Journal of Theoretical and Applied Information Technology
15th December 2018. Vol.96. No 23
© 2005 – ongoing JATIT & LLS

ISSN: **1992-8645** www.jatit.org E-ISSN: **1817-3195**

WEATHER MONITORING SERVICES BASED ON INTERNET OF THINGS AND CLOUD COMPUTING BY USING TEMPERATURE, HUMIDITY AND RAINFALL ENVIRONMENT PARAMETERS

[1]PUTRI SEROJA NASUTION, [2]AHMAD NURUL FAJAR
[1,2]Information Systems Management Department, BINUS Graduate Program-Master of Information Systems Management, Bina Nusantara University Jakarta, Indonesia 11480
E-mail: [1]putri.nasution@binus.ac.id, [2]afajar@binus.edu

ABSTRACT

Internet of Things (IoT) technology cannot be separated from the use of Cloud computing in terms of access, and it has many advantages when compared with traditional computing, such as: cost saving, reliability and scalability. IoT can be applied in various fields such as smart city, smart transportation, agriculture, weather monitoring and other fields. In this research, design and development of Cloud computing based Internet of Things for weather monitoring based on temperature, humidity and rainfall of surrounding environment parameters as message, and using MQTT which is IoT/M2M connectivity protocol. There are three main components of MQTT: the first component that generates the message is called a publisher, the second that consumes messages is called a subscriber, and the last component that handles the interaction between the publisher and the subscriber is called a broker or intermediary. Usability testing is also done from each component of the system using the black box approach that can represent overall functionality. Based on the design model of weather monitoring system, end-user can monitor weather conditions according to the placement node device of location and through Web-based applications installed on the cloud server. The amount of data message formatted into JSON is ± 120 bytes, and is published by node with time interval per ± 30 seconds.

Keywords: *Internet of Things, Cloud, MQTT Broker, Publish/Subscribe, JSON*

1. INTRODUCTION

Through Internet technology various devices can be widely connected, so they can communicate with each other, interact, exchange data or information and predicted by 2020 the number of devices that will connect to the Internet reaches more than 50 billion [1]. The next generation of Internet technology is referred to as Internet of Things (IoT) which is closely related to cloud computing technology that has the concept not only to connect various devices but also connect smart city, smart agriculture, smart home or building, smart healt and others [2]. In addition cloud computing technology uses the concept of resources pooling/shared pool scalable system, so the demand of system can be fulfilled as soon as possible [3].

Kevin Ashton claims that he was the first to use the phrase of "Internet of Things" when he working on Auto-ID Massachusetts Institute of Technology (MIT), while giving his presentation about connecting RFID information from P & G's supply chain to the Internet in 1999 [4]. With the convergence between the Internet of Things Technology with cloud computing is expected to provide a positive impact that is beneficial in everyday life, one of which is like the application of IoT and cloud on weather monitoring as one important factor to support various activities to be lived, such as work, school and even on vacation [5].

Other research related to this topic such as research [6] discusses IoT and Cloud integration solutions monitoring patient voice pathology through voice signals, body temperature and humidity around and sending them via Bluetooth technology to patient smartphones. Research [7] IoT case studies on meteorologists use simulators to represent temperature, humidity, wind speed and other data considered as sensors. Research [5] IoT prototyping weather monitoring utilizes weather cloud service and displays it through WeMos devices. Research [8] that evaluates performance as well as choosing the appropriate technology to build an IoT system. Research [9] IoT scenario on supply chain for identification and tracking flow logistics in realtime using GPRS, GPS and RFID devices. The last research [10] analyze IoT monitoring and remote control of business processes in realtime on food supply chain by virtualizing them.

Based on these studies, in general there are issues or questions such as about the middleware aspect to connect IoT devices with cloud, interoperability aspects of IoT and client devices to exchange data, and dynamic scalability of the system when resources need to be upgraded. In this research the design of Internet of things based cloud computing for weather monitoring services, data formatted into JSON generated by the node device consist of *embedded microcontroller unit* ATmega328 is integrated with DTH11 module as temperature sensor and humidity, rainfall sensor module using LM393 IC, SIM800L GPRS module to connect to internet network, and use VPS service as cloud server, so weather monitoring services can be considered as *cloud software as a service* (SaaS). Hopefully through the architecture or model of the results of this study can be applied in other IoT case studies in real terms.

