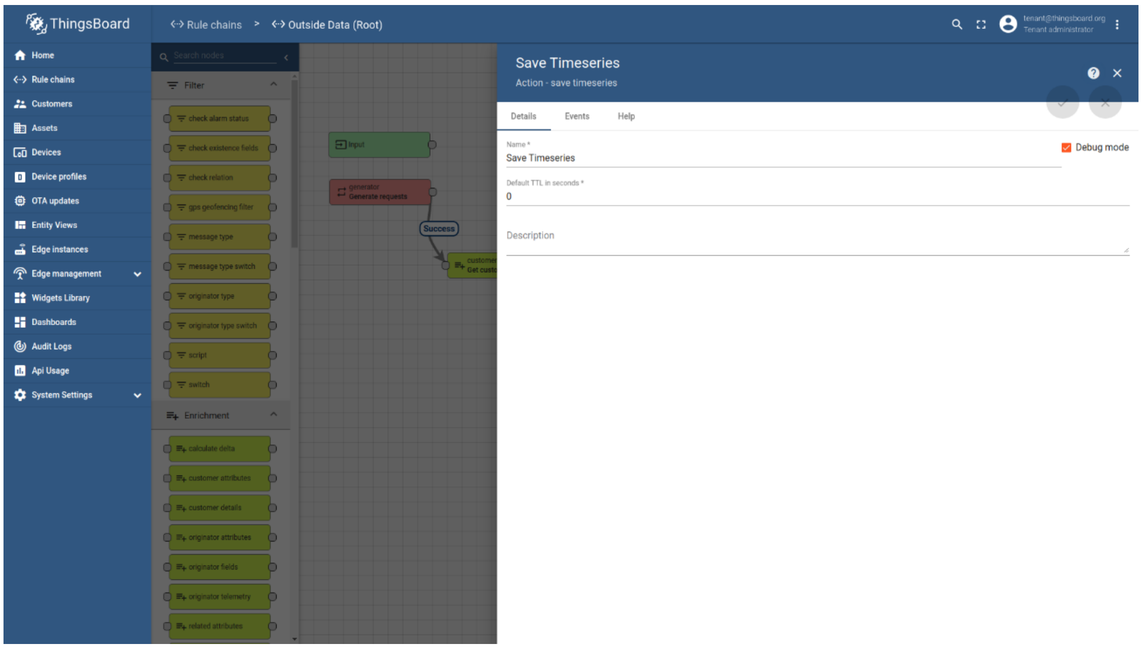
```
var newMsg = {
    "outsideTemp": msg.main.temp,
    "outsideMaxTemp": msg.main.temp_max,
    "outsideMinTemp": msg.main.temp_min,
    "outsideHumidity": msg.main.humidity,
    "outsidefeels_like":msg.main.feels_like,
    "outsidepressure":msg.main.pressure,
};


return {msg: newMsg, metadata: metadata, msgType: msgType};
```

}

Description

Figure 5-16: Script transformation node

- **Save time-series node:** Connect this node to the Script transformation node using a relation type of Success. This node will send the information into telemetry.



