

**PERANCANGAN SISTEM PENDETEKSI KEMATANGAN
BUAH DENGAN SEGMENTASI WARNA BERBASIS
MIKROKONTROLER MAPPI32**

**Oleh
DWINDY MONICA**

Skripsi

**Sebagai Salah Satu Syarat untuk Mencapai Gelar
SARJANA TEKNIK**

Pada

**Program Studi Teknik Informatika
Jurusan Teknik Elektro
Fakultas Teknik Universitas Lampung**



**FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG**

2024

ABSTRAK

PERANCANGAN SISTEM PENDETEKSI KEMATANGAN BUAH DENGAN SEGMENTASI WARNA BERBASIS MIKROKONTROLER MAPPi32

Oleh

DWINDY MONICA

Penelitian ini bertujuan untuk merancang sistem pendeteksi kematangan buah otomatis menggunakan mikrokontroler Mappi32 dan metode segmentasi warna HSV (*Hue, Saturation, Value*). Sistem ini dirancang sebagai solusi atas peningkatan produksi buah di Indonesia dan penyortiran manual yang masih banyak dilakukan, terutama di daerah pedesaan. Kematangan buah sangat mempengaruhi kualitas konsumsi, dengan warna sebagai salah satu cirinya. Dalam penelitian ini, dikembangkan dua model sistem. Model I memusatkan seluruh proses pada Mappi32, termasuk pemrosesan citra dan kontrol servo untuk menggerakkan lengan pemisah. Model II membagi beban kerja Mappi32 dengan pemrograman Python di komputer, sehingga Mappi32 difungsikan sebagai pemberi daya sensor kamera dan pengontrol servo, sementara pemrosesan data dilakukan menggunakan pemrograman Python. Hasil deteksi ditampilkan secara *real-time* melalui *dashboard* berbasis *WebSocket*. Metode yang digunakan meliputi segmentasi warna HSV, dan juga *thresholding* dengan kombinasi *adaptive* dan *Otsu threshold* serta *Gaussian filter*. Sedangkan, perancangan sistem menerapkan metode pengembangan *prototype*. Pengujian dilakukan pada objek buah stroberi dan jeruk yang disortir pada prototipe mini *conveyor*. Hasil pengujian menunjukkan bahwa Model I mengalami masalah *overheating* yang berimbas pada latensi pelepasan *buffer*. Model II terbukti lebih efisien dengan waktu pemrosesan lebih cepat, variabilitas waktu turun dari 0,94 detik ke 0,77 detik, dan tingkat keberhasilan deteksi meningkat dari 56,25% menjadi 100%. Sistem ini dapat memantau dan menghitung buah matang dan mentah secara *real-time*, sehingga meningkatkan efisiensi proses penyortiran di sektor pertanian.

Kata Kunci : Mappi32, Segmentasi HSV, *Thresholding*, Kematangan Buah

ABSTRACT

DESIGN OF A FRUIT RIPENESS DETECTION SYSTEM USING COLOR SEGMENTATION BASED ON MAPPI32 MICROCONTROLLER

By

DWINDY MONICA

This research aims to develop an automatic fruit ripeness detection system using the Mappi32 microcontroller and HSV (Hue, Saturation, Value) color segmentation. The system is designed as a solution to the increasing fruit production in Indonesia and the continued practice of manual sorting, especially in rural areas. Fruit ripeness, indicated by color, significantly affects consumption quality. In this study, two system models were developed. Model I handles all processes on the Mappi32, including image processing and servo control for sorting arm. Model II distributes the workload, using the Mappi32 as a power source for the camera sensor and a controller for the servo, while data processing is performed via Python programming on a computer. Detection results are displayed in real-time via a WebSocket-based dashboard. The methods used include HSV color segmentation, along with thresholding using a combination of adaptive and Otsu thresholding, as well as Gaussian filtering. The system design applied the prototype development method. Testing was conducted on strawberry and orange fruits that were sorted on a mini conveyor prototype. The test results showed that Model I experienced overheating issues, resulting in buffer release latency. Model II proved to be more efficient with faster processing time, a reduction in time variability from 0.94 seconds to 0.77 seconds, and an increase in detection success rate from 56.25% to 100%. The system is capable of monitoring and counting ripe and unripe fruits in real-time, thereby improving the efficiency of the sorting process in the agricultural sector.

Keywords : *Mappi32, HSV Segmentation, Thresholding, Fruit Ripeness*

Judul Skripsi : **PERANCANGAN SISTEM PENDETEKSI
KEMATANGAN BUAH DENGAN
SEGMENTASI WARNA BERBASIS
MIKROKONTROLER MAPPI32**

Nama Mahasiswa : **Dwindy Monica**

Nomor Pokok Mahasiswa : **2015061022**

Program Studi : **Teknik Informatika**

Jurusan : **Teknik Elektro**

Fakultas

Teknik



1. Komisi Pembimbing

Ir. M. Komarudin, S.T., M.T.
NIP. 196812071997031006

Ir. Titin Yulianti, S.T., M.Eng.
NIP. 198807092019032015

2. Mengetahui

Ketua Jurusan Teknik Elektro

Ketua Program Studi Teknik Informatika

Herlinawati, S.T., M.T.
NIP. 197103141999032001

Yessi Mulyani, S.T., M.T.
NIP. 197312262000122001

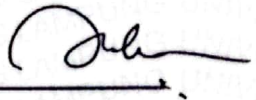
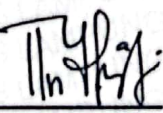

MENGESAIHKAN

1. Tim Penguji

Ketua : Ir. M. Komarudin, S.T., M.T.

