servirisma
JURNAL PENGABDIAN KEPADA MASYARAKAT

P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

SISTEM KONTROL PENCAHAYAAN BERBASIS IOT MENGINTEGRASI THINGSPEAK, ESP32, DAN KODULAR PADA KANDANG MERPATI DI DUSUN TULAR

Arief Yuli Nugroho [1], M. Dinda Christian Putri[2], Nisywani Hasna Priyandani[3], Arya Yusuf[4*]

[1,2,3]*Departemen Pendidikan Teknik Elektro, Fakultas Teknik, Universitas Negeri Yogyakarta*
[4]*Departemen Pendidikan Teknik Mesin, Fakultas Teknik, Universitas Negeri Yogyakarta*
*aryayusuf.2021@student.uny.ac.id

Abstrak

Penggunaan sakelar manual untuk mengontrol lampu kurang efektif, terutama ketika berada di luar rumah atau area yang sulit dijangkau. Meskipun sensor dapat mengatasi masalah ini, ada keterbatasan yang mengharuskan seseorang berada di dekat sensor. Oleh karena itu, sistem berbasis IoT telah dikembangkan, menawarkan potensi besar untuk mengotomatisasi dan mengontrol perangkat dari jarak jauh. Meskipun banyak penelitian telah dilakukan tentang IoT, masih sedikit studi yang mengeksplorasi integrasi ESP32, Kodular, dan ThingSpeak untuk pengembangan sistem IoT, padahal masing-masing alat ini memiliki keunggulannya sendiri. Tujuan penelitian ini adalah menilai apakah ketiga komponen tersebut dapat bekerja sama secara efektif untuk mengontrol dan memantau sistem pencahayaan dari jarak jauh. Metode pengembangan yang digunakan adalah model waterfall. Hasil penelitian menunjukkan bahwa integrasi ini berhasil memungkinkan kontrol pencahayaan jarak jauh, mengonfirmasi kelayakan penggabungan ESP32, Kodular, dan ThingSpeak untuk aplikasi IoT. Namun, kinerja sistem dibatasi oleh interval pembaruan data 15 detik pada versi gratis ThingSpeak, yang dapat menyebabkan keterlambatan. Untuk meningkatkan responsivitas, solusi seperti menambahkan timer hitung mundur di aplikasi atau beralih ke versi premium ThingSpeak dapat dipertimbangkan. Optimasi di masa depan perlu meningkatkan pemrosesan real-time dan keandalan dari perangkat keras.

Kata kunci: *ESP32, IoT, Kodular, Kontrol Pencahayaan, Thingspeak*

Abstract

The use of manual switches to control lights is less effective, especially when outside the house or in hard-to-reach areas. While sensors can address this issue, there is a limitation that requires someone to be near the sensor. Therefore, an IoT-based system has been developed, offering significant potential for automating and remotely controlling devices. While many studies have been conducted on IoT, there has been limited research exploring the integration of ESP32, Kodular, and ThingSpeak for IoT system development and each of these tools has its own advantages. The aim of this research is to assess whether these three components can effectively work together to control and monitor lighting systems remotely. The development method used is the waterfall model. The results demonstrate that this integration successfully enables remote lighting control, confirming the feasibility of combining ESP32, Kodular, and ThingSpeak for IoT applications. However, the system's performance is limited by the 15-second data update interval in ThingSpeak's free version, which may cause delays. To improve responsiveness, solutions such as adding a countdown timer in the application or upgrading to ThingSpeak's premium version. Future optimizations should enhance real-time processing and hardware reliability.

Keywords: *ESP32, IoT, Kodular, Lighting Control, Thingspeak*

P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

Pendahuluan

Dalam kehidupan sehari-hari saat ini, sakelar manual masih menjadi metode paling umum digunakan untuk mengontrol perangkat listrik seperti lampu. Sakelar berfungsi sebagai alat untuk menghubungkan atau memutus arus listrik dengan dua kondisi dasar yaitu menyala dan mati. Meskipun sederhana, metode ini sering dianggap tidak efisien dan tidak praktis, terutama dengan meningkatnya tuntutan akan kenyamanan dan penghematan energi. Contoh nyata masalah ini terjadi di ruang publik seperti ruang kelas atau kantor. Ketika seseorang memasuki ruangan gelap, mereka biasanya menyalakan lampu, namun sering kali lampu tetap menyala ketika mereka meninggalkan ruangan. Hal ini dapat menyebabkan lampu tetap menyala terus-menerus, yang tidak hanya membuang energi listrik tetapi juga membuat lampu lebih rentan rusak (Yulisman *et al.*, 2021). Solusi lainnya untuk masalah ini adalah menggunakan sensor, salah satunya adalah *proximity sensor*. Namun masalah penggunaan sensor semacam ini adalah ketergantungannya pada keberadaan objek atau orang di dekat sensor, sehingga kurang efektif untuk mengelola pencahayaan baik di dalam maupun luar ruangan. Sensor lain juga bisa digunakan, namun tetap bergantung pada keberadaan orang di dekat sensor. Oleh karena itu, jika pemilik rumah sedang bepergian, solusi berbasis sensor menjadi kurang efektif. Inilah mengapa *Internet of Things* (IoT) digunakan karena memungkinkan pemilik rumah yang sering bepergian untuk mengontrol sistem dari jarak jauh. Teknologi *Internet of Things* (IoT) menawarkan solusi efektif untuk masalah-masalah ini. *Internet of Things* (IoT) merupakan perkembangan teknologi yang memungkinkan perangkat untuk mengumpulkan data, memprosesnya, dan mengirimkan informasi ke pengguna (Ridwan & Sari, 2021). IoT memungkinkan koneksi berbagai perangkat dan sistem melalui internet, memfasilitasi pertukaran data menggunakan perangkat lunak, sensor, dan teknologi lainnya (Tamrakar *et al.*, 2022). Tujuan IoT adalah membuat tugas manusia menjadi lebih sederhana, efisien, dan efektif (Ayuningtyas, 2022).

Dikembangkan oleh Espressif Systems di Shanghai, Cina, ESP32 menyediakan fitur dan kemampuan yang kuat, menjadikannya pilihan ideal untuk aplikasi IoT (Hercog *et al.*, 2023). Dengan memanfaatkan mikrokontroler ESP32, sistem pencahayaan dapat dikontrol secara otomatis melalui internet. ESP32 memiliki konektivitas *Wi-Fi*, memungkinkan perangkat terhubung ke berbagai sensor dan aktuator. Untuk memfasilitasi operasi dan kontrol jarak jauh, dikembangkan aplikasi *mobile* menggunakan Kodular. Kodular adalah *platform* berbasis *web* yang menyediakan alat untuk membangun aplikasi *android* menggunakan metode pemrograman visual dengan konsep pemrograman blok *drag-and-drop* (Setiawan, 2020). Kodular juga bersifat *open-source*, sehingga dapat diakses oleh siapa saja (Herlianus & Gunadi, 2022). Kodular menyederhanakan pengembangan aplikasi tanpa memerlukan keahlian pemrograman tingkat lanjut, sehingga cocok bahkan untuk anak-anak mempelajari *platform* ini (Kholifah & Imansari, 2022). Keunggulan lain Kodular adalah kemampuannya untuk menguji aplikasi tanpa perlu mengunduh atau mengekspornya terlebih dahulu, menggunakan Kodular Companion (Furima *et al.*, 2022). Dalam sistem IoT, penyimpanan data operasional seperti status lampu, waktu penggunaan, dan pengaturan pencahayaan juga sangat penting. Oleh karena itu, sistem memerlukan *database* untuk menyimpan data yang dikumpulkan dari perangkat IoT. Melalui *database*, data dapat diakses dan digunakan untuk integrasi aplikasi (Adriansyah, 2024). *Database* yang digunakan dalam penelitian ini adalah ThingSpeak, sebuah *website* yang menyediakan layanan *cloud* untuk menyimpan data seperti pembacaan suhu, yang dapat diakses dari mana saja menggunakan *channel ID* dan *read key* (Ramisetty *et al.*, 2020).

