Jurnal Ilmu Komputer Agri-Informatika

Tersedia secara daring di:
https://jurnal.ipb.ac.id/index.php/jika

Volume 12 Nomor 2 halaman 155–166
eISSN: 2654-9735, pISSN: 2089-6026

Pengembangan Sistem Terintegrasi *Real-time Monitoring* dan Irigasi Cerdas pada Sawah IPB 4.0

Development of an Integrated Real-time Monitoring and Smart Irrigation System in SAWAH IPB 4.0

ABI SUPIYANDI, KARLISA PRIANDANA*, HENDRA RAHMAWAN

Abstrak

Sawah IPB 4.0 merupakan konsep *smart farming* yang dikembangkan oleh IPB sebagai model pertanian padi modern di Indonesia. Berbagai teknologi diterapkan dalam konsep ini, antara lain *real-time monitoring*, irigasi cerdas, mekanisasi, perlindungan tanaman cerdas, dan *multi-UAV*. Pada penelitian sebelumnya, dua teknologi utama, yaitu *real-time monitoring* dan irigasi cerdas, telah dikembangkan dan diimplementasikan dalam bentuk prototipe berbasis *Internet of Things* (IoT). Namun, kedua sistem tersebut masih beroperasi secara terpisah dan belum dilengkapi dengan fitur manajemen perangkat yang terintegrasi. Penelitian ini bertujuan untuk mengintegrasikan kedua sistem tersebut melalui pengembangan modul kontrol manual dan modul manajemen perangkat. Pengembangan sistem dilakukan menggunakan metode *prototyping* dan diimplementasikan dalam bentuk aplikasi mobile dengan pola arsitektur *Model–View–ViewModel* (MVVM) serta desain *single activity architecture*. Evaluasi sistem dilakukan melalui pengujian *black box* untuk menguji fungsionalitas dan pengujian kinerja yang berfokus pada waktu eksekusi. Hasil pengujian *black box* menunjukkan bahwa seluruh fungsi aplikasi berjalan dengan baik, sedangkan pengujian kinerja menunjukkan bahwa waktu respons aplikasi bervariasi bergantung pada fungsi yang dijalankan.

Kata Kunci**:** integrasi, irigasi cerdas, *real-time monitoring*, Sawah IPB 4.0

Abstract

Sawah IPB 4.0 is a smart farming concept developed by IPB University as a model for modern rice cultivation in Indonesia. This concept integrates various technologies, including real-time monitoring, smart irrigation, mechanization, smart plant protection, and multi-UAV systems. In previous studies, two core technologies—real-time monitoring and smart irrigation—were developed and implemented as Internet of Things (IoT)-based prototypes. However, these systems operated independently and lacked integrated device management capabilities. This study aims to integrate the two previously separated systems by developing a manual control module and a device management module. The system was developed using the prototyping method and implemented as a mobile application based on the Model–View–ViewModel (MVVM) architectural pattern with a single-activity architecture design. System evaluation was conducted through black-box testing to assess functionality and performance testing focused on execution time. The results of black-box testing indicate that all application features function correctly, while performance testing shows that application response times vary depending on the executed functions.

*Keywords**:** integration, real-time monitoring, Sawah IPB 4.0, smart irrigation*

PENDAHULUAN

Sawah IPB 4.0 merupakan salah satu implementasi *smart farming* pada lahan sawah yang menjadi bagian dari inisiatif Agro-Maritim IPB 4.0. Proyek ini dicirikan oleh penggunaan teknologi 4.0 (Institut Pertanian Bogor 2019). Arsitektur Sawah IPB 4.0 mencakup berbagai aplikasi utama, seperti pemantauan waktu nyata (*real-time monitoring*), irigasi cerdas,

Program Studi Ilmu Komputer, Sekolah Sains Data, Matematika, dan Informatika, Institut Pertanian Bogor, Bogor 16680;
*Penulis Korespondensi: Tel/Faks: 0251-8625584, Surel: karlisa@apps.ipb.ac.id

mekanisasi, perlindungan tanaman cerdas, serta penggunaan multi-UAV. Implementasi ini didukung oleh integrasi teknologi sensor, pengindraan jauh (*remote sensing*), kecerdasan buatan (*artificial intelligence*), teknologi awan (*cloud technology*), dan *Internet of Things* (IoT).

Salah satu penelitian mengenai sistem irigasi cerdas dilakukan oleh Priandana et al. (2024). Sistem tersebut dikembangkan menggunakan dua jenis *node* utama: *node* sensor dan *node* aktuator (Hafiduddin 2022). *Node* sensor mengintegrasikan komponen multivariabel untuk mengukur parameter tanah, curah hujan, suhu, kelembaban, serta jarak melalui sensor ultrasonik. Sementara itu, *node* aktuator dilengkapi motor servo yang berfungsi sebagai katup otomatis untuk mengatur aliran air. Perangkat IoT ini terhubung ke layanan *cloud* Thinger.io melalui internet dengan memanfaatkan *Application Programming Interface* (API) *endpoint* sebagai jalur komunikasi data. Untuk mengoptimalkan otomatisasi pengairan, sistem ini mengimplementasikan algoritma *Fuzzy Inference System* (FIS) sebagai basis pengambilan keputusan cerdas.

Selain sistem irigasi, pengembangan sistem pemantauan waktu nyata (*real-time monitoring*) berbasis seluler juga dilakukan oleh Mochtar (2023). Penelitian ini berfokus pada pengembangan aplikasi *mobile* yang memungkinkan pengguna memantau kondisi lahan sawah secara praktis. Secara teknis, sistem ini mengadopsi arsitektur perangkat keras dan layanan *cloud* yang selaras dengan sistem irigasi cerdas sebelumnya. Infrastruktur perangkat keras tetap mengandalkan dua jenis *node*, yaitu sensor untuk akuisisi data dan aktuator untuk tindakan fisik. Data yang dikumpulkan kemudian disimpan dan dikelola melalui *platform cloud* Thinger.io, yang juga berfungsi sebagai jembatan informasi menuju aplikasi pengguna. Komunikasi antara aplikasi *mobile* dengan *server* Thinger.io dilakukan secara terintegrasi melalui API *endpoint* untuk memastikan pertukaran data yang cepat dan akurat.