Sekretaris : Ir. Titin Yulianti, S.T., M.Eng.

Penguji : Dr. Eng. Ir Mardiana, S.T., M.T., IPM

2. Dekan Fakultas Teknik



Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc.)

NIP. 19750928200112 1 002

Tanggal Lulus Ujian Skripsi : 21 November 2024

SURAT PERNYATAAN

Yang bertanda tangan di bawah ini :

Nama : Dwindy Monica

NPM : 2015061022

Dengan ini menyatakan bahwa skripsi saya dengan judul “Perancangan Sistem Pendeteksi Kematangan Buah dengan Segmentasi Warna Berbasis Mikrokontroler Mappi32” dibuat oleh saya sendiri. Selain itu, sepanjang pengetahuan saya tidak ada karya atau pendapat yang pernah diterbitkan oleh orang lain, kecuali secara tertulis diacu dalam naskah ini sebagaimana yang disebutkan dalam daftar pustaka. Semua hasil yang termuat dalam skripsi ini telah mengikuti kaidah penulisan karya tulis ilmiah Universitas Lampung. Apabila di kemudian hari terbukti bahwa skripsi ini merupakan salinan atau dibuat oleh orang lain, maka saya bersedia menerima sanksi sesuai dengan ketentuan hukum atau akademik yang berlaku.

Bandar Lampung, 28 November 2024

Pembuat pernyataan,



Dwindy Monica

NPM. 2015061022

RIWAYAT HIDUP



Dwindy Monica lahir di Desa Semuli Raya, Kec. Abung Semuli, Kab. Lampung Utara pada tanggal 03 Februari 2024. Penulis merupakan anak kedua dari dua bersaudara, oleh pasangan Bapak Jarnani dan Ibu Wigati. Penulis telah menyelesaikan pendidikan formal di SD Negeri 1 Semuli Raya pada tahun 2014, kemudian SMP Negeri 1 Abung Semuli pada tahun 2017, dan menyelesaikan pendidikan menengah di SMA Negeri 1 Abung Semuli pada tahun 2020. Pada tahun yang sama, penulis berhasil diterima sebagai mahasiswa program studi Teknik Informatika melalui jalur SNMPTN sekaligus sebagai penerima beasiswa KIP-K. Selama berkuliah, penulis sempat aktif terlibat pada berbagai kegiatan. Di antaranya adalah bergabung sebagai anggota Departemen Pengembangan Keteknikan pada Himpunan Mahasiswa Teknik Elektro (HIMATRO) periode 2021/2022, Reporter Cetak pada UKPM Teknokra tahun 2021, dan Manajer Usaha pada UKPM Teknokra tahun 2022.

Selama periode tersebut, penulis sempat lolos dalam 8 besar Opini Nasional oleh LPM Teropong UMSU dengan tema kekerasan seksual. Dalam konteks akademis, penulis juga terlibat sebagai asisten Laboratorium Teknik Komputer pada tahun 2022-2024. Penulis juga aktif mengikuti beberapa kegiatan penelitian dan pengabdian kepada masyarakat bersama dosen terkait dengan fokus penerapan *embedded system*. Selain itu, penulis sempat mengikuti kegiatan Studi Independen oleh Kampus Merdeka Batch 3 di Mitra PT. Chairos International Ventures dengan proyek *Cloud Computing for Jobseekers* menggunakan AWS pada tahun 2022 dan melaksanakan kerja praktik di PT. Queen Network Nusantara dengan proyek *Border Gateway Protocol (BGP) Network* menggunakan MikroTik dan Quagga pada tahun 2023. Penulis juga sempat lolos dalam sertifikasi kompetensi BNSP sebagai *Junior Network Administrator*.

MOTTO

لَئِنْ شَكَرْتُمْ لَأَزِيدَنَّكُمْ

“Sesungguhnya jika kamu bersyukur, niscaya Aku akan menambah (nikmat) kepadamu”

(QS. Ibrahim : 7)

“Untuk mendapatkan apa yang kamu suka, pertama kamu harus sabar dengan apa yang kamu tidak suka”

(Imam Al-Ghazali)

“Hati, jiwa, dan pribadi yang sehat adalah yang tidak berprasangka buruk terhadap orang lain”

(Adzando Chrisdavema Zaelani)

“Hiduplah sesuai nilai diri sendiri dan melangkahlah setapak demi setapak. Meski mungkin membutuhkan waktu lebih lama, jangan merasa kalah. Kita akan baik-baik saja selama berada di jalur yang benar (jalurmu sendiri)”

(DK dan Hoshi SEVENTEEN)

“Hiduplah seperti dandelion: mampu beradaptasi dan tumbuh dengan baik di manapun kamu berada”

(Dwindy Monica)

PERSEMBAHAN



Alhamdulillahirabbil'aalamiin, sujud syukur dan segala puji kepada Allah SWT, Tuhan Yang Maha Agung dan Maha Tinggi atas segala rahmat dan hidayah-Nya. Tak lupa sholawat teriring salam kepada Nabi Muhammad SAW yang selalu menjadi teladan dalam kehidupanku. Semoga keberhasilan ini menjadi satu langkah awal untuk masa depanku dalam meraih cita-cita.