2. STATE OF THE ART

Message Queue Telemetry Transports Protocol (MQTT) is *lightweight protocol* connectivity for *machine-to-machine* (M2M) or IoT, has the ability to transmit and receive data in accordance with existing bandwidth capabilities or conditions, and is designed for the use of small voltages [11], so it is used as a primary consideration, as well as for optimization. MQTT was first developed by Andy Stanford-Clark (IBM) and Arlen Nipper in 1999 to monitor the oil pipeline. In handling data/message exchange, MQTT uses architecture or publish and subscribe methods, unlike HTTP using the concept of request and response [5].

Publish and subscribe method is event-driven which means there is a client in the form of node or user acting as publisher and subscriber. When the publisher sends data/messages to a topic, it will be handled by the MQTT broker who acts as an intermediary, and then the message will be forwarded to the subscriber based on interest or following topic as shown in Figure 1.

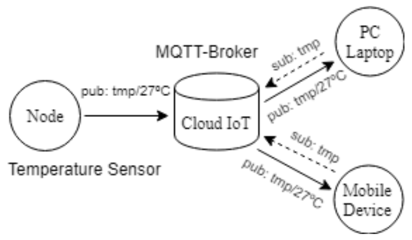


Figure 1: MQTT Publish And Subscribe Method

In MQTT there are 3 levels of QoS that can guarantee the data/message exchange on unreliable network conditions.

1. QoS 0 at most once that is the message will be sent only once with the best effort, at this level the possibility of missing messages may occur.
2. QoS 1 at lease once that is the message is guaranteed acceptable, but data duplication can occur.
3. QoS 2 exactly once that is message guarantee received exactly once, but will result in increased bandwidth usage.

Characteristics of cloud computing is not only limited to devices or services that can be accessed through the Internet network. Cloud computing has the meaning of a model provider of computing resources in the form of hardware, software, storage, network, power and other connected to one, and can be in management through interaction processing to a minimum [12]. Here can be described as massive-resources that are connected to each other and can be used as needed or on-demand. The cloud depiction describes the hiding of the complexities of various resources and interconnected infrastructure.

Based on the National Institute of Standards and Technology (NIST) there are 3 basic cloud computing services model [13] as follows.

1. *Software as a Service* (SaaS) is distribution model of applications offered by subscription (pay per-use), without to have a computing resource to run the selected application, because it can be accessed via the Internet network. For examples such as Microsoft Office 365, DropBox and Google Drive.
2. *Platform as a Service* (PaaS) PaaS is a paradigm for providing runtime from various programming languages and other support. This model is like a hosting environment so users can directly build and run applications on it via the Internet network, without having to

install and configure such as operating system, networking and other.

3. *Infrastructure as a Service* (IaaS) a service model that enables to get support for operations such as storage, hardware, servers and networking that can be accessed through the Internet.

The following in Figure 2 describes the management comparison of each service model of *resource cloud computing*, which consists of SaaS service model, PaaS and IaaS models from the user or customer perspective and from the cloud computing service provider perspective.

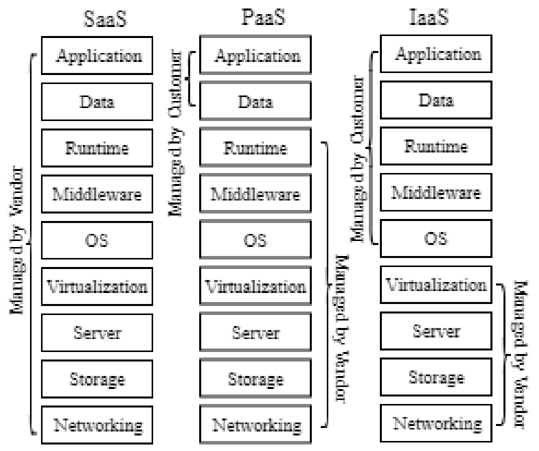


Figure 2: Comparison of Resource Cloud Computing Service Model Management [13}

There are various advantages offered by using cloud computing services, especially in this research development of IoT services, including as follows.

1. Scalability or flexibility: can easily increase or decrease resource which used for application or user needs and also reduces infrastructure complexity and developer workload.
2. Cost saving: to reduce especially does not require the procurement of infrastructure and computing resources that are not cheap (capital investment), and may require only a few experts in the field of IT and also the concept of cloud systems generally is pay-per-use.
3. Maintenance: mostly the system updates, repair or fixes are only done by the cloud computing provider, so it is possible the user does not need to buy the latest technology of resource (cost saving).
4. Portability or mobile access: users can work from anywhere and anytime via the internet network, so it can increase the productivity.

