Smart Cluster Housing Monitoring with ESP32, ESP-Mesh and Django

DOI:10.36909/jer.ASSEEE.16099

Mochamad Fajar Wicaksono*[a], Myrna Dwi Rahmatya[b]

[a]Program Studi Teknik Komputer, Universitas Komputer Indonesia,

[b]Program Studi Manajemen Informatika, Universitas Komputer Indonesia,

Jl. Dipati Ukur 112-116 Bandung, Indonesia.

*Email: mfajarw@email.unikom.ac.id; Corresponding Author.

ABSTRACT

The purpose of this study is to create a monitoring system for cluster housing. The method used in this research was experimental. Monitoring carried out in this system was installed and tested on ten houses. This system used an ESP-MESH for communication between ESP32 boards in residential areas. The process of sending data to the online server used two ESP32 boards that function as a mesh server and a gateway for data transmission. The data sent is in the form of data from ultrasonic sensors, PIR1 and PIR2 sensors, Flame sensors, Gas sensors, and Photos. The photo result was taking by ESP32CAM when the system detects an intruder. When something bad happens, such as fire detection, intruders, and gas leaks, residents will receive text and image notifications through the LINE application. The head of security will also get notifications via the web application. The web application is built using the Django framework. The results of this study are a monitoring system for cluster housing using ESP32, ESP-MESH, and Django. Based on the test results, this system has been running well by the initial objectives.

Keywords: ESP32, Mesh Protocol, Monitoring, Line, Django, Cluster Housing.

INTRODUCTION

This paper describes a system built for monitoring cluster housing. This system aims to create a cluster housing monitoring system using the ESP-Mesh protocol and IoT. Each house will be monitored centrally on the server, independently in each user account, and LINE messaging application. The main idea of the research in this paper is a smart home with IoT. Lots of related research that addresses similar themes. In 2016, a study made smart home monitoring with IoT and low power wireless using the ZigBee module (Kalaivanan et al., 2016). However, this research only focuses on one house and requires many controller modules and ZigBee modules, so it is less economical. Another related research is research on the smart home using NodeMCU ESP8266 in 2017 (Wicaksono, M. F., 2017.).

However, there was no notification via the application, and there was no relationship with the local security chief. In 2018 there was research related to this topic (Yuen, M. et al., 2018). However, this research only focuses on an application in one house even though the system already uses a server. Other research in 2018 has also made smart homes (Jadhav, L. et al., 2018). However, the research only focused on gas leaks and fires for one house, and it was clear that the network system was centralized. Besides, this study uses an Intel Galileo board which is relatively expensive for the system. In 2020 there is research using Arduino and ESP32CAM (Wicaksono, M. F., et al., 2020). However, in this study, the controller board is very dependent on an internet connection or must be completed online. So, if the tool is reproduced, it will require a large internet connection. Primarily, the two studies only focused on one house.

In general, the above studies only specifically monitor one house, and if it is to be developed or applied to many houses, there will likely be problems with wireless connection. Because in previous studies using traditional WiFi networks. In a traditional WiFi network connection, the system is centralized to an access point (AP). Meanwhile, the AP has a limited range, so that it becomes a limitation for stations that can be connected. Besides, traditional Wi-FI networks are prone to overloading due to the capacity limitations of the AP (Espressif Company, 2021).

In this research, a cluster housing monitoring system will be built using the ESP-MESH protocol for ESP32. Monitoring carried out in this system was installed and tested on ten houses. ESP-MESH is a network protocol built on the Wi-Fi protocol. ESP-MESH allows us to connect many nodes both inside and outside the room connected in one WLAN network. ESP-MESH has the advantage of being able to organize itself, capable of self-healing so that networks can be built and maintained independently (Guo, Z. et al., 2021). The meaning of self-healing here is that when one node cannot be connected, other nodes can still be connected through another intermediary node so that the network does not just disconnect (Lech, P. et al., 2017). By utilizing ESP-MESH, each controller board (ESP32) in each house is connected to receive and send data to the server. In other words, all controllers are connected to WLAN. Two ESP32s are used to send data to the server online. One serves as a mesh server, and the other serves as a gateway that is responsible for sending data to the server online. The result is that each house will be monitored centrally on the server, independently in each user account, and can send messages via the LINE messaging application by utilizing the LINE API.

METHOD

This research used experimental research methods. In this method, various designs and experiments were carried out directly based on theoretical studies from various literature to obtain the expected research results. The system block diagram is shown in Figure 1.