Meskipun kedua *prototype* sistem tersebut dapat berfungsi dengan baik, keduanya masih beroperasi secara terpisah. Sistem irigasi cerdas, misalnya, mampu melakukan irigasi otomatis namun belum dilengkapi fungsi pemantauan (*monitoring*). Sebaliknya, sistem pemantauan berbasis *mobile* hanya terbatas pada fungsi observasi tanpa kemampuan kendali (*control*). Guna mengatasi keterbatasan tersebut, diperlukan integrasi sistem untuk menciptakan satu kesatuan fungsi yang terpadu. Menurut García et al. (2020), integrasi mencakup penggabungan perangkat keras, perangkat lunak, maupun data guna memastikan seluruh komponen bekerja secara sinkron. Tujuan utama dari langkah ini adalah menciptakan efisiensi yang lebih tinggi dibandingkan komponen yang bekerja secara individual, sekaligus meningkatkan fungsionalitas sistem secara keseluruhan (Bellman *et al.* 2019).

Strategi integrasi yang diterapkan dalam penelitian ini adalah pengembangan aplikasi berbasis *mobile* yang mengonsolidasikan seluruh fungsi dari sistem sebelumnya. Dengan menjadikan sistem *real-time monitoring* sebagai fondasi, modul kontrol manual dikembangkan untuk memberikan fleksibilitas tambahan dalam merespons kondisi lapangan (Devare dan Hajare 2019; Solomakha dan Khizhnyak 2020). Fleksibilitas ini sangat penting, terutama pada siklus budidaya sawah yang tidak terakomodasi oleh aturan otomatisasi yang sudah ada.

Menurut Hartono et al. (2017), siklus padi sawah di Indonesia meliputi fase pembibitan, penanaman, pertumbuhan, hingga panen, yang didahului oleh persiapan lahan seperti membajak dan menggaru (Singh *et al.* 2016). Saat ini, sistem yang ada baru mampu mengairi lahan pada fase pertumbuhan dan panen saja. Oleh karena itu, fitur kontrol manual dikembangkan sebagai mekanisme cadangan (*backup*) untuk mengatasi kondisi darurat (Rohith et al. 2021; Agarwal dan Rai 2022), serta memfasilitasi kebutuhan perawatan atau kalibrasi perangkat di lapangan (Vijayakumar 2022).

Salah satu fitur utama yang dikembangkan dalam aplikasi ini adalah modul manajemen perangkat. Fitur ini menjadi krusial mengingat jumlah petak sawah yang dikelola dapat berubah sewaktu-waktu, yang berimplikasi pada fluktuasi jumlah perangkat di lapangan. Saat ini, sistem yang ada belum memiliki fleksibilitas untuk menambah atau mengurangi perangkat secara dinamis. Meskipun sistem *real-time monitoring* telah menyediakan tombol akses untuk beberapa lahan, tombol-tombol tersebut masih bersifat *hard-coded* (statis). Akibatnya, setiap

penambahan lahan baru memerlukan pembaruan kode sumber dan instalasi ulang aplikasi, yang tentu saja tidak efisien bagi pengguna.

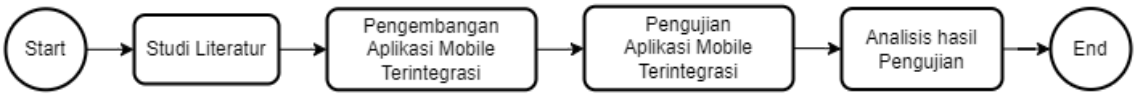
Seiring dengan meningkatnya kompleksitas sistem melalui penambahan fungsi-fungsi baru, aspek keamanan menjadi prioritas yang tidak dapat diabaikan. Dalam penelitian ini, penguatan aspek keamanan diusulkan melalui skema pembagian tugas atau hak akses di antara pengguna aplikasi. Oleh karena itu, dilakukan analisis mendalam terhadap kebutuhan pengguna guna memetakan peran masing-masing. Hasil analisis ini akan menjadi fondasi bagi pengembangan fitur keamanan pada penelitian selanjutnya.

Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk memberikan solusi melalui dua fokus utama. Fokus pertama adalah pengembangan modul kontrol manual pada aplikasi yang telah tersedia guna meningkatkan fleksibilitas operasional. Fokus kedua adalah pengembangan modul manajemen perangkat sebagai pelengkap fungsional, sehingga sistem dapat beradaptasi dengan perubahan jumlah perangkat di lapangan secara lebih dinamis.

METODE

Instrumen penelitian meliputi *node* sensor dan *node* aktuator yang dikembangkan pada penelitian sebelumnya (Priandana 2024; Hafiduddin 2022). Untuk pengembangan aplikasi berbasis *mobile*, digunakan Android Studio versi Koala 2024.1.1, sedangkan implementasi kode program pada *node* perangkat menggunakan Arduino IDE versi 2.3.2. Prosedur penelitian terdiri dari empat tahapan utama, yaitu: (1) studi literatur, (2) pengembangan aplikasi, (3) pengujian aplikasi, dan (4) analisis hasil pengujian, sebagaimana disajikan pada Gambar 1.

Tahap studi literatur bertujuan untuk memetakan sejauh mana sistem yang telah ada dikembangkan, sekaligus mendefinisikan fungsi-fungsi baru yang akan diintegrasikan. Pengembangan aplikasi kemudian dilakukan menggunakan metode *prototyping*. Proses pengumpulan kebutuhan (*requirements gathering*) dilaksanakan secara berkesinambungan dengan studi literatur terhadap arsitektur sistem yang sudah ada. Selanjutnya, pembuatan *prototype* awal dilakukan dengan merujuk pada desain asli, dibarengi dengan perbaikan *minor* untuk menjaga konsistensi arsitektur. Tahap terakhir dalam siklus ini adalah evaluasi terhadap *prototype* awal, yang diikuti dengan perbaikan (*refining*) berdasarkan umpan balik evaluasi guna memastikan sistem memenuhi spesifikasi yang diharapkan.



