UDP Pervasive Protocol Integration with IoT for Smart Home Environment using LabVIEW

Mochammad Hannats Hanafi Ichsan, Wijaya Kurniawan, Sabriansyah Rizqika Akbar
Computer Engineering, Computer System and Robotics Laboratory, Brawijaya University, Indonesia

Article Info

Article history:

Received Jan 2, 2018
Revised Jun 11, 2018
Accepted Jun 28, 2018

Keyword:

Internet of Things
LabVIEW
Pervasive Protocols
Smart Home
UDP

ABSTRACT

Pervasive computing is an environment which is used and integrated into every object and activities to meet human needs and its existence isn't perceived as something specific. The concept of Smart Home is to assist human needs in an everyday object that performs controls or being controlled. Based on previous research the used communication protocol is UDP (User Datagram Protocol) and the programming language is LabVIEW. UDP is used because it does not require handshaking in the broadcast process, as well as on the use of memory more efficient than other protocols. Devices which perform controls called Host and which is controlled called Client. Both of them (Things) have an ability to send data to the Internet without any human interaction. So this research wants to conduct pervasive protocol between Host and Client which each device is integrated with the Internet of Things (IoT). Data are posted at dweet.io that is a cloud server website that contains a simple online data submission which has free services. This research is conducted to measure the communication performance between host to client, host to cloud server and client to cloud server that represents household equipment.

Corresponding Author:

Mochammad Hannats Hanafi Ichsan,
Computer Engineering, Faculty of Computer Science,
Brawijaya University,
Jl. Veteran no 8, Ketawanggede, Kec. Lowokwaru, Malang, Jawa Timur, Indonesia.
Email: hanas.hanafi@ub.ac.id

1. INTRODUCTION

Pervasive computing is not only like desktop computing, but also could integrate with any device, any location, anytime and any data format across any network environment [1]. That device could perform any task such as monitoring and controlling other device [2]. Pervasive computing is used in many areas for its usability mobility usage, method and computation size of the pervasive device [3]. One of six technologies that continuously evolve until 2025 is the Internet of Things (IoT) [4]. IoT could be utilized to any environment such as a home. Lighting, heat, or another device that can be controlled remotely via the internet is Smart Home concept [5]. Smart Home is a very suitable object that can be implemented by pervasive computing, because these technologies should not felt and has interference with human activities [6]. Daily used household equipment like television, air conditioner, rice cooker etc. is very convergence, the used sensor, used hardware is very convergence and called a Things. Identified things and personalities should connect and communicate with the user, environment, and social user context [7].

The use of sensors and distribution of raw data is also increasing. The communication protocol is used to transmit, collect and process data [8]. Based on previous research, the used communication protocol is UDP [9]. UDP could perform data transmission at any device such as Host and Client. The Host could communicate with the Client with several rules. UDP is used because Smart Home environment does not need a large amount of data [10]. It also very efficient because does not need to validate data that is sent to the Host or the Client.

LabVIEW is used because it can easily connect with devices such as Arduino, microcontroller even computer networks [11] and could perform simulated input/output as well as real hardware [12]. At industrial words they have such as MyRIO, CompactRIO, and Elvis etc. to develop prototyped systems [13]. Pervasive computing sometimes is called by ubiquitous computing that has difficult task which is how to become an automatic identification from the distributed environment such as computation, communication and its process. It needs storage on the cloud where the data from either the Host or the Client being saved and visualized. One of the most cloud that free to save the data is dweet.io. Dweet.io is a free web service/cloud server that has facilitation to save simple data format online from every device that is connected [14]. Simpler data transfer format that used by dweet.io is JSON which include a timestamp, device ID and data value [15].

Previous research is conducted by designing and implementing UDP Pervasive on LabVIEW [9] and implementing on MyRIO [16] this research is originally extended from that previous system that communicate over cloud server. This system is designed by having a Host and a Client. The Host is designed by one device that could connect with many clients. The Client could communicate pervasively to the host. Our research is to measure the communication between Host, Client and dweet.io. Each of the Host and the Client will communicate with dweet.io by the internet and between Host and Client will be communicating locally so it can be measured how the service provided and are affected by the complexity of the network configuration and architecture.

2. RESEARCH METHOD

This section will explain about the communication design between the Client, the Host and dweet.io. Based on previous research, a pervasive discovery protocol is performed between Client and Host [9]. After both of Client and Host connected, they will communicate each other and both of them will send data to dweet.io. Each step by step state will be explained in this section, started from Client performing a broadcasted data until they can communicate. At communication state, a parallel process will be performing to send data to dweet.io and data format which is used is JSON. Then the communication between Client and Host; Client and dweet.io; and Host and dweet.io will be measured.