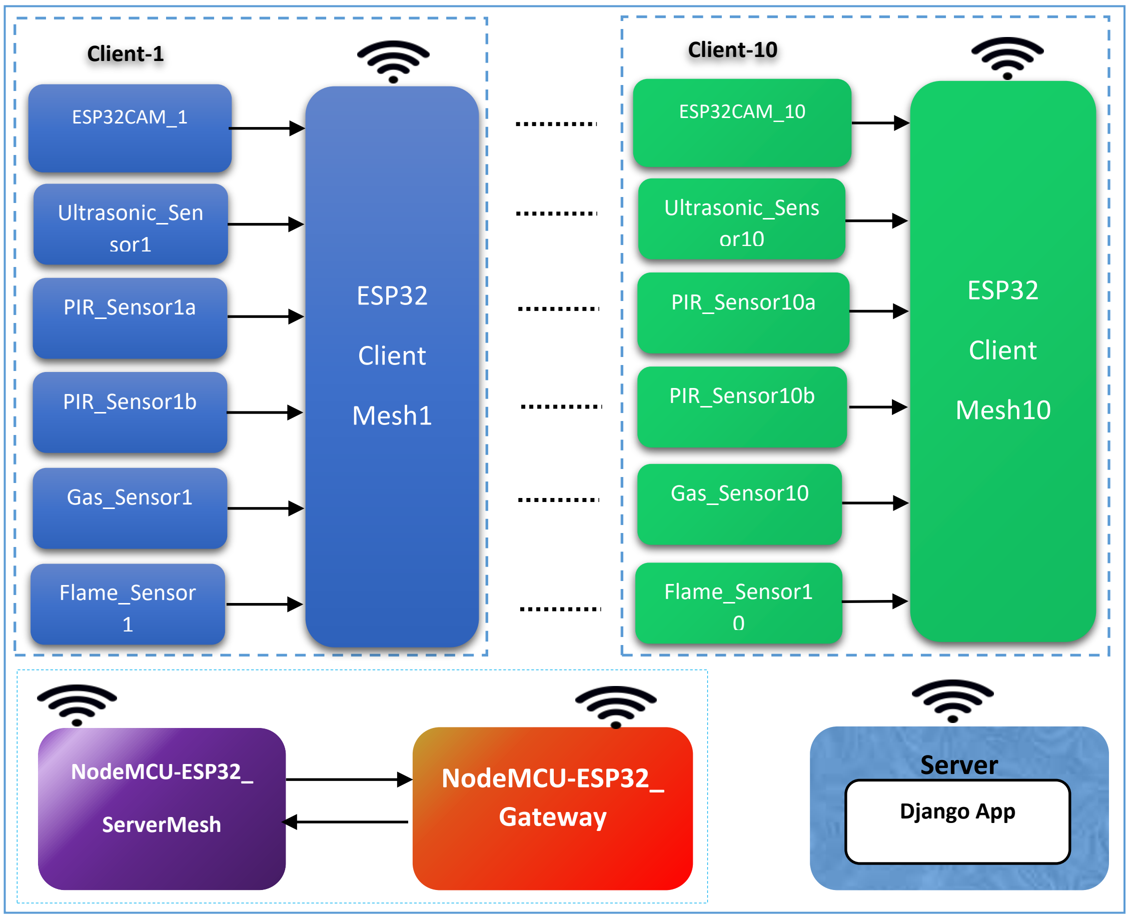


Figure 1 Block Diagram SCHMV1.0

This system is divided into three main parts: Server Mesh with Gateway, Client Mesh, and Server. All board controllers used in this study are NodeMCU ESP32. The ESP32 is an SoC equipped with various features, such as Wi-Fi (2.4GHz band), Bluetooth, dual-core with two 32-bit Tensilica Xtensa LX6 microprocessors, and multi peripherals (Mandanka, K. et al., 2017).

Client Mesh is each ESP32 NodeMCU board (Client1-Client10) and all the sensors and modules connected to it. In this study, 10 Client Mesh was made. In other words, the system created was applied to 10 houses.

Each Client Mesh is connected to two PIR sensors. This sensor is used to detect human movement (Saleh, S.B. et al., 2018). This study's type of sensor module was the HC-SR501 which has a detection distance of up to 7 meters and a detection angle of 120 degrees (Sysala, T. et al., 2017). The delay for each detection in this module can be adjusted from a range of 2 seconds to 4 seconds (Firdaus, R. et al., 2018). Two HC-SR501s are used in this system. The two sensors are placed on the right and left of the front of the house. The next sensor connected is the ultrasonic sensor. The type of sensor module used HC-SR04, the sensor is placed in the center of the house. The ultrasonic sensor will emit ultrasonic waves with a frequency of 40KHz (Zaw, H.N. et al., 2018). This sensor module has a measuring distance between 2cm to 400cm (Mutinda, M. et al., 2020). If one of the PIR sensors or both PIR sensors detect movement and the distance detected by the ultrasonic sensor is less than 200, the ESP32 NodeMCU will send a command to the ESP32CAM module to take photos. ESP32 CAM is a board built on the ESP32 (Mandanka, K. et al., 2017). The ESP32 CAM board is equipped with an OV2640 camera and a ten-pin GPIO that can be used to connect this board with other peripherals (Priyanka et al., 2020). The resulting photo will be sent in base64 format.

Furthermore, the Client Mesh also has a gas sensor. Where in this study, the MQ2 gas sensor module is used. This module is commonly used to detect leakage of LPG gas, isobutane (C4H10), propane (C3H8), methane (CH4), ethanol alcohol, hydrogen (H), smoke) (Heyasa, B.B.L., et al., 2017). This sensor module has high sensitivity and a very fast response time (Ansary, M. et al., 2018). In this study, the MQ-2 module was placed in the kitchen to detect LPG leaks. The output of this module is an analog value.

Furthermore, the flame sensor is used to detect fires. The sensor module used in this study is the KY-026 module. This flame sensor is built with an IR photodiode (Ansary, M. et al., 2018). This sensor module can detect infrared waves from 760nm to 11000nm (Zaw, H.N. et al., 2018). In this study, the KY-026 module was placed in the kitchen to detect fires. The output of this module is a digital value.