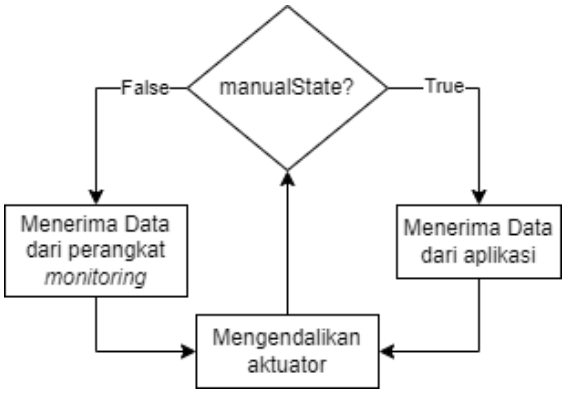
Gambar 1 Tahapan penelitian

Pengujian fungsional aplikasi dilakukan dengan menggunakan metode *black-box testing.* Metode ini bertujuan untuk memverifikasi setiap fungsi di dalam aplikasi guna memastikan seluruh fitur dapat berjalan sesuai spesifikasi tanpa melihat detail kode programnya. Dokumentasi pengujian disusun secara sistematis dalam tabel kasus uji. Tabel tersebut mencakup informasi komprehensif yang meliputi deskripsi fitur, skenario pengujian, hasil yang diharapkan, serta hasil aktual berupa status keberhasilan atau kegagalan (*pass/fail*). Setiap baris dalam tabel mewakili satu kasus uji spesifik untuk setiap fungsi yang dikembangkan.

Selain pengujian fungsional, pengujian performa juga dilakukan untuk mengukur efisiensi aplikasi. Parameter utama yang diukur adalah waktu eksekusi, yaitu durasi yang dibutuhkan aplikasi untuk memberikan respons terhadap masukan pengguna. Pengukuran ini dilakukan dengan memanfaatkan fitur *Network Inspector* pada Android Studio. Alat ini memungkinkan pemantauan lalu lintas data secara waktu nyata (*real-time*) sehingga waktu yang diperlukan untuk pertukaran data antara aplikasi, layanan *cloud*, dan perangkat dapat diukur secara akurat.

HASIL DAN PEMBAHASAN

Integrasi input kontrol dari aplikasi menuntut adanya modifikasi pada logika operasional perangkat aktuator. Pembaruan ini dilakukan dengan mengimplementasikan variabel *manualState* dan meningkatkan kemampuan perangkat untuk memproses input dari berbagai sumber data secara simultan. Algoritma baru dikembangkan untuk menggantikan logika pada sistem sebelumnya, yang strukturnya divisualisasikan dalam bentuk diagram alir pada Gambar 2. Berdasarkan diagram tersebut, langkah awal yang dilakukan oleh algoritma adalah melakukan verifikasi terhadap status *manualState*. Jika *manualState* bernilai *true* (benar), perangkat akan memprioritaskan instruksi yang diterima langsung dari aplikasi untuk mengendalikan aktuator. Sebaliknya, jika *manualState* bernilai *false* (salah), perangkat akan beralih ke mode otomatis dengan mengendalikan aktuator berdasarkan data sensor yang diterima dari perangkat *monitoring*. Mekanisme ini memastikan adanya fleksibilitas kendali antara sistem otomatis dan intervensi manual pengguna.



Gambar 2 Algoritma baru untuk kontrol aktuator

Strategi integrasi dalam penelitian ini menitikberatkan pada aspek skalabilitas, yang memungkinkan pengelolaan perangkat di setiap petak sawah dilakukan secara dinamis melalui aplikasi. Mekanisme manajemen perangkat, seperti fitur penambahan dan penghapusan, diintegrasikan untuk meningkatkan fleksibilitas sistem. Mengingat adanya penambahan fungsi kendali dan pengelolaan tersebut, penerapan fitur keamanan menjadi sangat krusial. Oleh karena itu, penelitian ini mengimplementasikan mekanisme otorisasi pengguna guna membatasi akses terhadap fungsi-fungsi kritis berdasarkan hasil analisis kebutuhan pengguna.

Analisis kebutuhan dalam penelitian ini mencakup analisis pengguna dan analisis fungsional. Langkah ini dilakukan untuk mengatasi keterbatasan pada sistem sebelumnya yang hanya berfokus pada fungsi pemantauan (*monitoring*) tanpa adanya kemampuan kendali (*control*) maupun manajemen perangkat. Selain itu, sistem terdahulu belum mendefinisikan peran pengguna secara spesifik. Analisis ini berhasil mengidentifikasi tiga peran utama, yaitu *admin*, *operator*, dan *guest*, di mana masing-masing peran memiliki hak akses yang berbeda terhadap fitur aplikasi.

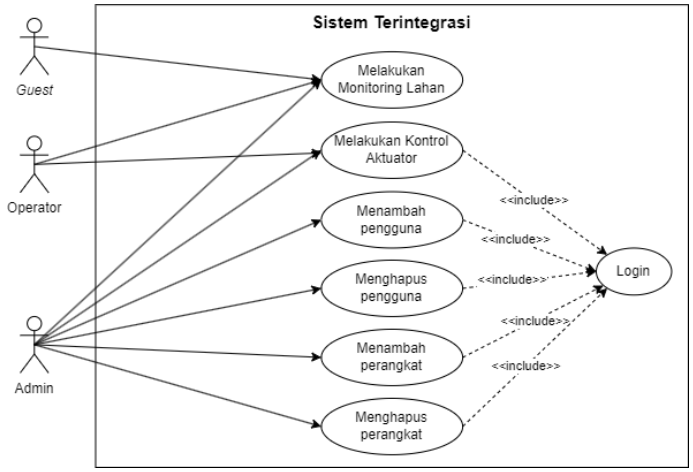
Hierarki pengguna disusun berdasarkan tanggung jawab dan tingkat risiko akses. *Admin* memiliki otoritas tertinggi dengan akses penuh ke seluruh fungsi, termasuk manajemen pengguna dan konfigurasi perangkat. *Operator* memiliki tanggung jawab operasional untuk mengontrol aktuator; meskipun secara *default* sistem berjalan otomatis menggunakan *Fuzzy Inference System* (FIS), *operator* diberikan wewenang untuk melakukan intervensi manual sesuai kondisi lapangan. Sementara itu, *guest* berada pada hierarki terendah dengan akses terbatas hanya untuk pemantauan lahan tanpa proses *login*, guna menyediakan informasi cepat tanpa risiko perubahan konfigurasi sistem.

