Indonesian Journal of Computer Science
ISSN 2302-4364 (*print*) dan 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam, No. 1, Padang, Indonesia, Telp. (0751) 7056199, 7058325
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

Pemantauan dan Deteksi Penyakit Daun Tomat Berbasis IoT dan CNN dengan Aplikasi Android

Suharyadi Pancono[1], Narwikant Indroasyoko[1], Asep Irfan Setiawan[1]
suharyadi@ae.polman-bandung.ac.id, asyoko@polman-bandung.ac.id, asepirfans.9c@gmail.com
[1]Politeknik Manufaktur Bandung

Informasi Artikel

Diterima : 31 Mei 2024
Direview : 5 Jun 2024
Disetujui : 30 Jun 2024

Kata Kunci

Aplikasi Android, CNN, DenseNet169, Internet of Things, Tanaman Tomat

Abstrak

Tomat merupakan komoditas bernilai tinggi dalam pertanian, sehingga para petani melakukan berbagai upaya untuk memastikan produksi tomat yang segar dan siap konsumsi. Namun, petani sering menghadapi kesulitan dalam memantau kondisi tomat karena masih menggunakan metode manual dan memiliki keterbatasan pengetahuan dalam mendeteksi penyakit pada daun tomat. Penelitian ini menawarkan solusi dengan memanfaatkan *transfer learning* dan *fine-tuning Convolutional Neural Network* (CNN) menggunakan arsitektur DenseNet169, serta teknologi *Internet of Things* (IoT). Model ini diimplementasikan dalam aplikasi Android menggunakan TensorFlow di *platform* Flutter setelah dikonversi ke format tflite. Hasil pengujian menunjukkan tingkat akurasi model mencapai 94%, sedangkan akurasi aplikasi dalam mendeteksi penyakit daun tomat mencapai 92.80% dan memiliki *response time* sekitar 1077.56 ms. Selain itu, aplikasi dapat memantau kondisi tanaman secara *realtime* dengan memiliki *delay* sebesar 1998 ms.

Keywords

Android App, CNN, DenseNet169, Internet of Things, Tomato Plants

Abstract

Tomatoes are a high-value commodity in agriculture, so farmers make various efforts to ensure the production of fresh and ready-to-consume tomatoes. However, farmers often face difficulties in monitoring tomato growth because they still use manual methods and have limited knowledge in detecting diseases on tomato leaves. This research offers a solution by utilizing transfer learning and fine-tuning Convolutional Neural Network (CNN) using DenseNet169 architecture, as well as Internet of Things (IoT) technology. The model is implemented in an Android application using TensorFlow on the Flutter platform after being converted to tflite format. The test results show that the accuracy of the model reaches 94%, while the accuracy of the application in detecting tomato leaf diseases reaches 92.80% and has a response time of about 1077.56 ms. In addition, the application can monitor plant conditions in realtime by having a delay of 1,998 ms.

A. Pendahuluan

Tomat (Solanum Lycopersicon) merupakan salah satu komoditas sayuran yang memiliki nilai ekonomi tinggi dan potensial ekspor yang besar [1]. Tanaman tomat berperan penting untuk kebutuhan masyarakat seperti konsumsi sehari-hari, pembuatan campuran bahan olahan, dan industri pengolahan makanan. Namun, untuk memenuhi permintaan yang terus meningkat, tanaman tomat masih membutuhkan penanganan yang serius. Peningkatan permintaan tomat ini seringkali tidak sebanding dengan peningkatan produksi [2].

Para petani tomat melakukan berbagai upaya untuk memastikan bahwa dapat menghasilkan tomat yang segar dan siap untuk dikonsumsi, bebas dari hama, dan penyakit lainnya yang dapat mengurangi produktivitas dan kualitas produksi [3]. Tanaman tomat tidak akan bertahan hidup jika tanah terlalu basah karena akar tomat akan mudah membusuk dan tidak dapat mengekstraksi nutrisi dari tanah, sirkulasi udara yang buruk di sekitar akar tomat dapat menyebabkan tanaman mati [4]. Untuk pertumbuhan tanaman tomat membutuhkan tanah yang subur dengan nilai pH tanah sekitar 5 - 6, serta memiliki kelembapan tanah optimal antara 60% - 80% agar tidak terlalu kering maupun basah. Suhu yang ideal untuk pertumbuhan tanaman tomat sekitar 24-28°C karena apabila suhunya terlalu tinggi maka tomat akan cenderung berwarna kuning dan jika suhu terlalu fluktuatif warna tomat tidak akan merata [3].

Untuk memenuhi kebutuhan tanaman tomat, air sangat penting dalam proses fotosintesis [5]. Proses penyiraman tanaman tomat merupakan kegiatan yang sangat penting untuk menghindari terjadinya gagal panen. Salah satu komponen yang harus diperhatikan adalah kelembapan tanah, karena berperan dalam proses transfer unsur hara dan senyawa lain dari media tanah ke tanaman, menjaga suhu tanaman dan mengoptimalkan kematangan daun dan buah [6]. Saat ini, sistem penyiraman yang dilakukan oleh petani masih banyak dilakukan secara manual [7]. Namun cara ini kurang efisien karena apabila tidak dipantau dapat mengakibatkan meningkatnya kadar air di dalam tanah [8].

Selain itu, daun tanaman tomat juga memegang peran krusial dalam proses pertumbuhan tanaman. Salah satu faktor yang mempengaruhi proses pertumbuhan tanaman adalah penyakit pada daun [9]. Sekitar 80% - 90% penyakit tanaman terjadi pada daun yang disebabkan oleh jamur, bakteri dan virus. Beberapa diantaranya seperti bakteri, penyakit busuk daun, bercak daun, mosaik tomat, penyakit kuning melengkung dan sebagainya [10]. Penyakit ini dapat dilihat secara visual karena mempunyai warna dan tekstur yang unik. Identifikasi penyakit secara visual oleh petani mempunyai kelemahan yaitu memakan waktu lama dan tidak akurat dalam identifikasi karena adanya kesamaan antara masing-masing jenis penyakit dengan penyakit lainnya [11].

Oleh karena itu, pemantauan tanaman dan deteksi dini penyakit sangat penting karena dapat mencegah kerugian yang lebih besar dan memungkinkan adopsi tindakan yang cepat sebelum penyakit tanaman menyebar dengan lebih luas. Praktik ini tidak hanya dapat menghemat waktu petani dalam proses pengelolaan tanaman, tetapi juga membantu mengurangi dampak negatif terhadap produktivitas dan kualitas hasil pertanian [12].

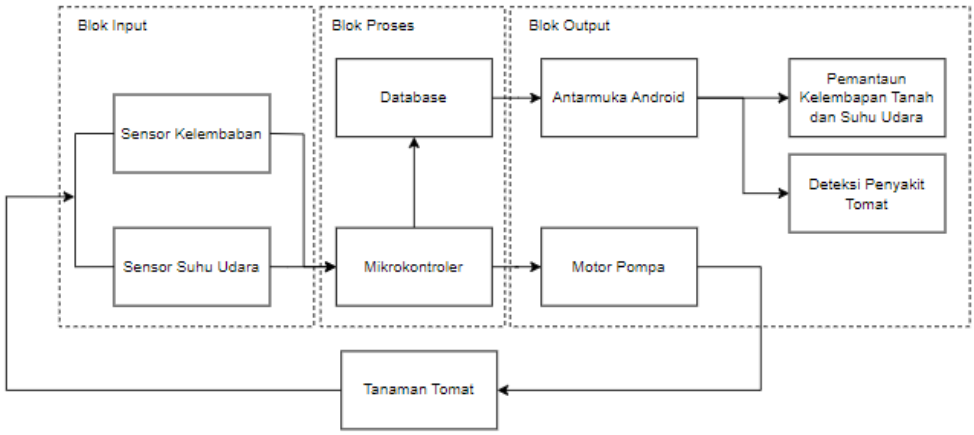
Penelitian terkait menunjukkan bahwa kondisi tanaman seperti kelembapan tanah, suhu udara, dan pH tanah dapat dipantau jarak jauh melalui *web browser*