The Mesh Server functions to broadcast his id and receive all data from the client (client mesh id-data, PIR1, and PIR2 sensors, ultrasonic sensors, gas sensors, flame sensors, and photos from ESP32CAM) from each Client Mesh then forward the data to the Gateway serially. The data received by the Gateway will then be sent to the server online.

On the server-side, there is a web application that will receive and process the data. The web application used in this research was built using the Django framework, which used the Python programming language. Django itself uses the MVT architecture (model, view, template) (Vamsi, K.M. et al., 2021). In other words, by using Django, we can combine the back-end and front-end. It allows us to write python code on the HTML page (Jiaying, L. et al., 2018). It also makes Django very efficient (Knaflic, C.N., 2018). In this web application, there is a special account for the head of security and a special account for each user (homeowner). When something goes wrong in one of the houses, the app will provide security and user feedback. The feedback to the head of security is in text and images and sirens on the server. Meanwhile, feedback for users is in the form of text messages through the LINE application.

The next part is the software design from the hardware side, which includes Server Mesh, Gateway, and Client Mesh (Client1-Client10). The flowchart for Client Mesh is shown in Figure 2.

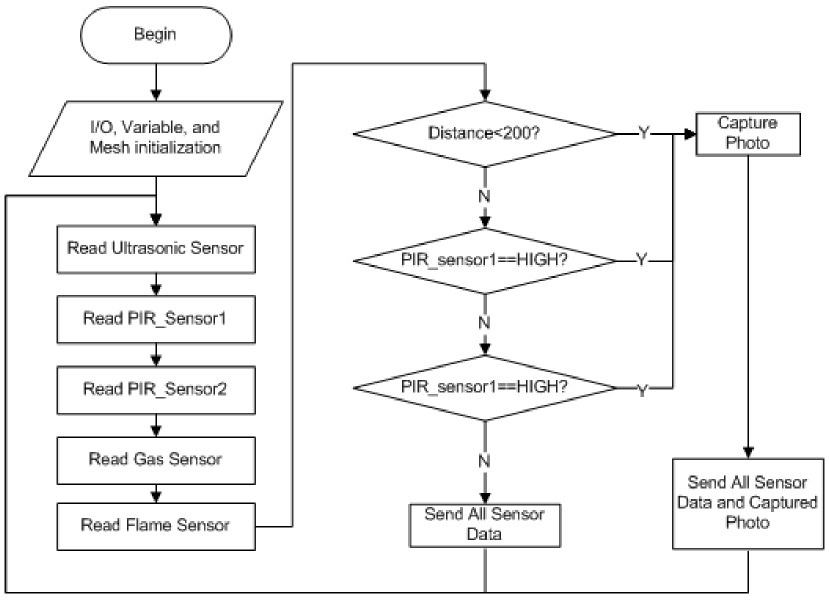


Figure 2 Flowchart Server Mesh

From Figure 2, it can be broadly explained that after the initialization process, Client Mesh will read all sensors (Ultrasonic, PIR_sensor1, PIR_sensor2, Gas, and Flame). If the distance is 200cm smaller, Client Mesh will take photos using ESP32CAM. Furthermore, if PIR_sensor1 or PIR_sensor2 is HIGH, then Client Mesh will take a photo using ESP32CAM. However, if the distance is greater than 200, then PIR_sensor1 and PIR_sensor2 are LOW, then Client Mesh will not take the photo. Finally, the Mesh Client will send all of this data to the Mesh Server. Next is an overview of the process flow that occurs in Server Mesh and Gateway. Figure 3 shows the flowchart for the Mesh Server and Gateway.

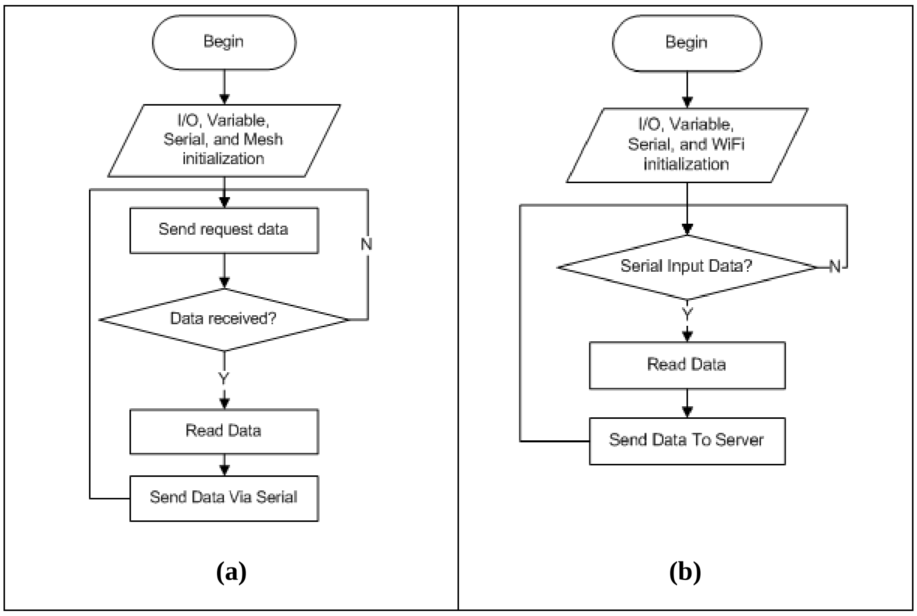


Figure 3 Flowchart (a) Server Mesh and (b) Gateway

Based on Figure 3, it can be explained that EPS32, which acts as Server Mesh, will broadcast its ID to all Client Mesh regarding data requests. If there is data coming from the Client Mesh, the data will be processed and forwarded to the gateway using serial communication. ESP32, which acts as a gateway, will continue to wait for serial port data. If there is incoming data, the data will be processed and sent to the server.

