Accredited Ranking SINTA 2
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online on: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 7 No. 1 (2023) 66 - 71 ISSN Media Electronic: 2580-0760

IoT Microcontroller Application Prototype as Data Transceiver from Network to USB Device

Rieke Adriati Wijayanti[1], M. Syirajuddin S[2], Abdul Rasyid[3], Ahmad Wilda Y[4]
[1,2,3,4] Electrical Engineering, Polytechnic State of Malang
[1]riekeaw@polinema.ac.id, [2]syirajuddin@polinema.ac.id, [3]abdul.rasyid@polinema.ac.id, [4]ahmadwildan@polinema.ac.id

Abstract

Telecommunications technology in the early 2000s until now has experienced a rapid increase. Starting with complex devices to microcomputer devices that are able to connect to the network. With the development of networking technology, it will leave behind non-network-support devices that can still be used. The existence of previous research on "The interface between the IoT microcontroller (ESP32) and the Max3421e USB Host" can be taken advantage of developing a device that can facilitate a non-support-network device into a support-network electronic device. So that non-support-network electronic devices do not become electronic waste and can also become a device that can be used in the present. In this study, a prototype of a data receiver from WiFi was designed, then the data that has been received is reorganized into rows of data that are ready to be sent to non-support-network electronic devices. This developed tool uses an ESP32 IoT microcontroller connected to the USB Host max3421e which has been packaged in the form of a USB host shield module using SPI protocol communication. The result obtained is that data from the network can be sent correctly to the USB Host max3421e via the ESP32 microcontroller.

Keywords: interface, microcontroller, USB Host, USB device, network

1. Introduction

Telecommunications technology in the early 2000s until now has developed very quickly. Starting with complex devices to microcomputer devices capable of being connected to the network. This development is supported by the development of silicon manufacturing technology as the basic material for computer components and technology in the internet world or better known as the Internet of Things (IoT) [1], [2].

Currently IoT is widely applied to aspects of industry, agriculture, animal husbandry, fisheries and many other aspects. With the development of networking technology, non-network-supported electronic devices will be left behind even though these devices can still be used [3], [4].

Departing from these problems, a solution emerged that can overcome communication problems between non-network-supported electronic equipment via Wifi. With this communication, users who will access via a computer or smartphone are not bothered by connecting using a USB cable.

The existing solution begins with the creation of an IoT microcontroller interface prototype (ESP32) with a USB Host max3421e by utilizing the SPI protocol that has been implemented. It was found that the IoT microcontroller (ESP32) and USB Host max3421e can communicate well and are able to recognize USB devices completely as well as USB devices connected to computers [5]. By utilizing the existing WiFi facilities on the IoT microcontroller (ESP32) [6], it is hoped that the prototype will not only be able to recognize a connected USB device, but also function as a data processor and sender from a computer or smartphone.

The development carried out from research [5] is to build an interface between the ESP32 microcontroller and the OLED Display to display the status of the equipment [7] and setting the ESP32 microcontroller data reception as a Wifi Station [8]. The setting in question is connecting to the network through the existing Access Point (AP) so that the ESP32 microcontroller can receive data from the Wifi network.

This study utilizes a microcontroller unit that can receive data from WiFi where the received data is rearranged into a data line that is ready to be sent to non-support-network electronic devices via a USB connection.

Accepted: 01-10-2022 | Received in revised: 25-01-2023 | Published: 03-02-2023

The data receiver from WiFi uses an IoT ESP32 microcontroller which is connected to a USB Host maxim integrated max3421e which has been packaged in the form of a USB Host shield module using the SPI protocol communication [9] [10]. With this USB Host shield, the received SPI protocol data will be converted into a USB protocol.

It is hoped that with this development, data communication between non-network-supported USB electronic devices and computers or smartphones can be done via Wifi.

2. Research Methods

The research that will be carried out is included in the type of science and technology development research, namely creating prototypes of tools that can communicate data between non-network-supported electronic equipment and data sending equipment such as computers or smartphones via WiFi.