maupun aplikasi *mobile* secara *real-time* sehingga dapat memudahkan petani dalam pengambilan keputusan. Tanaman tomat yang diteliti menunjukkan kondisi yang ideal, dengan kelembapan tanah berkisar antara 30%-80% [3][13]. Selanjutnya penelitian mengenai implementasi metode *Convolutional Neural Network* (CNN) untuk mendeteksi penyakit pada daun tomat, menunjukkan bahwa metode ini dapat menghasilkan tingkat akurasi uji mencapai 94% untuk gambar dari galeri dan 80% untuk gambar langsung dari kamera [14]. Berikutnya mengenai aplikasi deteksi penyakit menggunakan model terlatih berbasis android, penelitian ini membuat aplikasi deteksi penyakit daun tomat menggunakan *Convolutional Neural Network* (CNN) dengan bantuan Web *Teachable Machine* untuk melatih model menjadi *Tensorflow lite*, hasil uji coba menunjukkan bahwa aplikasi ini dapat mengidentifikasi penyakit daun dengan baik, meskipun dipengaruhi oleh spesifikasi kamera, sudut pengambilan gambar, dan pencahayaan [15].

Berdasarkan permasalahan di atas perlu dilakukan penelitian ini untuk merancang sebuah sistem rekayasa teknologi. Sistem ini diharapkan dapat membantu aktivitas petani, terutama dalam pemantauan kondisi tanaman tomat dari jarak jauh dan deteksi dini penyakit pada daun tanaman. Selain itu, sistem ini dirancang untuk memberikan informasi melalui sebuah aplikasi android yang sudah terintegrasi. Informasi yang di dapat berupa deteksi penyakit pada daun tomat dan kondisi tanaman tomat, seperti kelembapan tanah dan suhu udara.

B. Metode Penelitian

1. Gambaran Umum Sistem



Gambar 1. Gambaran Umum Sistem

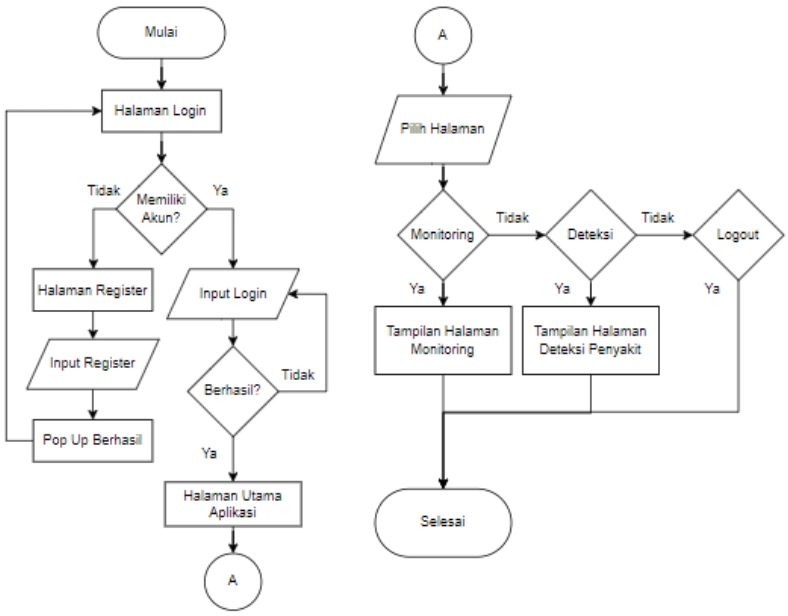
Sistem yang ditunjukkan pada Gambar 1 mengintegrasikan teknologi IoT dan CNN untuk memantau dan mendeteksi penyakit daun tomat melalui aplikasi Android. Sistem ini terdiri dari sensor kelembapan dan suhu udara yang mengukur kondisi lingkungan sekitar tanaman tomat, mengirimkan data ke mikrokontroler, yang kemudian mengunggahnya ke *database* melalui internet. Data ini dapat dipantau secara *realtime* oleh pengguna melalui antarmuka Android dan dilengkapi dengan fitur deteksi penyakit daun tomat menggunakan model CNN. Ketika kelembapan tanah turun di bawah ambang batas yang ditentukan, motor pompa diaktifkan untuk menyirami tanaman, dan akan dimatikan ketika kelembapan mencapai level yang telah ditentukan. Fitur deteksi penyakit memungkinkan

pengguna mengambil atau mengunggah foto daun tomat untuk dianalisis, memberikan diagnosis penyakit dan rekomendasi perawatan. Sistem ini menawarkan solusi efisien untuk mengelola kesehatan tanaman tomat, meningkatkan produktivitas dan kualitas hasil panen.

2. Perancangan *Software*

Metode RAD digunakan untuk mempercepat proses pengembangan sistem informasi serta migrasi dari sistem lama ke sistem baru dengan melibatkan aktif partisipasi dari pengguna dan pemangku kepentingan. Pendekatan ini bertujuan untuk memungkinkan pengembangan yang cepat dan responsif terhadap kebutuhan yang mungkin berubah dalam lingkungan pengembangan yang dinamis. Metode RAD terdiri dari tiga tahap yaitu *Requirements Planning, Design Workshop,* dan *implementation* [15].

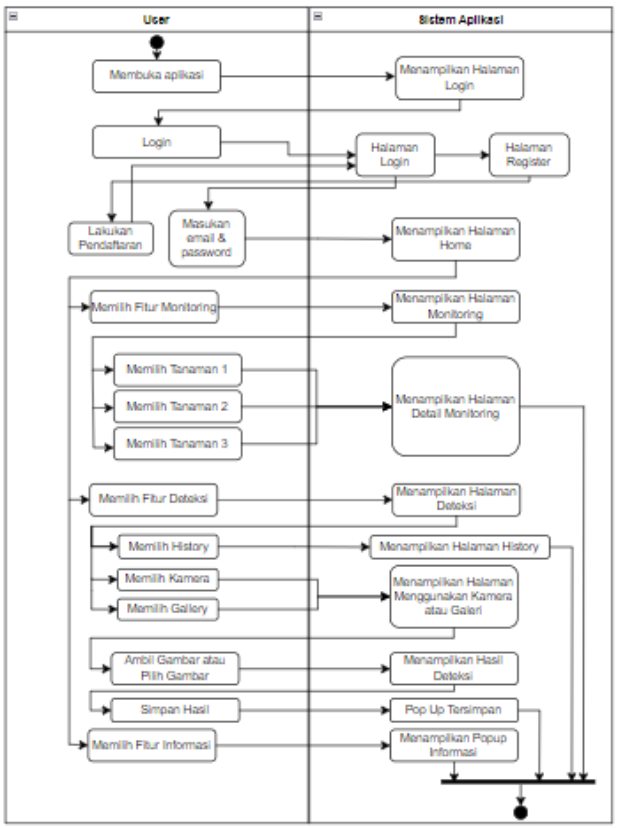
a. *Flowchart* Aplikasi



Gambar 2. *Flowchart* Aplikasi

Gambar 2 merupakan alur penggunaan aplikasi. Saat pertama membuka aplikasi pengguna harus *login* terlebih dahulu, apabila tidak memiliki akun maka bisa *register* terlebih dahulu. Apabila pengguna sudah berhasil *login* maka akan dialihkan ke halaman utama. Di dalam halaman utama terdapat tiga fitur yaitu monitoring, deteksi penyakit, dan informasi. Apabila pengguna memilih halaman monitoring maka akan menampilkan halaman monitoring, halaman tersebut terdapat beberapa tanaman tomat yang dipantau, dalam halaman *detail* monitoring disajikan data-data kelembapan tanah dan suhu udara. Selanjutnya apabila pengguna memilih halaman deteksi maka akan menampilkan halaman deteksi penyakit. Jika pengguna menekan tombol yang berada di informasi maka akan menampilkan informasi mengenai kelembapan tanah, suhu, dan deteksi yang berkaitan dengan tanaman tomat sebagai media edukasi. Berikutnya terdapat tombol *logout* apabila pengguna akan keluar dari aplikasi.

