**Jurnal Sistem Informasi dan Sistem Komputer,** Vol. 9, No. 2, Juli 2024
**ISSN:** 2715-906X (Online)
10.51717/simkom.v9i2.392

# PERANCANGAN KENDALI *SERVER* LINUX MENGGUNAKAN MIKROKONTROLER ESP 32

**Arif Wicahyanto[1], Hari Mulyanto[2], Diaz Arya Satya Pratama[3], Pramono[4]**
[1]22010322@mhs.udb.ac.id, [2]210103174@mhs.udb.ac.id, [3]210103149@mhs.udb.ac.id,
[4]pramono@udb.ac.id
[1,2,3,4]Fakultas Ilmu Komputer, Universitas Duta Bangsa Surakarta

## Abstrak

Pengelola *server* Linux memberikan perintah pada sistem operasi agar sistem operasi Linux dapat memenuhi kebutuhan pengelola *server*. Pengelola *server* harus melakukan login ke *server* baik melalui terminal ataupun melalui antar muka web. Penelitian ini bertujuan untuk merancang sebuah sistem kendali yang dapat mengatur konfigurasi *server* berdasarkan data intensitas cahaya yang diterima sensor cahaya yang terhubung pada mikrokontroler ESP 32. Perubahan cahaya dengan menyalakan dan memadamkan lampu ruangan akan membuat perubahan konfigurasi pada *server* Linux. Pada penelitian ini digunakan metode pengembangan *prototype*. Sistem dirancang dengan menggunakan sebuah papan pengembangan ESP 32, modul sensor cahaya dan sebuah *server* Linux. Pada perubahan kondisi ruangan, perangkat ESP 32 akan mengirim permintaan ke aplikasi Python yang dipasang pada *server* Linux, selanjutnya aplikasi akan melakukan konfigurasi *firewall* sesuai kondisi yang ditentukan. Berdasarkan dari hasil pengujian yang dilakukan menunjukkan bahwa sistem dapat mengontrol konfigurasi *firewall* sesuai perubahan kondisi intensitas cahaya.
.

**Kata kunci:** ESP 32, Kendali, Cahaya, Linux, Python

## Abstract

*Linux server administrator give commands to the operating system so that the Linux operating system can meet the needs of server administrator. The server administrator must login to the server either through the terminal or through the web interface. This research aims to design a control system that can adjust the server configuration based on light intensity data received by the light sensor connected to the ESP 32 microcontroller. Changes in light by turning on and off the room lights will make configuration changes on the Linux server. In this research, the prototype development method is used. The system is designed using an ESP 32 development board, a light sensor module and a Linux server. When the room conditions change, the ESP 32 device will send a request to the Python application installed on the Linux server, then the application will configure the firewall according to the specified conditions. Based on the results of the tests carried out, it shows that the system can control the firewall configuration according to changes in light intensity conditions.*

***Keywords:*** *ESP 32, Control, Light, Linux, Python*

## 1. Pendahuluan

Di era saat ini, pengelolaan *server* menjadi aspek penting dalam memastikan kelancaran operasional dan ketersediaan layanan berbasis teknologi informasi. Dalam mengelola *server*, pengelola *server* Linux diharuskan melakukan autentikasi ke dalam *server*, baik melalui terminal ataupun melalui antar muka *website* tertentu [1]. Kegiatan konfigurasi tersebut membutuhkan waktu dan ketelitian pengelola *server* yang akan memakan waktu untuk menyelesaikannya.

Penelitian ini dilakukan oleh adanya kebutuhan untuk membuat sistem kendali *server* Linux yang dapat berubah sesuai dengan lingkungan fisiknya. Dengan memanfaatkan teknologi sensor, khususnya sensor cahaya, peneliti berupaya mengembangkan sebuah mekanisme yang memungkinkan *server* Linux merespons perubahan lingkungan, seperti perubahan intensitas cahaya, yang dapat diinterpretasikan sebagai indikator aktivitas atau kondisi operasional tertentu.