2.1 System Block Diagram

The block diagram of the system carried out is shown in Figure 1:

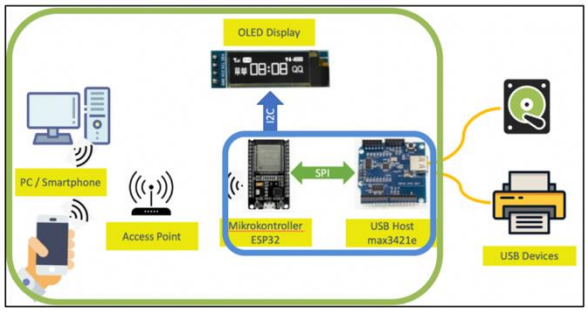


Figure 1. System Block Diagram

Figure 1 describes the block diagram of the system as a whole including the research that has been done [5] (blue box). This ongoing research is only limited to data communication from a computer or smartphone to a USB Host max3421e (green box). The system as a whole consists of a USB Device, USB Host max3421e, ESP32 Microcontroller, OLED Display and a computer or smartphone.

A computer or smartphone in order to communicate with non-network-supported electronic equipment must be connected using a USB cable and like a printer, the user also needs to perform the printer driver installation process first. If there are many devices that you want to use, there are also many drivers that need to be installed.

This device can minimize the installation process of various kinds of drivers, namely by utilizing WiFi. The device you want to use is connected to the interface between the USB Host and the ESP32 microcontroller [11] which is equipped with a WiFi module via SPI. Then, ESP32 will display the status of the equipment in the form of an IP Address obtained by the USB Device to the OLED Display via the I2C protocol [12].

Then the PC or smartphone that wants to communicate with the non-network-supported equipment can access it via WiFi, so users are not limited to using a USB cable.

2.2 System Configuration

Based on the block diagram of the system above, the system that has been developed in this study is shown in Figure 2:

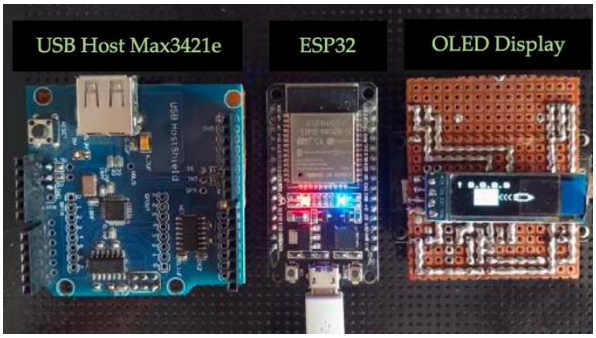


Figure 2. System Configuration

There are 3 parts of software development, namely: ESP32 microcontroller interface with USB Host max3421e already done in previous research, ESP32 microcontroller interface with OLED Display, and ESP32 microcontroller data reception settings as Wifi station. The setup of connecting to the network through an existing Access Point needs to be done so that the ESP32 microcontroller can receive data from the Wifi network.

2.3. Interface between microcontroller ESP32 with OLED Display

The development of the interface between the ESP32 microcontroller and the OLED Display is carried out so that the status of the equipment in the form of the IP Address obtained by the USB Device can be displayed.

The ESP32 microcontroller sends a binary raster initial display notification after the ESP32 microcontroller boots up. The results obtained during this process are shown in Figure 3.

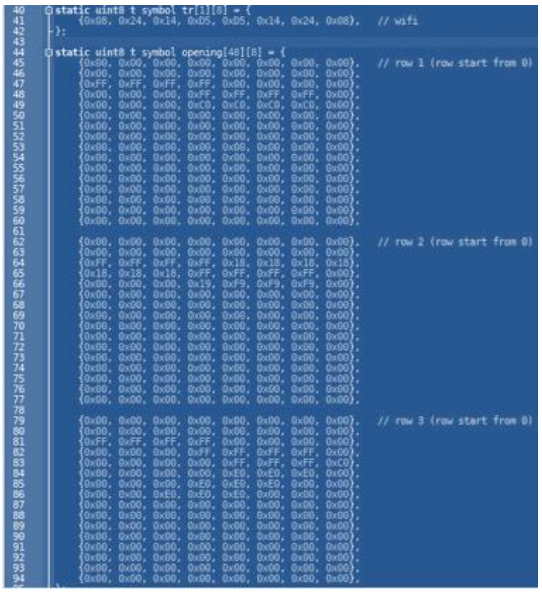
The notification displayed indicates the process of connecting the USB Device to the USB Host max3421e and the display of the ESP32 microcontroller when it has not obtained an IP address.