b. *Activity* Diagram



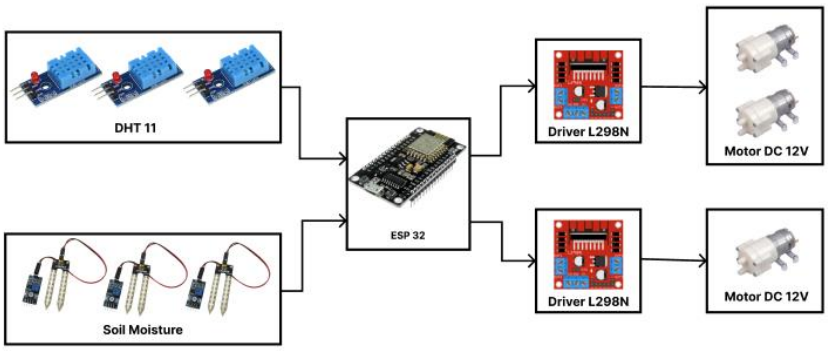
Gambar 3. *Activity* Diagam

Dalam rangkaian aktivitas yang tergambar pada gambar 3, terdapat dua tabel yang secara jelas memisahkan alur antara interaksi pengguna dan respons aplikasi. Saat pengguna menginisiasi aplikasi, fase awal ditandai dengan tampilan halaman *login*. Di dalam halaman *login* apabila pengguna sudah mempunyai akun maka bisa secara langsung memasukkan email dan *password*, tetapi apabila pengguna tidak memiliki akun maka dapat beralih ke halaman *register*, setelah pengguna mendaftarkan akunnya maka bisa masuk ke halaman *login*. Setelah pengguna berhasil *login* akan masuk ke halaman utama. Pada halaman utama terdapat 3 fitur utama yang dapat diakses, yaitu fitur monitoring, fitur deteksi, dan fitur informasi. Pengguna diberikan kebebasan untuk memilih sesuai kebutuhan. Begitu pengguna membuat pilihan, aplikasi dapat menampilkan konten yang sesuai dengan fitur yang telah dipilih oleh pengguna.

3. Perancangan *Hardware*

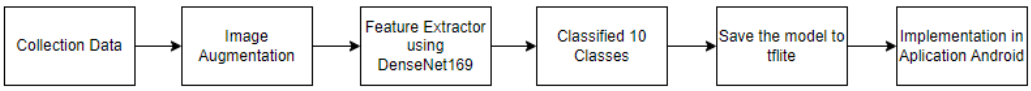
Dalam rancangan *hardware*, terdapat ESP32 yang bertujuan untuk menyimpan data sensor ke dalam *database* karena memiliki modul WiFi yang dapat terkoneksi dengan internet. Sensor yang digunakan adalah DHT11 untuk memantau kondisi suhu udara dan sensor *soil moisture* untuk memantau kelembapan tanah. Selain itu, terdapat motor *driver* L298N untuk mengendalikan motor pompa DC 12V. Selanjutnya, terdapat pompa DC 12V sebagai aktuator untuk menyiram tanaman

secara otomatis berdasarkan nilai kelembapan tanah. Gambar 4 menunjukkan peracangan *hardware.*



Gambar 4. Rancangan *Hardware*

4. Perancangan CNN



Gambar 5. Rancangan Pembuatan Model CNN

Gambar 5 menunjukkan alur kerja untuk pengembangan dan implementasi model CNN. Berikut penjelasan *detail* setiap langkahnya:

4.1. Collection Data

Langkah pertama dalam pembuatan model CNN adalah pengumpulan data. Data diambil berdasarakna data primer dan sekunder. Untuk data sekunder diambil melalui sumber terbuka yaitu *website* kaggle.com dari PlantVillage, sedangkan data primer diambil dari *google* dengan mencari setiap penyakit daun tomat. Dataset terdiri dari sepuluh kelas, termasuk sembilan kelas penyakit dan tanaman sehat. Jenis penyakit tersebut *meliputi yellow leaf curl virus, mozaik virus, septoria leaf spot, bacterial spot, target spot, early blight, late blight, leaf mold, and two-spot spider mite*. Dengan mengumpulkan data yang komprehensif dan representatif, model akan memiliki informasi yang cukup untuk belajar dan mengenali berbagai kondisi tanaman.

4.2. Image Augmentation

Langkah berikutnya adalah augmentasi gambar, di mana teknik manipulasi gambar digunakan untuk meningkatkan variasi dan jumlah gambar dalam dataset. Teknik augmentasi ini bisa mencakup rotasi, pemotongan, perubahan pencahayaan, dan transformasi lainnya. Dengan melakukan augmentasi gambar, variasi data latih meningkat, yang membantu model menjadi lebih *robust* dan mampu mengenali objek dari berbagai sudut dan kondisi. Penggunaan teknik augmentasi yang tepat dapat signifikan meningkatkan performa model dengan menciptakan lebih banyak variasi pada data latih.

4.3. Feature Extractor

Setelah augmentasi gambar, gambar-gambar tersebut diproses menggunakan DenseNet169 sebagai *feature extractor*. DenseNet169 adalah arsitektur jaringan saraf dalam yang dikenal karena kemampuannya dalam mengekstraksi fitur dari gambar secara efisien. Pada tahap *feature extraction*, tujuannya adalah

mengekstraksi fitur dari gambar dalam bentuk angka-angka yang mendefinisikan gambar tersebut. Tahap ini sangat penting karena fitur-fitur ini akan digunakan oleh bagian *fully connected* dari CNN untuk mengenali objek dan bentuk, menghasilkan identifikasi objek yang akurat.

4.4. Classification

Setelah fitur diekstraksi, langkah berikutnya adalah mengklasifikasikan gambar ke dalam sepuluh kelas yang sudah ditentukan. Proses klasifikasi ini dilakukan oleh *layer fully connected* di akhir jaringan saraf yang mengeluarkan probabilitas untuk setiap kelas, kemudian menentukan kelas yang paling mungkin. Dengan menggunakan arsitektur DenseNet169, model mampu mengenali dan mengklasifikasikan gambar dengan tingkat akurasi yang tinggi.

4.5. Save the model

Model yang telah dilatih kemudian disimpan dalam format *TensorFlow Lite* (tflite). Format ini dirancang khusus untuk implementasi model *machine learning* di perangkat dengan sumber daya terbatas seperti *smartphone*. Mengkonversi model ke format tflite membuatnya lebih ringan dan efisien untuk dijalankan di perangkat tersebut. Ini memungkinkan penggunaan model dalam aplikasi *mobile* tanpa mengorbankan performa.

4.6. Implementation

Langkah terakhir adalah implementasi model ke dalam aplikasi *mobile*. Dalam perancangan aplikasi ini menggunakan *framework* Flutter dan *library* tflite untuk memudahkan integrasi dan penggunaan model *machine learning* di dalam aplikasi. Dengan implementasi ini, pengguna dapat memanfaatkan kecerdasan buatan untuk melakukan prediksi atau klasifikasi gambar secara langsung di perangkat Android, yang sangat berguna dalam aplikasi pertanian untuk mendeteksi penyakit tanaman secara cepat dan akurat.