**Jurnal Sistem Informasi dan Sistem Komputer,** Vol. 9, No. 2, Juli 2024
**ISSN:** 2715-906X (Online)
10.51717/simkom.v9i2.392

Melalui penerapan metode pengembangan *prototype*, penelitian ini bertujuan untuk merancang dan mengimplementasikan sebuah sistem kendali yang dapat mengubah konfigurasi *server* Linux berdasarkan perubahan intensitas cahaya. ESP32 merupakan mikrokontroler dengan biaya terjangkau, hemat energi serta memiliki modul Wi-Fi dan Bluetooth terintegrasi. EPS 32 merupakan penerus dari mikrokontroler ESP8266 yang dikembangkan oleh *Espressif Systems*, perusahaan berbasis di Shanghai, Tiongkok [2] Sistem ini menggunakan ESP32, yang tidak hanya berfungsi sebagai jembatan antara dunia fisik dan sistem digital tetapi juga memfasilitasi komunikasi dua arah antara *server* dan lingkungan eksternal [3]. Integrasi ini memungkinkan *server* untuk melakukan penyesuaian konfigurasi secara otomatis berdasarkan input sensor cahaya LDR [4], menciptakan hal baru dalam pengelolaan *server*.

Beberapa penelitian terkait telah dilakukan dalam area ini, termasuk penelitian tentang penggunaan sensor cahaya pada mikrokontroler untuk aplikasi IoT [5], serta penelitian tentang pengaturan konfigurasi *server* Linux melalui antar muka *web* [6]. Akan tetapi, penelitian yang secara khusus menggabungkan konsep konfigurasi server Linux dengan sistem kendali ESB 32 tidak ditemukan.

Penelitian ini memiliki keunikan, yaitu menggabungkan konsep kendali berdasarkan kondisi lingkungan yaitu intensitas cahaya dengan pengaturan *server* Linux. Pendekatan ini memungkinkan *server* Linux secara otomatis menyesuaikan konfigurasi berdasarkan kondisi lingkungan sekitar yang belum banyak diteliti sebelumnya.

## 2. Metode

Dalam penelitian ini dibuat sebuah sistem kendali yang menggunakan papan pengembangan ESP 32, modul sensor cahaya dan *server* Linux Centos Stream 8. Aplikasi yang dibangun meliputi aplikasi pada perangkat pengembangan ESP32 dengan bahasa C++ dimana papan pengembang ini memiliki modul WiFi terintegrasi yang digunakan untuk melakukan koneksi jaringan [7] serta aplikasi dengan bahasa pemrograman Python yang dijalankan pada *server* Linux.

### 2.1. Rancangan Alur Kerja

Pada bagian ini dijelaskan gambaran kerja sistem secara umum dimana pada sistem ini terdapat 3 bagian utama. Bagian input berupa sumber cahaya yang berasal dari lampu penerangan ruangan, bagian mikrokontroler beserta sensor cahaya, dan bagian *server* Linux yang digunakan untuk menjalankan program Python sekaligus menjalankan perintah pada *server* Linux tersebut.


Gambar 1. Diagram Alur Kerja

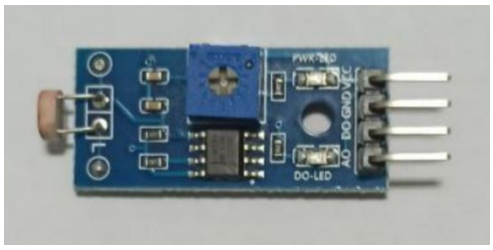
Mikrokontroler ESP 32 secara terus menerus membaca nilai intensitas cahaya di ruangan. Ditentukan sebuah nilai dimana nilai tersebut menjadi batas antara kondisi gelap dan kondisi terang. Saat terjadi perubahan kondisi dari gelap menjadi terang atau sebaliknya, mikrokontroler ESP 32 akan melakukan permintaan ke *server* Linux melalui jaringan nirkabel. Permintaan yang dikirim berisi data token untuk proses autentikasi serta perintah kendali. Data token yang diterima dibandingkan dengan data token yang ada di program Python. Jika data pada token yang dikirim sesuai, maka program Python akan menjalankan perintah kendali. Pada penelitian ini, digunakan perintah konfigurasi *firewall server* Linux dengan *iptables*, dimana akan melakukan blokir pada *chain input* dengan tujuan *port* 80 dan perintah untuk membuka blokir pada *chain input* dengan tujuan *port* 80 [8].