2.4 Microcontroller ESP32 as a Wifi Station Setting

The first time this system was run, the ESP32 microcontroller did not store the SSID and password data from the access point, so the ESP32 microcontroller did not get an IP address. To obtain the IP address of the access point, an ESP32 microcontroller setup is required to become an access

DOI: https://doi.org/10.29207/resti.v7i1.4555

point via a computer or smartphone [13][14]. The settings made are: The computer or smartphone is connected to an ESP32 access point named "USBWifi", open the internet browser, and enter the address //192.168.4.1, after that a display will appear as shown in Figure 4, ffter the "Connect" button in F is active, the ESP32 microcontroller will store the SSID and password in the ESP internal flash-rom area, then the ESP32 will restart, At the time of booting, the ESP32 microcontroller will use the SSID and password that have been stored to get the connection from the existing access point.



(a)

(b)

Figure 3. (a) Binary raster initial display notification after ESP32 microcontroller boots; (b) OLED Display display when ESP32 microcontroller boots

If the IP address have been obtained from the access point, the ESP32 microcontroller will send the IP address on the OLED Display. In this case, when the IP address has been obtained, it means that the USB Device has been installed on the system as shown in Figure 5.

This is evidenced by the display on the serial monitor of the computer connected to the system, displaying the same IP address as obtained, namely 192.168.1.100 as shown in Figure 6.

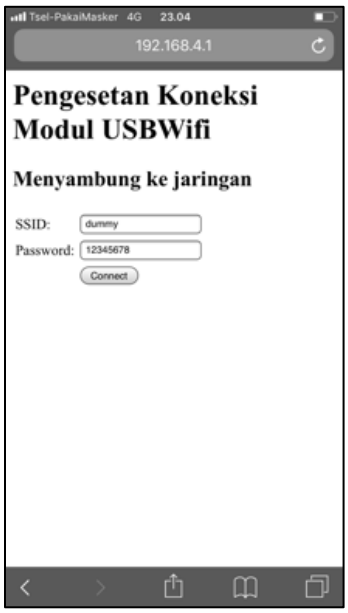


Figure 4. Initial view for connecting to the network

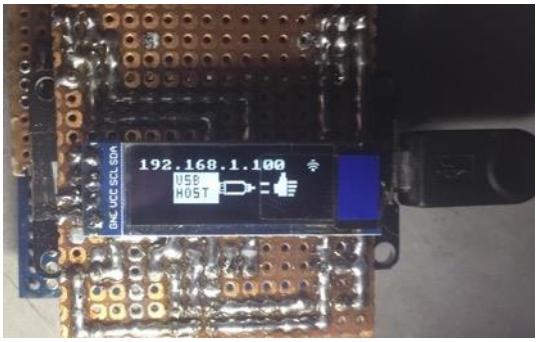


Figure 5. OLED Display display when you have obtained an IP address and the USB Device is connected

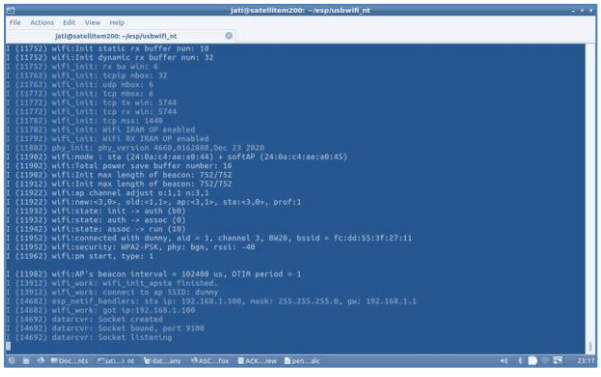


Figure 6. Serial display of the monitor from ESP32 to check the obtained IP address