Kupersembahkan dengan tulus karya ini untuk Kedua Orang Tuaku, terutama teruntuk Ibuku tercinta, yang senantiasa melangitkan doa-doa sepanjang hidupnya untukku. Teruntuk Nenek, dan Kakakku Verdyna Wulandari yang senantiasa membantu dan mendukungku dalam setiap keputusanku. Kuucapkan terima kasih sebesar-besarnya karena telah memberikan banyak kebaikan yang tidak akan pernah bisa terbalaskan. Skripsi ini merupakan bentuk perjuangan, cinta, kasih sayang, dan terima kasihku atas segala yang telah kalian berikan kepadaku. Semoga selepas ini, aku bisa mencapai apa yang aku dan kalian semua impikan.

Tak lupa juga, kepada seluruh dosen, yang senantiasa membimbing dan mengarahkanku, serta selalu melibatkanku dalam setiap kegiatan, sehingga berhasil membentuk diriku menjadi pribadi yang lebih semangat, termotivasi dan percaya diri. Kepada rekan-rekan seperjuanganku yang selalu kebersamai setiap langkahku. Serta, kepada seluruh civitas Jurusan Teknik Elektro Universitas Lampung. Semoga sedikit ilmu yang tertuang dalam Skripsi ini, dapat menjadi landasan yang bermanfaat kedepannya.

Terakhir, terima kasih pada diriku sendiri, **Dwindy Monica**.

SANWACANA

Alhamdulillah *rabbi'l'alamiin*, segala puji dan syukur ke hadirat Allah SWT. atas segala rahmat, karunia, dan hidayah-Nya, sehingga saya dapat menyelesaikan skripsi dengan judul **“Perancangan Sistem Pendeteksi Kematangan Buah dengan Segmentasi Warna Berbasis Mikrokontroler Mappi32”**. Shalawat serta salam senantiasa tercurah kepada Nabi Muhammad SAW yang telah menerangi jalan umatnya melalui dakwah-dakwahnya. Dalam proses penyusunan skripsi ini, tentu tidak lepas dari bantuan, bimbingan serta dukungan dari berbagai pihak, baik secara langsung maupun tidak langsung. Oleh karena itu, saya ucapkan terima kasih kepada semua pihak, khususnya kepada :

1. Dekan Fakultas Teknik Universitas Lampung, Bapak Dr. Eng. Helmy Fitriawan, S.T., M.Sc.,
2. Ibu Herlinawati, S.T., M.T. selaku Ketua Jurusan Teknik Elektro Universitas Lampung.
3. Ibu Yessi Mulyani, S.T., M.T selaku Ketua Program Studi Teknik Informatika Universitas Lampung, sekaligus dosen Pembimbing Akademik yang selalu memberikan ilmu dan masukan selama menjalani masa perkuliahan.
4. Bapak Ir. Muhammad Komarudin, S.T., M.T. selaku pembimbing utama yang telah membantu dalam proses pengerjaan skripsi ini dengan memberikan banyak afirmasi positif, ide-ide, saran, bimbingan, semangat dan juga motivasi.
5. Ibu Ir. Titin Yulianti, S.T., M.Eng. selaku dosen pembimbing pendamping yang telah meluangkan waktu untuk membantu, membimbing, memberikan banyak ide baru, dukungan, motivasi, afirmasi positif, berbagi pengalaman dan selalu mendengarkan keluh kesah, baik sebelum maupun selama proses

penyelesaian skripsi ini.

6. Ibu Dr. Eng. Ir. Mardiana, S.T., M.T., I.P.M. selaku dosen penguji yang telah banyak membantu dengan memberikan saran, masukan, dan motivasi dalam pengembangan skripsi ini.
7. Seluruh Dosen pada Jurusan Teknik Elektro yang telah membekali saya dengan berbagai ilmu pengetahuan dan pengalaman.
8. Mba Rika Asliana selaku Admin Program Studi Teknik Informatika yang telah membantu dalam segala penyelesaian proses administrasi saya.
9. Keluarga tercinta, terutama Ibu, bersama Kakak, Nenek, Kakek, Paman, Bibi, dan Ayah yang telah mendidik dan membesarkan saya, serta tiada hentinya memberikan semangat, dukungan, doa, bantuan dalam segala hal, tidak pernah menuntut dan selalu mendahulukan keputusan yang saya buat hingga skripsi ini dapat terselesaikan dengan baik.
10. Sahabat tercinta, Tia Safitri yang senantiasa menemani setiap langkah, mendengarkan keluh-kesah, memberikan nasihat, dan dukungan, sehingga skripsi ini dapat diselesaikan bersama dengan mental yang membaik.
11. Lutfi Nur Latifah (Umik) sebagai teman sekamar yang selalu menghibur, menenangkan, membantu dan mengetahui berbagai suka dan duka saya.
12. Teman-teman terkasih, yaitu Renata, Anisa Fitriyani, Regita Agnes, Afifah, Putri Pratiwi, Elda, Asha, Dwi Putri, Zeri dan Maylan yang selalu menemani, membantu, dan dengan sabar menanggapi sikap dan suasana hati saya yang berubah-ubah.
13. M. Naufal Rizqullah yang selalu menghibur dan memberikan waktu, pendapat, serta dukungan selama masa perkuliahan hingga skripsi ini dapat diselesaikan dengan baik.
14. Serta kepada seluruh civitas akademika Unila, staff administrasi, dan pihak-pihak yang tidak bisa saya sebutkan satu persatu.