Saat kondisi pencahayaan ruangan berubah dari gelap ke terang, *server* Linux akan menerima koneksi dengan tujuan *port* 80, sedangkan saat kondisi ruangan berubah dari terang ke gelap, *server* Linux akan menutup koneksi ke *server* Linux dengan tujuan *port* 80.

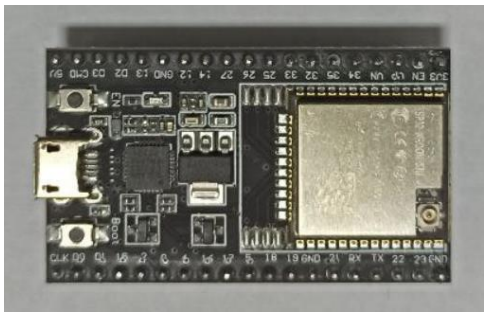
**Jurnal Sistem Informasi dan Sistem Komputer,** Vol. 9, No. 2, Juli 2024
**ISSN:** 2715-906X (Online)
10.51717/simkom.v9i2.392

## 2.2. Rancangan Perangkat Keras

Perancangan perangkat keras yang digunakan pada penelitian ini meliputi penggunaan modul sensor cahaya yang dihubungkan dengan mikrokontroler ESP 32.

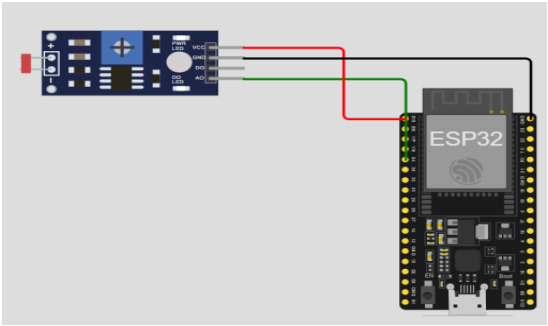



Gambar 2. Modul Sensor Cahaya



Gambar 3. Mikrokontroler ESP 32

Adapun alur koneksi antara sensor cahaya dan mikrokontroler ESP 32 adalah pin AO pada sensor cahaya dihubungkan ke pin 34 pada mikrokontroler ESP 32, PIN VCC pada sensor cahaya dihubungkan ke pin 3V3 pada mikrokontroler ESP 32 dan pin GND pada sensor cahaya dihubungkan ke pin GND pada mikrokontroler ESP 32 [9].

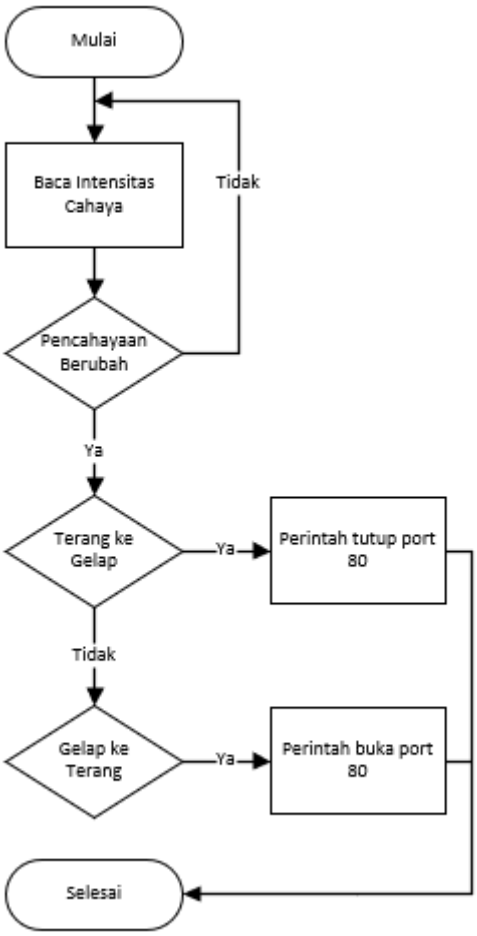



Gambar 4. Skema Koneksi Sensor Cahaya dan Mikrokontroler ESP 32

## 2.3. Rancangan Perangkat Lunak

Perancangan perangkat lunak dilakukan pada mikrokontroler ESP 32 yang berfungsi membaca data intensitas cahaya dan mengirimkan permintaan ke *server* Linux. Pada *server* Linux, dirancang aplikasi Python yang berperan menerima data dari mikrokontroler ESP 32 dan memberi perintah ke *server* Linux untuk membuka dan menutup koneksi ke *port* 80.