On the server, there is a web application built using the Django framework. Data received from the gateway will be saved to the database. Furthermore, the data will be processed again to be displayed on the user's page. It provides feedback according to the conditions that occur either through the web page or through the LINE application. In this web application, the head of security and users can log into the application and see their profile and home conditions based on sensor values and information displayed on the web page.

RESULTS AND DISCUSSION

The tests carried out are divided into two major parts, namely hardware testing and software testing. Hardware testing includes testing related to Server Mesh, Gateway, and all Client Mesh. The first test is a test that determines whether the output of all sensors and camera modules can be read by Client Mesh. The test results are shown in Table 1.

Table 1 Client Mesh functionality test

Client Mesh	**Sensors**					
	Ultrasonic	**PIR_ sensor1**	**PIR_ sensor2**	**Gas**	**Flame**	**ESP32CAM**
1	100%	100%	100%	100%	100%	100%
2	100%	100%	100%	100%	100%	100%
3	100%	100%	100%	100%	100%	100%
4	100%	100%	100%	100%	100%	100%
5	100%	100%	100%	100%	100%	100%
6	100%	100%	100%	100%	100%	100%
7	100%	100%	100%	100%	100%	100%
8	100%	100%	100%	100%	100%	100%
9	100%	100%	100%	100%	100%	100%
10	100%	100%	100%	100%	100%	100%

Functionality tests related to reading all sensors and modules in each Client Mesh are carried out ten times each. Each data field in table 1 is a summary of 10 tests performed. Each Client Mesh successfully reads data from each sensor and module connected to it based on the test results. The next test is testing sensor data and images from each Client Mesh to the Mesh Server. The results of this data transmission test are shown in Figure 4.

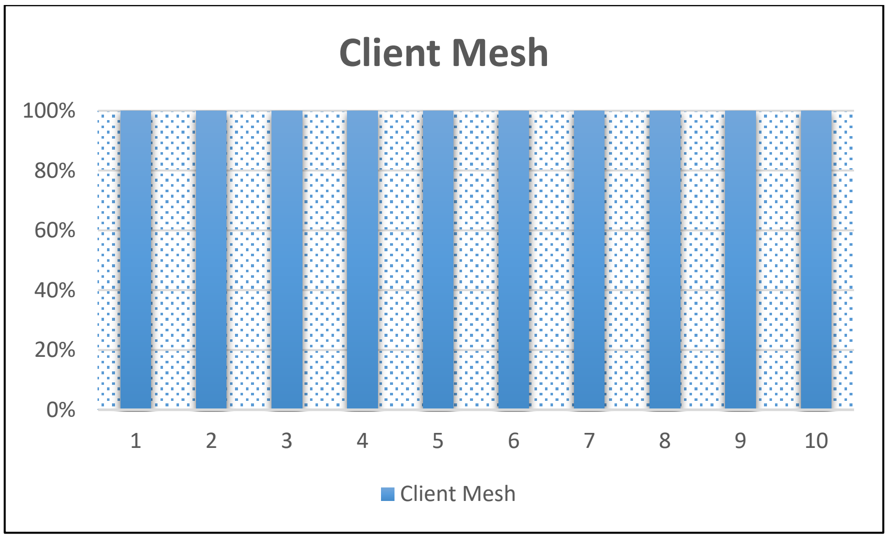


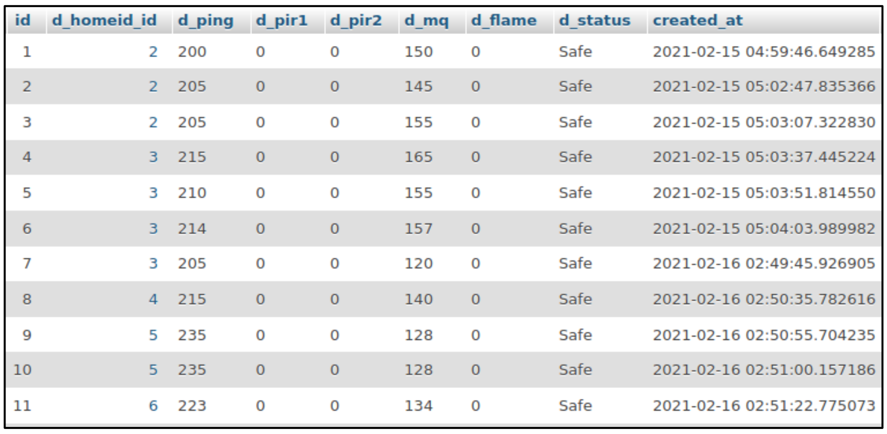
Figure 4 The percentage of test results for sending data

Testing of sending data from each Client Mesh is carried out ten times each. The 100% figure shows that out of 10 attempts, all of them are well received on the Mesh Server. The next test tests the delivery from Mesh Server to Gateway to Server and stored in the database. The test results are shown in Table 2.