Save Timeseries
Action - save timeseries

Name *
Save Timeseries

Debug mode ✓

Default TTL in seconds *
0

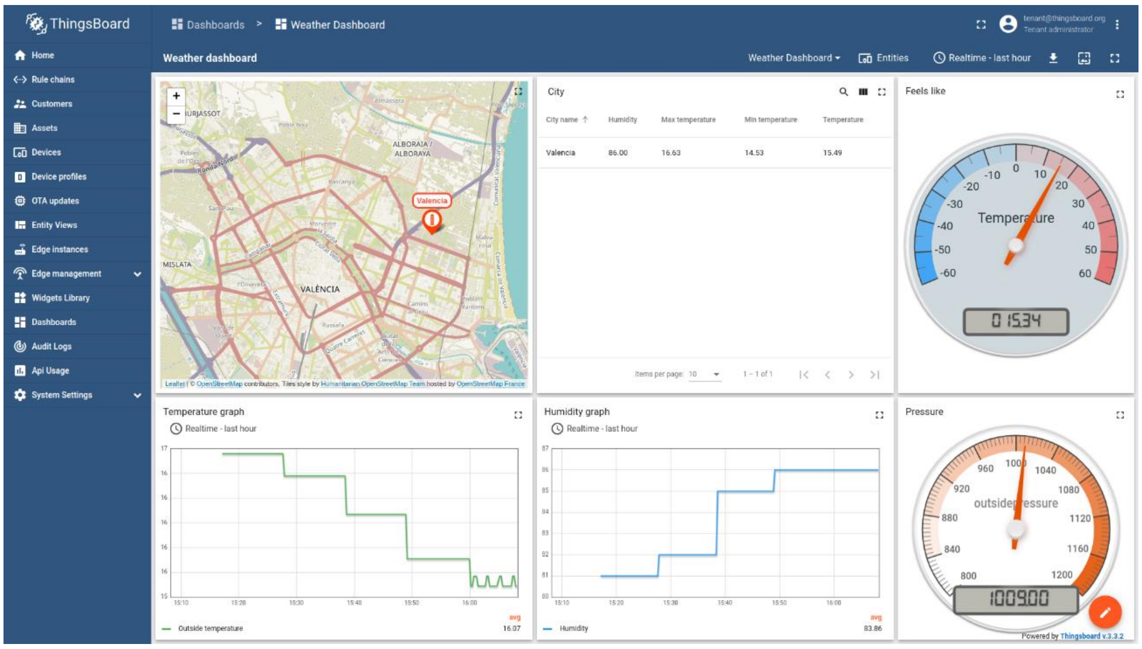
Description

Figure 5-17: Save time-series node

- **Setting up dashboard:** In order to add the data dashboard that we require, download and then import the JSON file that is provided in the ThingsBoard dashboard tutorial.

5.3 RESULT

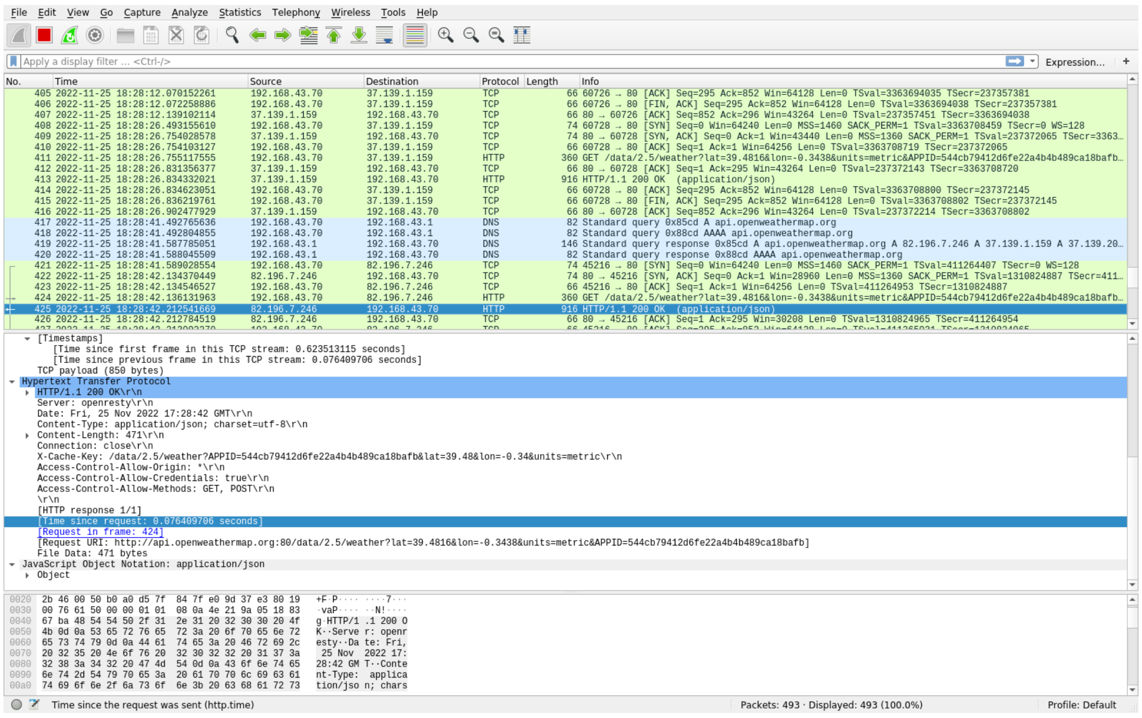
The results are presented in the following format:



City name	Humidity	Max temperature	Min temperature	Temperature
Valencia	86.00	16.63	14.53	15.49

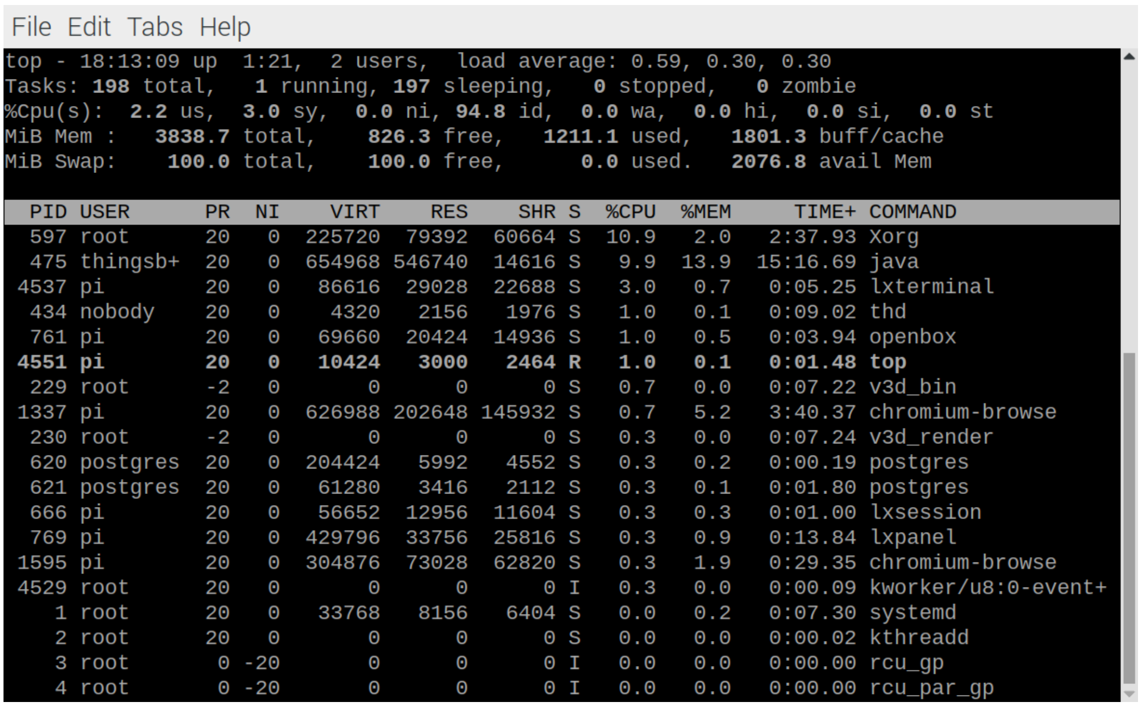
Figure 5-18: Dashboard result

This project uses the network packet capture tool Wireshark to track the time spent obtaining data in order to more precisely measure system performance. According to Wireshark, the response time is 7.6 milliseconds. The use of the CPU and memory is 9.9% and 13.9%, respectively. OpenweatherMap provides information on the current temperature, body temperature, likelihood of rain in the next hour, wind speed, and other variables.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