JSON is a lightweight data format, easy to read and write by humans, and easily translated and generated by the computer [14]. JSON is a text format that does not depend on any programming language because it uses a simple language style, and almost supported by all programming languages such as C, C++, C#, Java, JavaScript, Python and PHP. So this is what makes JSON as the ideal data exchange language format to handle interoperability. Here in Figure 3 the formation of JSON data formatting in the form of objects consisting of a pair of names or values and not ordered.

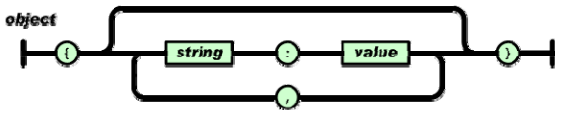


Figure 3: Format Object JSON [14]

In Figure 3 the JSON object format consists of a pair of names or values that do not have to be sorted. The object begins with "{" open curly brackets and ends with "}"closing braces. Each name follows: "." colon and each name or value pair separated by "," comma.

The testing outline is one of the important aspects of the system to find or know the functionality or usability of the system can run properly. Usability or User-based testing is basically a black-box testing techniques also referred to as testing with a focus on the behavior of the system user perspective [15,16]. Besides that, the system architecture has been done by [17] to anticipate dynamic change ecosystem. By performing black-box testing techniques through various functions that represent various things and impact to design interaction of user in accessing weather monitoring services. hopefully by doing usability testing using black-box testing approach can provide benefits from various things such as the following.

1. Usability testing to find errors during hardware setup, in software or application development, and also during the process of integration various components.
2. Usability testing can help to fixing errors that affected to user interface before releasing application.

3. Usability testing can identify the formed database structure.

3. RESEARCH METHOD

The following hardware and software requirements that are used in the development weather monitoring service system consist of hardware and software.

1. Node Hardware: Arduino Pro Mini 328, SIM800L GPRS Module, Module DTH11 for temperature also humidity sensors, and rainfall sensor using ICLM393 with plate probe.
2. Cloud Server: using VPS server (Virtual Private Server) with specification 1 Core CPU, Storage 30 GB and Memory 512MB.
3. *Software* or application: Editor Arduino IDE, NodeJS with packet mqttclient, ExpressJs, mongoose and Socket.IO; Operating System Ubuntu 14.04 Server Edition; Mosquitto MQTT Broker and MongoDB.

In this study, data structure of weather monitoring services based IoT and cloud for the process of sending and receiving messages consists of 3 parts, that is data topic structure format, initialization range value of temperature sensor, humidity and rainfall, also payload or message data formatting consist of weather data parameter, identity and coordinate node formatted into JSON.

Below in Table 1 is format data topic in string form used by broker to filter messages from each connected client. Each topic can consist of one or more levels separated by a slash.

Table 1: Format Structure Data Topic

Topic Name	Description
weather/node_id	To get weather data from node
initialize/node_id	To get node status online or offline
broker	To get broker status online or offline
toggle_led/node_id	To change node led state become on or off

The following in Table 2 is about classification range value of temperature sensor in Celsius units to define cold, warm and hot conditions that applied in this study.

Table 2: Initialize Range Value of Sensor Temperature

Temperature	Condition
<= 22 ℃	Cold
>= 22 ℃ or <= 35 ℃	Warm
>= 35 ℃	Hot

The following in Table 3 that is about classification range value of humidity sensor in Celsius units to define dry, ideal or normal and moist conditions that applied in this study.

Table 3: Initialization Range Value of Sensor Humidity

Humadity	Condition
<= 40 %	Dry
>= 40% or <= 60%	Ideal
>= 60 %	Moist

The rainfall sensor has a range of decimal values from 0 to 1023. This range value converted to percent to adjust in order to have the same range degree as other value such as weather parameters, and also to facilitate the present data in the form of graph charts. Conversion of rainfall sensor values into percentage form is done in coding software with the following simple equations.

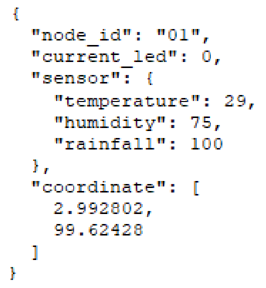
$$\text{rainfall} = \text{sensor\_value}/1023 * 100 \quad (1)$$

The following in Table 4 is about defining the range of rainfall sensor values in the percentage value to define the condition of rain weather, drizzle and rain condition that used in this study.