C. Hasil dan Pembahasan

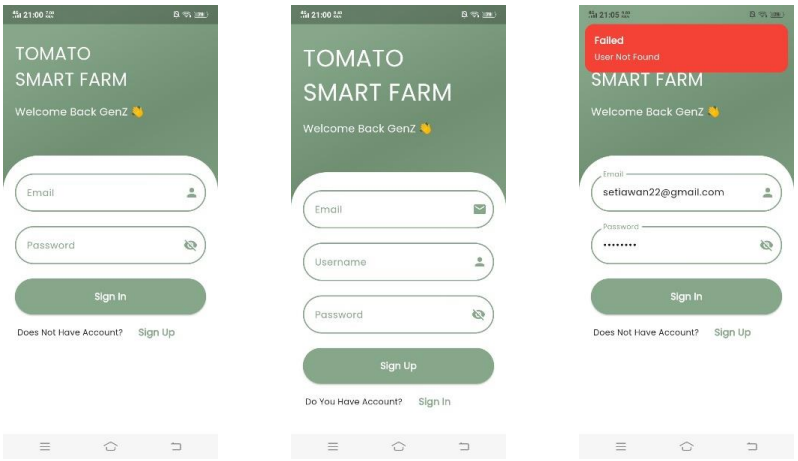
1. Implementasi Sistem

Gambar 6 menunjukkan hasil dari sistem pemantauan kondisi untuk tanaman tomat, yang terdiri dari beberapa bagian. Pertama, terdapat *box* panel yang berfungsi sebagai tempat penyimpanan mikrokontroler serta *power supply*. Selain itu, sistem ini dilengkapi dengan sensor untuk memantau kondisi tanaman, seperti kelembaban tanah dan suhu, serta pompa air kecil untuk menyiram tanaman. Air untuk penyiraman diambil dari ember yang berfungsi sebagai sumber air. Tanaman tomat ditanam dalam *polybag* dan terhubung dengan sistem penyiraman *drip irrigation*, yang menyiram langsung ke akar tanaman tomat, memastikan penyiraman yang efisien dan tepat sasaran. Sistem ini dirancang untuk memantau dan mengelola kondisi tanaman secara otomatis.

Gambar 6. Implementasi Sistem

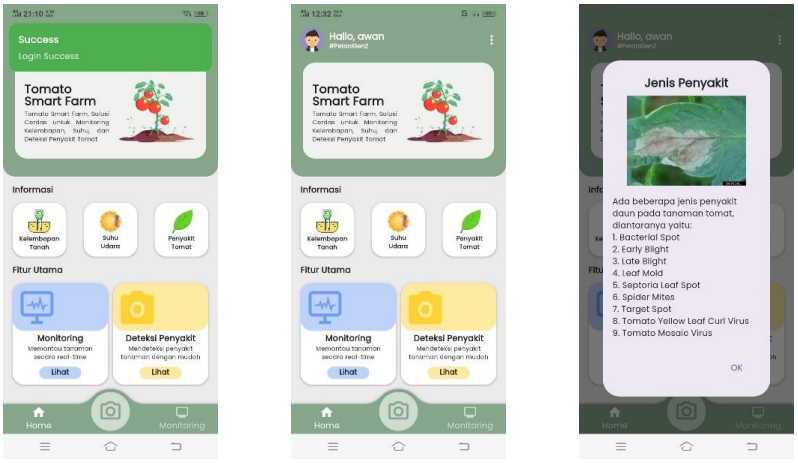
2. Implementasi Aplikasi

Saat pengguna membuka aplikasi untuk pertama kalinya, mereka akan diarahkan ke halaman login. Pengguna diminta untuk memasukkan *email* dan kata sandi yang valid, apabila tidak valid maka akan muncul *popup* peringatan. Selain itu, terdapat halaman register apabila pengguna akan membuat akun terlebih dahulu. Dapat dilihat pada gambar 7.



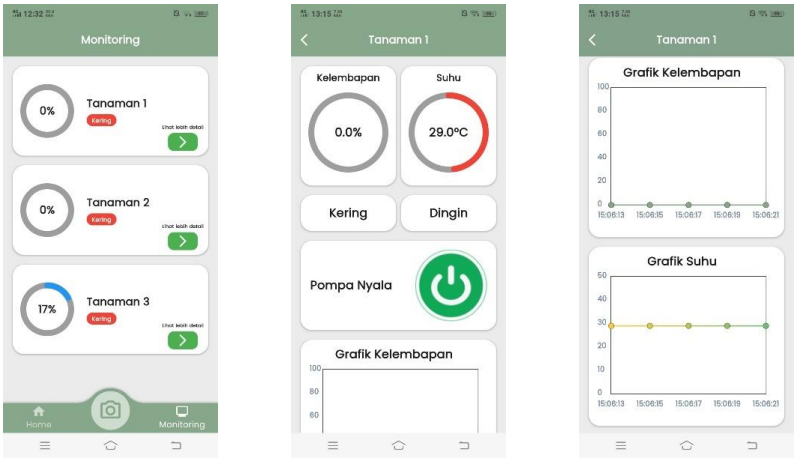
Gambar 7. Halaman *Login* dan Registrasi

Setelah *login* maka pengguna akan diarahkan ke halaman utama, terdapat menu informasi, dalam menu ini akan diberikan edukasi mengenai kelembapan tanah, suhu udara, serta jenis penyakit tomat yang ditampilkan melalui *popup*. Selain itu terdapat fitur utama yaitu monitoring dan deteksi penyakit. Terakhir di terdapat *icon* titik tiga itu ketika ditekan akan menampilkan menu *logout*.