Dusun Tular merupakan salah satu dari delapan dusun yang ada di Desa Seloboro, Kecamatan Salam, Kabupaten Magelang, Provinsi Jawa Tengah. Desa Seloboro terbagi menjadi delapan dusun yang dilewati oleh sungai putih dan terbagi menjadi dua wilayah yaitu wilayah barat (Dusun Sukowati, Dusun

servirisma
JURNAL PENGABDIAN KEPADA MASYARAKAT
P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

Kuncen, Dusun Klumpukan, dan Dusun Tular) dan wilayah Timur (Dusun Krapyak, Dusun Gajahan, Dusun Nabin Benteng, dan Dusun Seloboro). Sebagai dusun yang paling padat penduduknya di Desa Seloboro, Dusun Tular memiliki kehidupan sosial yang sangat aktif sehingga banyak dinamika sosial yang ada. Salah satunya adalah memelihara burung merpati yang nantinya biasanya akan digunakan untuk lomba adu kecepatan dan lain sebagainya. Terdapat warga yang mengeluh bahwa kandang burung merpati sangat tertutup sehingga ketika ingin menyalakan lampu diharuskan untuk membawa alat penerangan maupun meraba-raba karena bagian dalamnya yang gelap. Selain itu karena sering ditinggal untuk bekerja sering kali pemilik bisa lupa memadamkan maupun menyalakan lampu. Hal tersebut menjadi latar belakang tim melakukan penelitian untuk membantu menyelesaikan masalah dengan membuat sistem *internet of things* serta mencoba kelayakan dari integrasi ESP32, Kodular, dan Thingspeak. Adapun tujuan dari pembuatan sistem ini adalah untuk menyalakan dan memadamkan lampu dari jarak jauh.

Meskipun banyak peneliti yang telah mengeksplorasi pengembangan IoT menggunakan berbagai mikrokontroler (seperti ESP32, ESP8266, NodeMCU), aplikasi (seperti Kodular, Blynk), dan *database* (seperti Firebase, ThingSpeak), namun belum ada penelitian sejenis yang mengintegrasikan ESP32, ThingSpeak, dan Kodular sebagai kombinasi yang menawarkan keunggulan unik. Kodular menyederhanakan pengembangan antarmuka pengguna dengan pemrograman berbasis blok yang intuitif, sementara ThingSpeak menyediakan *database* visual yang ramah pengguna. Meskipun masing-masing alat ini memiliki keunggulan sendiri, kelayakan penggabungannya untuk aplikasi IoT masih belum teruji. Penelitian ini difokuskan pada evaluasi kelayakan integrasi ESP32, Kodular, dan ThingSpeak untuk aplikasi IoT sederhana, dengan perangkat akan diuji dan diimplementasikan dalam kandang merpati untuk mengontrol lampu dari perangkat.

Metode

Penelitian ini menggunakan metode *waterfall* untuk merancang sistem IoT. Metode *waterfall* merupakan pendekatan tradisional dalam pengembangan perangkat lunak yang dilakukan secara berurutan melalui tahapan-tahapan tertentu. Setiap tahap harus diselesaikan sepenuhnya sebelum melanjutkan tahap berikutnya (Khan, 2023). Model ini banyak digunakan dalam rekayasa perangkat lunak untuk memastikan keberhasilan suatu proyek (Senarath, 2021). Metode ini disebut "*waterfall*" (air terjun) karena proses pengembangannya menyerupai aliran air yang mengalir ke bawah secara bertahap. Pendekatan ini memastikan bahwa setiap langkah pengembangan diselesaikan sepenuhnya sebelum beralih ke fase berikutnya. Berikut (Gambar 1) merupakan tampilan metode *waterfall* yang digunakan:

Gambar 1 menggambarkan metode *Waterfall*, dengan setiap fase dijelaskan menurut (Wahid, 2020) sebagai berikut:

A. *Requirement Analysis* (Analisis Kebutuhan)

Pada tahap ini, komunikasi antara pengembang sistem dan pengguna sangat penting untuk memahami harapan terhadap perangkat lunak beserta batasan-batasannya.

B. *System Design* (Desain Sistem)

Tahap ini bertujuan merancang sistem secara menyeluruh berdasarkan kebutuhan yang telah dikumpulkan pada fase sebelumnya. Fokus utama tahap ini adalah memastikan sistem dirancang secara efisien, memenuhi semua persyaratan, dan dapat diimplementasikan dengan teknologi yang sesuai. Dalam pengembangan ini, desain akan terdiri dari lima jenis sebagai berikut:

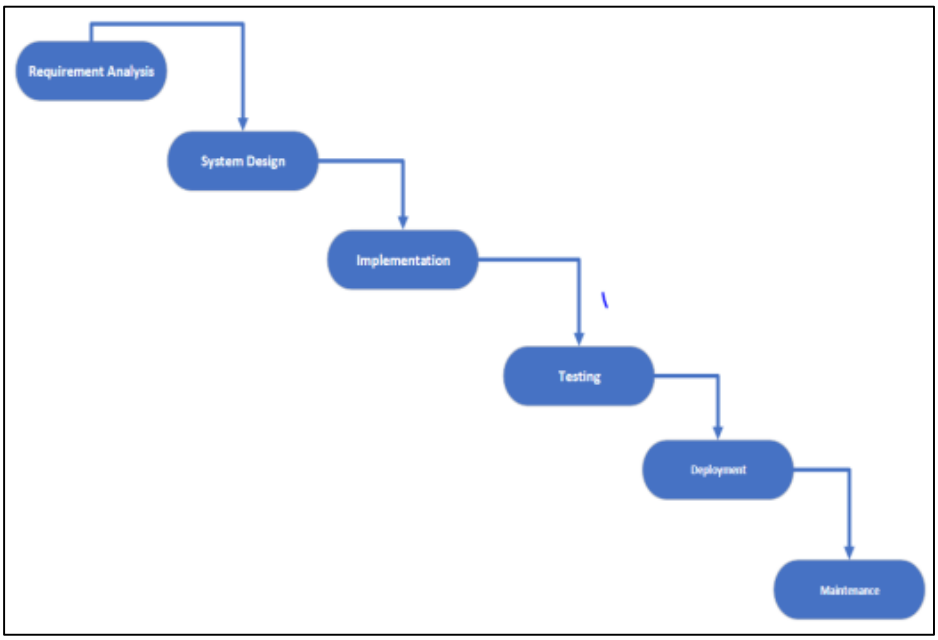
servirisma
JURNAL PENGABDIAN KEPADA MASYARAKAT
P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