3.1. Network Design

The used standard design is from ETSI [17] M2M (Machine to Machine) area network domain and at this research that design will be extended to communicate via the internet. The pervasive service discovery protocol using UDP is shown in

Figure *1* which has Client and Host and several communication states. Start from the Client performing broadcast, until the broadcast are listened by the Host. Data which is sent during the broadcast process is Client Name, IP and Service that the Client has. After host listen the broadcast, the Host performs checking about device duplication. If the client is already known, it will give the same IP. But if the Client is not already known, the Host would give a new IP to the Client at Send ACK process. Send ACK process is a process that Host replying broadcast for the Client. A client receiving data from the host and the Client turn of broadcast process. After several processes completely performed, they will communicate which is performed by Data Transfer.

The data transfer process is performed periodically. The delay is set for 2 seconds. While the Host and the Client perform data transfer, both of them transfer the data to dweet.io. Parallel Process paradigm is used in this communication method. Its process will run if the communication success, while the Client send data to Host, at the same time the Client is also send data to a cloud server, the Host perform that process in the same way. One of any web service that provides free cloud server is dweet.io. Data is sent to dweet.io in JSON format. In

Figure *2* is an example of JSON data which is sent from Host to cloud server. That data contains data status explains with “this”, “getting” and “dweets” [18] explains data transmission status between success or not. On the data that is sent, contain “thing”, the “thing” is written by the thing names. The other is “created”, the “created” data contains date and time which indicated the time when the data is sent by "thing". The last structure is “content”, the “content” contains data sent or generated by "thing". The “date” format that sent by Host is milliseconds precision.

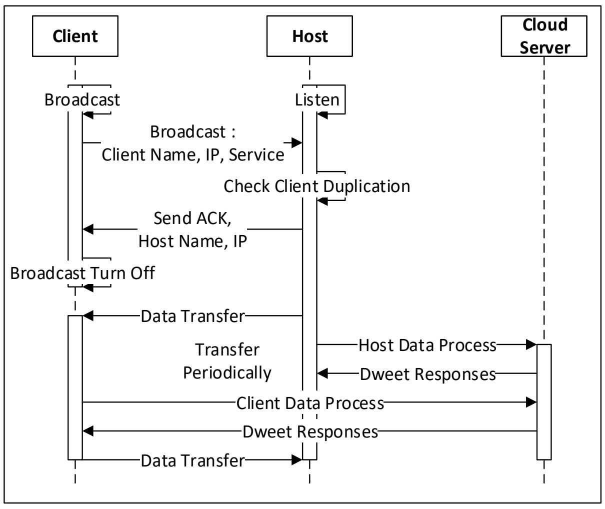
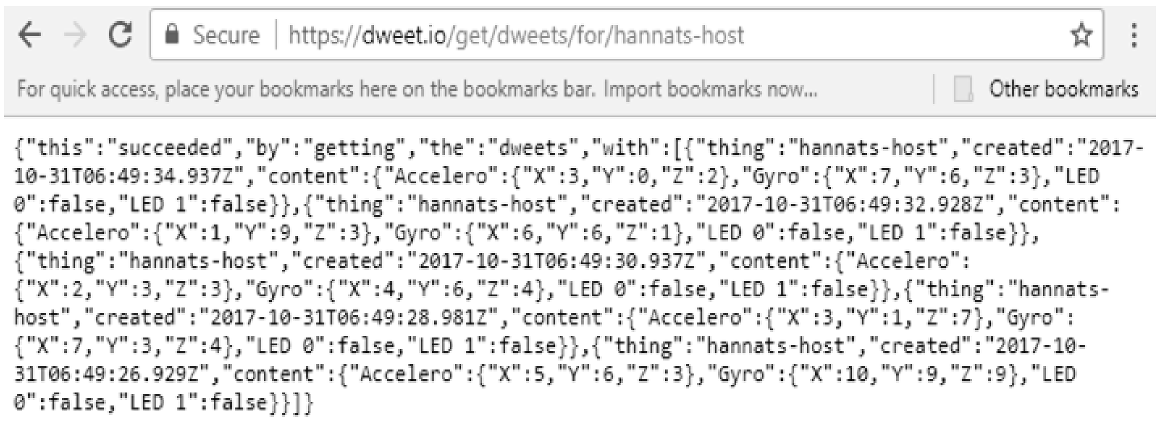