Seluruh analisis kebutuhan fungsional ini dimodelkan menggunakan *Unified Modelling Language* (UML) dalam bentuk *use case diagram* dan *sequence diagram*. Detail mengenai daftar *use case* aplikasi disajikan pada Tabel 1.

Tabel 1 Daftar *use case* dalam aplikasi terintergrasi

Nama *Use Case*	Deskripsi *Use Case*	Pengguna
Login ke aplikasi	Melakukan *login* ke aplikasi.	Admin, Operator
Menambah operator	Menambah *user* dengan peran operator ke dalam sistem.	Admin
Menghapus operator	Menghapus *user* dengan peran operator dari sistem.	Admin
Melakukan pemantauan	Melakukan pemantauan terhadap kondisi lahan.	Admin, Operator, *Guest*
Melakukan kontrol terhadap aktuator	Melakukan kontrol terhadap aktuator di lahan.	Admin, Operator
Menambah perangkat ke dalam sistem	Mendaftarkan perangkat baru ke dalam sistem.	Admin
Menghapus perangkat dari sistem	Menghapus perangkat yang telah terdaftar di sistem.	Admin

Dari Tabel 1 dapat dibuat suatu diagram *use case*. Diagram *use case* ini disajikan pada Gambar 3. Pada diagram ini, terdapat satu relasi <<include>> antar *use case*, yaitu pada kontrol aktuator, menambah pengguna, menghapus pengguna, menambah perangkat, dan menghapus perangkat kepada *use case* login. Relasi ini memiliki arti bahwa *use case* login adalah bagian yang harus dijalankan sebelum *use case* melakukan kontrol aktuator, menambah pengguna, menghapus pengguna, menambah perangkat, dan menghapus perangkat dapat dieksekusi. Dengan kata lain, pengguna harus melakukan *login* terlebih dahulu untuk dapat menjalankan lima *use case* tersebut.



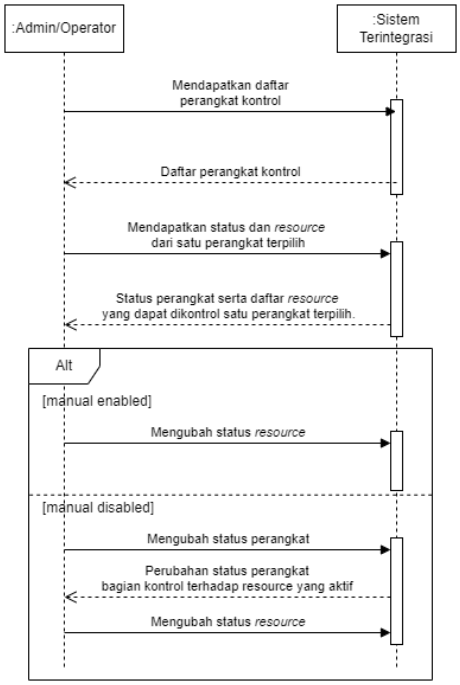
Gambar 3 Diagram *use case* untuk pengguna admin, operator, dan *guest*

Pengguna dengan peran *admin* diwajibkan melakukan *login* untuk mengakses sistem. Setelah otentikasi berhasil, sistem akan menampilkan halaman *dashboard* yang mengintegrasikan fungsi pemantauan lahan, kontrol aktuator, serta manajemen pengguna dan perangkat. Fitur manajemen ini mencakup kemampuan untuk membuat (*create*), memperbarui (*update*), dan menghapus (*delete*) data pengguna maupun perangkat di dalam ekosistem Sawah IPB 4.0.

Sementara itu, *operator* memiliki hak akses untuk memantau kondisi lahan dan mengendalikan aktuator setelah melalui proses *login*. Berbeda dengan kedua peran sebelumnya, pengguna *guest* dirancang untuk akses informasi cepat; mereka dapat langsung memantau kondisi lahan tanpa harus melakukan proses *login*, namun tidak memiliki otoritas untuk melakukan perubahan pada sistem.

Alur interaksi kontrol perangkat digambarkan melalui *sequence diagram* pada Gambar 4. Fungsi kontrol ini eksklusif bagi *admin* dan *operator*. Melalui halaman utama, pengguna dapat mengakses daftar perangkat kontrol dan memilih unit tertentu untuk melihat detail status serta *resource* yang tersedia. Terdapat dua status operasional perangkat: *manual enabled* dan *manual disabled*. Status *manual enabled* menunjukkan bahwa perangkat berada dalam mode manual, sehingga pengguna memiliki kendali penuh untuk mengubah status *resource*. Sebaliknya, status *manual disabled* menandakan perangkat beroperasi secara otomatis di bawah kendali algoritme *fuzzy*. Jika pengguna ingin melakukan intervensi manual pada perangkat yang berstatus *manual*

disabled, sistem mengharuskan pengguna mengubah status menjadi *manual enabled* terlebih dahulu sebelum perintah kontrol dapat dieksekusi.