1. Desain perangkat lunak

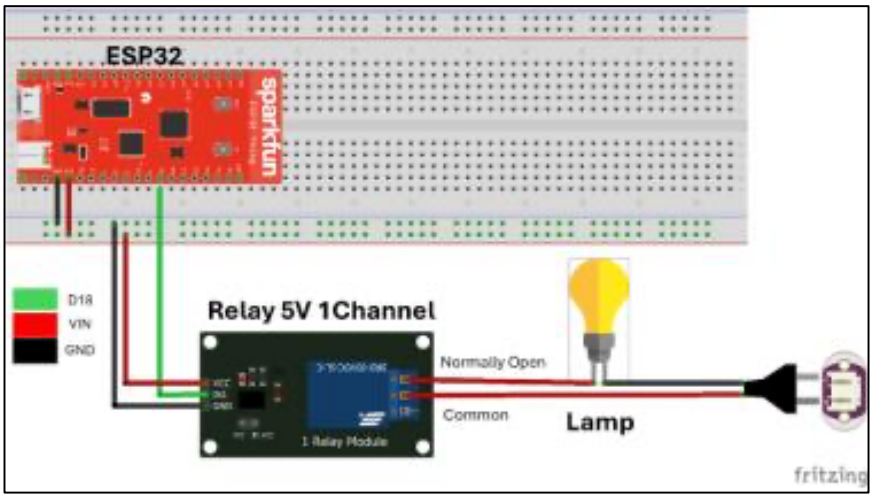
Pengembangan aplikasi ini dimulai setelah fase analisis kebutuhan selesai dilakukan. Aplikasi dibangun menggunakan platform Kodular yang berbasis pemrograman *hybrid* dan menawarkan beberapa keunggulan. Salah satu keunggulan utama Kodular adalah kemampuannya memungkinkan pengembangan aplikasi bahkan tanpa memerlukan pengetahuan pemrograman mendalam dari pengguna.

2. Desain perangkat keras

Selain pengembangan aplikasi, alur proses untuk desain perangkat keras juga direncanakan. Rancangan rangkaian (Gambar 2) yang akan digunakan meliputi:



Gambar 1. Fase Metode Waterfall yang mengadaptasi dari Khan (Khan, 2023)



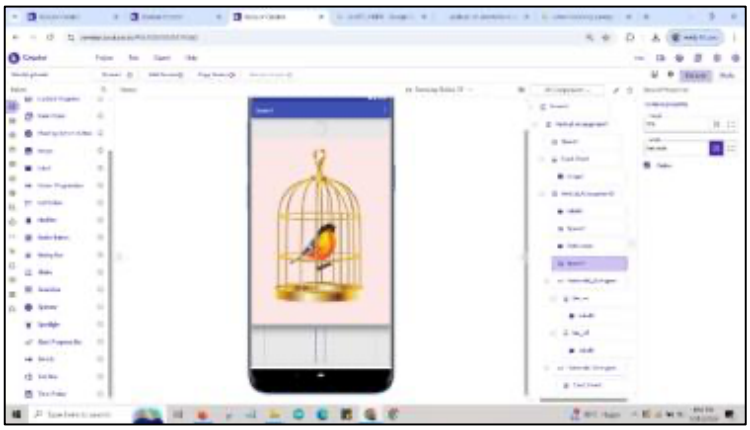
Gambar 2. Diagram sirkuit

servirisma
JURNAL PENGABDIAN KEPADA MASYARAKAT
P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

Penjelasan sederhana dari gambar di atas adalah: pin digital 18 (kabel hijau) dari ESP32 terhubung ke pin *input* Relay. Kemudian, *output* COM Relay terhubung ke terminal negatif sumber daya. Selanjutnya, pin NO (*Normally Open*) *relay* terhubung ke lampu yang dirangkai secara seri dengan sumber daya. Untuk menyalakan lampu, *input* yang harus dikirim ke *relay* adalah 1, yang dapat didefinisikan secara fleksibel sesuai kebutuhan.

3. Desain antarmuka

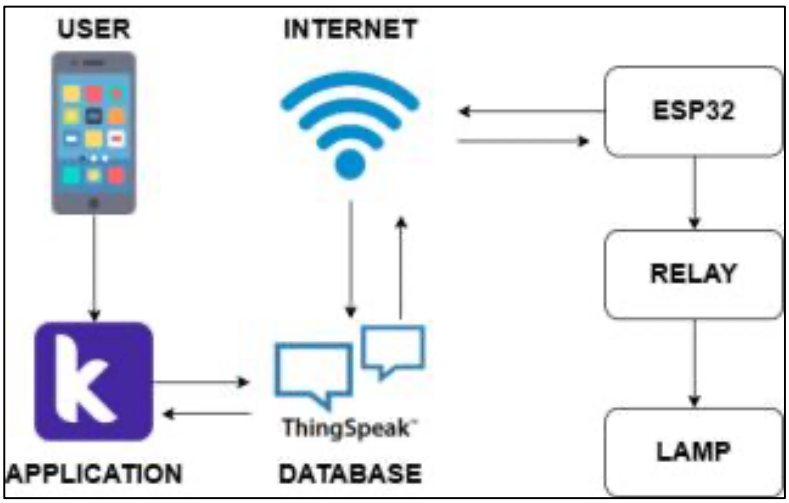
Desain antarmuka aplikasi dibuat sederhana karena hanya berfungsi untuk mengontrol lampu menyala/padam. Dengan demikian, tidak diperlukan penambahan fitur lain dalam aplikasi ini (Gambar 3).



Gambar 3. Pembuatan aplikasi di Kodular

4. Diagram Blok

Gambar berikut ini memperlihatkan diagram blok sistem:



Gambar 4. Blok Diagram Sistem

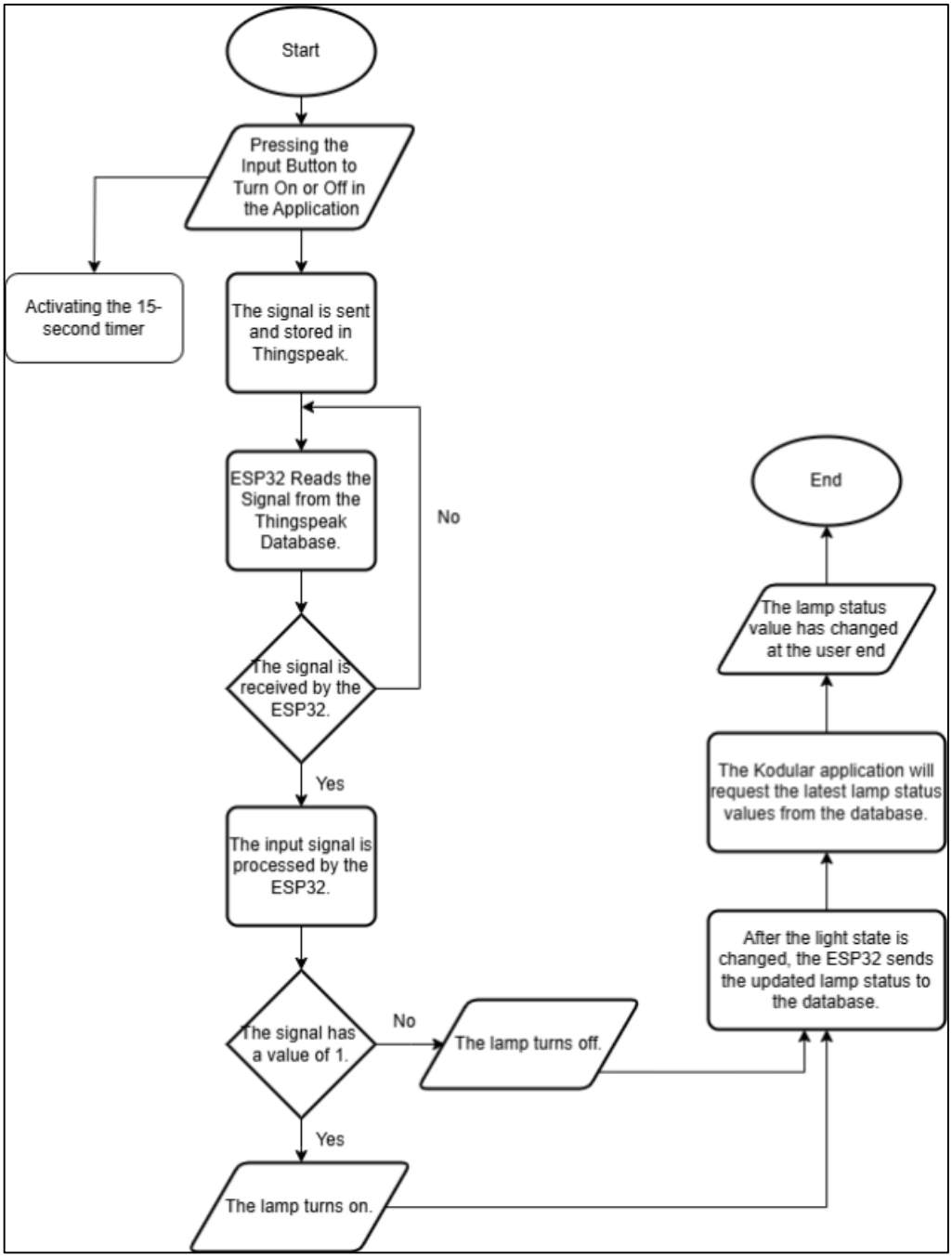
Gambar 4 di atas menjelaskan alur kerja sistem. Proses dimulai ketika pengguna mengakses aplikasi yang dikembangkan di Kodular. Aplikasi ini berkomunikasi dengan sistem database ThingSpeak. Sementara itu, modul ESP32 terhubung ke internet sehingga dapat mengakses database ThingSpeak juga. Dengan demikian, data yang dikirim melalui aplikasi Kodular dapat diterima oleh