Figure 1. Machine to machine area network (Extended) *[17]*

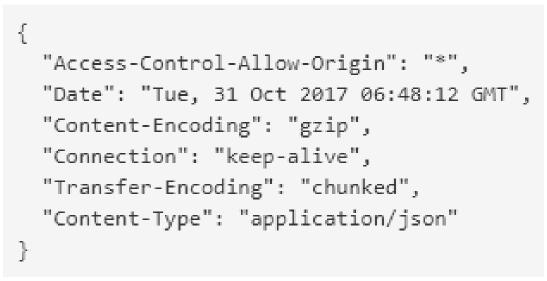

Secure
https://dweet.io/get/dweets/for/hannats-host
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...
Other bookmarks

{"this":"succeeded","by":"getting","the":"dweets","with":[{"thing":"hannats-host","created":"2017-10-31T06:49:34.937Z","content":{"Accelero":{"X":3,"Y":0,"Z":2},"Gyro":{"X":7,"Y":6,"Z":3},"LED 0":false,"LED 1":false}},{"thing":"hannats-host","created":"2017-10-31T06:49:32.928Z","content":{"Accelero":{"X":1,"Y":9,"Z":3},"Gyro":{"X":6,"Y":6,"Z":1},"LED 0":false,"LED 1":false}},{"thing":"hannats-host","created":"2017-10-31T06:49:30.937Z","content":{"Accelero":{"X":2,"Y":3,"Z":3},"Gyro":{"X":4,"Y":6,"Z":4},"LED 0":false,"LED 1":false}},{"thing":"hannats-host","created":"2017-10-31T06:49:28.981Z","content":{"Accelero":{"X":3,"Y":1,"Z":7},"Gyro":{"X":7,"Y":3,"Z":4},"LED 0":false,"LED 1":false}},{"thing":"hannats-host","created":"2017-10-31T06:49:26.929Z","content":{"Accelero":{"X":5,"Y":6,"Z":3},"Gyro":{"X":10,"Y":9,"Z":9},"LED 0":false,"LED 1":false}}]}

Figure 2. Data sent from host to dweet.io

After dweet.io received the data, they will send response data to Host. Data that is sent by dweet.io has JSON format too. The data contains "date", the "date" represent the date and time when dweet.io sent a response in

Figure *3*. When the Host disconnect with the Client, the entire process is stopped. The host will not send data to Client and cloud server, so does the Client. The host would be back on the broadcast process, the client back to listen process. If client have been detected, the Host will check duplication for the Client. If there is no duplication, Host will perform sent ACK etc. But if there is duplication client, the Host will clear his memory, the Host will perform broadcast again. The detail of the scheme is explained in next section.



```
{
  "Access-Control-Allow-Origin": "*",
  "Date": "Tue, 31 Oct 2017 06:48:12 GMT",
  "Content-Encoding": "gzip",
  "Connection": "keep-alive",
  "Transfer-Encoding": "chunked",
  "Content-Type": "application/json"
}
```

Figure 3. Dweet.io responses to host

In this research, network which is used between Host and Client is connected locally. But they will be doing communication to cloud server via internet. This research is focused to measure availability between Host to Client, Host to dweet.io and Client to dweet.io. Between the Hosts to Client it will measure their availability and their delay with milliseconds time precision.

3.2. Host and Client Design

This design on this section is the resulting design from previous research [9]. That design on

Figure *4* has already tested by functional testing scenario and works well. Between expected output and the real output, it matches. But the previous research doesn't measure the availability of host and client. At this research the design is extended so it has some feature which is:

1. Idle: this process is turning on the Host and performs opening UDP port. After port opened, the Host sending broadcast data.
2. Listening: at this state, the host listen is there any data returned from the client. If the data returned by the client, it goes to Check Device Duplication state. But if the host doesn't receive any data, it will stay at Wait for Broadcast State.
3. Check Device Duplication: this state is checking Client. Did the Client was known before, if the Client has not been connected to the Host, then the Host sending ACK and goes to ACK Sent state. If the client has been known before, the Host directly goes to Control Process state.
4. Send ACK: this state performing for checking Client hardware status. The process of what services are owned by the Client. ACK that is sent contains Host Name and Host IP.
5. Control Process: this state is when the Host performing control Client.
6. Dweet.io: this state is extending state from previous research. This state is working parallel with Control Process State when the Host connected with the Client. If the Client disconnected from the Host then this state stops working too.