**Jurnal Sistem Informasi dan Sistem Komputer,** Vol. 9, No. 2, Juli 2024
**ISSN:** 2715-906X (Online)
10.51717/simkom.v9i2.392

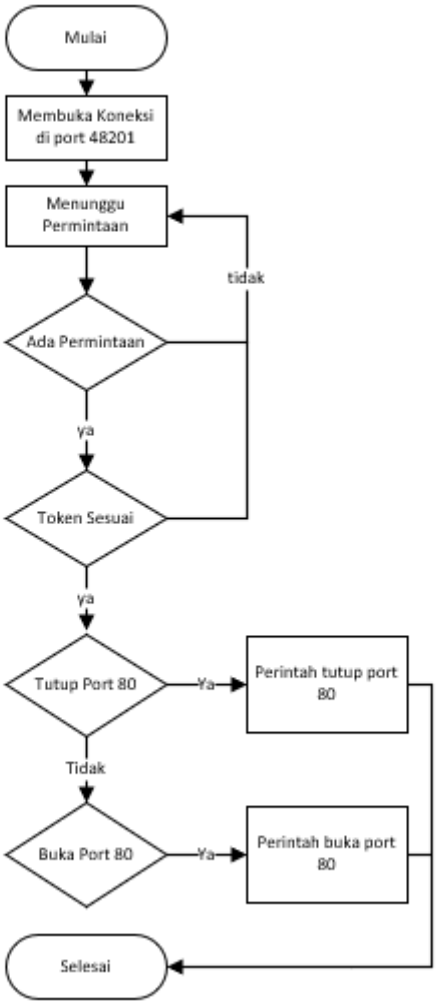



Gambar 5. Alur Aplikasi pada Mikrokontroler ESP 32

Pada Gambar 5 dijelaskan alur kerja perangkat lunak pada mikrokontroler ESP 32. Perangkat mikrokontroler ESP 32 membaca data intensitas cahaya melalui sensor cahaya [10]. Pada ruangan, dilakukan percobaan pembacaan intensitas cahaya untuk mengetahui batas nilai kondisi ruangan gelap dan terang. Dari hasil percobaan ditentukan nilai batas kondisi gelap dan terang yaitu nilai 2000. Jika hasil nilai pembacaan sensor cahaya di bawah 2000 maka kondisi ruangan adalah terang, sedangkan jika nilai pembacaan sensor cahaya di atas nilai 2000 kondisi ruangan adalah gelap.

Dari proses pemantauan intensitas cahaya, pada program dibuat mekanisme untuk mengetahui adanya perubahan dari kondisi gelap ke kondisi terang. Saat terjadi perubahan kondisi cahaya ruangan, aplikasi melakukan permintaan berjenis *post* melalui protokol HTTP [11] ke *server* Linux dengan *port* tujuan 48201. Pada permintaan tersebut memuat data dalam format JSON [12] berupa sebuah variabel yang bernama 'token' yang bernilai '5a64aJLK' dan variabel yang bernama 'perintah' yang bersisi perintah 'buka' atau 'tutup'. Setelah melakukan permintaan, aplikasi akan kembali membaca intensitas cahaya melalui sensor cahaya secara terus menerus untuk mendeteksi perubahan kondisi cahaya ruangan.

**Jurnal Sistem Informasi dan Sistem Komputer,** Vol. 9, No. 2, Juli 2024
**ISSN:** 2715-906X (Online)
10.51717/simkom.v9i2.392




Gambar 6. Alur Aplikasi Python pada *Server* Linux

Pada Gambar 6, dijelaskan alur kerja aplikasi Python yang dijalankan pada *server* Linux. Aplikasi Python menerima koneksi TCP pada *port* 48201 [13]. Saat ada permintaan koneksi masuk berjenis *post*, aplikasi akan memeriksa data yang diterima. Jika data sesuai dengan parameter yang ditentukan, program akan melakukan pengiriman perintah ke blokir koneksi atau tutup koneksi sesuai dengan parameter yang diterima.