servirisma
JURNAL PENGABDIAN KEPADA MASYARAKAT
P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

ESP32 melalui ThingSpeak. ESP32 kemudian memproses data ini untuk mengontrol *relay* untuk mengatur lampu yang terhubung. Selain itu, karena ESP32 memiliki konektivitas internet, modul ini dapat mengirim status terkini lampu (*ON/OFF*) kembali ke ThingSpeak. Aplikasi Kodular kemudian mengambil data terbaru ini dari ThingSpeak, memungkinkan penyesuaian nilai secara *real-time* pada antarmuka aplikasi.

5. *Flowchart*

Selain itu, ada juga *flowchart* yang menunjukkan cara kerja aplikasi *smart home* sederhana seperti berikut:



Gambar 5. *Flowchart* Sistem

servirisma
JURNAL PENGABDIAN KEPADA MASYARAKAT
P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

Penjelasan gambar 5 di atas adalah: pertama, pengguna akan menekan tombol *ON* atau *OFF* dalam aplikasi *smart home*. Ketika pengguna menekan tombol, *timer* menghitung mundur 15 detik akan aktif. Data kemudian akan dikirim ke *database* Thingspeak. Pada saat yang sama, ESP32 diprogram untuk terus meminta data status tombol lampu dari Thingspeak. Ketika data diterima oleh ESP32, data akan diproses. Jika nilai data yang diterima adalah 1, *relay* akan menyala sehingga lampu menyala. Sebaliknya, jika data yang diterima adalah 0, *relay* akan mati dan lampu padam. Setelah status lampu berubah, ESP32 memperbarui status lampu terbaru di *database*. Secara bersamaan, aplikasi mengambil data lampu yang diperbarui dari database, memastikan nilai yang ditampilkan di antarmuka aplikasi mencerminkan keadaan saat ini.

C. *Implementation* (Implementasi)

Fase implementasi adalah proses desain sistem yang telah dirumuskan diterjemahkan ke dalam kode program. Tidak hanya program, simulasi dari desain kasar perangkat juga dibuat untuk mempermudah proses verifikasi.

D. Testing (Pengujian)

Fase pengujian bertujuan untuk memastikan bahwa sistem yang dikembangkan berfungsi sesuai dengan kebutuhan pengguna dan bebas dari kesalahan.

E. *Deployment* (Pengaplikasian)

Deployment adalah tahap dalam pengembangan perangkat lunak dari sistem atau produk yang telah selesai dan diuji dipersiapkan untuk digunakan oleh pengguna akhir. Proses ini melibatkan serangkaian langkah untuk memastikan bahwa perangkat lunak beroperasi dengan lancar di lingkungan produksi, memenuhi kebutuhan dan harapan pengguna.

F. Maintenance (Pemeliharaan)

Fase ini terjadi setelah sistem digunakan. Pengembang melakukan pemeliharaan untuk memastikan sistem tetap berfungsi sesuai kebutuhan pengguna. Pemeliharaan mencakup perbaikan *bug*, pembaruan fitur, dan penyesuaian terhadap perubahan kebutuhan atau lingkungan teknologi.

Hasil dan Pembahasan

A. *Requirement Analysis*

Proses analisis kebutuhan masalah dari pemelihara burung merpati, langkah pertama adalah melakukan observasi ke kediaman warga yang berada di Dusun Tular RT.07/ RW.06. Observasi di sini ditujukan untuk melihat kondisi serta harapan dari pemelihara tentang jalannya sistem nanti. Hasil dari observasi ini akan menjadi acuan dari pembuatan alat dan aplikasi yang akan dibuat. Tabel berikut adalah rangkuman dari masalah dan solusinya:

Tabel 1. Analisis Masalah dan Solusi

Masalah	Solusi
Kandang merpati sangat tertutup sehingga keadaan di dalam sangat gelap. Menyebabkan susah untuk mencari tempat sakelar berada. Serta Pemilik selalu pergi pagi untuk bekerja sehingga seringkali lupa dalam memadamkan atau menyalakan lampu kandang.	Dibuatkan sistem *Internet of Things* sehingga Pemilik dapat mengatur nyala padam lampu secara bebas dan dari jarak yang jauh.

servirisma
JURNAL PENGABDIAN KEPADA MASYARAKAT
P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

Di dalam kandang Pemilik juga masih ingin terdapat sakelar manual yang bisa digunakan.	Pada rangkaian perangkat keras alat IoT akan diparalelkan dengan sakelar manual, sehingga ketika perangkat IoT tidak digunakan atau rusak bisa menggunakan sakelar manual.

B. *System Design*

Berdasarkan tujuan tersebut, telah ditetapkan spesifikasi dengan perangkat lunak dan perangkat keras yang akan digunakan (Tabel 2 dan Tabel 3):

Tabel 2. Hardware

No.	*Hardware*	Tipe-Deskripsi
1	Laptop	Digunakan untuk mengembangkan dan memprogram ESP32 dari Arduino IDE.
2	ESP32 Lolin 32 *Wifi Bluetooth*	Berfungsi sebagai pengontrol utama, menerima perintah dari aplikasi seluler dan mengendalikan *relay*.
3	*Relay* 5V 1 *Channel Output* 250VAC 30VDC 10 A	Berfungsi sebagai sakelar otomatis yang menyalakan atau mematikan lampu berdasarkan perintah dari ESP32.
4	*ProjectBoard*	*Platform* untuk merakit dan menghubungkan semua komponen sistem.
5	PVC *Board*	Wadah/rumah untuk menyimpan dan melindungi rangkaian elektronik.

Tabel 3. Software

No.	*Software*	Tipe-Deskripsi
1	Arduino IDE	Digunakan untuk memprogram ESP32 dengan kode yang diperlukan agar dapat berkomunikasi dengan aplikasi seluler dan *relay*.
2	Kodular	Sebagai *platform* pengembangan untuk membangun aplikasi seluler yang memungkinkan pengguna mengontrol lampu.
3	ThingSpeak	*Platform* berbasis Cloud untuk menyimpan data dari aplikasi seluler, yang kemudian diakses oleh ESP32 guna memantau sistem secara *real-time*.