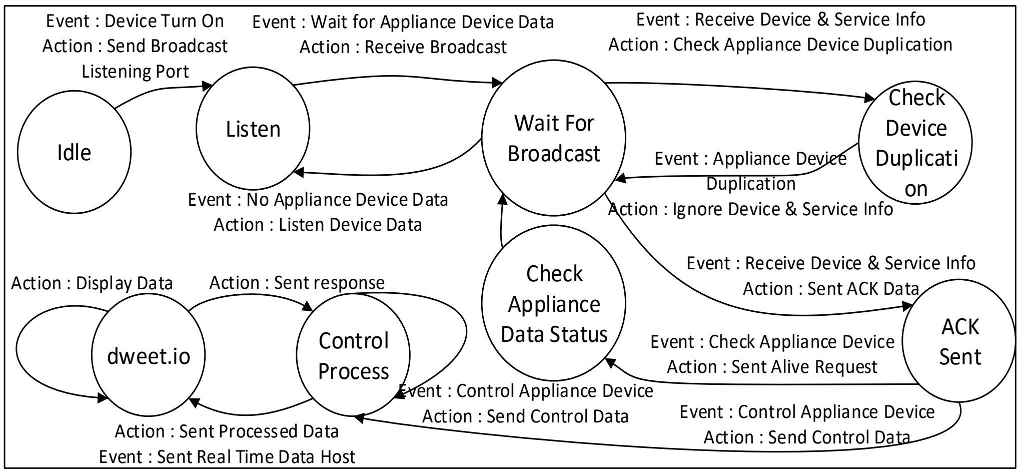

Event : Device Turn On
Action : Send Broadcast
Listening Port
Event : Wait for Appliance Device Data
Action : Receive Broadcast
Event : Receive Device & Service Info
Action : Check Appliance Device Duplication
Idle
Listen
Wait For Broadcast
Check Device Duplicati on
Event : Appliance Device Duplication
Action : Ignore Device & Service Info
Event : No Appliance Device Data
Action : Listen Device Data
Event : Receive Device & Service Info
Action : Sent ACK Data
Action : Display Data
Action : Sent response
dweet.io
Control Process
Check Appliance Data Status
ACK Sent
Event : Check Appliance Device
Action : Sent Alive Request
Event : Control Appliance Device
Action : Send Control Data
Event : Control Appliance Device
Action : Send Control Data
Action : Sent Processed Data
Event : Sent Real Time Data Host

Figure 4. Host state machine diagram

One cycle of processes is performed in number 1 until 5 by the Host is the entire process that described. Every cycle is performed to identify one Client. If there is more than one client, the other client should wait the previous cycle is completed. The 6th state, the dweet.io state is performed to send data that generated by the Host. The data contains a host name, dweet status, and data that received from the client.

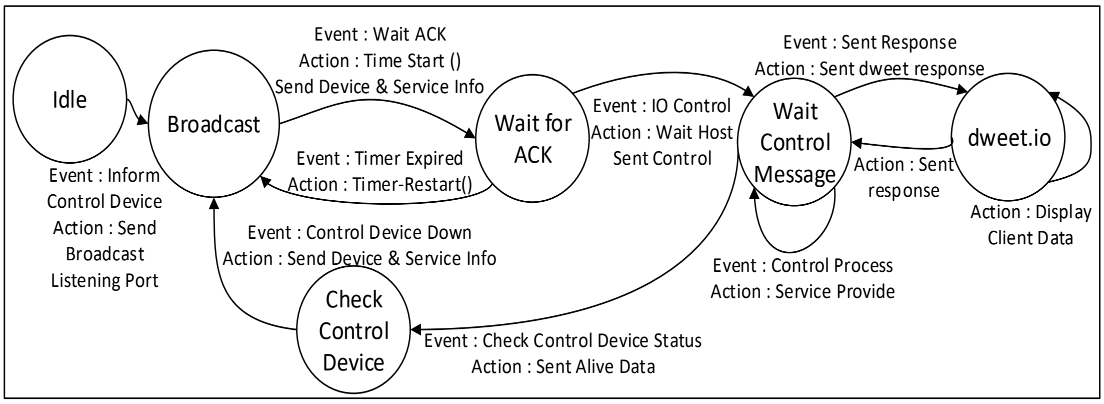


Figure 5. Client state machine diagram

Figure 5 representing state machine for the Client. The Client design is an extended design from previous research too. The Client has several states that specifically describes and already tested. The Client state which has been designed is:

1. Idle: this state was designed when the Client turned on. In this state, the client performs preparation process to perform broadcast data. The broadcasted data are Service Name, IP Client and Service Number.
2. Broadcast: in this state, the Client broadcast data for 250 milliseconds. Then it will move to wait for ACK State.
3. Wait for ACK: while the Client went to this state, the Broadcast process is turned off. If the Client receiving an ACK, then goes to Check Control Device state. But if the Client doesn't receive ACK after several seconds, the Client goes back to Broadcast State.
4. Check Control Device: this state is checking the services that owned by the Client itself. If the services ready to control, the Client sent data to the host about the Client status.
5. Wait for Control Message: in this state, the Client waiting for the Host to be controlled. If the Client has data that is not to be controlled such as gyro meter or accelerometer, the Client directly sent data to the Host. This state is performed when the Client communicates with the Host.
6. Dweet.io: as done by the Host, the Client will send data to dweet.io. This process is parallel with Wait Control state. If disconnected with the Host, the Client would not send data to dweet.io as a cloud server. Cloud server would send data status if they received data from the Client such as the Host do.

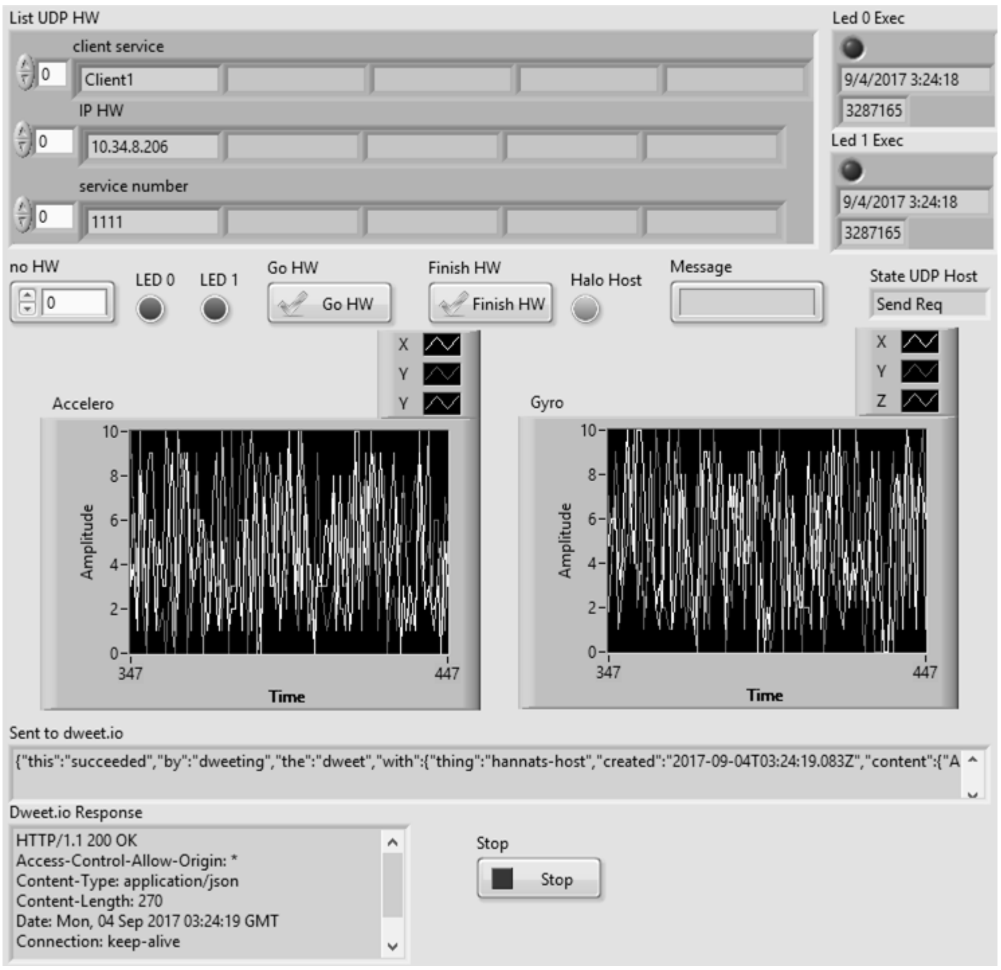
3. RESULTS AND TESTING

Implementation of the Host and the Client interface has been rebuild based on system design. There are several changes to user interface such as time, data which is sent to cloud server, cloud server responses etc. This section would be described the user interface that contains much information about system development (the Host and the Client) and contains system testing. System testing would be conducted to measure the availability of the systems between the Host, the Client and cloud server.

3.1. Result

Based on previous research and the extended methods that created in this research, interface implementation would be explained in this chapter.