Penelitian ini masih jauh dari kata sempurna, dikarenakan terbatasnya pengalaman dan pengetahuan yang saya miliki. Oleh karena itu, diharapkan segala bentuk saran, masukan dan kritik yang membangun dari berbagai pihak. Semoga skripsi ini dapat bermanfaat bagi para pembaca dan semua pihak, khususnya dalam bidang pengembangan *embedded system* dan *image processing*.

Bandar Lampung, 28 November 2024

Penulis,

A handwritten signature in black ink, appearing to read 'Dwindy Monica', with a large, stylized initial 'D' at the top.

Dwindy Monica

NPM. 2015061022

DAFTAR ISI

	Halaman
DAFTAR ISI	vi
DAFTAR TABEL	ix
DAFTAR GAMBAR	x
DAFTAR SINGKATAN	xiv
I. PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah.....	3
1.3 Tujuan Penelitian.....	3
1.4 Manfaat Penelitian.....	4
1.5 Batasan Masalah.....	4
1.6 Sistematika Penulisan.....	4
II. TINJAUAN PUSTAKA	7
2.1 Kematangan Buah	7
2.2 Pengolahan Citra Digital	7
2.3 Warna Digital	8
2.4 Segmentasi Warna HSV	10
2.5 <i>Thresholding</i>	12
2.5.1 Histogram.....	13
2.5.2 <i>Adaptive Threshold</i>	14
2.5.3 <i>Otsu Threshold</i>	15
2.5.4 <i>Gaussian Filter</i>	16
2.6 OpenCV.....	17
2.7 Metode <i>Prototype</i>	18

2.8	Protokol HTTP dan <i>WebSocket</i>	20
2.9	Mappi32	21
2.10	Modul Kamera OV2640.....	23
2.11	Motor Servo.....	25
2.12	Modul LM2596	26
2.13	Motor DC	27
2.14	Arduino IDE.....	28
2.15	Visual Studio Code.....	28
2.16	Jeruk	29
2.17	Stroberi.....	31
2.18	Penelitian Terkait	32
III.	METODOLOGI PENELITIAN	39
3.1	Waktu dan Tempat	39
3.2	Alat dan Bahan Penelitian	39
3.2.1	Alat.....	39
3.2.2	Bahan	42
3.3	Tahap Penelitian.....	42
3.4	Rancangan Sistem	45
3.4.1	Algoritma Segmentasi Warna.....	45
3.4.2	Blok Diagram Sistem.....	53
3.4.3	Model Komunikasi Sistem.....	55
3.4.4	Model Algoritma Pemrograman Sistem	59
3.4.5	Desain Sistem	68
3.4.6	<i>Flowchart</i> Cara Kerja Sistem	69
3.5	Pengujian Sistem	71
3.5.1	Pengujian Komponen.....	71
3.5.2	Pengujian Keseluruhan Sistem	72
IV.	HASIL DAN PEMBAHASAN	73
4.1	Pengolahan Sampel Warna.....	73
4.1.1	Akuisisi Citra	74
4.1.2	Pengolahan Citra Sampel.....	75
4.1.3	Output Histogram Stroberi.....	79

4.1.4	Output Histogram Jeruk.....	81
4.1.5	Output <i>Combined Threshold</i> Citra.....	83
4.1.6	Penetapan Implementasi <i>Threshold</i>	85
4.2	Kalibrasi Jarak Kamera	89
4.3	Perancangan Mesin <i>Conveyor</i>	93
4.4	Perancangan Model I Sistem Pendeteksi Kematangan Buah.....	99
4.4.1	Rangkaian Sistem Pendeteksi Kematangan Buah Model I.....	99
4.4.2	<i>Display</i> Hasil Hitung Buah Model I	109
4.5	Pengujian Sistem Model I	110
4.5.1	Pengujian Komponen Model I.....	110
4.5.2	Pengujian Waktu Proses Model I.....	112
4.5.3	Hasil Tingkat Keberhasilan Model I.....	121
4.6	Perancangan Model II Sistem Pendeteksi Kematangan Buah.....	123
4.6.1	Rangkaian Sistem Pendeteksi Kematangan Buah Model II	125
4.6.2	<i>Display</i> Hasil Hitung Buah Model II.....	136
4.7	Pengujian Sistem Model II.....	139
4.7.1	Pengujian Komponen Model II.....	139
4.7.2	Pengujian Waktu Proses Model II	141
4.7.3	Hasil Tingkat Keberhasilan Sistem Model II	145
4.8	Analisis Keseluruhan Sistem.....	147
V.	SIMPULAN DAN SARAN.....	151
5.1	Simpulan.....	151
5.2	Saran.....	152
	DAFTAR PUSTAKA	153
	LAMPIRAN.....	159