## 3. Hasil dan Pembahasan

Setelah dilakukan perancangan sistem kendali Linux dengan mikrokontroler ESP 32, proses dilakukan dengan pengujian sistem yaitu dengan cara menguji fungsi-fungsi yang telah dirancang sebelumnya apakah dapat berjalan sesuai dengan rancangan yang telah dibuat sebelumnya.

### 3.1 Pengujian Pembacaan Intensitas Cahaya

Pengujian dilakukan untuk mengetahui nilai hasil pembacaan intensitas cahaya oleh sensor cahaya apakah sudah sesuai dengan nilai batas yang ditentukan yaitu di bawah 2000 untuk kondisi lampu menyala dan di atas 2000 untuk kondisi lampu mati. Pengujian dilakukan dengan mengamati data yang ditampilkan pada terminal oleh program yang telah dimasukkan ke mikrokontroler ESP 32.

**Jurnal Sistem Informasi dan Sistem Komputer,** Vol. 9, No. 2, Juli 2024
**ISSN:** 2715-906X (Online)
10.51717/simkom.v9i2.392

Tabel 1. Pengujian Pembacaan Kondisi Ruangan

| Percobaan | Kondisi Lampu Ruangan | Nilai Pembacaan | Hasil |
|---|---|---|---|
| 1 | Menyala | 753 | Sesuai |
| 2 | Menyala | 720 | Sesuai |
| 3 | Menyala | 723 | Sesuai |
| 4 | Menyala | 721 | Sesuai |
| 5 | Menyala | 725 | Sesuai |
| 6 | Menyala | 720 | Sesuai |
| 7 | Mati | 2103 | Sesuai |
| 8 | Mati | 2214 | Sesuai |
| 9 | Mati | 2242 | Sesuai |
| 10 | Mati | 2232 | Sesuai |
| 11 | Mati | 2222 | Sesuai |
| 12 | Mati | 2215 | Sesuai |

### 3.2 Pengujian Perubahan Kondisi Cahaya Ruangan

Pengujian dilakukan untuk mengetahui apakah aplikasi yang dibuat pada perangkat ESP 32 dapat bekerja untuk mendeteksi perubahan kondisi dari gelap ke terang dan perubahan kondisi dari terang ke gelap. Pengujian dilakukan dengan menyalakan dan mematikan lampu ruangan kemudian mengamati hasilnya pada terminal.

```
10:34:48.799 -> Nilai Analog Sensor Cahaya = 860 => Terang
10:34:49.797 -> Nilai Analog Sensor Cahaya = 954 => Terang
10:34:50.805 -> Nilai Analog Sensor Cahaya = 2785 => Gelap
10:34:50.837 -> Status Berubah
10:34:50.870 ->  => Menjadi Gelap
```

Gambar 7. Perubahan Kondisi dari Terang ke Gelap

```
10:37:38.516 -> Nilai Analog Sensor Cahaya = 2263 => Gelap
10:37:39.516 -> Nilai Analog Sensor Cahaya = 2225 => Gelap
10:37:40.541 -> Nilai Analog Sensor Cahaya = 2224 => Gelap
10:37:41.530 -> Nilai Analog Sensor Cahaya = 846 => Terang
10:37:41.563 -> Status Berubah
10:37:41.595 ->  => Menjadi Terang
```

Gambar 8. Perubahan Kondisi dari Gelap ke Terang

Tabel 2. Pengujian Pembacaan Kondisi Ruangan

| Percobaan | Perubahan Kondisi | Pembacaan | Hasil |
|---|---|---|---|
| 1 | Gelap ke Terang | Terdeteksi | Sesuai |
| 2 | Gelap ke Terang | Terdeteksi | Sesuai |
| 3 | Gelap ke Terang | Terdeteksi | Sesuai |
| 4 | Gelap ke Terang | Terdeteksi | Sesuai |
| 5 | Gelap ke Terang | Terdeteksi | Sesuai |
| 6 | Gelap ke Terang | Terdeteksi | Sesuai |
| 7 | Terang ke Gelap | Terdeteksi | Sesuai |
| 8 | Terang ke Gelap | Terdeteksi | Sesuai |
| 9 | Terang ke Gelap | Terdeteksi | Sesuai |
| 10 | Terang ke Gelap | Terdeteksi | Sesuai |
| 11 | Terang ke Gelap | Terdeteksi | Sesuai |
| 12 | Terang ke Gelap | Terdeteksi | Sesuai |