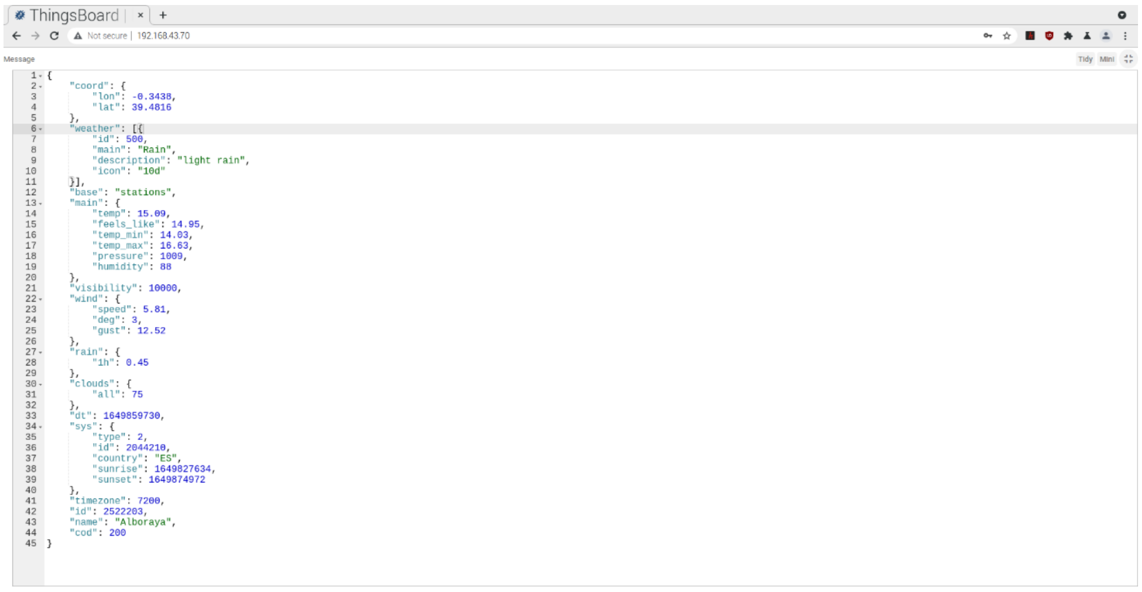
```
No.  Time                          Source          Destination     Protocol Length Info
 405 2022-11-25 18:28:12.070152261 192.168.43.70   37.139.1.159    TCP       66 60726 → 80 [ACK] Seq=295 Ack=852 Win=64128 Len=0 TSval=3363694035 TSecr=237357381
 406 2022-11-25 18:28:12.072258886 192.168.43.70   37.139.1.159    TCP       66 60726 → 80 [FIN, ACK] Seq=295 Ack=852 Win=64128 Len=0 TSval=3363694038 TSecr=237357381
 407 2022-11-25 18:28:12.139102114 37.139.1.159    192.168.43.70   TCP       66 80 → 60726 [ACK] Seq=852 Ack=296 Win=43264 Len=0 TSval=237357451 TSecr=3363694038
 408 2022-11-25 18:28:26.493155610 192.168.43.70   37.139.1.159    TCP       74 60728 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3363708459 TSecr=0 WS=128
 409 2022-11-25 18:28:26.754028578 37.139.1.159    192.168.43.70   TCP       74 80 → 60728 [SYN, ACK] Seq=0 Ack=1 Win=43440 Len=0 MSS=1360 SACK_PERM=1 TSval=237372065 TSecr=3363…
 410 2022-11-25 18:28:26.754103127 192.168.43.70   37.139.1.159    TCP       66 60728 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3363708719 TSecr=237372065
 411 2022-11-25 18:28:26.755117555 192.168.43.70   37.139.1.159    HTTP     360 GET /data/2.5/weather?lat=39.4816&lon=-0.3438&units=metric&APPID=544cb79412d6fe22a4b4b489ca18bafb…
 412 2022-11-25 18:28:26.831356377 37.139.1.159    192.168.43.70   TCP       66 80 → 60728 [ACK] Seq=1 Ack=295 Win=43264 Len=0 TSval=237372143 TSecr=3363708720
 413 2022-11-25 18:28:26.834332021 37.139.1.159    192.168.43.70   HTTP     916 HTTP/1.1 200 OK  (application/json)
 414 2022-11-25 18:28:26.834623051 192.168.43.70   37.139.1.159    TCP       66 60728 → 80 [ACK] Seq=295 Ack=852 Win=64128 Len=0 TSval=3363708800 TSecr=237372145
 415 2022-11-25 18:28:26.836219761 192.168.43.70   37.139.1.159    TCP       66 60728 → 80 [FIN, ACK] Seq=295 Ack=852 Win=64128 Len=0 TSval=3363708802 TSecr=237372145
 416 2022-11-25 18:28:26.902477929 37.139.1.159    192.168.43.70   TCP       66 80 → 60728 [ACK] Seq=852 Ack=296 Win=43264 Len=0 TSval=237372214 TSecr=3363708802
 417 2022-11-25 18:28:41.492765636 192.168.43.70   192.168.43.1    DNS       82 Standard query 0x85cd A api.openweathermap.org
 418 2022-11-25 18:28:41.492804855 192.168.43.70   192.168.43.1    DNS       82 Standard query 0x88cd AAAA api.openweathermap.org
 419 2022-11-25 18:28:41.587785051 192.168.43.1    192.168.43.70   DNS      146 Standard query response 0x85cd A api.openweathermap.org A 82.196.7.246 A 37.139.1.159 A 37.139.20…
 420 2022-11-25 18:28:41.588045509 192.168.43.1    192.168.43.70   DNS       82 Standard query response 0x88cd AAAA api.openweathermap.org
 421 2022-11-25 18:28:41.589028554 192.168.43.70   82.196.7.246    TCP       74 45216 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=411264407 TSecr=0 WS=128
 422 2022-11-25 18:28:42.134370449 82.196.7.246    192.168.43.70   TCP       74 80 → 45216 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1360 SACK_PERM=1 TSval=1310824887 TSecr=411…
 423 2022-11-25 18:28:42.134546527 192.168.43.70   82.196.7.246    TCP       66 45216 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=411264953 TSecr=1310824887
 424 2022-11-25 18:28:42.136131963 192.168.43.70   82.196.7.246    HTTP     360 GET /data/2.5/weather?lat=39.4816&lon=-0.3438&units=metric&APPID=544cb79412d6fe22a4b4b489ca18bafb…
 425 2022-11-25 18:28:42.212541669 82.196.7.246    192.168.43.70   HTTP     916 HTTP/1.1 200 OK  (application/json)
 426 2022-11-25 18:28:42.212784519 82.196.7.246    192.168.43.70   TCP       66 80 → 45216 [ACK] Seq=1 Ack=295 Win=30208 Len=0 TSval=1310824965 TSecr=411264954

    [Timestamps]
       [Time since first frame in this TCP stream: 0.623513115 seconds]
       [Time since previous frame in this TCP stream: 0.076409706 seconds]
    TCP payload (850 bytes)
 Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
    Server: openresty\r\n
    Date: Fri, 25 Nov 2022 17:28:42 GMT\r\n
    Content-Type: application/json; charset=utf-8\r\n
    Content-Length: 471\r\n
    Connection: close\r\n
    X-Cache-Key: /data/2.5/weather?APPID=544cb79412d6fe22a4b4b489ca18bafb&lat=39.48&lon=-0.34&units=metric\r\n
    Access-Control-Allow-Origin: *\r\n
    Access-Control-Allow-Credentials: true\r\n
    Access-Control-Allow-Methods: GET, POST\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.076409706 seconds]
    [Request in frame: 424]
    [Request URI: http://api.openweathermap.org:80/data/2.5/weather?lat=39.4816&lon=-0.3438&units=metric&APPID=544cb79412d6fe22a4b4b489ca18bafb]
    File Data: 471 bytes
 JavaScript Object Notation: application/json
    Object

0020  2b 46 00 50 b0 a0 d5 7f  84 7f e0 9d 37 e3 80 19   +F·P···· ····7···
0030  00 76 61 50 00 00 01 01  08 0a 4e 21 9a 05 18 83   ·vaP···· ··N!····
0040  67 ba 48 54 54 50 2f 31  2e 31 20 32 30 30 20 4f   g·HTTP/1 .1 200 O
0050  4b 0d 0a 53 65 72 76 65  72 3a 20 6f 70 65 6e 72   K··Serve r: openr
0060  65 73 74 79 0d 0a 44 61  74 65 3a 20 46 72 69 2c   esty··Da te: Fri,
0070  20 32 35 20 4e 6f 76 20  32 30 32 32 20 31 37 3a    25 Nov  2022 17:
0080  32 38 3a 34 32 20 47 4d  54 0d 0a 43 6f 6e 74 65   28:42 GM T··Conte
0090  6e 74 2d 54 79 70 65 3a  20 61 70 70 6c 69 63 61   nt-Type:  applica
00a0  74 69 6f 6e 2f 6a 73 6f  6e 3b 20 63 68 61 72 73   tion/jso n; chars

Time since the request was sent (http.time)        Packets: 493 · Displayed: 493 (100.0%)        Profile: Default
```