DAFTAR TABEL

	Halaman
Tabel 2.1 Spesifikasi Mappi32.....	23
Tabel 2.2 Spesifikasi ESP32-Cam	25
Tabel 2.3 Spesifikasi Motor Servo SG90.....	26
Tabel 2.4 Spesifikasi Motor DC <i>Gearbox</i>	27
Tabel 2.5 Perubahan Warna Buah Jeruk	30
Tabel 2.6 Perubahan Warna Buah Stroberi.....	32
Tabel 2.7 Penelitian Terkait	35
Tabel 3.1 Jadwal Pengerjaan Skripsi	39
Tabel 3.2 Alat Penelitian.....	40
Tabel 3.3 Daftar <i>Library</i> dan Modul.....	41
Tabel 3.4 Letak Komponen Mesin.....	69
Tabel 4.1 Hasil <i>Combined Threshold</i>	84
Tabel 4.2 <i>Threshold HSV</i>	89
Tabel 4.3 Kalibrasi Jarak Kamera ke Buah.....	90
Tabel 4.4 Koneksi Komponen Penggerak.....	97
Tabel 4.5 <i>Wiring</i> Komponen Model I.....	100
Tabel 4.6 Pengujian Komponen Model I.....	110
Tabel 4.7 Pengujian Waktu Proses Model I.....	114
Tabel 4.8 Tingkat Keberhasilan Sistem Model I	121
Tabel 4.9 <i>Wiring</i> Komponen Model II.....	125
Tabel 4.10 Pengujian Komponen Model II.....	140
Tabel 4.11 Pengujian Waktu Proses Model II	141
Tabel 4.12 Hasil Tingkat Keberhasilan Sistem Model II.....	145
Tabel 4.13 Perbandingan Hasil Pengujian	149

DAFTAR GAMBAR

	Halaman
Gambar 2.1 Ruang Warna HSV	9
Gambar 2.2 Skala HSV	10
Gambar 2.3 Spesifikasi Board Mappi32	22
Gambar 2.4 Modul OV2640	24
Gambar 2.5 ESP32-Cam	24
Gambar 2.6 Motor Servo SG90	25
Gambar 2.7 Modul LM2596	26
Gambar 2.8 Motor DC <i>Gearbox</i>	27
Gambar 2.9 <i>Dashboard</i> Arduino IDE	28
Gambar 2.10 <i>Dashboard</i> Visual Studio Code	29
Gambar 2.11 Buah Jeruk	30
Gambar 2.12 Buah Stroberi	31
Gambar 3.1 Diagram Alir Penelitian	43
Gambar 3.2 Alur Pencarian <i>Threshold</i>	46
Gambar 3.3 Segmentasi HSV pada Sistem	51
Gambar 3.4 Blok Diagram Sistem Model I	54
Gambar 3.5 Blok Diagram Sistem Model II	54
Gambar 3.6 Diagram Komunikasi Model I	56
Gambar 3.7 Diagram Komunikasi Model II	57
Gambar 3.8 Skema Peran Komponen pada Sistem	59
Gambar 3.9 Algoritma Program Model I	61
Gambar 3.10 Algoritma Program Kamera dan Mappi32 Model II	63
Gambar 3.11 Algoritma Program Python dan Dashboard Model II	66
Gambar 3.12 Sketsa Mesin Sortir (Desain Pribadi)	69
Gambar 3.13 Cara Kerja Sistem	70

Gambar 4.1 Sampel Citra Stroberi Matang.....	75
Gambar 4.2 Sampel Citra Jeruk Matang.....	75
Gambar 4.3 Sampel Buah Mentah.....	75
Gambar 4.4 Konversi dan Ekstraksi Warna.....	76
Gambar 4.5 Program Visualisasi Histogram.....	76
Gambar 4.6 Program Metode <i>Gaussian Filter</i>	77
Gambar 4.7 Program Metode <i>Otsu Thresholding</i>	77
Gambar 4.8 Program Metode <i>Adaptive Thresholding</i>	78
Gambar 4.9 Program Menghitung <i>Combined Threshold</i>	78
Gambar 4.10 Program Bentuk <i>Binary Combined Threshold</i>	79
Gambar 4.11 Histogram Citra Stroberi_1.....	79
Gambar 4.12 Histogram Citra Stroberi_2.....	80
Gambar 4.13 Histogram Citra Stroberi_Mentah.....	81
Gambar 4.14 Histogram Citra Jeruk_1.....	82
Gambar 4.15 Histogram Citra Jeruk_2.....	82
Gambar 4.16 Histogram Citra Jeruk_Mentah.....	83
Gambar 4.17 Uji <i>Saturation</i> dan <i>Value</i>	86
Gambar 4.18 Program Kalibrasi HSV dan H.....	87
Gambar 4.19 Kalibrasi I HSV dan H.....	88
Gambar 4.20 Kalibrasi II HSV dan H.....	88
Gambar 4.21 Kalibrasi III HSV dan H.....	88
Gambar 4.22 Hasil <i>Stream</i> Tes Kamera.....	90
Gambar 4.23 Perakitan Badan <i>Conveyor</i>	93
Gambar 4.24 Perakitan Tiang <i>Conveyor</i>	94
Gambar 4.25 <i>Bracket</i> Sensor Kamera.....	94
Gambar 4.26 Perakitan <i>Roller Conveyor</i>	95
Gambar 4.27 Perakitan <i>Belt Conveyor</i>	95
Gambar 4.28 Potensiometer LM2596.....	96
Gambar 4.29 Skematik Diagram Penggerak <i>Conveyor</i>	97
Gambar 4.30 Sistem Mini <i>Conveyor</i>	98
Gambar 4.31 Tampilan Objek di Bawah Kamera.....	98
Gambar 4.32 Skematik Diagram Model I.....	100