3. Results and Discussions

From the configuration of the system that has been built on and the software development that has been carried out, it is found that between a computer or a smartphone can send data through the Wifi network. To find out the success of the data sent, it is necessary to test the

DOI: https://doi.org/10.29207/resti.v7i1.4555

system. The testing carried out on this system consists of three processes, namely:

3.1 Authentication Data Reception from Wifi Network to ESP32 Microcontroller

The process carried out in this testing stage is to send data from Wifi to the ESP32 Microcontroller as shown in Figure 7.

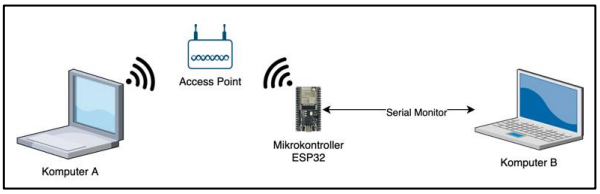


Figure 7. Data transmission from Wifi to ESP32 Microcontroller

In the picture above, there is computer A that has been equipped with PuTTY software (terminal emulator) and is connected to the access point wirelessly. In Figure 8, the following is shown the display of computer A that sends data, namely "halloo!" with the number of characters sent is seven characters, then translated into ASCII code to "68 61 6c 6c 6f 6f 21".

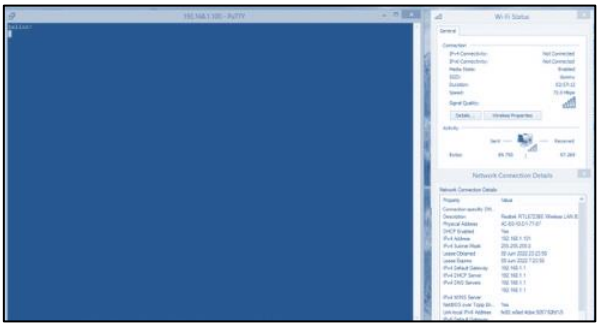


Figure 8. View from computer A (data sender)

On the other hand, there is a computer B connected to the ESP32 serial monitor receiving the same result which is "68 61 6C 6C 6F 6F 21" with a character length of 9 characters. There is an addition of two characters from the initial data because there is an addition of the character "0D 0A" [15] as shown in Figure 9.

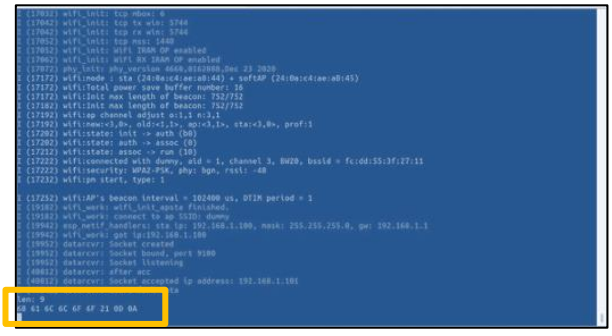


Figure 9. Serial display of the monitor from ESP32 on computer B (data receiver)

The two additional characters "0D 0A" are ASCII codes from cr (carriage return) and lf (line feed) in order. Cr and lf data are generated from PuTTY software when pressing the "enter" button on the computer keyboard which marks the data command being sent. In actually, for example data printing, there is no character "0D 0A" as the command sends the data. But the data receiver (printer) has its own standards in processing data. By ignoring the 0D and 0A characters, it can be concluded that the transmission of data from computer A to the ESP32 Microcontroller was successfully carried out.

3.2 Authentication of Data Reception from ESP32 Microcontroller to Max3421e Host USB

The different between the first authentication is the usage of USB Host max3421e that have been connected to microcontroller ESP32 using the SPI protocol. The process carried out in this testing stage is to send data from Wifi to the USB Host max3421e through the ESP32 Microcontroller. Here in Figure 10, there is computer A that has been equipped with PuTTY software (terminal emulator) and is connected to the access point wirelessly.