Table 4: Initialization Range Value Of Rainfall Sensor

Rainfall	Condition
<=50%	Rain
> 50% and < 85%	Drizzle
>= 85%	Sunny

The formatting of weather monitoring parameters into JSON forms consists of data identity and led status, data sensor temperature, humidity, rainfall and coordinates data of latitude and longitude with fixed values. Below in Figure 4 sample formatting the weather parameter data into JSON as a message or payload that will be used.



```
{
  "node_id": "01",
  "current_led": 0,
  "sensor": {
    "temperature": 29,
    "humidity": 75,
    "rainfall": 100
  },
  "coordinate": [
    2.992802,
    99.62428
  ]
}
```

Figure 4: Data Weather Parameter JSON Format

Below in Figure 4, system architecture design or Internet of Things model which implemented on weather monitoring services.

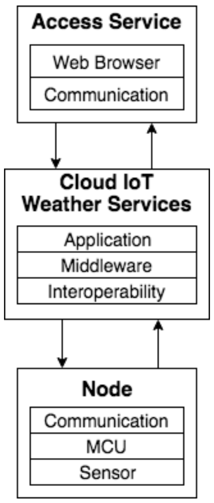


Figure 5:System Architecture Design

The following is a description of Figure 5 on the various layers with components contained in the system architecture design of the Internet of Things cloud-based weather monitoring services.

1. Node layer consists of microcontroller unit (MCU) integrated with sensor in order to sensing and acquisition data temperature, humidity and rainfall environment. Further data or message is sent to the cloud server using GPRS communications serially. Communication component not only serves to send message, but can serve to receive message.
2. Cloud IoT weather services layer is a cloud server that is accessed through the Internet network and is built starting from the installation and configuration of the operating system, and other supporting applications. In this layer there is a middleware component that is responsible for handling or managing how the various nodes with the user to be connected and communicate with each other to send and receive messages. The outline of middleware component itself is consist of MQTT and Node.js that comes with the Socket.IO package. Then the interoperability component is how to deal formatting the data message both data generated by the node and user formatted into the form of JavaScript object notation (JSON). The last component is an web-based weather service monitoring application that develop from Node.js platform using an additional package that is ExpressJS. In this layer is also equipped with MongoDB database applications so that data generated from the Node device can be stored on the cloud server.
3. Access service layer enables users to get temperature, humidity and rainfall information that sent from various node devices, and accessed via cloud IoT weather service layer. This can be done because on this layer there is a Web-based application and using Socket IO to spread node messages. So by using end-user devices, for example smartphones and of course that have the ability to connect to the Internet network and support Web Browse applications, then the various information generated by the node device can be accessed remotely.

On the node device consists of 2 parts design that is hardware and software. Below in the Figure 6 schematic diagram of the hardware node that describes the linkage between the module, component and the microcontroller unit (MCU).

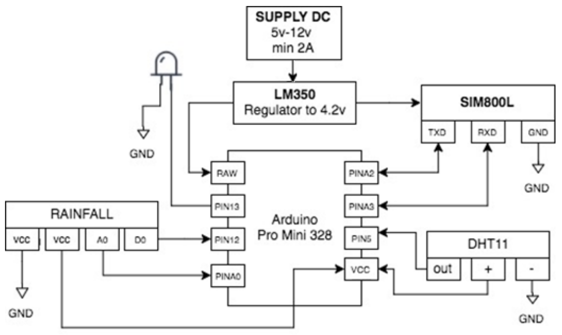


Figure 6: Schematic Hardware Node

Atmega328 as the main controller device has clock speed of 16 MHz, 32 KB flash memory, 2 KB SRAM and 1 KB EEPROM. Power supply for this MCU about 5V - 12V and one of the most important thing is the required power supply SIM800L from 3.7v - 4.2v with the use of the current of 2A obtained from the datasheet. So it takes the use of LM350 regulator and also as safety of various components. In the circuit there is also the addition of an LED indicator that represents data reception by the node for subsequent development work.

The following in the Figure 7 result of designing the hardware node according to the specification and schematic.

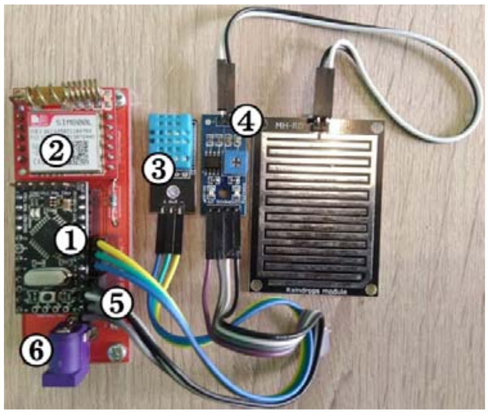


Figure 7: Node Hardware

The following in Figure 8 design of node software consists of several process blocks written into the flowchart.