Gambar 4.33 <i>Setup</i> Sensor Kamera	101
Gambar 4.34 <i>Looping</i> Sensor Kamera	102
Gambar 4.35 <i>Tools Sketch</i> Kamera	103
Gambar 4.36 Sensor Gagal Diinisialisasi	104
Gambar 4.37 Sensor Berhasil Berjalan	104
Gambar 4.38 <i>Setup</i> Mappi32	104
Gambar 4.39 <i>Looping</i> Mappi32	105
Gambar 4.40 Konversi RGB ke HSV Mappi32	106
Gambar 4.41 Implementasi <i>Threshold</i> Mappi32	107
Gambar 4.42 Hitung Piksel dan Persentase Warna Mappi32	107
Gambar 4.43 Deteksi Kematangan Mappi32	108
Gambar 4.44 <i>Display</i> Hasil Hitung Model I	110
Gambar 4.45 Objek Pengujian	113
Gambar 4.46 Proses Pengujian	114
Gambar 4.47 Output Serial Monitor	119
Gambar 4.48 <i>Output</i> JSON	120
Gambar 4.49 Skematik Komponen Model II	125
Gambar 4.50 <i>Setup</i> Sensor Kamera Model II	127
Gambar 4.51 Output <i>Stream Link</i>	128
Gambar 4.52 <i>Setup</i> Mappi32 Model II	128
Gambar 4.53 Kontrol Servo Model II	129
Gambar 4.54 Inisialisasi Variabel	130
Gambar 4.55 Program Akuisisi Citra Model II	131
Gambar 4.56 Konversi RGB ke HSV Model II	131
Gambar 4.57 Implementasi <i>Threshold</i> Model II	132
Gambar 4.58 Segmentasi Citra Python	132
Gambar 4.59 Klasifikasi Kematangan Buah	133
Gambar 4.60 Inisialisasi Server <i>WebSocket</i>	134
Gambar 4.61 <i>Framing, Encoding, dan Sending</i> Data	135
Gambar 4.62 Elemen DOM JavaScript	136
Gambar 4.63 Membuat Koneksi <i>WebSocket</i>	137
Gambar 4.64 Penanganan Pesan <i>WebSocket</i>	137

Gambar 4.65 Fungsi <i>Capture</i> Gambar	138
Gambar 4.66 <i>Error-Handling WebSocket</i>	138
Gambar 4.67 <i>Dashboard Counting</i> Buah Model II	139
Gambar 4.68 <i>Dashboard Hasil Counting</i>	144
Gambar 4.69 Contoh Hasil <i>Capture</i>	144

DAFTAR SINGKATAN

API	: <i>Application Programming Interface</i>
CIELAB	: <i>Commission Internationale de l'Eclairage Lab</i>
CMYK	: <i>Cyan Magenta Yellow Black</i>
CNN	: <i>Convolutional Neural Network</i>
CPU	: <i>Central Processing Unit (Unit Pengolahan Pusat)</i>
CSS	: <i>Cascading Style Sheets</i>
DC	: <i>Direct Current</i>
DMIPS	: <i>Dhrystone Million Instructions per Second</i>
DOM	: <i>Document Object Model</i>
ESP IDF	: <i>Espressif IoT Development Framework</i>
FCM	: <i>First Cumulative Moment</i>
FPS	: <i>Frames per Second</i>
GND	: <i>Ground</i>
GPIO	: <i>General-Purpose Input/Output</i>
Ha	: Hektar (satuan luas lahan)
HSB	: <i>Hue Saturation Brightness</i>
HSV	: <i>Hue Saturation Value</i>
HTML	: <i>HyperText Markup Language</i>
HTTP	: <i>Hyper Text Transfer Protocol</i>
IC	: <i>Integrated Circuit</i>
IDE	: <i>Integrated Development Environment</i>
IO	: <i>Input Output</i>
IoT	: <i>Internet of Things</i>
IP	: <i>Internet Protocol</i>
JPEG	: <i>Joint Photographic Experts Group</i>
JSON	: <i>JavaScript Object Notation</i>

KB	: Kilobyte
Kgf	: <i>Kilogram-force</i>
KHz	: Kilohertz
KiB	: Kibibyte
LCD	: <i>Liquid Crystal Display</i>
Li-Po	: <i>Lithium Polymer</i>
LM2596	: <i>Linear Module 2596</i>
LoRa	: <i>Long Range</i>
Ma	: Miliampere
MHz	: Megahertz
MJPEG	: <i>Motion JPEG</i>
MP	: Mega Pixel
NodeMCU	: <i>Node Microcontroller Unit</i>
OpenCV	: <i>Open Source Computer Vision</i>
OS	: <i>Operating System</i>
OTA	: <i>Over-the-Air</i>
OV2640	: <i>OmniVision 2640</i>
PCB	: <i>Printed Circuit Board</i>
PSRAM	: <i>Pseudo Static Random Access Memory</i>
PVC	: <i>Polyvinyl Chloride</i>
PWM	: <i>Pulse Width Modulation</i>
QVGA	: <i>Quarter Video Graphics Array</i>
QQVGA	: <i>Quarter Quarter Video Graphics Array</i>
RAM	: <i>Random Access Memory</i>
RGB	: <i>Red Green Blue</i>
ROM	: <i>Read-Only Memory</i>
RPM	: <i>Revolutions per Minute</i>
RST	: Reset
RX	: <i>Receive</i>
SD	: <i>Secure Digital (Memory Card)</i>
SDM	: <i>Sumber Daya Manusia</i>
SPI	: <i>Serial Peripheral Interface</i>