Table 2 Test results of sending and storing data to the server

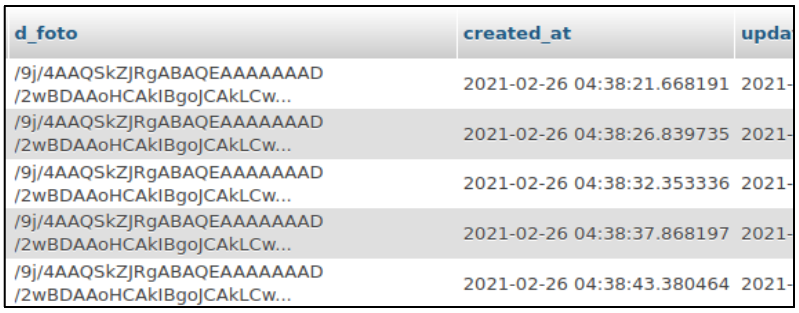
No	**Mesh Server (send)**	**Gateway (received and send to server)**	**Database (saved)**
1	√	√	saved
2	√	√	saved
3	√	√	saved
4	√	√	saved
5	√	√	saved
6	√	√	saved
7	√	√	saved
8	√	√	saved
9	√	√	saved
10	√	√	saved

Data transmission from the Mesh Server to the Gateway is done using serial communication. Meanwhile, data transmission from the gateway to the server is done wirelessly using an internet connection. Table 1 shows that all data was successfully received and successfully saved to the database. Figures 5 and 6 show examples of data that were successfully stored in the database.



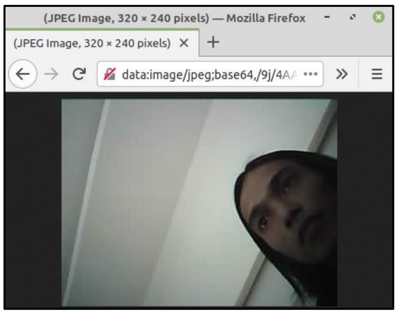
id	d_homeid_id	d_ping	d_pir1	d_pir2	d_mq	d_flame	d_status	created_at
1	2	200	0	0	150	0	Safe	2021-02-15 04:59:46.649285
2	2	205	0	0	145	0	Safe	2021-02-15 05:02:47.835366
3	2	205	0	0	155	0	Safe	2021-02-15 05:03:07.322830
4	3	215	0	0	165	0	Safe	2021-02-15 05:03:37.445224
5	3	210	0	0	155	0	Safe	2021-02-15 05:03:51.814550
6	3	214	0	0	157	0	Safe	2021-02-15 05:04:03.989982
7	3	205	0	0	120	0	Safe	2021-02-16 02:49:45.926905
8	4	215	0	0	140	0	Safe	2021-02-16 02:50:35.782616
9	5	235	0	0	128	0	Safe	2021-02-16 02:50:55.704235
10	5	235	0	0	128	0	Safe	2021-02-16 02:51:00.157186
11	6	223	0	0	134	0	Safe	2021-02-16 02:51:22.775073

Figure 5 Sensor data that has been saved to the database



d_foto	created_at	upda
/9j/4AAQSkZJRgABAQEAAAAAAAD /2wBDAAoHCAkIBgoJCAkLCw...	2021-02-26 04:38:21.668191	2021-
/9j/4AAQSkZJRgABAQEAAAAAAAD /2wBDAAoHCAkIBgoJCAkLCw...	2021-02-26 04:38:26.839735	2021-
/9j/4AAQSkZJRgABAQEAAAAAAAD /2wBDAAoHCAkIBgoJCAkLCw...	2021-02-26 04:38:32.353336	2021-
/9j/4AAQSkZJRgABAQEAAAAAAAD /2wBDAAoHCAkIBgoJCAkLCw...	2021-02-26 04:38:37.868197	2021-
/9j/4AAQSkZJRgABAQEAAAAAAAD /2wBDAAoHCAkIBgoJCAkLCw...	2021-02-26 04:38:43.380464	2021-

(a)



(JPEG Image, 320 × 240 pixels) — Mozilla Firefox

(b)

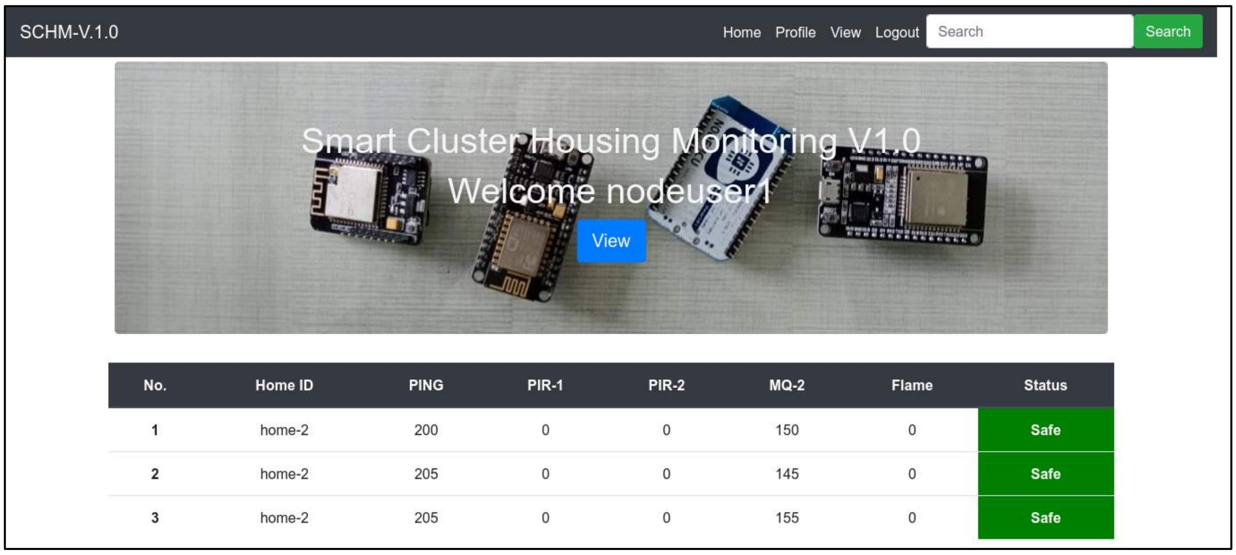
Figure 6. Image data (a) base64 type (b) testing on browser