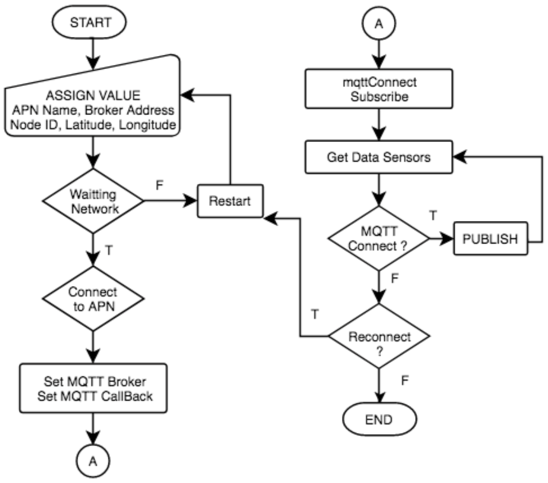
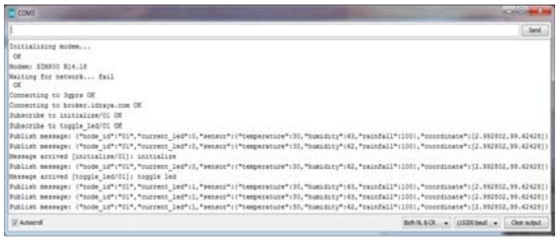


Figure 8: Flowchart Software Node

In Figure 8 shown how the node software can connect and send weather parameter messages to the broker from step assign value access point name (APN) variable according to subscriber indentation (SIM) card which is used in GPRS module, mqtt broker server address, node identity, latitude and longitude of coordinate node. Then the registration phase matches APN and connects to the internet network, subscribe to initialize and toggle led topics, sensing and getting data sensors then format it into JSON and publish it.

The following in Figure 9 results of designing node software based on the flowchart and has compiled, uploaded to node hardware and executed it, and displayed it via Serial Monitor feature form Arduino IDE.



```
Initializing modem...
 OK
Modem: SIM800 R14.18
Waiting for network... fail
 OK
Connecting to 3gprs OK
Connecting to broker.idraya.com OK
Subscribe to initialize/01 OK
Subscribe to toggle_led/01 OK
Publish message: {"node_id":"01","current_led":0,"sensor":{"temperature":30,"humidity":63,"rainfall":100},"coordinate":[2.992802,99.62428]}
Publish message: {"node_id":"01","current_led":0,"sensor":{"temperature":30,"humidity":62,"rainfall":100},"coordinate":[2.992802,99.62428]}
Message arrived [initialize/01]: initialize
Publish message: {"node_id":"01","current_led":0,"sensor":{"temperature":30,"humidity":62,"rainfall":100},"coordinate":[2.992802,99.62428]}
Message arrived [toggle_led/01]: toggle led
Publish message: {"node_id":"01","current_led":1,"sensor":{"temperature":30,"humidity":63,"rainfall":100},"coordinate":[2.992802,99.62428]}
Publish message: {"node_id":"01","current_led":1,"sensor":{"temperature":30,"humidity":63,"rainfall":100},"coordinate":[2.992802,99.62428]}
Publish message: {"node_id":"01","current_led":1,"sensor":{"temperature":30,"humidity":62,"rainfall":100},"coordinate":[2.992802,99.62428]}
```

Figure 9: Software Node

The folloing in Figure 10 software installation on a VPS cloud server for weather monitoring services in accordance with the system architecture in order to work as weather service layer and access service layer.

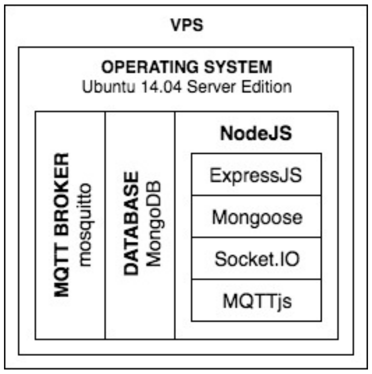


Figure 10: Software Server

The following in Figure 11 design of the weather service layer software consist of several process blocks written into the flowchart and based on the system architecture.