SRAM	: <i>Static Random Access Memory</i>
TCP	: <i>Transmission Control Protocol</i>
TTL	: <i>Transistor-Transistor Logic</i>
TX	: <i>Transmit</i>
UART	: <i>Universal Asynchronous Receiver-Transmitter</i>
URL	: <i>Uniform Resource Locator</i>
USB	: <i>Universal Serial Bus</i>
UXGA	: <i>Ultra Extended Video Graphics Array</i>
V	: <i>Volt (satuan)</i>
VCC	: <i>Voltage Common Collector</i>
VSCode	: <i>Visual Studio Code</i>
WiFi	: <i>Wireless Fidelity</i>
YCbCr	: <i>Luminance, Blue-difference, Red-difference</i>
YUV	: <i>Luminance and Chrominance</i>
ZCM	: <i>Zeroth Cumulative Moment</i>

I. PENDAHULUAN

1.1 Latar Belakang

Indonesia adalah salah satu negara penghasil buah-buahan dengan jenis yang cukup beragam. Hampir di seluruh wilayah Indonesia merupakan daerah yang menghasilkan bermacam-macam jenis tanaman buah karena didukung oleh iklim tropis. Berdasarkan data Statistik Pertanian oleh Badan Pusat Statistik dan Direktorat Jenderal Hortikultura Republik Indonesia (2022), produksi buah-buahan di Indonesia cenderung meningkat dari 19.643.616 Ton pada tahun 2017 menjadi 25.975.508 Ton pada tahun 2021. Luas lahan panen (*Harvested Area*) untuk jenis tumbuhan buah-buahan pun meningkat dari tahun 2017-2022, yaitu sekitar 279.767 Ha [1]. Keanekaragaman dan kekayaan jenis buah-buahan di Indonesia, tidak hanya berpengaruh pada aspek perekonomian saja, melainkan berdampak juga terhadap kesehatan masyarakatnya. Buah-buahan mengandung berbagai nutrisi yang sangat penting untuk kesehatan. Terlebih lagi apabila buah yang dikonsumsi atau diolah memiliki tingkat kematangan yang sesuai. Hal ini dikarenakan tingkat kematangan buah cukup menentukan kualitas dan mutu dari buah tersebut.

Oleh sebab itu, menentukan tingkat kematangan buah-buahan dalam proses pemanenan akan sangat penting. Penentuan tingkat kematangan buah dapat terdiri dari berbagai parameter, salah satu ciri yang mudah ditemukan adalah adanya perubahan warna. Perubahan warna pada sayur dan buah disebabkan oleh adanya pigmen yang dikandungnya. Warna adalah salah satu karakteristik yang digunakan untuk menentukan kematangan buah dan menentukan keseluruhan dari kualitas buah tersebut [2]. Meskipun demikian, tingkat kematangan buah masih dapat berubah meski telah dilakukan pemanenan. Perubahan tersebut dapat

disebabkan oleh metabolisme atau proses biokimia di dalam buah yang belum sepenuhnya berhenti. Artinya, tetap diperlukan adanya proses pasca-panen seperti penyortiran buah untuk memilah buah mana yang dapat langsung dikonsumsi, dapat diolah ataupun akan didistribusikan lagi/diekspor. Proses penyortiran sendiri termasuk dalam kategori pasca-panen primer, yang sarana dan teknologinya akan sangat memengaruhi mutu produk hasil pertanian [3].

Saat ini, proses sortir buah di Indonesia masih banyak yang dilakukan secara manual/tradisional [3], [4]. Rendahnya penggunaan sarana dan teknologi ini umumnya disebabkan oleh kualitas SDM yang masih rendah serta kurangnya sarana dan teknologi pasca-panen, terutama di pinggiran kota ataupun pedesaan [3]. Metode ini tentunya kurang efektif baik dari segi waktu, biaya dan tenaga kerja, terutama ketika hasil produksi buah mencapai skala yang cukup besar. Oleh karena itu, diperlukan pemanfaatan teknologi yang dapat menjadi solusi untuk menjawab permasalahan tersebut. Salah satu teknologi yang dapat diimplementasikan adalah sistem tertanam berbasis mikrokontroler dan sensor. Pemanfaatan komponen elektronik dalam sektor pertanian adalah salah satu langkah yang tepat untuk membantu mengoptimalkan proses, meningkatkan efisiensi, menekan biaya, serta memberikan solusi untuk menjawab berbagai tantangan terkait pertanian. Berdasarkan topik tersebut, maka akan dirancang sebuah sistem pendeteksi kematangan buah untuk proses penyortiran secara otomatis.

Untuk merancang sistem pendeteksi kematangan buah tersebut, mikrokontroler Mappi32 dipilih sebagai mikrokontroler utama karena Mappi32 memiliki spesifikasi yang cukup memadai. Meskipun memiliki harga yang terjangkau, Mappi32 dilengkapi dengan frekuensi CPU yang cukup besar, yaitu 240MHz dengan kapasitas SRAM 520 KB. Selain itu, Mappi32 mendukung penggunaan sebagai server HTTP tanpa memerlukan modul tambahan, serta dapat digunakan pada beberapa platform pemrograman, sehingga memudahkan dalam memproses program yang digunakan [5]. Mappi32 juga merupakan produk inovasi generasi muda Indonesia yang layak untuk diuji coba untuk implementasi di berbagai sektor, termasuk pertanian.