Image quality taken by using this ESP32CAM is QVGA 320x240px. The image data shown in Figure 6a is stored in base64, while Figure 6b shows the string test into the browser to show that the received string is not truncated and can be converted back into an image.

Next is web application testing. This test includes the login process, user page views and monitoring pages, and application notifications when something bad happens. Before using the smart cluster housing monitoring system, the homeowner or user must log in first by entering a username and password. If the user enters the wrong username or password, a login failure notification will appear. If the entered username and password match, the user will enter the homepage. Several menus on the homepage that the user can access include the profile and view.

The user can see the username, full name, age, house code, address, and last login data on the profile page. The user cannot change these data, and the admin can only change this data.

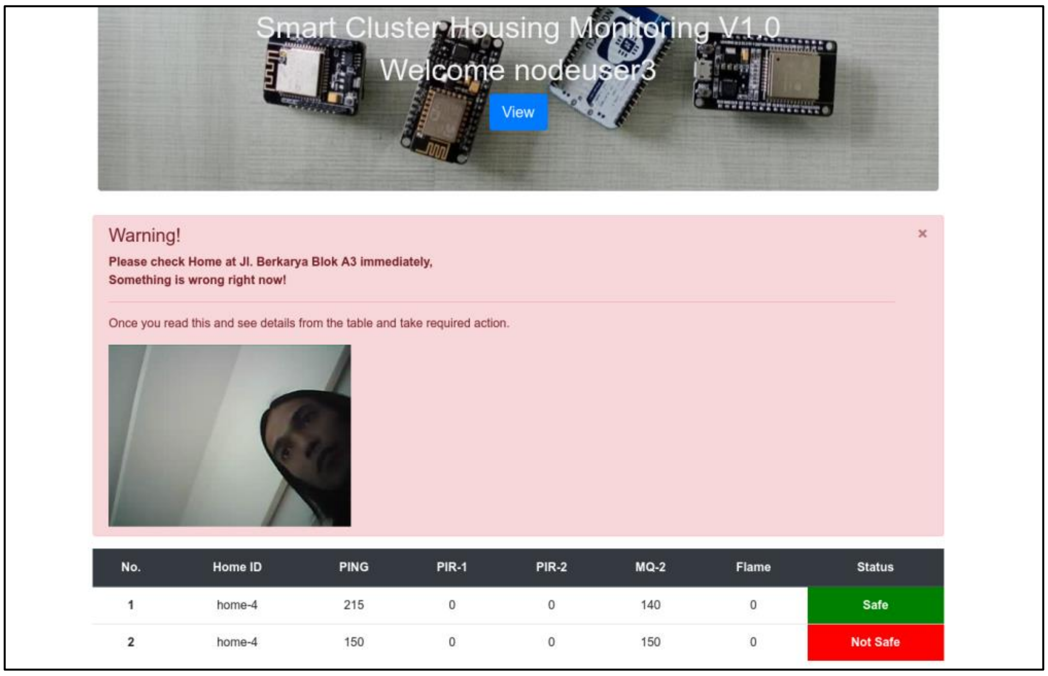
On the view page, users can see sensor data that represents the condition of their house. Figure 7 shows five sensors: PING, PIR-1, PIR-2, MQ-2, and Flame, displaying the sensor readings. The status shows the conclusion of the readings from the five sensors. Safe status indicates that the house is in a safe condition. Meanwhile, the head of security can also monitor the condition of each house through a similar page.



No.	Home ID	PING	PIR-1	PIR-2	MQ-2	Flame	Status
1	home-2	200	0	0	150	0	Safe
2	home-2	205	0	0	145	0	Safe
3	home-2	205	0	0	155	0	Safe

Figure 7 View page

However, if one of the sensors produces a value outside the specified limit, the web application will immediately change the status to Not Safe. If the status shows Not Safe, the user will get a notification on the view page, as shown in Figure 8. In Figure 8, the bottom view of the house is not safe because there are intruders. The head of security will also see the same notification accompanied by a siren sound.



Warning!