Figure 5-19: Response time using Wireshark



```
top - 18:13:09 up  1:21,  2 users,  load average: 0.59, 0.30, 0.30
Tasks: 198 total,   1 running, 197 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.2 us,  3.0 sy,  0.0 ni, 94.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3838.7 total,    826.3 free,   1211.1 used,   1801.3 buff/cache
MiB Swap:    100.0 total,    100.0 free,      0.0 used.   2076.8 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  597 root      20   0  225720  79392  60664 S  10.9   2.0   2:37.93 Xorg
  475 thingsb+  20   0  654968 546740  14616 S   9.9  13.9  15:16.69 java
 4537 pi        20   0   86616  29028  22688 S   3.0   0.7   0:05.25 lxterminal
  434 nobody    20   0    4320   2156   1976 S   1.0   0.1   0:09.02 thd
  761 pi        20   0   69660  20424  14936 S   1.0   0.5   0:03.94 openbox
 4551 pi        20   0   10424   3000   2464 R   1.0   0.1   0:01.48 top
  229 root      -2   0       0      0      0 S   0.7   0.0   0:07.22 v3d_bin
 1337 pi        20   0  626988 202648 145932 S   0.7   5.2   3:40.37 chromium-browse
  230 root      -2   0       0      0      0 S   0.3   0.0   0:07.24 v3d_render
  620 postgres  20   0  204424   5992   4552 S   0.3   0.2   0:00.19 postgres
  621 postgres  20   0   61280   3416   2112 S   0.3   0.1   0:01.80 postgres
  666 pi        20   0   56652  12956  11604 S   0.3   0.3   0:01.00 lxsession
  769 pi        20   0  429796  33756  25816 S   0.3   0.9   0:13.84 lxpanel
 1595 pi        20   0  304876  73028  62820 S   0.3   1.9   0:29.35 chromium-browse
 4529 root      20   0       0      0      0 I   0.3   0.0   0:00.09 kworker/u8:0-event+
    1 root      20   0   33768   8156   6404 S   0.0   0.2   0:07.30 systemd
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.02 kthreadd
    3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
```