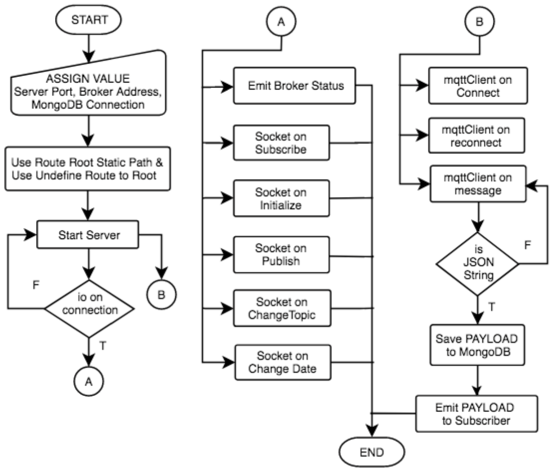
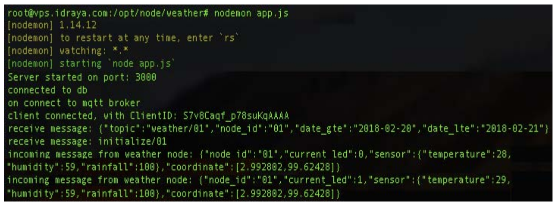


Figure 11: Weather Service Layer Flowchart

In Figure 11 explains how the weather service layer can serve as middleware and interoperability. The task of the broker component to be responsible for managing the message or payload traffic between the publisher and subscriber. If there is a message received, it will be validated by checking the structure of JSON data format, then if the validation result is correct, the message is stored fist into the database MongoDB and then forwarded to the subscriber. On this layer there are triggers to check the status of brokers and nodes whether in offline or online conditions.

The following in Figure 12 result of designing weather service layer software based on flowchart and has been run on the server and built on NodeJS platform based JavaScript and including by various libraries such as ExpressJs to create web application easily and with middleware, mongoose as the connecting driver between NodeJs and MongoDB database, and also Socket.IO as WebSocket for connecting between service layer and access service layer that occurs bidirectional.



```
root@vps.idraya.com:/opt/node/weather# nodemon app.js
[nodemon] 1.14.12
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
Server started on port: 3000
connected to db
on connect to mqtt broker
client connected, with ClientID: S7v8Caqf_p78suKqAAAA
receive message: {"topic":"weather/01","node_id":"01","date_gte":"2018-02-20","date_lte":"2018-02-21"}
receive message: initialize/01
incoming message from weather node: {"node id":"01","current led":0,"sensor":{"temperature":28,
"humidity":59,"rainfall":100},"coordinate":[2.992802,99.62428]}
incoming message from weather node: {"node_id":"01","current_led":1,"sensor":{"temperature":29,
"humidity":59,"rainfall":100},"coordinate":[2.992802,99.62428]}
```

Figure 12: Weather Service Layer Apps

The following in Figure 13 design of access service layer software consist of several process blocks written into the flowchart and based on the system architecture.

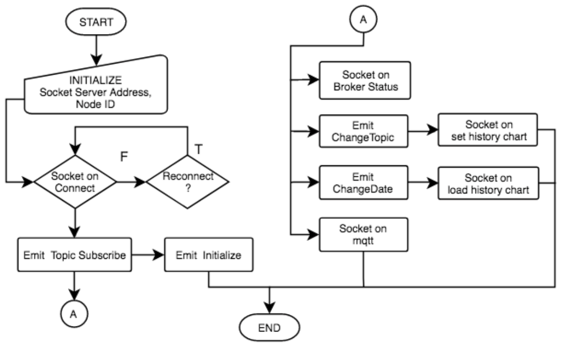


Figure 13: Access Service Layer Flowchart

Figure 13 describes how access service layers can represent or function as an application layer, starts from process of initialization socket server address and node id variables at the time application is accessed by user. Then the socket client connects to the socket server, next subscribe to the weather topic and publish it to topic initialize to check the selected node status is online or offline. While on the broker, topic is send by socket server when the client socket connected, and various functions such as to display or plot to chart the history of node weather data that stored in MongoDB database.