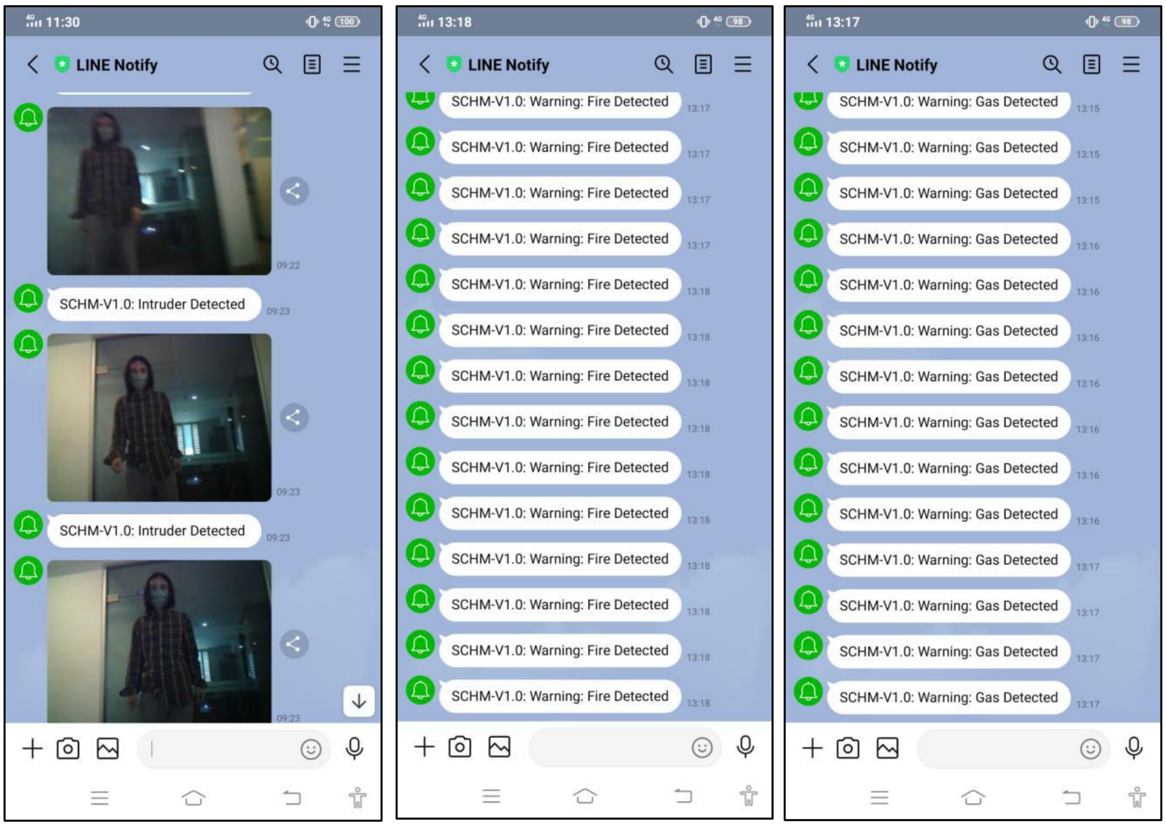
Please check Home at Jl. Berkarya Blok A3 immediately,
Something is wrong right now!

Once you read this and see details from the table and take required action.

No.	Home ID	PING	PIR-1	PIR-2	MQ-2	Flame	Status
1	home-4	215	0	0	140	0	Safe
2	home-4	150	0	0	150	0	Not Safe

Figure 8 Notification when there is an intruder

In addition to getting notifications on the view page, users will get notifications in the form of messages via LINE. When an intruder is detected, the householder will get a message in a warning text and intruder’s picture. When there is a gas leak or fire, the residents will get a text message. The text message will continue to be obtained as long as the sensor detects a gas leak or fire. The web application automatically sends all notifications. Figure 9. shown messages can be in the form of images if there is an intruder, notification if a fire is detected, and gas leaks.



SCHM-V1.0: Intruder Detected 09:23
SCHM-V1.0: Intruder Detected 09:23

SCHM-V1.0: Warning: Fire Detected 13:17
SCHM-V1.0: Warning: Fire Detected 13:17
SCHM-V1.0: Warning: Fire Detected 13:17
SCHM-V1.0: Warning: Fire Detected 13:17
SCHM-V1.0: Warning: Fire Detected 13:18
SCHM-V1.0: Warning: Fire Detected 13:18
SCHM-V1.0: Warning: Fire Detected 13:18
SCHM-V1.0: Warning: Fire Detected 13:18
SCHM-V1.0: Warning: Fire Detected 13:18
SCHM-V1.0: Warning: Fire Detected 13:18
SCHM-V1.0: Warning: Fire Detected 13:18
SCHM-V1.0: Warning: Fire Detected 13:18
SCHM-V1.0: Warning: Fire Detected 13:18
SCHM-V1.0: Warning: Fire Detected 13:18

SCHM-V1.0: Warning: Gas Detected 13:15
SCHM-V1.0: Warning: Gas Detected 13:15
SCHM-V1.0: Warning: Gas Detected 13:15
SCHM-V1.0: Warning: Gas Detected 13:16
SCHM-V1.0: Warning: Gas Detected 13:16
SCHM-V1.0: Warning: Gas Detected 13:16
SCHM-V1.0: Warning: Gas Detected 13:16
SCHM-V1.0: Warning: Gas Detected 13:16
SCHM-V1.0: Warning: Gas Detected 13:16
SCHM-V1.0: Warning: Gas Detected 13:16
SCHM-V1.0: Warning: Gas Detected 13:17
SCHM-V1.0: Warning: Gas Detected 13:17
SCHM-V1.0: Warning: Gas Detected 13:17
SCHM-V1.0: Warning: Gas Detected 13:17

(a) (b) (c)

Figure 9 Message notification when detected (a) intruder (b) fire, and (c) gas leak

Based on all the test results, it can be concluded that the smart cluster housing monitoring with ESP32, ESP-MESH, and Django has been successfully built. In addition, it can run according to the original purpose, which is monitoring cluster housing with ESP-Mesh Protocol and IoT. In this system, ten houses will be monitored centrally on the server, independently in each user account and LINE messaging application. Meanwhile, in the previous study, it only specifically monitors one house, and if it is developed or applied to many houses, it will have collided with wireless connection problems (Wicaksono, M. F., 2017 & Yuen, M. et al., 2018 & Jadhav, L. et al., 2018 & Wicaksono, M. F., et al., 2020).