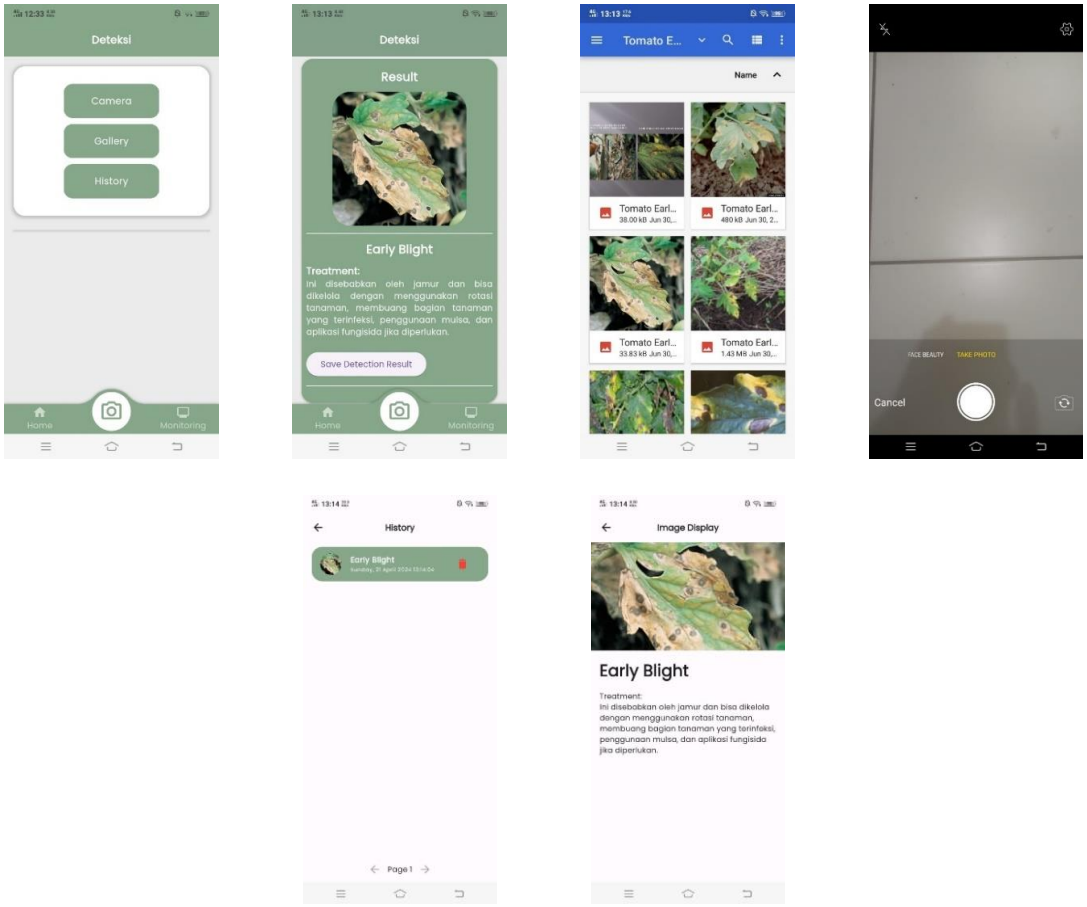
Gambar 8. Halaman Utama

Berikutnya merupakan halamam monitoring, di dalam halaman ini menampilkan informasi 3 tanaman yang akan dipantau, ketika pengguna menekan tombol "lihat lebih detail" maka akan ditampilkan halaman *detail* untuk mengetahui informasi lebih lengkap dan divisualisasikan melalui *circle chart* dan grafik. Informasi yang ditampilkan berupa nilai sensor kelembapan tanah dan suhu udara.



Gambar 9. Halaman Monitoring

Selanjutnya halaman deteksi, di dalam halam deteksi ini terdapat beberapa tombol yaitu *camera, gallery,* dan *history*. Apabila pengguna memilih camera maka untuk mendeteksi penyakit daun tomat akan di arahkan ke kamera *smartphone* secara langsung. Berikutnya apabila pengguna memilih *gallery* maka akan diarahkan ke dalam folder untuk mencari foto penyakit yang akan dideteksi. Setelah melakukan proses deteksi berikutnya terdapat tombol *save* yang digunakan untuk menyimpan hasil deteksi sebagai *history*, sehingga pengguna dapat melihat kembali hasil yang telah dideteksi, Di dalam *history* terdapat list hasil deteksi yang telah dilakukan.



Gambar 10. Halaman Deteksi Penyakit

3. Pengujian

3.1. Pengujian *Hardware*

Pengujian *hardware* yang telah dilakukan bertujuan untuk mengevaluasi kinerja sistem dalam memantau kelembapan tanah secara otomatis menggunakan sensor kelembapan dan perangkat pengendali pompa air. Hasil pengujian ini diharapkan dapat memberikan informasi yang akurat mengenai efektivitas dan efisiensi sistem dalam menjaga kondisi optimal kelembapan tanah untuk tanaman. Berikut adalah tabel yang menunjukkan hasil dari pengujian *hardware.*

Tabel 1. Pengujian *Hardware*

Pengujian	SM1	SM2	SM3	DHT1	DHT2	DHT3	Keterangan Kondisi Tanah	Pompa air 1	Pompa air 2	Pompa air 3
1	35%	40%	43%	26.7	27.1	27	Kering	ON	ON	ON
2	65%	73%	65%	26.7	27.1	27	Lembab	OFF	OFF	OFF
3	64%	73%	65%	26.7	27.1	26.7	Lembab	OFF	OFF	OFF
4	64%	71%	65%	26.7	26.8	26.7	Lembab	OFF	OFF	OFF
5	62%	71%	63%	26.2	26.8	26.7	Lembab	OFF	OFF	OFF
6	62%	68%	63%	26.2	26.8	26.7	Lembab	OFF	OFF	OFF
7	61%	65%	62%	26.2	26.8	26.2	Lembab	OFF	OFF	OFF
8	61%	65%	62%	26.2	26.2	26.2	Lembab	OFF	OFF	OFF
9	61%	65%	61%	26.2	26.2	26.2	Lembab	OFF	OFF	OFF
10	61%	63%	61%	26.2	26.2	26.2	Lembab	OFF	OFF	OFF

Pada tabel di atas menujukkan sistem dapat bekerja dengan baik, ketika nilai sensor kelembapan tanah kurang dari ambang batas maka pompa akan menyala. Pada penelitian ini nilai ambang batas dari sensor kelembapan tanah yaitu 60%, sehingga terbukti pada kondisi awal tanaman tomat sebelum disiram mendapatkan nilai kelembapan 35% untuk tanaman pertama, 40% untuk tanaman kedua, dan 43% untuk tanaman ketiga, kondisi tersebut membuat pompa menyala. Pada pengujian berikutnya, ketika nilai sensor kelembapan tanah lebih dari 60% maka pompa akan mati. Oleh karena itu, kinerja alat ini berfungsi dengan baik karena mampu merespons perubahan kelembapan tanah dengan menyalakan atau mematikan pompa sesuai dengan *setpoint* dari nilai kelembapan tanah yang telah ditentukan.

3.2. Pengujian *Software*

Pengujian *functional suitability* pada *software* menggunakan metode *black box.* Pengujian ini dilakukan untuk mengetahui tingkat keberhasilan fitur – fitur pada aplikasi *mobile*. Fungsi aplikasi diujikan pada 3 metrik yaitu *Functional Adequecy* (FA), *Functional Implementation Coverage* (FIC), *Functional Implementation Completeness* (FICM) [16]. Berikut merupakan hasil dari pengujian *software* yang dilakukan pada tabel

Tabel 2. Pengujian *Software*

Fitur	FA	FIC	FICM
Login, Register dan *Logout*	1	1	1
Halaman *Home*	1	1	1
Monitoring Kelembapan Tanah	1	1	1
Monitoring Suhu	1	1	1
Halaman Deteksi	1	1	1
Simpan Hasil Deteksi	1	1	1
Riwayat Deteksi	1	1	1
Delete Hasil Deteksi	1	1	1
Popup Informasi	1	1	1

Tabel 2 menampilkan pengujian fungsional untuk mengevaluasi keberhasilan fitur dalam aplikasi Android. Penilaiannya berkisar dari 0 (buruk) hingga 1 (baik), dengan nilai antara 0 hingga 1 ($0 \leq X \leq 1$). FA menunjukkan kecukupan fungsional dari fungsi-fungsi aplikasi, FIC menggambarkan cakupan implementasi fungsional dari fungsi-fungsi aplikasi, dan FICM mencerminkan kelengkapan implementasi fungsional dari fungsi-fungsi aplikasi.

3.3. Pengujian *Response Time* Sensor Terhadap *Interface* Aplikasi

Pengujian *response time* diperlukan dalam sebuah sistem yang terdapat komunikasi data, hal ini bertujuan untuk mengetahui waktu tunda (*delay*) pengiriman dan penerimaan data. Pengujian dilakukan dengan bantuan program C++ dengan mengambil waktu *startTime* saat mulai mengambil data dengan *endTime* saat data telah diterima sehingga mendapatkan selisih waktu, dalam program tersebut menggunakan fungsi *millis* sehingga waktu dapat ditampilkan melalui serial monitor Arduino IDE Berikut merupakan hasil pengujian *response time*.