Figure *6* represent Host user interface that has feature and function based on system design. The host has the ability to control more than one device at one time. Each feature which is owned by the Client could pervasively control and monitored by the Host.



```
client service: 0 | Client1
IP HW: 0 | 10.34.8.206
service number: 0 | 1111
Led 0 Exec: 9/4/2017 3:24:18 | 3287165
Led 1 Exec: 9/4/2017 3:24:18 | 3287165
no HW: 0
```

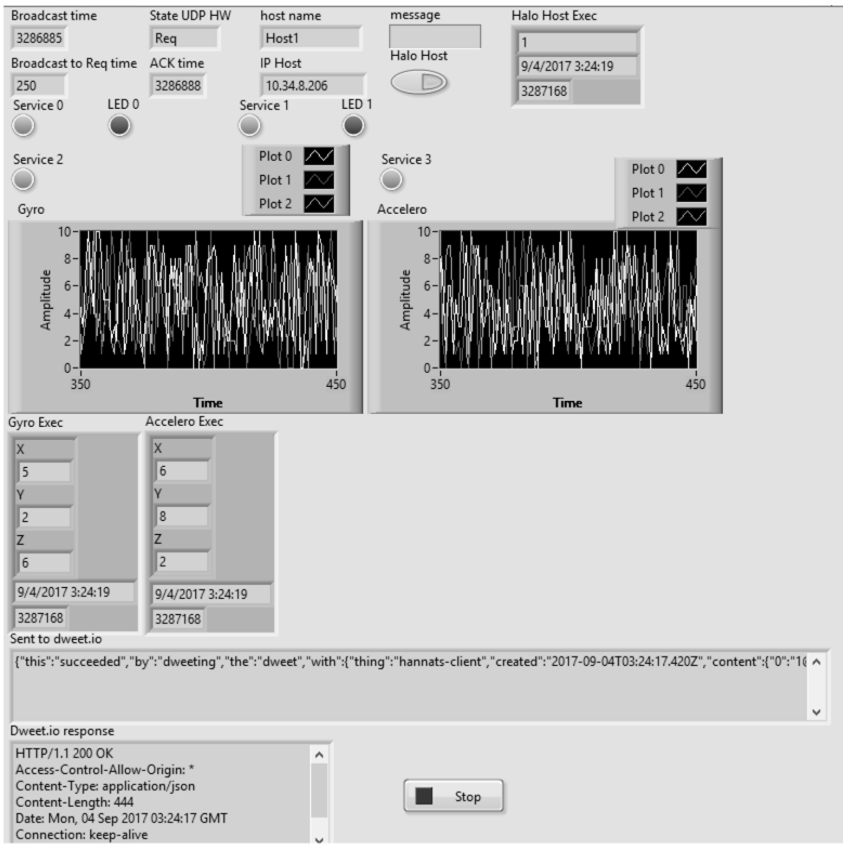
```
{"this":"succeeded","by":"dweeting","the":"dweet","with":{"thing":"hannats-host","created":"2017-09-04T03:24:19.083Z","content":{"A
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
Content-Length: 270
Date: Mon, 04 Sep 2017 03:24:19 GMT
Connection: keep-alive
```

Figure 6. The Host user interface

At this research, the feature that is owned by the Client represents by 2 LED, 1 accelerometer sensor and 1 gyroscope sensor. If the client doesn't have one of several sensors, the Client service would not appear on the Host interface. To measure the availability of the Host, it provided the time that represents when data is executed and sent from the client. To measure communication time of cloud server, it provided two text boxes. First text box "sent to dweet.io" is data that sent by the Host to cloud server which contains the Host time. "Dweet.io Response" is responses from cloud sensor that contain time when data is sent by cloud server. Accelerometer and gyroscope sensor is represented by waveform chart that generated by the Client. At the Host, the process of sending data was done in the parallel process while sending data to the Client and Dweet.io.

Figure *7* is Client user interface that has feature and function based on system design and is extended from previous research. The Client provided services that could monitor and controlled by the Host and could receive a message from the Host. At this research, the client has 2 LED, 1 generated XYZ signal for each gyroscope and accelerometer sensors. Such as the Host, the Client has "sent to dweet.io" feature that indicates the Client could communicate with a cloud server. The message which is sent to cloud server are same with the message that sent to the Host that works parallels too. Then "Dweet.io response" is a function that sent by cloud server to the Client. So it can measure communication time between the Clients with a cloud server. "Halo Host exec" is used to display message from the host, it contains time that is sent by the Host that used to measure communication delay. So it can measure availability between the Host – the Client, the Host – cloud server and the Client – cloud server.