**Jurnal Sistem Informasi dan Sistem Komputer,** Vol. 9, No. 2, Juli 2024
**ISSN:** 2715-906X (Online)
10.51717/simkom.v9i2.392

### 3.3 Pengujian Kendali ke *Server* Linux

Pengujian dilakukan untuk mengetahui apakah aplikasi yang dibuat pada perangkat ESP 32 dapat mengirimkan data ke *server* Linux, serta menguji program Python pada *server* Linux, apakah setelah menerima data dari mikrokontroler ESP 32, program akan menjalankan kendali ke *firewall* Linux melalui perintah *iptables*. Pengujian dilakukan dengan melakukan pengamatan *rule iptables* pada *server* Linux setelah terjadi perubahan kondisi cahaya. Pada pengujian dilakukan pula pengamatan waktu eksekusi perintah.

```
13:10:05.225 -> Status Berubah
13:10:05.257 ->  => Menjadi Gelap
13:10:05.257 -> HTTP Response code: 200
13:10:05.314 -> Data recv
```

Gambar 9. Mikrokontroler ESP 32 Berhasil Mengirimkan Perintah Tutup ke *Server* Linux

```
Received JSON data: {'token': '5a64aJLK', 'perintah': 'tutup'}
Port 80 blocked successfully.
103.23.224.172 - - [05/Apr/2024 13:10:05] "POST /execute HTTP/1.1" 200 -
```

Gambar 10. Program Python Menerima Data dan Menjalankan Perintah Kendali Tutup

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:80
```

Gambar 11. Aturan Blokir *Port* 80 Ditambahkan Pada *Chain Input*

```
13:11:40.310 -> Status Berubah
13:11:40.353 ->  => Menjadi Terang
13:11:41.238 -> HTTP Response code: 200
13:11:41.271 -> Data recv
```

Gambar 12. Mikrokontroler ESP 32 Berhasil Mengirimkan Perintah Buka ke *Server* Linux

```
Received JSON data: {'token': '5a64aJLK', 'perintah': 'buka'}
# Warning: iptables-legacy tables present, use iptables-legacy to see them
103.23.224.172 - - [05/Apr/2024 13:11:40] "POST /execute HTTP/1.1" 200 -
```

Gambar 13. Program Python Menerima Data dan Menjalankan Perintah Kendali Buka

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
```

Gambar 14. Aturan blokir *port* 80 pada *chain input dihapus*

**Jurnal Sistem Informasi dan Sistem Komputer,** Vol. 9, No. 2, Juli 2024
**ISSN:** 2715-906X (Online)
10.51717/simkom.v9i2.392

Tabel 3. Pengujian Kendali *Server* Linux

| Percobaan | Kondisi | Perintah Kendali | Hasil | Waktu Eksekusi |
|---|---|---|---|---|
| 1 | Gelap ke Terang | Berhasil | Sesuai | 2 detik |
| 2 | Gelap ke Terang | Berhasil | Sesuai | 4 detik |
| 3 | Gelap ke Terang | Berhasil | Sesuai | 3 detik |
| 4 | Gelap ke Terang | Berhasil | Sesuai | 2 detik |
| 5 | Gelap ke Terang | Berhasil | Sesuai | 4 detik |
| 6 | Gelap ke Terang | Berhasil | Sesuai | 3 detik |
| 7 | Terang ke Gelap | Berhasil | Sesuai | 3 detik |
| 8 | Terang ke Gelap | Berhasil | Sesuai | 1 detik |
| 9 | Terang ke Gelap | Berhasil | Sesuai | 3 detik |
| 10 | Terang ke Gelap | Berhasil | Sesuai | 4 detik |
| 11 | Terang ke Gelap | Berhasil | Sesuai | 2 detik |
| 12 | Terang ke Gelap | Berhasil | Sesuai | 2 detik |