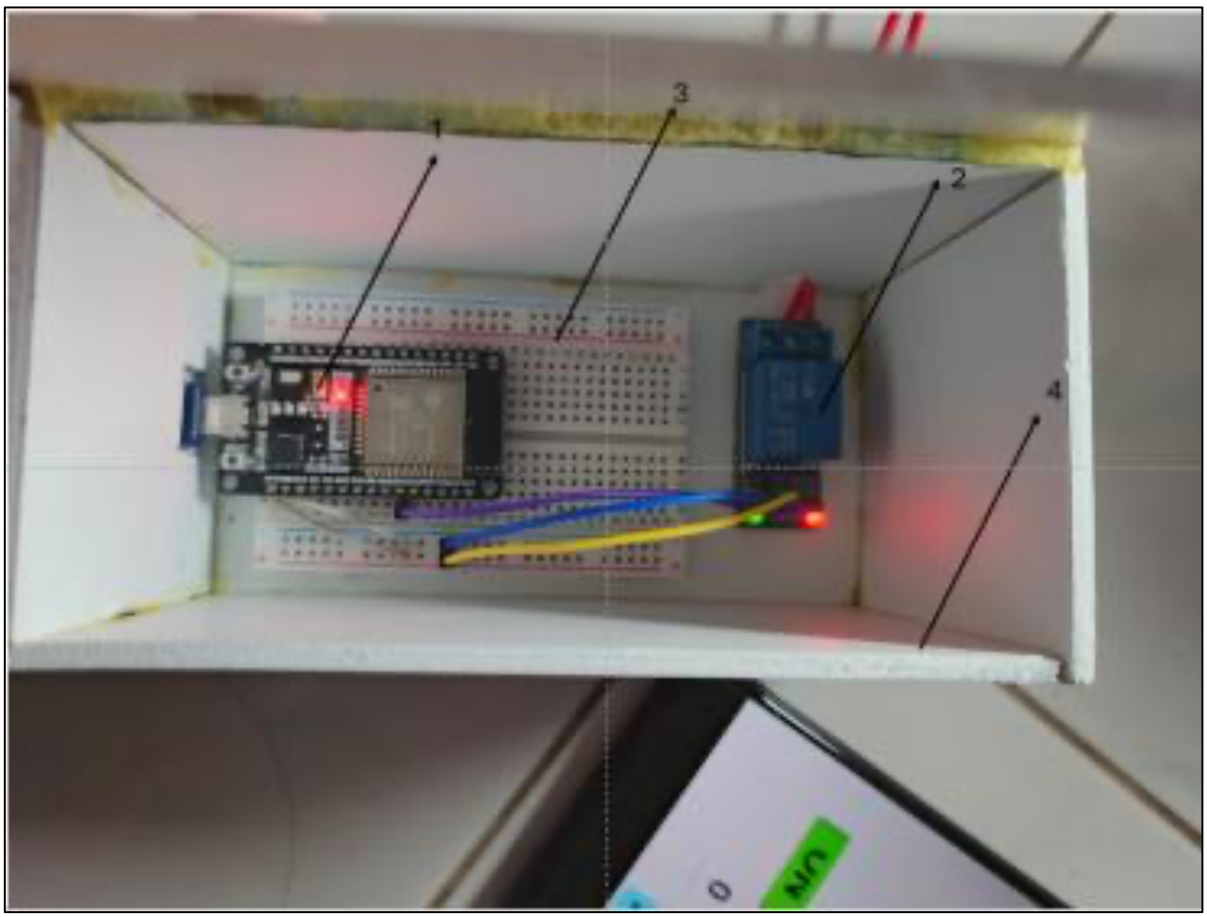
servirisma
JURNAL PENGABDIAN KEPADA MASYARAKAT
P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

Sistem ini hanya dapat berfungsi optimal dengan integrasi ESP32, Kodular, dan ThingSpeak. Oleh karena itu, penggunaan perangkat lain dapat mengakibatkan perbedaan fungsi. Namun, perangkat keras atau perangkat lunak pada kedua tabel di atas selain ketiga komponen tersebut dapat diubah sesuai kondisi yang ada.

C. *Testing*

Pada tahap pengujian ini, rangkaian simulasi dirakit secara nyata dan aplikasi yang telah dibuat juga akan diuji untuk mendeteksi adanya kesalahan. Berikut adalah rangkaian yang telah dirakit beserta proses pengujiannya:

Gambar 6 menunjukkan implementasi perangkat keras dari simulasi rangkaian yang telah dibuat dengan seluruh rangkaian dibangun dan terhubung sesuai dengan simulasi. PIN ESP32 (1) terhubung ke pin (2) semua komponen terpasang pada papan proyek (3) serta di dalam papan PVC (4).

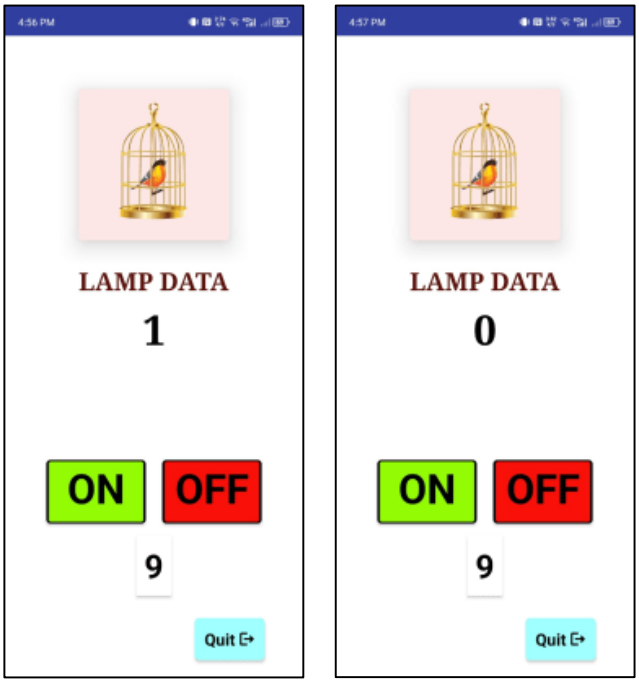


Gambar 6. Implementasi dari *Hardware*

Gambar 7 menunjukkan aplikasi yang sedang diuji untuk menyalakan prototipe lampu sebelum diterapkan langsung. (a) Menunjukkan bahwa ketika tombol “on” ditekan, nilai data lampu berubah menjadi 1 karena mengambil nilai dari *database*. Kemudian, muncul nilai 9 yang menunjukkan *timer cooldown* sebelum tombol lain dapat ditekan. (b) Menunjukkan bahwa ketika tombol “off” ditekan, nilai data lampu berubah menjadi 0 karena mengambil nilai dari *database*. Kemudian muncul nilai 9 yang

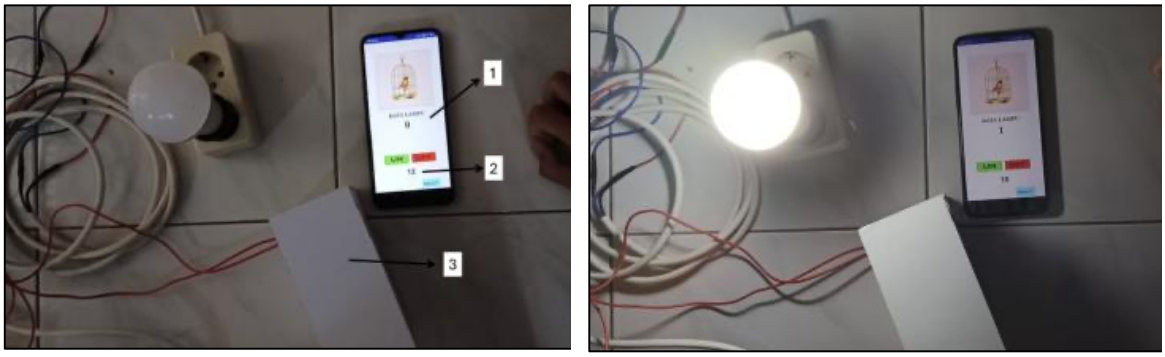
servirisma
JURNAL PENGABDIAN KEPADA MASYARAKAT
P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

menunjukkan *timer cooldown* sebelum tombol lain dapat ditekan. Desain aplikasi menyertakan *timer* karena layanan ThingSpeak yang digunakan masih gratis, sehingga diperlukan jeda 15 detik sebelum data dapat dikirim ke ThingSpeak. Untuk implementasi baik aplikasi maupun perangkat keras, terdapat pada gambar di bawah ini.