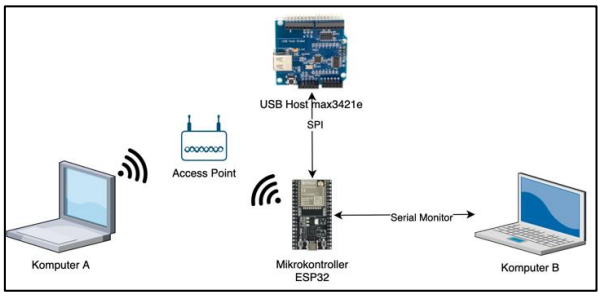


Figure 10. Sending data from Wifi to USB Host max3421e via ESP32 Microcontroller

In Figure 11, the following is shown the display of computer A that sends data, namely "hello world!!!!!" with the number of characters sent is 16 characters, then translated into ASCII code to "68 65 6C 6C 6F 20 77 6F 72 6C 64 21 21 21 21 21".

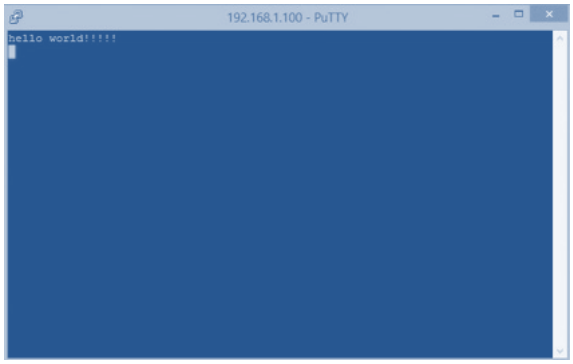


Figure 11. View from computer A (data sender)

On the other hand, there is a computer B connected to the ESP32 serial monitor receiving the same result which is "68 65 6C 6C 6F 20 77 6F 72 6C 64 21 21 21 21 21" with a character length of 18 characters. There is an addition of two characters from the initial data due to

DOI: https://doi.org/10.29207/resti.v7i1.4555

the addition of the character "0D 0A" as shown in Figure 12.

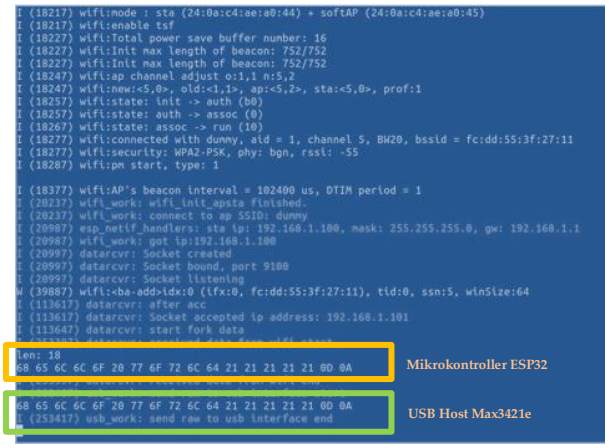


Figure 12. Serial display of monitors from ESP32 and USB Host max3421e on computer B (data receiver)

In Figure 12, there are 2 rows of data barrage displayed. The first row of data (boxed orange) shows the data received in the ESP32 microcontroller and the second row (boxed in green) shows the data received in the USB Host max3421e. It can be seen on computer B that the rows of data received by the two components are the same, both in terms of the number of characters and the order of the characters have corresponded to the data sent from computer A.

3.3 Data Buffer Testing on the ESP32 Microcontroller

The data buffer for this study has been set to 256, which means that in 1 data transmission it is limited to 256 bits (starting from index 0 to index 255), according to the program code as follows:

```
static void do_retransmit(const int sock)
 {
      uint8_t len;
      uint8_t rx_buffer[256];
       ESP_LOGI(TAG, "start fork data");
       …
 }
```

Testing this buffer is carried out to see whether the data sent from the computer is correctly sized according to the buffer that has set the value. The process carried out is the delivery of the character 'a' in the amount of 992 characters, the same as the process of testing the second point, as shown in Figure 13.