The following in Figure 14 results of designing weather service layer software running on a web server that represents user applications for monitoring weather services based on temperature, humidity, rainfall through surrounding environments based on the location of node device placement

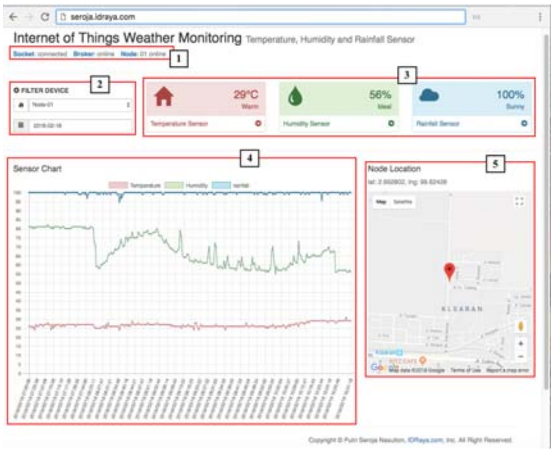


Figure 14: Access Service Layer Apps

Based on Figure 14 above the weather data is published by the node through weather topic and formatted into JSON, then after forwarded by the broker the data is parsed to be displayed to the user via a computer device such as laptop, smartphone or table using a web browser without having to depend on the operating system used.

From this layer the user can also view the broker status, enable or disable the led switch through toggle_led topic example as publisher, and displays weather data history based on node and selected date.

4. RESULTS AND DISCUSSION

usability testing is done to answer all specifications on the weather monitoring system that built can be fulfilled or not neglected, both in terms of hardware and software, and usability testing using a black-box approach through functionality that represents the overall weather monitoring system based on the internet of things and the cloud that implemented.

The following in Table 5 usability testing black box approach on the node device that is done in accordance with the results of system design either in the form of software or hardware..

Table 5: Node Device Black-Box Testing

Type of Test	Configuration	Observation
Compile & upload source code	Connect Arduino to usb computer via serial communication.	Success compiling and code able to write into arduino/node.
Connect to internet	Insert SIM Card to SIM800L and set APN	Able to get internet access connection via GPRS/2G
Connect to broker	Set broker address to localhost and use port 1883	Successful connect to broker identify by message on server
Read sensor	Read module DHT11 and rainfall per ±10 seconds	Able to get data temperature, humidity and rainfall
Create JSON data	Create object for node_id and current_led; nested object for sensor temperature, humidity and rainfall; and nested array for posistion latitude and longitude	Successful creating JSON data formatted with size ± 120 bytes
Publish	Topic weather/node_id, initialize/node_id and toggle_led/node_id by using QoS level 0	Successful publishing message to broker
Subscribe	Topic initialize/node_id and, toggle_led/node_id by using QoS level 1	Successful subscribing to all topics.

Analysis result of usability testing node device by using black-box approach is done thoroughly representing the functionality of hardware and software can run as expected. This is because before testing is done every component both node device, server device and user can connect to each other, and can run according to its function, and then testing is done gradually in accordance with the design of both hardware and software.

The following in Table 6 usability testing black box approach on server services performed in accordance with the design of system weather service layer and access service layer that represents the overall functionality.

Table 6: Server Service Black-Box Testing

Type of Test	Configuration	Observation
Start MongoDB	Execute by command mongod --dbpath /data/db vai terminal	Successful start MongoDB server
Run app.js file	Execute by command nodemon app.js via terminal	Code apps successful run on server via 3000 port
Start mosquitto mqtt broker	Execute by command /etc/init.d/mosquitto start via terminal	Successful start mosquitto broker on server
Receive message	Occurred when mqttclient on message function	Successful receive message based on each publisher topics
Save message to MongoDB	Occurred when mqttclient on message function then create Node save function	Successful save receive message published by node.
Forward message	Occoured when mqttclient on message and all socket server function, then socket server emit or mqttclient publish to subscriber	Successful forward message or status to subscriber based on their interest topics
Access service layer as IoT Weather apps	Occoured when end-user open IoT weater services application via URL or Server Address	End user successful receive message such as broker status, node and weather information based on selected node_id

Analysis result of usability testing server service by using black-box approach is done thoroughly representing the functionality of weather service layer and access service layer software can run as expected. This is because before testing is done from any layer or application both on the server side and node devices can connect to each other, and can run according to its function, , and then testing is done gradually in accordance with the design of both hardware and software by selecting filter device action. Through the user application interface various indicators of classification weather parameters such as temperature, humidity and rainfall can be seen or identified.

5. CONCLUTIONS

Based on research that has been done both from the results of implementation, testing and observation can be concluded that.