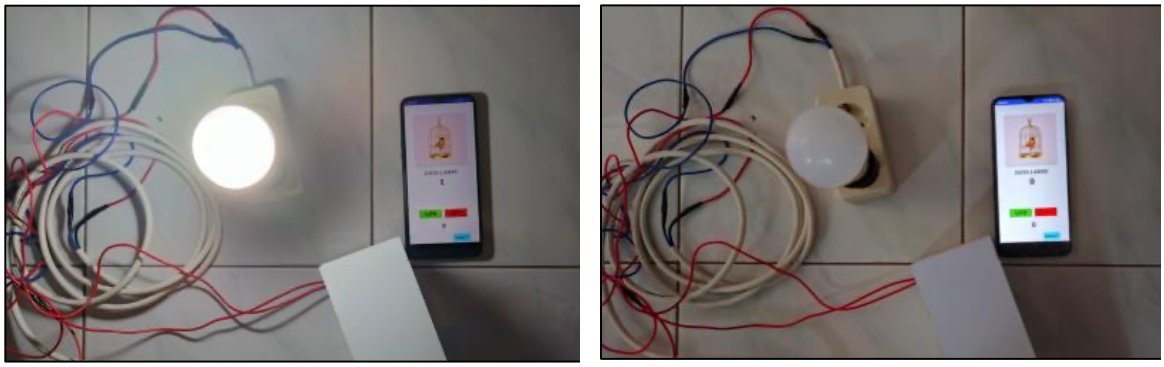
(a) (b)

Gambar 7. Pengujian Aplikasi: (a) Menyalakan, (b) Memadamkan



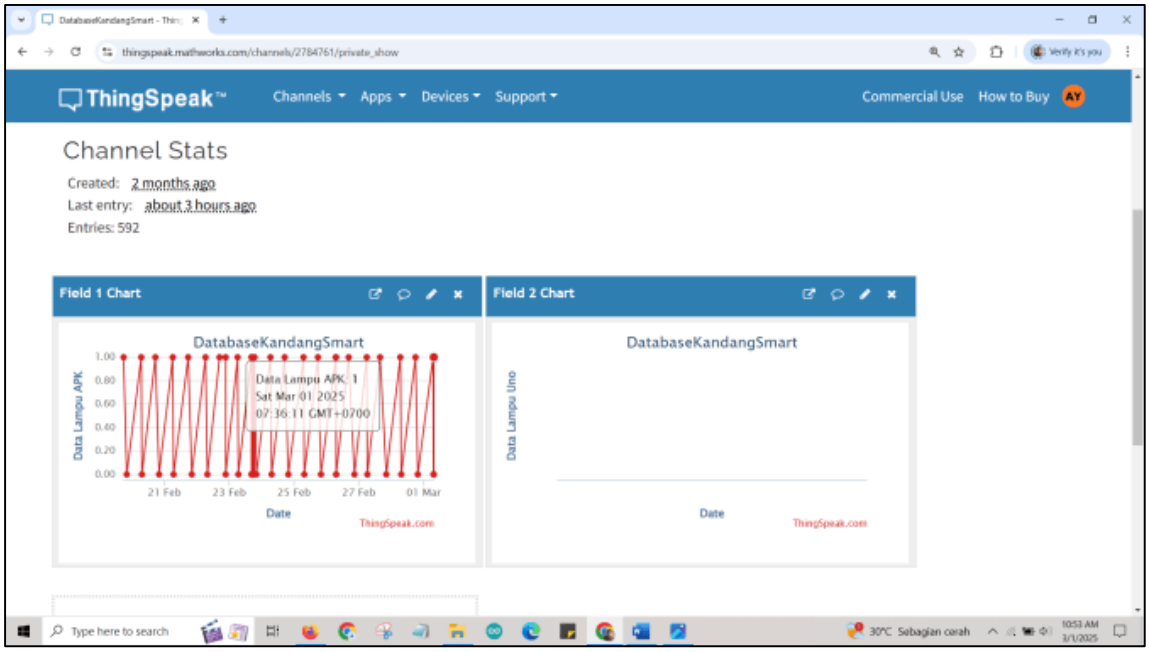
(a) (b)

servirisma
JURNAL PENGABDIAN KEPADA MASYARAKAT
P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126



(c) (d)

Gambar 8. Pengujian Rangkaian dan Aplikasi: (a) Kondisi awal: Padam, [1] Data lampu, [2] Timer, [3] Perangkat keras



Gambar 9. Isi dari Database *ThingSpeak*

Gambar 8 menunjukkan pengujian simultan aplikasi dan rangkaian. (a) Kondisi awal terjadi ketika lampu padam dan nilai data lampu (1) menunjukkan 0. (b) Menunjukkan bahwa ketika tombol "on" ditekan, nilai data lampu dalam aplikasi berubah menjadi 1 dan lampu menyala, *timer* kemudian memulai urutan hitung mundur. Sebaliknya, ketika tombol "off" ditekan seperti yang ditunjukkan pada (d), lampu padam dan data lampu dalam aplikasi juga berubah menjadi 0.

Gambar 9 tersebut menampilkan data dari aplikasi setelah Pemilik menggunakan perangkat dari tanggal 21 Februari hingga 1 Maret 2025, yang tersimpan dalam *database* ThingSpeak. Menurut Pemilik, perangkat digunakan saat mereka berada di tempat kerja yang berjarak 2 km dari lokasi penempatan alat, dan sistem berfungsi dengan baik serta pemeriksaan data secara keseluruhan mudah dilakukan karena disimpan dalam bentuk grafik. Namun, di luar integrasi ESP32, Kodular, dan ThingSpeak, terdapat masalah pada kabel yang digunakan.

servirisma
JURNAL PENGABDIAN KEPADA MASYARAKAT
P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

Dalam *database*, terdapat *field* yang menyimpan data dari aplikasi berupa nilai "on" sebesar 1 dan nilai "off" sebesar 0, yang kemudian dibaca oleh ESP32. Titik-titik merah pada *field* menunjukkan perubahan data yang dilakukan melalui aplikasi. Hanya satu jenis data yang disimpan karena sejak tahap perencanaan awal, Pemilik menginginkan sistem yang sederhana, hanya untuk menyalakan dan memadamkan lampu. Namun, sistem ini masih dapat dikembangkan dengan menambahkan data tambahan seperti memeriksa apakah lampu sudah menyala atau belum, serta fungsionalitas lainnya.