## 4. Kesimpulan dan Saran

Berdasarkan hasil pengujian rancangan kendali *server* Linux dengan mikrokontroler ESP 32 dapat diambil kesimpulan bahwa sistem yang dirancang dapat melakukan kendali pada *server* Linux. Kendali yang dilakukan dengan cara memberikan perintah buka *port* 80 dan tutup *port* 80 dengan menggunakan perintah *iptables*, sesuai dengan kondisi lingkungan yang ditentukan yaitu perubahan dari terang ke gelap dan dari gelap ke terang. Sistem kendali yang dirancang diharapkan dapat mempermudah tugas pengelola *server* khususnya pada konfigurasi *server* yang berhubungan dengan kondisi lingkungan. Saran untuk penelitian selanjutnya untuk meneliti dan mengoptimalkan waktu eksekusi program kendali serta dapat dikembangkan untuk melakukan otomatisasi

## Daftar Pustaka

[1] W. Yahya, A. Bhawiyuga, and E. S. Pramukantoro, *Administrasi Sistem Server Berbasis Linux: Konsep dan Praktik*. 2019.
[2] Espressif, "ESP32." https://www.espressif.com/en/products/socs/esp32
[3] Y. Basir, M. R. A. Pratama, and M. W. Aminullah, "Perancangan Sistem Pendeteksi Dan Penanggulangan Banjir Menggunakan Esp32 Berbasis Iot," *J. Ilm. Giga*, vol. 26, no. 1, p. 11, 2023, doi: 10.47313/jig.v26i1.2127.
[4] B. Eko Cahyono *et al.*, "Karakterisasi Sensor LDR dan Aplikasinya pada Alat Ukur Tingkat Kekeruhan Air Berbasis Arduino UNO (Characterization of the LDR Sensor and Its Application in an Arduino UNO-Based Water Turbidity Meter)," *J. Teor. dan Apl. Fis.*, vol. 7, no. 2, pp. 179–186, 2019.
[5] Y. Natasya and H. Santoso, "Prototipe Aplikasi Smart Lighting Untuk Mengontrol Lampu Jalan Berbasis Android Menggunakan Esp32," *Sibatik J. Vol.*, vol. 2, no. 8, pp. 2581–2598, 2023, [Online]. Available: https://publish.ojs-indonesia.com/index.php/SIBATIK
[6] K. Hitchcock, *Linux System Administration for the 2020s*. 2022.
[7] R. S. Hadikusuma, V. W. Mahyastuty, M. Siregar, and L. Hardine, "Pengaruh Delay Pada Modul Wifi 802.11B/G/N Esp 32 Di Rentang Frekuensi 2.4 Ghz Terhadap Smart Control Solar Home System," *Transm. J. Ilm. Tek. Elektro*, vol. 25, no. 4, pp. 149–155, 2023, doi: 10.14710/transmisi.25.4.149-155.
[8] M. G. Mihalos, S. I. Nalmpantis, and K. Ovaliadis, "Design and implementation of firewall security policies using Linux iptables," *J. Eng. Sci. Technol. Rev.*, vol. 12, no. 1, pp. 80–86, 2019, doi: 10.25103/jestr.121.09.
[9] A. Budijanto, S. Winardi, and K. E. Susilo, *INTERFACING ESP32*. 2021.
[10] M. F. Wicaksono and M. D. Rahmatya, "Implementasi Arduino dan ESP32 CAM untuk Smart Home," *J. Teknol. dan Inf.*, vol. 10, no. 1, pp. 40–51, 2020, doi: 10.34010/jati.v10i1.2836.

[11] M. Babiuch, P. Foltynek, and P. Smutny, "Using the ESP32 microcontroller for data processing," *Proc. 2019 20th Int. Carpathian Control Conf. ICCC 2019*, no. May 2019, 2019, doi: 10.1109/CarpathianCC.2019.8765944.
[12] Mozilla Foundation, "Working with JSON." https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON (accessed Apr. 25, 2024).
[13] D. Seftyanto, D. F. Priambodo, and T. Yulita, *Pemrograman Jaringan Dengan Python*. 2022.