1. Node device can send or publish message in the form of weather parameter data consisting of temperature, humidity and rainfall around the environment as well as coordinates location of node placement formatted into JSON using MQTT protocol to broker with delivery topic. The messages received by the cloud server as middleware is stored into the MongoDB database first, and then forwarded to the subscriber according to topics of interest.
2. Even though the node publishes messages using QoS level 0, it's almost never packaged that is not received or received by the cloud server, this is because the size of the message formatted into the JSON can be said that is small ± 120 bytes and in publish with time intervals per ± 30 seconds, also influenced the quality of the Internet network on the node is quite good.
3. Through the application interface of the weather monitoring service installed on the cloud server, environmental weather data parameters around the node device can be monitored by the user through the application of weather monitoring services using a web browser, and can be considered a Cloud SaaS service. This is certainly because interoperability between publisher and subscriber can be handled by using the agreed JSON data format and also makes it easier to parse data message.

Internet of things weather monitoring services are outlined based on the design of system architecture and other designs, so it is hoped that the results of this research model can be applied in other case studies, such as IoT for monitoring on the floodgates, vehicles tracking and others. For further work on the node device a GPS sensor is added to retrieve coordinate data, if the node is always on the move. The weather parameter data published by the node can only use QoS level 0, due to the unavailability of the other QoS level usage features for send in the mqtt library used. Nevertheless, the cloud server still passes it to the consumer according to the level of QoS subscribe to the topic, and for subsequent work to develop the mqtt library it can support other QoS levels.

REFERENCES:

[1] David Bradley, David Russell, Ian Ferguson, John Isaacs, Allan MacLeod, Roger White, “The Internet of Things – The future or the end of mechatronics”, ELSEVIER in Mechatronics, Volume 27, April 2015, Pages 57-74.

[2] M. Agus Triawan, H. Hindersah, et.al, "Internet of things using publish and subscribe method cloud-based application to NFT-based hydroponic system", *International Conference on System Engineering and Technology*, 2016, pp. 98-104.

[3] Santosh Kumar and R. H. Goudar, “Cloud Computing – Research Issues, Challenges, Architecture, Platforms and Applications: A Survey”, *IEEE in International Journal of Future Computer and Communication* vol. 1, no. 4 pp. 356-360, 2012

[4] Kevin Ashton, “That ‘Internet of Things’ Thing,” *RFID Journal*, 22 June 2009.

[5] R. K. Kodali and A. Sahu, “An IoT based weather information prototype using WeMos”. *IEEE International Conference on Contemporary Computing and Informatics (IC3I),* Noida, 2016, pp. 612-616.

[6] G. Muhammad, S. M. M. Rahman, A. Alelaiwi and A. Alamri, “Smart Health Solution Integrating IoT and Cloud: A Case Study of Voice Pathology Monitoring”, *IEEE in Communications Magazine*, vol. 55, no. 1, pp. 69-73, January 2017.

[7] A. Markus and A. Kertesz, "*Simulating IoT Cloud systems: A meteorological case study*," *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, Valencia, 2017, pp. 171-176.

[8] K. Vandikas and V. Tsiatsis, “Microservices in IoT clouds”, *in IEEE 2016 Cloudification of the Internet of Things (CIoT)*, Paris, 2016, pp. 1-6.

[9] Z. David R. Gnimpieba, Ahmed Nait-Sidi-Moh, David Durand, Jérôme Fortin, “Using Internet of Things Technologies for a Collaborative Supply Chain: Application to Tracking of Pallets and Containers”, *ELSEVIER in Procedia Computer Science* Volume 56, 2015, Pages 550-557.

[10] C.N. Verdouw, J. Wolfert, A.J.M. Beulens, A. Rialland, “Virtualization of food supply chains with the internet of things”, *ELSEVIER in Journal of Food Engineering* Volume 176, 2016, Pages 128-136.

[11] __________, “Frequently Asked Questions”, http://mqtt.org/faq, 01 November 2017.

[12] Santosh Kumar and R. H. Goudar, "Cloud Computing – Research Issues, Challenges,

Architecture, Platforms and Applications: A Survey", *International Journal of Future Computer and Communication* vol. 1, no. 4 pp. 356-360, 2012.

[13] _____, "The NIST Definition of Cloud Computing", SP 800-145 (September 2011).

[14] ________________, "Introduction JSON", https://www.json.org/index.html, 07 November 2017.

[15] G. J. Myers, T. Badgett and C. Sandler, "Usability (User) Testing. In The Art of Software Testing", 2015.

[16] Manuel Bacso, Omar Lenin Gutiérrez Gutiérrez, "Designing and executing a security and usability testing plan: IdeaClick Prototype", *HAAGA-HELIA University in Business Information Department Technology*, 2013.

[17]Fajar,A.N..,Budiardjo,E.K.,Hasibuan,Z.A. System Architecture in Dynamic Environment based on Commonality and Variability Business Processes. *8th ICCM IEEE 2012*