Gambar 4 Diagram *sequence* untuk fungsi menambah perangkat

Prototipe aplikasi *mobile* mengimplementasikan *single activity architecture* sebagai arsitektur strukturalnya. Pada *prototype* hanya terdapat satu *activity* sebagai wadah tampilan halaman atau antarmuka (*destination*). Setiap halaman diimplementasikan dalam suatu fragmen yang dikontrol dari *activity*. Navigasi antar fragmen dilakukan dengan mendefinisikan suatu graf navigasi. Terdapat 7 fragmen untuk setiap halaman yaitu: (1) HomeFragment; (2) monitorListFragment; (3) monitorDetailFragment; (4) controlListFragment; (5) controlPageFragment; (6) deviceListFragment; dan (7) addDeviceFragment. Sebagai contoh, halaman *monitoring* ditangani oleh fragmen MonitorListFragment. Halaman tersebut menampilkan daftar perangkat *monitoring* yang ada di dalam sistem.

Design pattern MVVM digunakan untuk menerapkan prinsip *separation of concern* dalam pengembangan aplikasi. *Model* (M) merepresentasikan data objek sebagai tanggapan dari sebuah pemanggilan API *endpoint*. Pemanggilan API *endpoint* sendiri dilakukan oleh *Viewmodel* (VM) yang mengimplementasikan *business logic* aplikasi dalam bentuk fungsi. Implementasi VM pada aplikasi dikembangkan dalam suatu berkas yang digunakan bersama yaitu SharedViewModel. Data yang dihasilkan dari VM tersebut kemudian digunakan oleh *View* (V) untuk ditampilkan dalam suatu halaman antarmuka. Pada aplikasi, V diimplementasikan dalam bentuk fragmen.

Sebagai contoh implementasi MVVM pada aplikasi adalah pada saat aplikasi menampilkan daftar perangkat *monitoring*. Fungsi refreshDevices dalam SharedViewModel melakukan pemanggilan API *endpoint* ListDevice. Fungsi ini mengembalikan suatu daftar yang berisikan objek dari model GetDeviceResponseItem. Daftar tersebut kemudian digunakan oleh fragmen MonitorListFragment untuk ditampilkan menggunakan tata letak tertentu.

Sebagai penghubung antara perangkat dan aplikasi, digunakan API Thinger.io. Sama dengan kedua sistem eksisting, komunikasi dilakukan oleh perangkat dan aplikasi dengan mengakses API *endpoint.* Pada sistem eksisting, hanya terdapat satu *endpoint* dengan nama AccessDeviceResources. *Endpoint* ini digunakan untuk mendapatkan nilai suatu *resource* yang ada di perangkat *monitoring*. Sistem baru mengimplementasikan beberapa *endpoint* tambahan untuk mendukung fungsionalitas sistem, terutama untuk mengakses *resource* dan *properties* yang ada pada perangkat aktuator. *Resource* dalam hal ini merupakan suatu data *real-time* yang berasal dari perangkat seperti nilai bacaan sensor, sedangkan *properties* merupakan data

persisten atau konfigurasi perangkat yang tersimpan di *server cloud*. Terdapat tambahan lima *endpoint* Thinger.io pada sistem baru yaitu: (1) ListDevice; (2) ReadDeviceProperty; (3) UpdateDeviceProperty; (4) UpdateDeviceResource; dan (5) addDevice. Terdapat tiga tipe dari *endpoint* yaitu *get*, *put*, dan *post*. *Endpoint* tipe *get* digunakan untuk mendapatkan nilai tertentu dari *server* atau perangkat. Tipe *put* dan *post* digunakan untuk menetapkan suatu nilai yang ada di server atau di perangkat. Keterangan detail dari setiap *endpoint* disajikan pada Tabel 2.

Tabel 2 Penggunaan *endpoint* pada sistem eksisting dan sistem baru

Nama *Endpoint*	Tipe	Link	Keterangan
AccesDeviceResource	*Get*	/v3/users/{user}/devices/{device}/resources/{resource}	Mengembalikan nilai *resource* dari perangkat.
ListDevice	*Get*	/v1/users/{user}/device	Mengembalikan semua perangkat yang terdaftar di sistem.
ReadDeviceProperty	*Get*	/v3/users/{user}/devices/{device}/properties/{property}	Mengembalikan nilai dari *property* tertentu di perangkat yang terdaftar di sistem.
UpdateDeviceProperty	*Put*	/v3/users/{user}/devices/{device}/properties/{property}	Menetapkan nilai dari *property* tertentu di perangkat yang terdaftar di sistem.
UpdateDeviceResource	*Post*	/v3/users/{userId}/devices/{deviceId}/resource/{resourceId}	Menetapkan nilai dari *resource* tertentu di perangkat yang terdaftar di dalam sistem.
AddDevice	*Post*	/v1/users/{userId}/devices	Mendaftarkan perangkat baru ke sistem.

Iterasi pertama

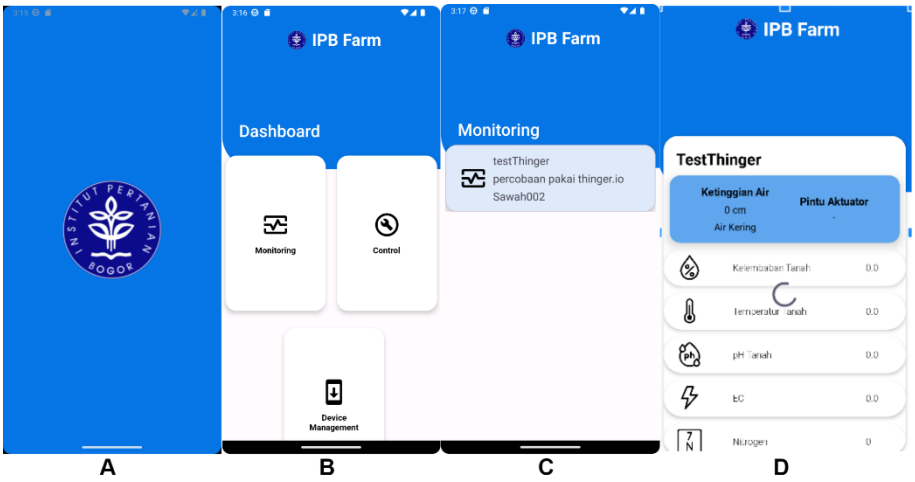
Iterasi pertama pengembangan aplikasi difokuskan pada pembangunan struktur navigasi utama dan repositori fungsi pemantauan. Salah satu pembaruan signifikan adalah implementasi ulang fungsi *splash screen* dengan memanfaatkan fitur *Theme* sebagai basis. Metode ini mengeliminasi kebutuhan akan berkas *layout* dan kode Kotlin khusus, sehingga komponen tersebut dapat dikeluarkan dari grafik navigasi (*navigation graph*). Pendekatan ini lebih efisien dibandingkan rancangan pada sistem sebelumnya yang masih menggunakan *layout* manual.