CONCLUSIONS

Smart cluster housing monitoring with ESP32, ESP-MESH, and Django has been completed. From the hardware side, sensor data from each Client Mesh has been successfully processed and successfully sent to the server. The data received can be saved from the server-side to the database, then it can be processed and displayed. The web application can also provide notifications on web pages, text, images, sound, and notifications to the LINE messaging application. Smart cluster housing monitoring with ESP32, ESP-MESH, and Django can assist the head of security in monitoring the cluster area and help residents to monitor their homes and get messages immediately when something happens.

REFERENCES

Kalaivanan, I.S., Sangeetha, M. 2016. Monitoring and Controlling of Smart Homes using IoT and Low Power Wireless Technology. Indian Journal of Science and Technology. 9 (31): 1-9.

Wicaksono, M. F. 2017. Implementasi Modul WiFi NodeMCU ESP8266 Untuk Smart Home. Komputika: Jurnal Sistem Komputer. 6(1): 1-6.

Yuen, M., Chu, S.Y., Chu, W.H., Cheng, H.S., Ho, L.N., Yuen, S.P. 2018. A Low-Cost IoT Smart Home System. International Journal of Engineering & Technology.7 (4): 3143-3147.

Jadhav, L., Pai, V. 2018. Smart Home Security Using Internet of Things. International Research Journal of Engineering and Technology (IRJET), 05(02): 617-621

Wicaksono, M.F., Rahmatya, M.D. 2020. Implementasi Arduino dan ESP32 CAM untuk Smart Home. Jurnal Teknologi dan Informasi, 10(1): 40-51.

Espressif Company. 2021. ESP32: ESP-IDF Programming Guide. Retrieved April 2021 from https://docs.espressif.com/projects/esp-idf/en/latest/esp32/esp-idf-en-v4.4-dev-960-

gcf457d4-esp32.pdf

Guo Z., Ma X., Zhang P., and Liu Z. (2021). A Dust Sensor Monitoring System Using Wi-Fi Mesh Network. Journal of Physics: Conference Series. 1754 (2021): 1-7.

Lech, P., and Lodarski P. (2017). Analysis of The IoT WiFi Mesh Network. In: 6th Computer Science On-line Conference. 574(2017): 272-280

Mandanka K., Mistry S. (2017). Design and Development of Wearable device using Bluetooth Low Energy. International Journal of Advance Research in Engineering, Science & Technology. 4(4): 873-880.

Saleh, S.B., Mazlan, S.B., Hamzah, N.I.B., Karim, A.Z.Z.B.A., Zainal, M.S.B., Hamzah, S.A.B., Nor, D.B.M., Poad, H.B.M. 2018. Smart Home Security Access System Using Field Programmable Gate Arrays. Indonesian Journal of Electrical Engineering and Computer Science. 11(1): 152-160.

Sysala, T., Fogl, D., Neumann, P. 2017. The family house control system based on Raspberry Pi. MATEC Web of Conferences. 02034 (2017): 1-7.

Firdaus, R., Mulyana, E. (2018). Smart Building Lighting System. IOP Conf. Ser.: Mater. Sci. Eng. 384(1). 1-7.

Zaw, H.N., Hla, T.T. 2018. Design and Implementation of Flame Sensor and Obstacle Detection for Automatic Fire Fighting Robot. International Journal of Scientific Engineering and Technology Research. 07(02): 262-266.

Mutinda, M., Kamweru, P.K. 2020. Arduino Uno, Ultrasonic Sensor HC-SR04 Motion Detector with Display of Distance in the LCD. International Journal of Engineering and Technical Research. 9(5):936-941.

Priyanka., Kantha, P. 2020. Realization of an IoT System to Ensure Doorway Security by Integrating ESP32-CAM with Cloud Server. International Research Journal of Engineering and Technology (IRJET). 07(10): 1235-1238.

Heyasa, B.B.L., Galarpe, V.R.K.R. 2017. Initial Development and Testing of Microcontroller-MQ2 Gas Sensor for University Air Quality Monitoring. IOSR Journal of

Electrical and Electronics Engineering (IOSR-JEEE). 12(3):47-53.

Ansary, M., Surid, F.A., Debnath, D., Rahman, G.M. 2018. GSM Based Fire Security Using Arduino. International Journal of Engineering Sciences & Management Research. 5(3): 3-7.

Vamsi, K.M., Lokesh, P., Reddy, K.N., Swetha, P. 2021. Visualization of Real World Enterprise Data using Python Django Framework. IOP Conf. Series: Materials Science and Engineering. 1042 (2021): 1-6.

Jiaying, L., Tao, T., Wei, W., Bo, X., Xiangjie, K. 2018. A survey of scholarly data visualization. IEEE Access. 6(2018): 19205-19221.

Knaflic C.N. 2015. Storytelling with data: A data visualization guide for business professionals. John Wiley & Sons