On the other hand in Figure 14, there is computer B connected to the ESP32 serial monitor receiving data as much as 4 times with each delivery limited according to the number of buffers that have been set before, which is 256 characters. So that for data delivery of 992 characters, first data delivery of 256 characters, second data delivery of 256 characters, third data delivery of 256 characters and last data delivery of 224 characters plus "0D 0A" characters at the end of the data.

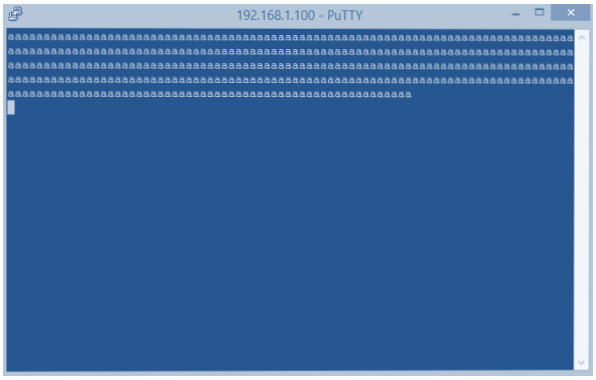


Figure 13. Display from computer A (data sender) with 992 characters

In Figure 14 there are 2 rows of data barrage displayed. The first row of data shows the data received in the ESP32 microcontroller and the second row shows the data received in the USB Host max3421e as shown below:

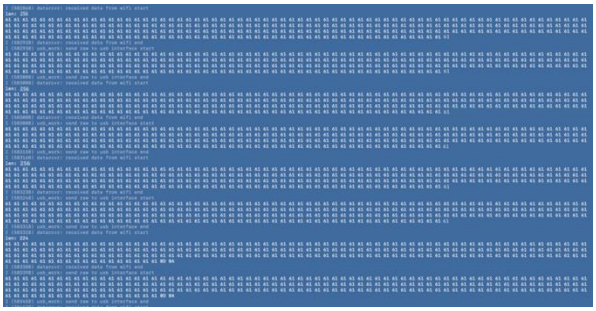


Figure 14. Serial display of the monitor from esp32 and USB host max3421e on computer B (data receiver) with the amount of data sent 992 characters

4. Conclusion

This research is a development of previous research that discussed the interface between the ESP32 microcontroller and the MAX3421e USB Host. The development carried out is the interface of the ESP32 microcontroller with the OLED Display, the data reception settings of the ESP32 microcontroller as a Wifi station. From the results of this development, it was found that sending data from a computer or smartphone to a USB device can be done properly and correctly in terms of the order of data and the amount of data. In addition, the data and status of the device connected to the ESP32 microcontroller can be displayed properly through the OLED Display. For further research, a data analysis process will be carried out in the form of processing data from the delivery results from the computer to the USB host max3421e based on predetermined defined class codes.

DOI: https://doi.org/10.29207/resti.v7i1.4555

Reference

[1] G. Hergika, Siswanto, and S. S, "Perancangan Internet Of Things (IoT) Sebagai Kontrol Infrastuktur Dan Peralatan Toll Pada Pt. Astra Infratoll Road," *PROSISKO J. Pengemb. Ris. Dan Obs. Sist. Komput.*, vol. 8, no. 2, pp. 86–98, Sep. 2021, doi: 10.30656/prosisko.v8i2.3862.

[2] M. Babiuch, P. Foltynek, and P. Smutny, "Using the ESP32 Microcontroller for Data Processing," in *2019 20th International Carpathian Control Conference (ICCC)*, Krakow-Wieliczka, Poland, May 2019, pp. 1–6. doi: 10.1109/CarpathianCC.2019.8765944.

[3] T. Darmanto and H. Krisma, "Implementasi Teknologi IOT Untuk Pengontrolan Peralatan Elektronik Rumah Tangga Berbasis Android," *J. Tek. Inform. Unika St. Thomas*, vol. 04, pp. 1–12, Jul. 2019, doi: https://doi.org/10.17605/jti.v4i1.505.