Figure 5-20: CPU and memory performance of the system



```
{
    "coord": {
        "lon": -0.3438,
        "lat": 39.4816
    },
    "weather": [{
        "id": 500,
        "main": "Rain",
        "description": "light rain",
        "icon": "10d"
    }],
    "base": "stations",
    "main": {
        "temp": 15.09,
        "feels_like": 14.95,
        "temp_min": 14.03,
        "temp_max": 16.63,
        "pressure": 1009,
        "humidity": 88
    },
    "visibility": 10000,
    "wind": {
        "speed": 5.81,
        "deg": 3,
        "gust": 12.52
    },
    "rain": {
        "1h": 0.45
    },
    "clouds": {
        "all": 75
    },
    "dt": 1649859730,
    "sys": {
        "type": 2,
        "id": 2044210,
        "country": "ES",
        "sunrise": 1649827634,
        "sunset": 1649874972
    },
    "timezone": 7200,
    "id": 2522203,
    "name": "Alboraya",
    "cod": 200
}
```

Figure 5-21: Data from OpenweatherMap

It turned out that the Raspberry Pi 4 was able to successfully connect to Thingsboard and get data from OpenweatherMap. This information was then utilised to monitor the temperature and humidity of the surrounding area, and it was displayed onto a dashboard for additional study. Users are able to examine the current weather conditions for a specific location through the use of this project. The information about the weather that is provided to users will always be up to date because it comes from OpenweatherMap.

6. CONCLUSIONES

The results of the study have demonstrated that a Raspberry Pi 4 can connect to Thingsboard and retrieve data from OpenweatherMap. This is an important discovery since it makes it possible to use this platform to create a wide range of applications that can use data from the OpenweatherMap API.

It was seen to be a rather simple process to connect the Raspberry Pi 4 to Thingsboard, and the Thingsboard documentation was found to be clear and helpful. The data from the OpenweatherMap API was successfully retrieved after the connection was made, and it was shown on the Thingsboard dashboard.