Fungsi pemantauan (*monitoring*) dari sistem lama diintegrasikan kembali ke dalam arsitektur baru melalui dua *endpoint* utama: ListDevice untuk mengakuisisi daftar perangkat dan AccessDeviceResource untuk membaca nilai data sensor pada *node* sensor. Pada *endpoint* ListDevice, sistem menerapkan pemfilteran otomatis agar hanya menampilkan daftar perangkat pemantauan.

Pengembangan dilanjutkan dengan implementasi antarmuka halaman utama (*Home*) yang menyediakan akses ke menu pemantauan, kontrol, dan manajemen perangkat. Berbeda dengan sistem lama yang bersifat *hard-coded* (statis), daftar perangkat pada iterasi ini diimplementasikan secara dinamis berdasarkan data yang terdaftar di server. Secara keseluruhan, iterasi pertama menghasilkan empat halaman purwarupa utama: *splash screen*, halaman utama, daftar perangkat dinamis, dan detail pemantauan. Dokumentasi hasil pengembangan ini disajikan melalui tangkapan layar purwarupa pada Gambar 5.

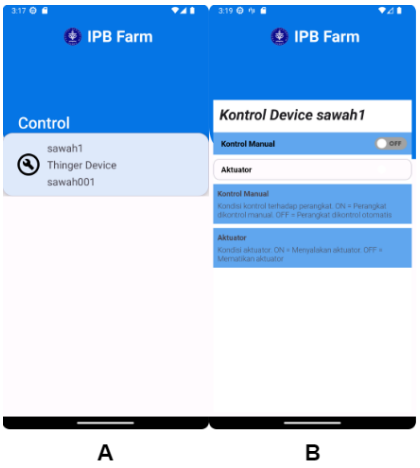
Iterasi kedua

Identifikasi kebutuhan pada iterasi kedua berfokus pada pengembangan modul kontrol, yang meliputi halaman daftar perangkat kontrol dan halaman kendali aktuator. Penambahan fungsi ini bertujuan agar aplikasi terintegrasi memiliki kemampuan penuh dalam mengoperasikan unit aktuator di lapangan. Secara teknis, pengembangan halaman daftar perangkat kontrol mengadopsi struktur yang serupa dengan modul pemantauan pada iterasi sebelumnya. Halaman ini diimplementasikan menggunakan *fragment* dengan pustaka **Epoxy** untuk manajemen komponen antarmuka yang dinamis. Perbedaan utamanya terletak pada penerapan filter data yang dikonfigurasi khusus untuk menampilkan kategori perangkat kontrol saja.



Gambar 5 Tangkapan layar prototipe iterasi pertama

Operasional pada iterasi kedua ini mengintegrasikan empat *endpoint* utama dari API Thinger.io. *Endpoint* ListDevice digunakan untuk mengakuisisi daftar seluruh perangkat yang terdaftar di server. Untuk manajemen mode kerja, digunakan ReadDeviceProperty untuk membaca status operasional aktuator dan UpdateDeviceProperty untuk mengubah status tersebut (misalnya beralih antara mode otomatis dan manual). Terakhir, *endpoint* UpdateDeviceResource digunakan sebagai jalur instruksi untuk mengontrol *solenoid valve* secara langsung. Hasil pengembangan pada iterasi kedua ini mencakup prototipe fungsional untuk daftar perangkat kontrol dan antarmuka kendali aktuator, sebagaimana didokumentasikan pada Gambar 6.

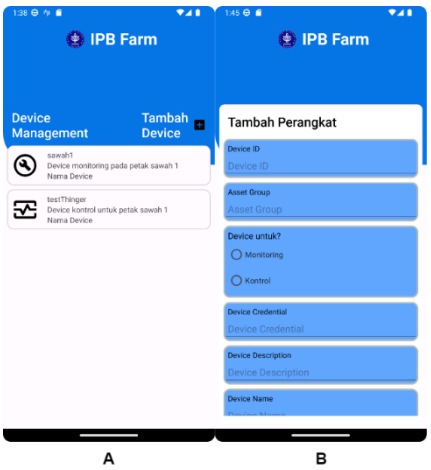


Gambar 6 Tangkapan layar hasil iterasi kedua

Iterasi ketiga

Iterasi ketiga difokuskan pada pengembangan modul manajemen perangkat yang mencakup halaman daftar perangkat menyeluruh dan fungsi penambahan perangkat baru. Implementasi fitur ini bertujuan untuk memberikan kemudahan bagi pengguna dalam mengelola infrastruktur perangkat keras langsung melalui aplikasi *mobile*. Berbeda dengan iterasi sebelumnya, halaman daftar perangkat pada tahap ini memanfaatkan *endpoint* ListDevice tanpa filter, sehingga mampu menampilkan seluruh inventaris perangkat pemantauan maupun kontrol secara komprehensif.