Tabel 3. Pengujian *Response Time*

Pengujian	Data Terbaca	Data Terkirim	Status	Selisih
1	06:50:28.125	06:50:30.062	Terkirim	1937
2	06:50:30.216	06:50:32.145	Terkirim	1929
3	06:50:32.199	06:50:34.143	Terkirim	1944
4	06:50:34.277	06:50:36.208	Terkirim	1931
5	06:50:36.298	06:50:38.243	Terkirim	1945
6	06:50:38.36	06:50:40.499	Terkirim	2139
7	06:50:40.645	06:50:42.577	Terkirim	1932
8	06:50:42.674	06:50:44.820	Terkirim	2146
9	06:50:44.942	06:50:46.874	Terkirim	1932
10	06:50:46.953	06:50:49.100	Terkirim	2147
Rata- rata selisih				1998

Tabel di atas menunjukkan hasil pengujian kecepatan pengiriman data yang dilakukan ketika sensor mengirimkan data dan dibaca oleh antarmuka melalui *smartphone*. Pengujian ini dilakukan sebanyak 10 kali untuk memastikan konsistensi hasil. Dari hasil pengujian, diperoleh rata-rata waktu tunda (delay) sebesar 1998 ms. Hasil ini menunjukkan efisiensi dalam proses pengiriman dan pembacaan data oleh sistem, memberikan indikasi bahwa sistem mampu menangani transmisi data secara cepat dan responsif.

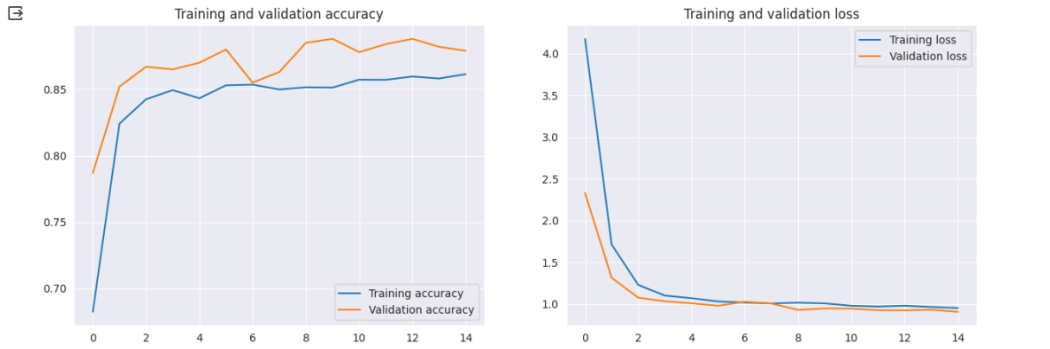
3.4. Hasil Pelatihan dan Pengujian Pada Google *Colaboratory*

Dalam penelitian ini menggunakan rasio perbandingan dataset untuk *training*, *validation*, dan *testing* sebesar 8:1:1, total data yang digunakan sebesar 10.000 dengan penyebaran 8000 data untuk *training*, 1000 data untuk *validation*, dan 1000 data untuk *testing*. Model ini dilakukan melalui 2 tahap yaitu tahap pertama ketika sebelum adanya *fine-tune* dan tahap kedua ketika dilakukan *fine-tune*.

a. Hasil Sebelum *Fine-Tune*

Pada tahap ini, dilakukan ekstraksi fitur (feature extraction) menggunakan arsitektur DenseNet169 yang telah dilatih sebelumnya. Proses ini tidak melibatkan pelatihan seluruh model, tetapi hanya fokus pada bagian akhir yang bertanggung jawab atas klasifikasi, karena bagian ini memiliki karakteristik khusus yang dapat digunakan untuk tugas klasifikasi yang sedang dihadapi. Selanjutnya, dilakukan pelatihan pada layer tambahan yang ditambahkan, yaitu BatchNormalization, Dense dengan 256 unit dan aktivasi ReLU, *Dropout* sebesar 0.45, seed 123, dan terakhir Dense dengan 10 kelas menggunakan aktivasi softmax. Pengoptimalisasi dilakukan menggunakan algoritma Adam dengan learning rate sebesar 0.001, dan fungsi *loss* yang digunakan adalah *categorical crossentropy*.

Selama pelatihan ini, jumlah epoch yang ditentukan adalah 15. Selain itu, menerapkan fungsi *callback Early Stopping* untuk mencegah *overfitting* selama proses pelatihan dan validasi. Berikut merupakan hasil dari tahap pertama.

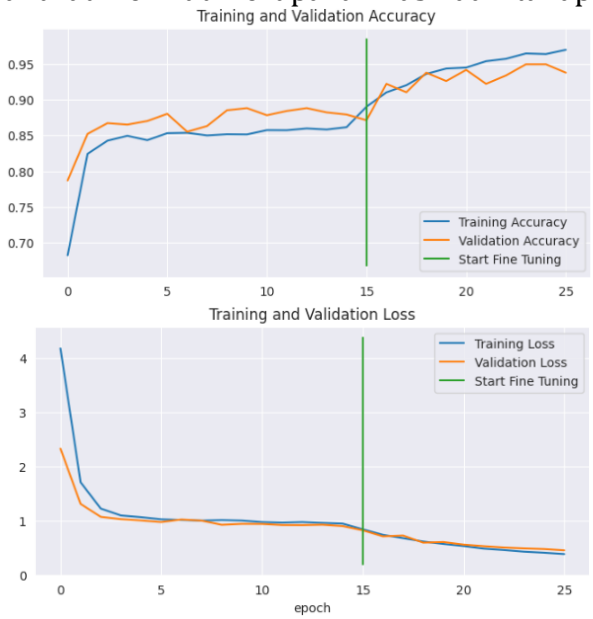


Gambar 11. Hasil Akurasi Sebelum *Fine-Tune*

Grafik di atas menunjukkan perkembangan akurasi dan *loss* selama pelatihan model. Dalam proses ini, model dilengkapi dengan fungsi *Callback Early Stopping* yang secara otomatis menghentikan pelatihan pada epoch ke-14. Di akhir pelatihan model menghasilkan nilai akurasi validasi sebesar 0.8790 dan nilai *loss* validasi sebesar 0.9071. Selanjutnya nilai akurasi dari testing sebesar 0.8840 dan nilai *loss* dari testing sebesar 0.8854.

b. Hasil Setelah *Fine-Tune*

Dalam tahap ini, dilakukan proses *fine-tuning* dengan melatih sebagian dari lapisan-lapisan teratas dari arsitektur DenseNet169 dan melatih lapisan-lapisan tambahan seperti yang telah dijelaskan sebelumnya. Tujuan dari *fine-tuning* ini adalah untuk mengoptimalkan model sehingga mampu meningkatkan nilai akurasi dalam menyelesaikan tugas klasifikasi atau prediksi yang spesifik. Dengan memperbarui representasi fitur tingkat tinggi pada lapisan-lapisan atas model DenseNet169 dan melatih kembali lapisan-lapisan tambahan, model dapat belajar pola-pola yang lebih spesifik dari data yang ditangani, sehingga dapat menghasilkan prediksi yang lebih akurat. Berikut merupakan hasil dari tahap kedua.