Pada tahap ini, semua komponen sistem pencahayaan otomatis diuji, termasuk uji perangkat saat lampu dinyalakan dan dipadamkan sebanyak tiga kali, kemudian diperiksa apakah lampu merespon sesuai perintah. Selama pengujian tidak terjadi kehilangan data, namun karena layanan ThingSpeak yang digunakan masih versi gratis, data yang dikirim dalam interval kurang dari 15 detik tidak akan diterima oleh ThingSpeak. Meskipun memiliki keterbatasan, sistem tetap memiliki potensi besar untuk pengembangan IoT dalam berbagai aplikasi karena masih mampu menerima *input* dan mengontrol *output* dari jarak jauh. Tabel 4 berikut merupakan kondisi pengujian tersebut:

Tabel 4. Pengujian menggunakan prototype

Kondisi	Data Lampu pada Aplikasi	Nilai pada *Database*	Kondisi Lampu
Tombol On pada aplikasi ditekan dengan *delay* ≥ 15 detik	1	1	On
Tombol Off pada aplikasi ditekan dengan *delay* ≥ 15 detik	0	0	Off
Tombol On pada aplikasi ditekan dalam waktu <15 detik	1	Tetap sama dengan nilai sebelumnya	Tetap dalam kondisi sebelumnya
Tombol Off pada aplikasi ditekan dalam waktu <15 detik	0	Tetap sama dengan nilai sebelumnya	Tetap dalam kondisi sebelumnya

Catatan Tambahan:

- Nilai 1: Menandakan lampu dalam keadaan menyala (ON)
- Nilai 0: Menandakan lampu dalam keadaan padam (OFF)

Pada tabel 4 tersebut, dapat diamati bahwa kondisi pertama dan kedua saat tombol ditekan setelah 15 detik dari penekanan sebelumnya, nilai pada aplikasi dan basis data adalah sama. Namun, pada kondisi ketiga dan keempat saat tombol ditekan sebelum 15 detik dari penekanan sebelumnya, nilai pada aplikasi dan basis data berbeda. Eksperimen dilakukan dalam empat kondisi berbeda dengan setiap kondisi diuji sebanyak tiga kali. Data yang diperoleh menunjukkan bahwa kondisi pertama dan kedua menghasilkan hasil yang konsisten, dengan nilai yang sesuai antara aplikasi dan basis data ketika kondisi dipatuhi. Sebagai contoh, pada kondisi pertama, jika tombol ON ditekan dan waktu tunda melebihi 15 detik, nilai data pada aplikasi adalah 1, dan nilai pada basis data juga menjadi 1. Namun, hasilnya berbeda pada kondisi ketiga dan keempat. Jika tombol ditekan sebelum 15 detik, nilai pada aplikasi akan tetap berubah menjadi 1, tetapi nilai pada basis data akan tetap pada nilai sebelumnya. Misalnya, jika nilai sebelumnya adalah 1, maka akan tetap 1, dan jika nilai sebelumnya adalah 0, maka akan tetap 0.

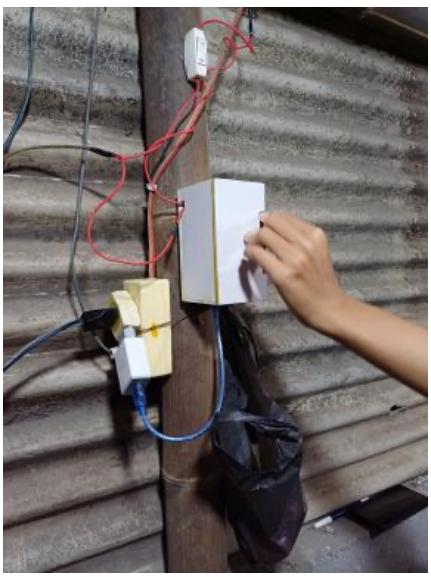
servirisma
JURNAL PENGABDIAN KEPADA MASYARAKAT
P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

Sehingga konsistensi data antara aplikasi dan basis data diamati sebanyak benar 6 kali dan salah sebanyak 6 kali dalam kondisi yang ada di dalam tabel. Hal ini terjadi karena aplikasi telah mengirim sinyal terbaru, tetapi basis data ThingSpeak yang digunakan adalah versi gratis, sehingga membatasi penerimaan data hanya sekali setiap 15 detik. Berdasarkan kondisi di atas, aplikasi itu sendiri menyertakan *timer* 15 detik memungkinkan pengguna untuk menentukan waktu yang tepat untuk menekan tombol lagi.

D. *Deployment*

Pada fase implementasi, hasil simulasi diaplikasikan dalam kondisi nyata dengan menggunakan tiga lampu. Lampu-lampu tersebut pertama-tama dihubungkan secara seri sebelum disambungkan ke *relay*, dan juga dipasang paralel dengan sakelar standar. Konfigurasi ini memastikan bahwa jika terjadi masalah pada *relay*, sakelar standar tetap dapat digunakan untuk menyalakan dan memadamkan lampu secara manual.

Gambar 10. Proses pembuatan alat didampingi pemilik kandang



Gambar 11. Pemasangan Alat

servirisma
JURNAL PENGABDIAN KEPADA MASYARAKAT
P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

Setelah instalasi, langkah berikutnya adalah melakukan pengujian lebih lanjut dengan menyalakan dan memadamkan lampu sebanyak dua kali, kemudian menguji sakelar manual sebanyak dua kali. Hasil uji lapangan kedua dapat dilihat pada tabel di bawah ini:

Tabel 5. Uji coba alat setelah dipasang

No.	*Scenario Test*	Hasil	Kesimpulan
1	Menyalakan perangkat melalui aplikasi	Lampu 1, 2, dan 3 menyala secara bersamaan	Valid
2	Mematikan perangkat melalui aplikasi	Lampu 1, 2, dan 3 mati secara bersamaan	Valid
3	Menyalakan perangkat menggunakan saklar manual	Lampu 1, 2, dan 3 menyala secara bersamaan	Valid
4	Mematikan perangkat menggunakan saklar manual	Lampu 1, 2, dan 3 mati secara bersamaan	Valid

Tabel 5 menyajikan hasil pengujian perangkat setelah pemasangan di lokasi penggunaan. Pengujian dilakukan untuk memastikan perangkat berfungsi sebagaimana mestinya dalam berbagai skenario penggunaan. Setiap skenario pengujian dilaksanakan sebanyak dua kali, dan sebuah skenario dinyatakan valid apabila terdapat konsistensi antara perintah (skenario) dengan respons (hasil uji). Pada seluruh skenario yang diuji, hasil menunjukkan kesesuaian antara perintah dan respons, mengonfirmasi bahwa semua pengujian valid.

E. *Maintenance*

Pada tahap akhir, setelah produk diimplementasikan, pemeliharaan akan berfokus pada kondisi perangkat seperti kabel dan *relay.* Beberapa masalah muncul selama pengujian atau penerapan alat ini. Masalah pertama adalah *database* ThingSpeak yang hanya dapat menerima data setiap 15 detik. Akibatnya, saat menerima data dari aplikasi seluler, sistem harus menunggu 15 detik lagi sebelum dapat mengirim data kembali dari aplikasi. Masalah berikutnya adalah pada kabel saat sumber *input* atau sinyal ke *relay* dan *output* tidak stabil, menyebabkan *relay* terkadang gagal menyala atau mati.

Berdasarkan hasil pada Tabel 3 dan Tabel 4, integrasi IoT menggunakan ESP32, Kodular, dan ThingSpeak telah berhasil diimplementasikan. Tabel-tabel tersebut menunjukkan bahwa sistem beroperasi dengan konektivitas yang baik, mampu menerima *input* dan mengontrol *output* meskipun dengan batasan waktu 15 detik.