Broadcast time: 3286885
State UDP HW: Req
host name: Host1
message:
Halo Host Exec: 1 / 9/4/2017 3:24:19 / 3287168
Broadcast to Req time: 250
ACK time: 3286888
IP Host: 10.34.8.206

Gyro Exec: X 5, Y 2, Z 6, 9/4/2017 3:24:19, 3287168
Accelero Exec: X 6, Y 8, Z 2, 9/4/2017 3:24:19, 3287168

Sent to dweet.io

```
{"this":"succeeded","by":"dweeting","the":"dweet","with":{"thing":"hannats-client","created":"2017-09-04T03:24:17.420Z","content":{"0":"1(
```

Dweet.io response

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
Content-Length: 444
Date: Mon, 04 Sep 2017 03:24:17 GMT
Connection: keep-alive
```

Figure 7. The Client user interface

3.2. Testing

This research conducted by two experiment scenario that all testing scenario ignore network condition. The first experiment scenario conducted by the Client sent to each Host and Cloud server. A data transfer that are measured is 15 (fifteen) times and each data conducted by different data sizes. Based on first experiment scenario, the average amount of data sent by the Client is 436.47 bytes, the biggest data is 453 bytes and the smallest data that is produced by the client is 421 bytes. The data that sent to the Host has an average delay 14.33 milliseconds which has maximum time 17 milliseconds and the smallest time that received by the host is 11 milliseconds. The data that is sent to Cloud Server is 1.093 second which the maximum time is 1.336 second and the smallest time that receives by Cloud server is 0.785 second that could be seen at Table 1.

The second testing scenario is conducted by the Host sent data to each Client and Cloud Server, data transfer that is measured are 15 (fifteen) times and each data conducted by different data sizes. The data that produced by the Host have an average 270.93 bytes with the biggest data size is 173 bytes and the smallest data size is 268 bytes

Table 1. Client Sent data to Host and to Cloud Server

No	Client to Data Size (bytes)	Host Delay (Milliseconds)	Cloud Server Delay (Second)
1	440	12	0.785
2	438	14	1.209
3	436	13	1.199
4	435	12	1.206
5	453	17	1.106
6	440	16	1.186
7	452	11	0.816
8	447	15	1.054
9	440	14	1.186
10	431	13	1.181
11	433	15	1.193

No	Client to Data Size (bytes)	Host Delay (Milliseconds)	Cloud Server Delay (Second)
12	430	15	1.229
13	427	15	1.264
14	424	16	1.301
15	421	17	1.336
Average	436.47	14.33	1.093

The data that received by the Client have an average delay is 64.6 milliseconds with the biggest delay is 81 second and the smallest delay is 49 milliseconds. The data that received by Cloud server has a delay too with average 0.869 seconds, the biggest delay is 0.881 second and the smallest delay is 0.84 second that could see in Table 2.

Table 2. Host Sent Data to The Client and Cloud Server

No	Host to Data Size (bytes)	Client Delay (Milliseconds)	Cloud Server Delay (Second)
1	270	63	0.852
2	269	49	0.949
3	269	51	0.861
4	271	56	0.863
5	270	59	0.881
6	269	61	0.868
7	268	55	0.935
8	271	65	0.829
9	274	60	0.872
10	269	72	0.874
11	272	70	0.861
12	272	73	0.855
13	273	76	0.85
14	273	78	0.845
15	274	81	0.84
Average	270.93	64.6	0.869

4. CONCLUSION AND FUTURE WORK

This research is successfully developed UDP Pervasive service and discovery protocol using LabVIEW. This research was conducted by one Host and one Client, each of them could communicate pervasively. When the Host and the Client communicate, each of them has parallel communication with the Cloud Server. The experience scenario conducted to measure availability between Host, Client and Cloud Server that ignore the network condition. Between two testing scenario it can be seen that the delay in data transmission between the Host to the Client (average delay 14.33 millisecond) bigger that the Client to the Host (average 64.6 millisecond) even though the data that sent is smaller. Average data size that sent by client is 436 bytes and average data size that the Host sent 270.93 bytes. This is due to when the Host sent data to the Client, the Client would accomplish his task first then receiving data from the Host. The state machine design take effect on the delay that resulting from the Client and the Host.