Gambar 12. Hasil Akurasi Sesudah *Fine-Tune*

Grafik di atas menunjukkan peningkatan yang signifikan dalam nilai akurasi model, mendekati nilai 1, serta penurunan yang besar dalam nilai *loss*, mendekati nilai 0. Proses pelatihan melanjutkan dari tahap pertama, dimulai dari epoch ke-15 hingga mencapai epoch ke-25. Melalui proses *fine-tuning*, terbukti bahwa nilai akurasi model dapat ditingkatkan, dengan mencapai akurasi validasi sebesar 0.9460 dan loss validasi sebesar 0.04437. Pada pengujian dengan dataset *testing*, model juga memperlihatkan kinerja yang sangat baik dengan akurasi sebesar 0.9430 dan *loss* sebesar 0.4253. Hasil ini menunjukkan bahwa *fine-tuning* berhasil meningkatkan kemampuan model dalam mengklasifikasikan data dengan akurat.

c. Evaluasi Model

Selanjutnya, tahap evaluasi menggunakan *confusion matrix* merupakan langkah penting untuk menganalisis tingkat keberhasilan dan kegagalan model klasifikasi yang telah dibuat. *Confusion matrix* menampilkan informasi tentang label aktual (benar) dan label prediksi dari model untuk setiap kelas atau kategori penyakit pada daun tomat. Dari *confusion matrix* ini, dapat menghitung berbagai metrik performa model seperti presisi, *recall*, dan skor F1 untuk setiap kelas. Untuk mengetahui nilai performa model dapat menggunakan rumus berikut:

$$Accuracy = \frac{(TN+TP)}{(TN+TP+FN+FP)} \quad (1)$$

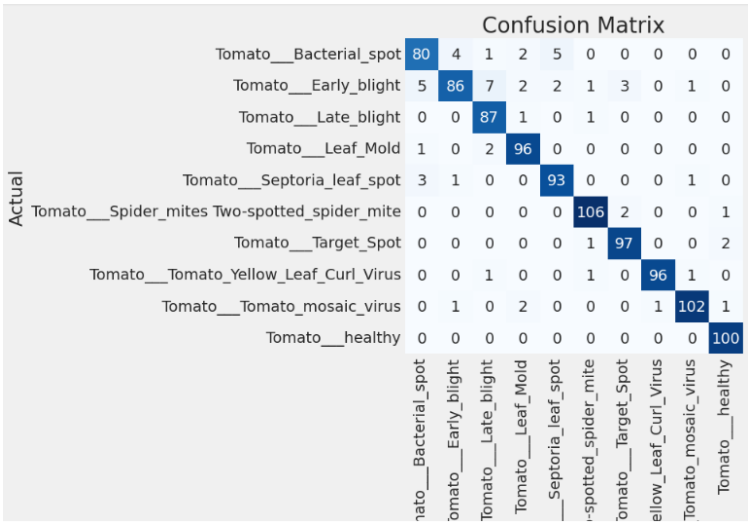
$$Precision = \frac{TP}{(TP+FP)} \quad (2)$$

$$Recall = \frac{TP}{(TP+FN)} \quad (3)$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{(Precision+Recall)} \quad (4)$$

Keterangan:
TP (True Positive) = Data citra daun tomat positif yang diprediksi benar
TN (True Negative) = Data citra daun tomat negatif yang diprediksi benar
FN (False Positive) = Data citra daun tomat negatif yang salah diprediksi positif
FP (False Negative) = Data citra daun tomat positif yang salah diprediksi negatif

Gambar 13 menujukkan hasil dari *confusion matrix* sedangkan tabel 4 merupakan hasil performa model secara lengkap untuk presisi, *recall*, dan skor F1 dari tiap kelas.



Gambar 13. *Confusion Matrix*

Berikut merupakan tabel dari hasil performa model, tabel ini menampilkan metrik performa utama seperti akurasi, presisi, *recall*, dan *F1-score*.

Tabel 4. Hasil Performa Model

Class	Precision	Recall	F1-Score	Support
Tomato__Bacterial_spot	0.90	0.87	0.88	92
Tomato__Early_blight	0.93	0.80	0.86	107
Tomato__Late_blight	0.89	0.98	0.93	89
Tomato__Leaf_mold	0.93	0.97	0.95	99
Tomato__Septoria_leaf_spot	0.93	0.95	0.94	98
Tomato__Spider_mites Two-spotted_spider_mite	0.96	0.97	0.97	109
Tomato__Target_spot	0.95	0.97	0.96	100
Tomato__Tomato_Yellow_Leaf_Curl_Virus	0.99	0.97	0.98	99
Tomato__Tomato_mosaic_virus	0.97	0.95	0.96	107
Tomato__healthy	0.96	1.00	0.98	100
Accuracy			0.94	1000

3.5. Pengujian Model Pada Aplikasi

Pengujian dilakukan menggunakan aplikasi yang telah diinstal pada *smartphone*. Setiap penyakit diuji menggunakan 5 gambar. Proses pengujian ini didasarkan pada hasil pelatihan model yang dilakukan di *Google Colaboratory*, yang kemudian dikonversi menjadi format TfLite untuk diimplementasikan pada platform Android.

Tabel 5. Pengujian Model Pada Aplikasi

Uji ke-	Response Deteksi (ms)	Data Uji	Hasil Prediksi	Akurasi
1	1080	Bacterial Spot	Bacterial Spot	99.32%
2	1078	Bacterial Spot	Bacterial Spot	91.92%
3	1014	Bacterial Spot	Bacterial Spot	99.92%
4	1067	Bacterial Spot	Bacterial Spot	99.88%
5	1079	Bacterial Spot	Bacterial Spot	73.30%
6	1083	Early Blight	Early Blight	99.76%
7	1076	Early Blight	Early Blight	80.80%
8	1066	Early Blight	Early Blight	86.73%
9	1065	Early Blight	Early Blight	99.80%
10	1085	Early Blight	Early Blight	97.97%
11	1080	Late Blight	Late Blight	99.73%
12	1072	Late Blight	Late Blight	87.59%
13	1071	Late Blight	Late Blight	99.59%
14	1085	Late Blight	Late Blight	99.58%
15	1269	Late Blight	Late Blight	99.83%
16	1077	Leaf Mold	Leaf Mold	99.94%
17	1015	Leaf Mold	Leaf Mold	95.53%
18	1020	Leaf Mold	Leaf Mold	99.17%
19	1004	Leaf Mold	Leaf Mold	37.60%
20	1145	Leaf Mold	Leaf Mold	99.94%
21	1027	Septoria Leaf Spot	Septoria Leaf Spot	95.82%
22	1107	Septoria Leaf Spot	Septoria Leaf Spot	94.57%
23	1073	Septoria Leaf Spot	Septoria Leaf Spot	99.88%
24	1013	Septoria Leaf Spot	Septoria Leaf Spot	97.03%
25	1122	Septoria Leaf Spot	Septoria Leaf Spot	59.96%
26	1092	Spider Mites	Spider Mites	98.74%

27	1078	Spider Mites	Spider Mites	99.92%
28	1090	Spider Mites	Spider Mites	99.78%
29	1076	Spider Mites	Spider Mites	98.69%
30	1081	Spider Mites	Spider Mites	99.97%
31	1039	Target Spot	Target Spot	99.94%
32	1093	Target Spot	Target Spot	40.65%
33	1132	Target Spot	Target Spot	58.67%
34	1177	Target Spot	Target Spot	79.42%
35	1108	Target Spot	Target Spot	78.25%
36	1084	Mosaic Virus	Mosaic Virus	99.98%
37	1120	Mosaic Virus	Mosaic Virus	99.98%
38	1096	Mosaic Virus	Mosaic Virus	99.67%
39	1081	Mosaic Virus	Mosaic Virus	99.94%
40	1035	Mosaic Virus	Mosaic Virus	99.95%
41	1079	Yellow Leaf Curl Virus	Yellow Leaf Curl Virus	99.95%
42	1080	Yellow Leaf Curl Virus	Yellow Leaf Curl Virus	99.99%
43	1046	Yellow Leaf Curl Virus	Yellow Leaf Curl Virus	95.17%
44	1082	Yellow Leaf Curl Virus	Yellow Leaf Curl Virus	99.84%
45	1060	Yellow Leaf Curl Virus	Yellow Leaf Curl Virus	99.74%
46	1111	Healthy	Healthy	99.39%
47	1032	Healthy	Healthy	99.96%
48	1067	Healthy	Healthy	99.99%
49	1016	Healthy	Healthy	99.94%
50	1050	Healthy	Healthy	97.46%