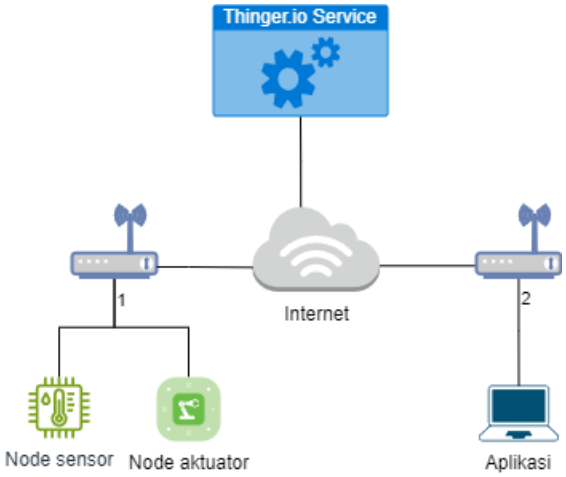
Selain fungsi pemantauan, modul ini juga menyediakan formulir entri data (*input form*) yang diperlukan untuk mendaftarkan perangkat baru ke dalam ekosistem sistem. Proses pendaftaran ini memungkinkan skalabilitas sistem yang lebih baik saat terjadi penambahan petak lahan sawah. Secara keseluruhan, hasil pengembangan pada iterasi ketiga mencakup halaman manajemen perangkat yang terintegrasi dan fungsi pendaftaran perangkat dinamis. Dokumentasi purwarupa untuk iterasi ini disajikan pada Gambar 7.



Gambar 7 Tangkapan layar hasil iterasi ketiga

Skenario pengujian dilakukan sebagaimana diilustrasikan pada Gambar 8, dengan menggunakan satu unit node sensor dan satu unit node aktuator. Secara infrastruktur, kedua perangkat tersebut terhubung ke internet melalui modem WiFi, sementara aplikasi pada emulator Android Studio menggunakan jaringan internet kampus IPB University. Perangkat keras yang digunakan merupakan instrumen dari penelitian sebelumnya, namun dengan firmware mikrokontroler yang telah diperbarui, khususnya pada bagian kredensial, protokol koneksi jaringan, serta optimasi algoritma pada node aktuator.

Kredensial pada setiap *node* mencakup parameter *username*, *device_id*, dan *device_credential* yang berfungsi sebagai basis otentikasi saat melakukan jabat tangan (*handshake*) dengan server Thinger.io. Pengaturan ini memastikan bahwa setiap pertukaran data antara perangkat keras dan aplikasi melalui platform *cloud* terjamin keamanannya dan terarah pada *device* yang tepat.



Gambar 8 Skema pengujian sistem

Pengujian dilakukan dengan mencoba setiap fungsi pada aplikasi. Fungsi tersebut dikelompokkan menjadi tiga, yaitu *monitoring*, kontrol, dan manajemen perangkat. Contoh tabel pengujian untuk fungsi pada kelompok kontrol disajikan pada Tabel 3. Sementara itu, pengukuran performa aplikasi dilakukan menggunakan *network profiler* (bagian dari Android *Profiler*) pada Android Studio. Proses pengujian adalah dengan memanggil fungsi di *emulator* aplikasi, *network profiler* akan menunjukkan waktu eksekusi fungsi tersebut.

Tabel 3 Hasil pengujian kelompok fungsi kontrol

Deskripsi	Skenario	Hasil yang diharapkan	Hasil
Akses terhadap halaman kontrol	Pengguna mengakses halaman kontrol dari *homescreen.*	Aplikasi berpindah halaman menjadi halaman kontrol dan menampilkan daftar perangkat yang dapat dikontrol.	Berhasil
Akses terhadap halaman kontrol suatu perangkat kontrol tertentu	Pengguna mengakses halaman kontrol perangkat tertentu dengan status manual kontrol bernilai *False.*	Halaman kontrol perangkat terpilih ditampilkan, saklar manual kontrol berada di posisi *OFF*, dan saklar untuk mengontrol aktuator tidak aktif.	Berhasil
Akses terhadap halaman kontrol suatu perangkat kontrol tertentu	Pengguna mengakses halaman kontrol perangkat tertentu dengan status manual kontrol bernilai *True*	Halaman kontrol untuk perangkat terpilih ditampilkan, saklar manual kontrol berada di posisi *ON*, dan saklar aktuator aktif, berada pada posisi *OFF.*	Berhasil
Perubahan mode kerja perangkat aktuator	Pengguna mengubah mode kerja perangkat aktuator dari *OFF* menjadi *ON.*	Saklar mode kerja bergeser ke kanan. Saklar untuk mengontrol aktuator menjadi aktif dan dapat diakses. LCD pada perangkat menampilkan nilai manualState=1.	Berhasil
Perubahan mode kerja perangkat aktuator	Pengguna mengubah mode kerja perangkat aktuator dari *ON* menjadi *OFF*.	Saklar mode kerja bergeser ke kiri. Saklar untuk mengontrol aktuator menjadi tidak dapat diakses. LCD pada perangkat menampilkan nilai manualState=0.	Berhasil
Pengguna mengubah kondisi aktuator pada perangkat	Pengguna mengubah kondisi aktuator dari *OFF* menjadi *ON*.	Saklar bergeser ke kanan dan menampilkan tulisan *ON*. LCD pada *node* aktuator menampilkan nilai *open percentage* 100% (90 derajat).	Berhasil
Pengguna mengubah kondisi aktuator pada perangkat	Penguna mengubah kondisi aktuator dari *ON* menjadi *OFF*.	Saklar bergeser ke kiri dan menampilkan tulisan *OFF*. LCD pada perangkat aktuator menampilkan nilai *open percentage* 0% (0 derajat).	Berhasil

Pengukuran dilakukan pada fitur untuk nampilkan daftar perangkat *monitoring*, daftar perangkat kontrol, daftar seluruh perangkat, mengubah mode manual, serta mengubah aktuator. Fitur untuk menampilkan daftar perangkat memiliki waktu eksekusi yang seragam dengan rata-rata waktu eksekusi di bawah satu detik. Fitur mengubah mode manual dan aktuator memiliki waktu eksekusi lebih tinggi dengan nilai lebih dari satu detik. Waktu eksekusi terlama adalah pada fungsi mengubah aktuator dengan rata-rata 6.75 detik dan standar deviasi 2.83 detik. Hal ini kemungkinan disebabkan karena rantai komunikasi yang lebih panjang antara API *server* dengan perangkat. Pada fitur untuk menampilkan perangkat, aplikasi hanya berkomunikasi dengan API *server*. Sedangkan untuk mengubah aktuator, aplikasi berkomunikasi hingga ke perangkat aktuator melalui API *server*. Hasil pengujian performa menggunakan waktu eksekusi fitur pada aplikasi disajikan pada Tabel 4.