The Client sent data to Host, the data contains data sensor which simulated by gyroscope sensor, accelerometer sensor and LED. The Host sent data to Host contains it too which simulated by LED. Each experiment scenario is successfully sent each service that sent to Cloud Server too. Cloud Server just stored data that sent them. This research could continue with bigger and ubiquitous environment. The pervasive discovery protocol using UDP could be implemented in several embedded device such as Arduino, raspberry or MyRIO. The embedded device could represent household equipment that could be installed such as smart device. So the user would not cause to be experienced with the existence of smart device at daily household equipment.

ACKNOWLEDGEMENTS

The team thanks to the Computer System and Robotics Laboratory, Computer Engineering Department, Faculty of Computer Science, Brawijaya University. This paper is an extension of work originally published and reported in “IAES 2016: International Conference on Electrical Engineering, Computer Science and Informatics (EECSI 2016)” at Semarang, East Java, Indonesia 22-25 November 2016, with the title “Lightweight UDP Pervasive Protocol in Smart Home Environment based on LabVIEW”.

REFERENCES

[1] S.R. Akbar, W. Kurniawan, M.H.H. Ichsan, I. Arwani, M.T. Handono, "Pervasive Device and Service Discovery Protocol In XBee Sensor Network," in *ICACSIS*, Malang, Indonesia, 2016.

[2] J. Froehlich, L. Findlater and J. Landay, "The Design of Eco-Feedback Technologu," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Atlanta, Georgia, USA, 2010.

[3] P. Dourish and G. Bell, Divining A Digital Future - Mess and Mythology in Ubiquitous Computing, London, England: The MIT Press, 2011.

[4] L. Atzori, A. Iera and G. Morabito, ""The Internet of Things: A survey"," *Elsevier Computer Networks,* pp. 2787-2805, 2010.

[5] O. Vermesan and P. Friess, Internet of Things - Global Technological and Societal Trends, Aalborg, Denmark: River Publisher, 2011.

[6] J. Ma, T.L. Yang, B.O. Apduhan, R. Huang, L. Barolli and M. Takizawa, "Towards a Smart World and Ubiquitous Intelligence: A Walkthrough from Smart Things to Smart Hyperspaces and UbicKids," *International Journal of Pervasive Computing and Communications,* vol. 1, no. 1, pp. 53-68, 2005.

[7] C. Aggarwal, N. Ashish and A. Sheth, The Internet of Things: A Survey from The Data-Centric Perspective, Book Chapter in "Managing and Mining Sensor Data", Springer, 2013.

[8] G. Tripathi, D. Singh and K. K. Loo, "EOI: Entity of Interest Based Network Fusion for Future Services," in *ICHIT 2011: Convergence and Hybrid Information Technology*, 2011.

[9] W. Kurniawan, M.H.H. Ichsan, S. R. Akbar and I. Arwani, "Lightweight UDP Pervasive Protocol in Smart Home Environment Based on Labview," in *IAES International Conference on Electrical Engineering, Computer Science and Informatics*, Semarang, Indonesia, 2016.

[10] A. Ford, C. Raiciu, M. Handley and O. Bonaventure, TCP Extensions for Multipath Operation with Multiple Addresses, RFC 6824, 2013.

[11] F.J. Jimenez and D.J. Frutos, "Virtual Instrument for Measurement, Processing Data, and Visualization of Vibration Patterns of Piezoelectric Devices," *Elsevier,* vol. 27, no. 6, pp. 653-663, 2005.

[12] M.H.H Ichsan, W. Kurniawan and M. Huda, "Water Quality Monitoring with Fuzzy Logic Control Based on Graphical Programming," *TELKOMNIKA,* vol. 14, no. 4, p. 1446~1453, 2016.

[13] L. Vanfretti, V. H. Aarstrand, M. S. Almas, V. S. Perić and J. O. Gjerde, "A software development toolkit for real-time synchrophasor applications," in *IEEE Grenoble Conference,* Grenoble, 2013.

[14] K. Fysarakis and S. Ntamapiras, "Hardware implementation of a system classifying the optoacoustic signature of insect wing flap," in *Conference: 22nd International Congress on Sound and Vibration (ICSV22),* Florence, Italy, 2015.

[15] A. Halldorsson, "IIoT data collection for OEE measurements," Reykjavík University Library, 2016.

[16] W. Kurniawan, M.H.H. Ichsan and S. R. Akbar, "UDP Pervasive Protocol Implementation for Smart Home Environment on MyRIO using LabVIEW," *International Journal of Electrical and Computer Engineering (IJECE),* vol. 8, no. 1, pp. 113-123, 2018.

[17] e. a. J. Holler, "From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence," in *Elsevier*, Waltham, MA, 2014.

[18] I. Bug Labs, "dweet.io," 2017. [Online]. Available: http://dweet.io/. [Accessed 1 11 2017].