Dari hasil tabel di atas, akurasi deteksi penyakit daun tomat berdasarkan pengujian terhadap 50 sampel daun tomat menunjukkan rentang akurasi antara 40.65% hingga 99.99%. Perhitungan akurasi ini dapat dipelajari lebih lanjut pada sumber [2] . Untuk menghitung akurasi aplikasi dalam mendeteksi penyakit pada daun tomat dari 50 sampel, digunakan persamaan berikut:

$$Akurasi = \frac{\sum presentasi\ deteksi}{jumlah\ sampel} \quad (5)$$

Dengan menggunakan persamaan di atas, diperoleh:

$$Akurasi = \frac{4640.14}{50} = 92.80\% \quad (6)$$

Hasil perhitungan tersebut menunjukkan bahwa tingkat akurasi aplikasi dalam mendeteksi penyakit daun tomat adalah 92.80%. Selain itu, rata-rata waktu yang dibutuhkan untuk melakukan deteksi adalah 1077.56 ms. Faktor-faktor yang mempengaruhi tingkat akurasi pendeteksian meliputi intensitas pencahayaan saat pengambilan gambar dan jarak kamera *smartphone* terhadap objek.

D. Simpulan

Penelitian ini berhasil mengimplementasikan sistem pemantauan dan deteksi penyakit tanaman dengan menggunakan *framework* Flutter sebagai antarmuka aplikasi dan *Convolutional Neural Network* (CNN) untuk mendeteksi penyakit tanaman. Performa arsitektur DenseNet169 menunjukkan hasil yang sangat baik dengan akurasi sebesar 94%. Implementasi CNN pada aplikasi Android juga memberikan hasil yang memuaskan, dengan 50 sampel daun tomat mencapai tingkat akurasi sekitar 92.80% dan waktu deteksi rata-rata 1077.56 ms. Selain itu, sistem ini berfungsi dengan baik dalam merespons perubahan kelembapan tanah dengan menyalakan atau mematikan pompa sesuai *setpoint* yang ditentukan. Aplikasi ini memungkinkan pemantauan kondisi tanaman secara *real-time* dengan

delay sekitar 1.998 ms. Dengan demikian, sistem yang dikembangkan dapat diandalkan untuk membantu petani dalam mendeteksi penyakit tanaman secara cepat dan akurat, serta memantau kondisi tanaman secara *realtime* sehingga memungkinkan tindakan pencegahan yang tepat untuk menjaga kesehatan tanaman.

E. Referensi

[1] S. Wales, S. M. T. Tulung, and R. Mamarimbing, "Growth And Production Of Tomato (Solanum Lycopersicum L.) On Several Types Of Growing Media," *Jurnal Agroekoteknologi Terapan*, vol. 4, no. 1, 2023.

[2] K. Muchtar, Chairuman, Yudha Nurdin, and Afdhal Afdhal, "Pendeteksian Septoria pada Tanaman Tomat dengan Metode Deep Learning berbasis Raspberry Pi," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 5, no. 1, pp. 107–113, Feb. 2021, doi: 10.29207/resti.v5i1.2831.

[3] R. Gunawan, T. Andhika, . S., and F. Hibatulloh, "Monitoring System for Soil Moisture, Temperature, pH and Automatic Watering of Tomato Plants Based on Internet of Things," *Telekontran : Jurnal Ilmiah Telekomunikasi, Kendali dan Elektronika Terapan*, vol. 7, no. 1, pp. 66–78, Apr. 2019, doi: 10.34010/telekontran.v7i1.1640.

[4] M. R. Rojat and A. F. Febriyansyah, "Pengembangan Sistem Kendali Cerdas Dan Monitoring Pada Tanaman Tomat," *Portaldata.org*, vol. 2, no. 7, 2022.

[5] S. B. Mursalin, H. Sunardi, and Zulkifli, "Sistem Penyiraman Tanaman Otomatis Berbasis Sensor Kelembaban Tanah Menggunakan Logika Fuzzy," *JURNAL ILMIAH INFORMATIKA GLOBAL*, vol. 11, no. 1, 2020.

[6] R. Nurhasanah, L. Savina, Z. M. Nata, and I. Zulkhair, "Design and Implementation of IoT based Automated Tomato Watering System Using ESP8266," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Jun. 2021. doi: 10.1088/1742-6596/1898/1/012041.

[7] I. D. G. J. Negara, K. Wiratama, I. N. Merdana, A. Supriyadi, and I. W. Yasa, "PENYULUHAN TENTANG IRIGASI TETES," *JMM (Jurnal Masyarakat Mandiri)*, vol. 7, no. 2, p. 1607, Apr. 2023, doi: 10.31764/jmm.v7i2.13906.

[8] Y. Irawan, E. Sabna, A. Fauzan Azim, R. Wahyuni, N. Belarbi, and M. Muncho Josephine, "Automatic Chili Plant Watering Based On Internet Of Things (IOT)," *Journal of Applied Engineering and Technological Science*, vol. 3, no. 2, pp. 77–83, 2022, [Online]. Available: www.pemuri.unaux.com

[9] R. C. Sigitta, R. H. Saputra, and F. Fathulloh, "Deteksi Penyakit Tomat melalui Citra Daun menggunakan Metode Convolutional Neural Network," *AVITEC*, vol. 5, no. 1, p. 43, Feb. 2023, doi: 10.28989/avitec.v5i1.1404.

[10] T. A. Salih, A. J. Ali, and M. N. Ahmed, "Deep Learning Convolution Neural Network to Detect and Classify Tomato Plant Leaf Diseases," *OAlib*, vol. 07, no. 05, pp. 1–12, 2020, doi: 10.4236/oalib.1106296.

[11] U. Khultsum and A. Subekti, "Penerapan Algoritma Random Forest dengan Kombinasi Ekstraksi Fitur Untuk Klasifikasi Penyakit Daun Tomat," *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. 5, no. 1, p. 186, Jan. 2021, doi: 10.30865/mib.v5i1.2624.

[12] L. R. Priya, G. I. Rajathi, and Dr. R. Vedhapriyavadhana, "Crop Disease Detection and Monitoring System," *International Journal of Recent Technology*

and Engineering (IJRTE), vol. 8, no. 4, pp. 3050–3053, Nov. 2019, doi: 10.35940/ijrte.D7857.118419.

[13] M. Muchlasin, M. Hasbi, and S. Siswanti, "Decision Tree Method for Automation of Plant Sprinklers and Monitoring Based On Soil Moisture," *Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI)*, vol. 12, no. 1, pp. 25–32, Mar. 2023, doi: 10.23887/janapati.v12i1.59075.

[14] C. R. Kotta, D. Paseru, and M. Sumampouw, "Implementasi Metode Convolutional Neural Network untuk Mendeteksi Penyakit pada Citra Daun Tomat," *Jurnal_Pekommas_Vol._7_No*, vol. 2, pp. 123–132, 2022.

[15] Y. H. Natbais and A. B. S. Umbu, "Aplikasi Deteksi Penyakit pada Daun Tomat Berbasis Android Menggunakan Model Terlatih Tensorflow Lite," *TEKNOTAN*, vol. 17, no. 2, p. 83, Aug. 2023, doi: 10.24198/jt.vol17n2.1.

[16] M. R. Oktaviani and Rizky Pradana, "Prototype Sistem Pakan Ikan dan Pemantauan PH Berbasis Android dengan Metode PLC," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 5, no. 4, pp. 729–738, Aug. 2021, doi: 10.29207/resti.v5i4.3193.