It was determined that the data from OpenweatherMap was reliable and current, which is essential for any programs that uses this data. In general, it was successful and reasonably simple to connect the Raspberry Pi 4 to Thingsboard and retrieve data from OpenweatherMap. We were able to quickly and easily obtain data from OpenweatherMap thanks to the success of this project.

6.1 FUTURE WORK

In the future, the initiative might be expanded to incorporate information from additional weather services, such the National Weather Service.

The project might also be expanded to incorporate information from different kinds of sensors, like air quality sensors.

The concept might be expanded to incorporate information from different kinds of gadgets, such as smart thermostats.

BIBLIOGRAPHY

[1] J. Kim, S. C. Choi, I. Y. Ahn, N. M. Sung, and J. Yun, "From WSN towardsWoT: Open API scheme based on oneM2M platforms," *Sensors (Switzerland)*, vol. 16, no. 10, Oct. 2016, doi: 10.3390/s16101645.

[2] A. Suryana, F. P. Lismana, R. M. Rachmat, S. D. Putra, and M. Artiyasa, "Implementation of Weather Station for the Weather Reality in A Room," in *6th International Conference on Computing, Engineering, and Design, ICCED 2020*, Oct. 2020. doi: 10.1109/ICCED51276.2020.9415799.

[3] Y. A. Ahmad, T. Surya Gunawan, H. Mansor, B. A. Hamida, A. Fikri Hishamudin, and F. Arifin, "On the Evaluation of DHT22 Temperature Sensor for IoT Application," in *Proceedings of the 8th International Conference on Computer and Communication Engineering, ICCCE 2021*, Jun. 2021, pp. 131–134. doi: 10.1109/ICCCE50029.2021.9467147.

[4] Amity University. Dubai Campus, Institute of Electrical and Electronics Engineers. UAE Section, and Institute of Electrical and Electronics Engineers, *Abstract proceedings of International Conference on Computation, Automation and Knowledge Management (ICCAKM-2020) : 9th-10th January 2020*.

[5] A. Gupta, A. Tripathi, R. Coutinho, R. Rodrigues, and P. Raut, "Smart Sustainable Mini Weather Station," in *IEEE Region 10 Humanitarian Technology Conference, R10-HTC*, 2021, vol. 2021-September. doi: 10.1109/R10-HTC53172.2021.9641672.

[6] Institute of Electrical and Electronics Engineers and IEEE Communications Society, *The 2019 International Symposium on Networks, Computers and Communications (ISNCC 2019) : 18-20 June, 2019, Istanbul, Turkey*.

[7] C. Dewi and R.-C. Chen, "Integrating Real-Time Weather Forecasts Data Using OpenWeatherMap and Twitter," 2019. [Online]. Available: http://ejournal.uksw.edu/ijiteb

[8] E. Okhovat and M. Bauer, “Monitoring the Smart City Sensor Data Using Thingsboard and Node-Red,” in *Proceedings - 2021 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Internet of People, and Smart City Innovations, SmartWorld/ScalCom/UIC/ATC/IoP/SCI 2021*, 2021, pp. 425–432. doi: 10.1109/SWC50871.2021.00064.

[9] Institute of Electrical and Electronics Engineers, *2018 IEEE Global Conference on Internet of Things (GCIoT).*

[10] M. Casillo, F. Colace, M. de Santo, A. Lorusso, R. Mosca, and D. Santaniello, “VIOT_Lab: A Virtual Remote Laboratory for Internet of Things Based on ThingsBoard Platform,” in *Proceedings - Frontiers in Education Conference, FIE*, 2021, vol. 2021-October. doi: 10.1109/FIE49875.2021.9637317.

[11] L. O. Aghenta and M. T. Iqbal, “Low-cost, open source IoT-based SCADA system design using thinger.IO and ESP32 thing,” *Electronics (Switzerland)*, vol. 8, no. 8, Aug. 2019, doi: 10.3390/electronics8080822.

[12] I. O. Suzanti, N. Fitriani, A. Jauhari, and A. Khozaimi, “REST API Implementation on Android Based Monitoring Application,” in *Journal of Physics: Conference Series*, Jul. 2020, vol. 1569, no. 2. doi: 10.1088/1742-6596/1569/2/022088.