Tabel 4 Hasil pengujian performa

	Waktu eksekusi fungsi (s)				
Percobaan ke-	*List Monitoring Device*	*List Control Device*	*List Device Management*	*Change Working Mode*	*Change Actuator*
1	0.63	0.71	0.63	1.12	9.52
2	0.65	0.55	0.73	1.51	7.99
3	0.59	0.71	0.76	1.19	8.43
4	0.52	0.55	0.58	1.26	7.30
5	0.55	0.66	0.64	1.71	3.53
6	0.65	0.75	1.68	1.51	3.31
7	0.77	0.68	0.62	1.28	2.91
8	0.66	0.58	0.51	1.58	4.21
9	0.73	1.80	0.67	1.40	10.7
10	0.69	0.67	0.57	1.80	9.62
Mean	**0.64**	**0.76**	**0.73**	**1.43**	**6.75**
Max	**0.77**	**1.80**	**1.68**	**1.80**	**10.7**
Min	**0.52**	**0.55**	**0.51**	**1.12**	**2.91**
Stdev	**0.07**	**0.35**	**0.32**	**0.21**	**2.82**

Hasil pengujian yang disajikan pada Tabel 4 memiliki sedikit perbedaan dengan hasil pengujian pada penelitian sebelumnya. Secara umum, waktu eksekusi aplikasi terintegrasi lebih lambat dibanding pada aplikasi *mobile* di sistem sebelumnya. Hal ini terjadi karena perbedaan skema pengujian dan juga spesifikasi dari jaringan komunikasi yang digunakan. Pengujian pada penelitian sebelumnya dilakukan dengan menghubungkan *node* sensor pada jaringan WiFi yang berbasis *broadband fiber optic* dengan kecepatan hingga 30 Mbps. Sedangkan pada penelitian ini, *node* sensor dan aktuator terhubung pada jaringan WiFi yang berbasis *mobile data*.

SIMPULAN

Integrasi sistem irigasi cerdas dan *real-time monitoring* pada Sawah IPB 4.0 berhasil dilakukan. Integrasi dilakukan dalam bentuk pengembangan aplikasi *mobile* menggunakan *design pattern single activity architecture* dan Model-View-Viewmodel. Modul kontrol manual dan modul manajemen perangkat berhasil dikembangkan dan diimplementasikan ke dalam aplikasi *mobile*. Aplikasi yang dikembangkan memiliki rata-rata waktu respon 0.71 detik pada fitur yang hanya terkait dengan API *server*. Sedangkan pada fitur yang terkait akses *resource* pada perangkat, aplikasi memiliki waktu respon yang lebih lama yaitu pada rentang 4.09 detik.

DAFTAR PUSTAKA

Agarwal H, Rai JN. 2022. Traffic Control System based on Density with Emergency Priority Mechanism. *2022 International Conference on Electronics and Renewable Systems (ICEARS)*:192–197.

Bellman KL, Gruhl C, Landauer C, Tomforde S. 2019. Self-Improving System Integration - On a Definition and Characteristics of the Challenge. *2019 IEEE 4th International Workshops on Foundations and Applications of Self Systems (FASW)*:1–3.

Devare J, Hajare N. 2019. A Survey on IoT Based Agricultural Crop Growth Monitoring and Quality Control. Di dalam: *2019 International Conference on Communication and Electronics Systems (ICCES)*:1624–1630.

García JRR, Lenz G, Haveman SP, Bonnema GM. 2020. State of the Art of Mobility as a Service (MaaS) Ecosystems and Architectures—An Overview of, and a Definition, Ecosystem and System Architecture for Electric Mobility as a Service (eMaaS). *World Electric Vehicle Journal*.11(1):1–19.

Hafiduddin M. 2022. Pengembangan Sistem Irigasi Cerdas untuk Tanaman Padi Varietas IPB3S pada Lahan Sawah IPB 4.0 Menggunakan Algoritme Fuzzy Sugeno [skripsi]. Bogor: Institut Pertanian Bogor.

Hartono N, Laurence, Johannes HP. 2017. Identification, measurement, and assessment of water cycle of unhusked rice agricultural phases: Case study at Tangerang paddy field, Indonesia. *IOP Conference Series: Materials Science and Engineering* Vol. 273: 1–7.

Institut Pertanian Bogor. 2019. *Agro-Maritim 4.0 Kontribusi Pemikiran IPB untuk Indonesia (Edisi Revisi 2019)*. Bogor: Penerbit IPB Press.

Mochtar K. 2023. Pengembangan Sistem Real-time Monitoring Lahan Sawah 4.0 IPB Menggunakan Sensor pH, EC, dan Kelembapan Tanah Berbasis Mobile Application [skripsi]. Bogor: Institut Pertanian Bogor.

Priandana K, Hafiduddin M, Mochtar K, Akbar AR, Hussin M. 2024. Development of Intelligent Irrigation System for Paddy Field in Indonesia. *Computer Science for Smart Agriculture, Education, Medical and Related Fields: Perspective from Indonesia and Malaysia*. Bogor: Penerbit IPB Press. hlm. 64–81.

Rohith M, Sainivedhana R, Fatima NS. 2021. IoT Enabled Smart Farming and Irrigation System. Di dalam: *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*: 434–439.

Singh UV, Kumar D, Moses DrS. 2016. Performance Evaluation of Manually Operated Paddy Drum Seeder in Puddled field. *IOSR J Agric Vet Sci*. 9(6):69–83.

Solomakha GM, Khizhnyak SV. 2020. Automated Control Module for a Production Process Monitoring System. *Software & Systems*. 33(3):516–522.

Vijayakumar J. 2022. Alternate of Manual Weeding Tools: A Research into an Automatic Weeding Control Strategies Enabled by Embedded Systems. *International Journal on Recent and Innovation Trends in Computing and Communication*. 10(2):1–6.