[4] A. Wagyana, "Prototipe Modul Praktik untuk Pengembangan Aplikasi Internet of Things (IoT)," *Setrum Sist. Kendali-Tenaga-Elektron.-Telekomun.-Komput.*, vol. 8, no. 2, p. 238, Dec. 2019, doi: 10.36055/setrum.v8i2.6561.

[5] Rieke Adriati Wijayanti, Ahmad Wilda Yulianto, Dianthy Marya, Muhammad Syirajuddin S., and Nurul Hidayati, "Antarmuka Mikrokontroller IoT (ESP32) Dengan USB Host max3421e," *J. Appl. Smart Electr. Netw. Syst.*, vol. 1, no. 02, pp. 70–75, Dec. 2020, doi: 10.52158/jasens.v1i02.141.

[6] Politeknik Negeri Bali *et al.*, "Perbandingan Kinerja Arduino Uno dan ESP32 Terhadap Pengukuran Arus dan Tegangan," *J. Otomasi Kontrol Dan Instrumentasi*, vol. 13, no. 1, pp. 35–47, 2021, doi: 10.5614/joki.2021.13.1.4.

[7] A. Y. Putra, H. Srihendayana, and N. Tjahjamooniarsih, "Monitoring Kamera Pengintai Jarah Jauh Terintegrasi dengan Google Drive Berbasis Raspberry Pi via Internet," *J. Tek. Elektro Univ. Tanjungpura*, vol. 2, no. 1.

[8] P. A. Nugroho, "Kontrol Lampu Gedung Melalui WIFI ESP8266 Dengan Web Server Lokal," *JEIS J. Elektro Dan Inform. Swadharma*, vol. 1, no. 2, pp. 1–11, 2021.

[9] C. Wootton, "Serial Peripheral Interface (SPI)," in *Samsung ARTIK Reference*, Berkeley, CA: Apress, 2016, pp. 335–349. doi: 10.1007/978-1-4842-2322-2_21.

[10] M. Iqbal, T. W. Widodo, and B. A. A. Sumbodo, "Sistem Pengendali Pengambilan Gambar Pada Kamera DSLR Melalui Protokol PTP," *IJEIS*, vol. 6, no. 2, pp. 117–128, 2016.

[11] N. V. A. Royani, M. J. Afroni, and B. D. Sulo, "E-Inventory pada Laboratorium Teknik Elektro di Universitas Islam Malang Menggunakan Barcode Scanner," *Inform. Electr. Electron. Eng. Infotron*, vol. 1, no. 2, p. 71, Jan. 2022, doi: 10.33474/infotron.v1i2.14787.

[12] A. Jazmi, S. R. Akbar, and E. R. Widasari, "Implementasi Multi – Channel Pada Wireless Sensor Network," *J. Pengemb. Teknol. Inf. Dan Ilmu Komput. Vol 2 No 4 2018*, 2017, [Online]. Available: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/1243

[13] P. Macheso, S. Chisale, C. Daka, N. Dzupire, J. Mlatho, and D. Mukanyirigira, "Design of Standalone Asynchronous ESP32 Web-Server for Temperature and Humidity Monitoring," in *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, Mar. 2021, pp. 635–638. doi: 10.1109/ICACCS51430.2021.9441845.

[14] A. Kurniawan, *Internet of Things Projects with ESP32: Build exciting and powerful IoT projects using the all-new Espressif ESP32*. Packt Publishing Ltd, 2019.

[15] F. A. Sitorus, N. B. Nugroho, and U. F. S. S. Pane, "Implementasi Algoritma Advanced Encryption Standard (AES) 128 Bit Untuk Keamanan Data Transaksi Penjualan Pada PT. MITSUBISHI ELECTRIC INDONESIA," *J. Cyber Tech*, vol. 4, no. 5, 2022.

DOI: https://doi.org/10.29207/resti.v7i1.4555