P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

Dalam hal skalabilitas, sistem awalnya dirancang hanya untuk menyalakan dan mematikan lampu. Namun sistem ini berpotensi untuk dikembangkan dengan fitur tambahan seperti pemantauan status dan kontrol lebih banyak perangkat karena konsep dasarnya tetap sama. Dari segi kegunaan, sistem menyediakan antarmuka yang sederhana dan *user-friendly* untuk mengontrol perangkat, serta cara mudah untuk melihat data secara komprehensif karena semua data disimpan dalam format grafik di *database* ThingSpeak. Temuan ini menunjukkan bahwa komponen-komponen bekerja efektif bersama-sama untuk mengontrol dan memantau sistem pencahayaan cerdas. ESP32 berkomunikasi dengan aplikasi seluler yang dibangun menggunakan Kodular dan mengambil/mengirim data melalui ThingSpeak. Integrasi ini menunjukkan kemampuan sistem dalam memungkinkan kontrol jarak jauh dan pemantauan data seperti yang diharapkan.

Namun terdapat keterbatasan yang signifikan dalam implementasi, terutama pada versi gratis ThingSpeak. Tingkat non-berbayar ThingSpeak membatasi transmisi data hanya sekali setiap 15 detik, yang dapat menyebabkan keterlambatan pembaruan status sistem. Keterbatasan ini mungkin membingungkan pengguna tentang kapan mereka dapat menekan tombol kontrol lagi setelah perintah awal. Sebagai solusi untuk mengatasi keterbatasan versi gratis ThingSpeak, penambahan *timer* hitung mundur 15 detik dalam aplikasi seluler dapat membantu mengurangi kebingungan pengguna dengan memberikan umpan balik yang jelas tentang kapan sistem siap menerima perintah berikutnya. *Timer* ini memastikan pengalaman pengguna yang lebih baik tanpa memerlukan biaya tambahan. Alternatif lain, *upgrade* ke paket premium ThingSpeak yang mengurangi interval pembaruan data menjadi 1 detik, menawarkan solusi untuk meningkatkan responsivitas sistem secara signifikan meskipun dengan biaya tambahan.

Kesimpulan

Berdasarkan hasil uji coba, integrasi IoT menggunakan ESP32, Kodular, dan ThingSpeak telah terbukti berhasil, dengan hasil uji coba ini digunakan untuk mengontrol lampu dari jarak 2 km dan secara umum berfungsi baik. Integrasi ini menunjukkan bahwa komponen-komponen tersebut dapat bekerja sama secara efektif untuk mencapai fungsionalitas yang diinginkan dan layak untuk digunakan. Sistem dapat beroperasi dengan baik secara konsisten namun masalah yang muncul berada di luar pembahasan utama, khususnya pada kabel. Meskipun demikian, sistem ini memiliki keterbatasan, terutama pada *database* ThingSpeak yang hanya dapat menerima data setiap 15 detik. Hal ini berpotensi membingungkan pengguna karena mereka tidak mengetahui waktu yang tepat untuk menekan tombol ON atau OFF kembali. Untuk mengatasi hal ini, diusulkan solusi seperti memasang *timer* hitung mundur 15 detik dalam aplikasi atau beralih ke paket premium ThingSpeak untuk transmisi data yang lebih cepat. Optimasi di masa depan perlu difokuskan pada peningkatan kemampuan pemrosesan data *real-time* serta pemilihan dan penggunaan komponen *hardware* yang tepat untuk meningkatkan keandalan sistem.

Ucapan Terima Kasih

Tim pelaksana mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu dalam penelitian serta penulisan artikel ini. Terima kasih kepada dosen pembimbing, Bapak Datu Jatmiko, S.Pd., M.A., yang telah memberikan bimbingan, masukan, dan dukungan selama proses penelitian ini. Tim pelaksana juga berterima kasih kepada Bapak Novan Dimas Eko Jati selaku pemilik kandang merpati karena telah mengizinkan untuk melakukan penelitian dan penempatan alat ini.

P-ISSN 2809-1442
E-ISSN 2829-4645
VOL. 5, NO. 2, NOVEMBER 2025
DOI: https://doi.org/10.21460/servirisma.2025.52.126

Daftar Pustaka

Adriansyah, A. A. (2024). A Study on the Important Role of Databases for Libraries. *Jurnal Ilmiah Nusantara (JINU)*, *1*(4), 488–496. https://doi.org/https://doi.org/10.61722/jinu.v1i4.1819

Ayuningtyas, A. A. (2022). Application of Internet of Things (IoT) in Efforts to Realize a Digital Library in the Era of Society 5.0. *Jurnal Ilmu Perpustakaan*, *11*(1), 29–36.

Furima, Y. A., Naibaho, J. P. P., & Suhendra, C. D. (2022). Learning and playing applications for early childhood using Kodular. *JISTECH: Journal of Information Science and Technology*, *11*(1), 47–58. https://doi.org/10.30862/jistech.v11i1.63

Hercog, D., Lerher, T., Truntič, M., & Težak, O. (2023). Design and Implementation of ESP32-Based IoT Devices. *Sensors*, *23*, 6739. https://doi.org/10.3390/s23156739

Herlianus, & Gunadi, G. (2022). Development of Learning Media for Animal and Human Motion Organs Based on Android Using Kodular. *Informatik : Jurnal Ilmu Komputer*, *18*(1), 88. https://doi.org/10.52958/iftk.v17i4.4605

Khan, S. M. A. (2023). *Waterfall Model Used in Software Development Reference: Software Requirements Engineering Waterfall Model*. 1–4. https://doi.org/10.13140/RG.2.2.29580.69764

Kholifah, U., & Imansari, N. (2022). BUILDING A MOBILE APPLICATION TRAINING USING KODULAR FOR STUDENTS OF SMPN 1 SELOREJO. *Abdimas Galuh*, *4*(1), 549–553. https://doi.org/10.25157/ag.v4i1.7259

Ramisetty, S. V., Akkineni, P., Sai, O. G., & P, P. S. (2020). Home Automation using Thingspeak. *International Journal of Innovative Technology and Exploring Engineering*, *10*(1), 219–223. https://doi.org/10.35940/ijitee.a8171.1110120

Ridwan, M., & Sari, K. M. (2021). Application of IoT for Automated Controlling System of Temperature, Humidity, and Acidity in Hydroponics. *Jurnal Teknik Pertanian Lampung*, *10*(4), 481–487. https://doi.org/10.23960/jtep-l.v10i4.481-487

Senarath, U. S. (2021). *Waterfall Methodology, Prototyping and Agile Development*. 1–25. https://doi.org/10.13140/RG.2.2.17918.72001

Setiawan, R. (2020). Design and Development of Android-Based Learning Media Without Coding as Easy as Assembling a Puzzle. *Jurnal Sistem Informasi Dan Sains Teknologi*, *2*(2), 1–7. https://doi.org/https://doi.org/10.31326/sistek.v2i2.729

Tamrakar, A. K., Shukla, A., Kalifullah, A. H., Reegu, F. A., & Shukla, K. (2022). Extended review on internet of things (IoT) and its characterisation. *International Journal of Health Sciences*, *6*(S2), 8490–8500. https://doi.org/10.53730/ijhs.v6ns2.7177

Wahid, A. A. (2020). Analysis of the Waterfall Method for Information System Development. *Jurnal Ilmu-Ilmu Informatika Dan Manajemen STMIK*.

Yulisman, Ikhsan, I., Febriani, A., & Melyanti, R. (2021). Application of Internet of Things (IoT) for Light Control Using NodeMCU ESP8266 and Smartphone. *Jurnal Ilmu Komputer*, *10*(2), 136–143. https://doi.org/10.33060/jik/2021/vol10.iss2.231