Dengan berbasiskan Mappi32, proses klasifikasi warna menggunakan metode pengolahan citra digital dengan model segmentasi warna HSV (*Hue, Saturation, Value*) dipilih untuk mendeteksi kematangan buah secara otomatis. Ruang warna HSV adalah model warna yang merepresentasikan warna asli seperti penglihatan mata manusia, sehingga mudah untuk mendeteksi adanya perubahan dan perbedaan warna. Model warna HSV juga dapat memberikan hasil yang lebih stabil dan konsisten meskipun terdapat variasi pencahayaan pada lingkungan sekitarnya [6]. Selain melakukan pemilahan buah berdasarkan warna, sistem ini juga mampu menampilkan hasil *counting* buah. Hal ini bertujuan untuk memudahkan pengguna dalam mengetahui jumlah buah yang matang dan belum matang. Hasil *counting* tersebut divisualisasikan melalui *dashboard* secara *real-time*.

1.2 Perumusan Masalah

Berdasarkan latar belakang tersebut, terdapat beberapa kajian masalah yang mendasari penelitian ini, yaitu :

1. Bagaimana merancang program pengolahan citra berbasis metode segmentasi HSV untuk mendeteksi kematangan buah berdasarkan warna?
2. Bagaimana menyesuaikan program pengolahan citra digital untuk mendeteksi kematangan buah berdasarkan warna pada mikrokontroler?
3. Bagaimana menampilkan visualisasi data hasil *counting* klasifikasi buah matang dan mentah pada *dashboard*?

1.3 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah :

1. Merancang program pengolahan citra digital berbasis metode segmentasi HSV untuk mendeteksi kematangan buah berdasarkan warna.
2. Merancang dan menyesuaikan program pada mikrokontroler yang mendukung pengolahan citra digital untuk mendeteksi kematangan buah berdasarkan warna.
3. Menampilkan visualisasi data hasil *counting* klasifikasi buah matang dan

mentah pada *dashboard*.

1.4 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat bagi pengguna untuk :

1. Mengetahui tingkat relevansi dari pengimplementasian mikrokontroler sebagai pembangun sistem pendeteksi kematangan.
2. Mengetahui dan mendeteksi tingkat kematangan buah berdasarkan perubahan warna, sehingga dapat melakukan pemilahan pasca-panen yang sesuai agar didapatkan hasil buah yang berkualitas.
3. Memonitor jumlah buah yang sudah matang dan belum matang melalui visualisasi data *counting* buah pada *dashboard* secara *real-time*.

1.5 Batasan Masalah

Adapun batasan masalah dalam penelitian ini adalah :

1. Parameter yang digunakan dalam penelitian adalah kematangan buah berdasarkan warna dengan metode pengolahan citra berbasis segmentasi citra HSV.
2. Buah yang digunakan sebagai objek penelitian adalah buah stroberi dan buah jeruk.
3. Keluaran penelitian berupa *prototype* sistem pendeteksi kematangan berbasis mikrokontroler.
4. Penelitian tidak berfokus pada analisis desain antarmuka dan *deployment dashboard*.
5. Tidak membahas terkait komunikasi data.

1.6 Sistematika Penulisan

Adapun sistematika penulisan laporan penelitian ini adalah :

1) PENDAHULUAN

Pada bagian ini membahas terkait permasalahan serta tujuan yang melatarbelakangi dilakukannya penelitian ini. Permasalahan yang dihadapi adalah

kurangnya pemanfaatan teknologi dalam proses sortir buah di Indonesia. Oleh sebab itu, dalam penelitian ini membahas terkait perancangan sistem penyortiran buah secara otomatis berbasis mikrokontroler. Selain itu, dalam bab ini juga dibahas terkait batasan masalah agar ruang lingkup dari topik penelitian ini tidak meluas dan memiliki fokus tertentu.

2) TINJAUAN PUSTAKA

Pada bagian ini membahas teori-teori yang terkait dengan penelitian serta digunakan sebagai penunjang penelitian. Teori pendukung ini dapat diambil dari berbagai sumber ilmiah seperti buku dan jurnal. Teori penunjang yang dibahas adalah Kematangan Buah, Pengolahan Citra Digital, Warna Digital, Segmentasi Warna HSV, *Thresholding*, Histogram, *Adaptive Threshold*, *Otsu Threshold*, *Gaussian Filter*, OpenCV, Metode *Prototype*, Protokol HTTP dan *WebSocket*, Mappi32, Modul Kamera OV2640, Motor Servo, Modul LM2596, Motor DC, Arduino IDE, Visual Studio Code, Jeruk, dan Stroberi. Selain membahas teori pendukung, pada bab ini juga menjelaskan seputar 10 penelitian terkait dari penelitian terdahulu secara singkat.

3) METODOLOGI PENELITIAN

Pada bagian ini membahas waktu penelitian dan tempat pengerjaan penelitian, tahapan penelitian, perancangan desain sistem sampai desain pengujian. Selain itu, dijelaskan juga analisa kebutuhan yang digunakan seperti alat dan bahan, serta metode yang digunakan dalam menyelesaikan penelitian termasuk rancangan-rancangan secara garis besar. Alat dan bahan yang dibahas adalah perangkat lunak, perangkat keras, serta objek penelitian yang digunakan untuk mendukung perancangan sistem termasuk parameter yang dibahas. Kemudian metode yang digunakan adalah metode *prototype* dengan pengolahan citra menggunakan algoritma segmentasi citra ruang HSV.

4) PEMBAHASAN

Pada bagian ini akan membahas hasil olah data penelitian. Hasil penelitian dapat berupa analisis implementasi dan pengujian sistem terhadap parameter yang diujikan. Adapun parameter yang dibahas adalah kematangan berdasarkan warna, jarak antara sensor kamera dengan objek, *delay*, serta hasil *counting*. Hasil

penelitian tersebut disertai dengan penjelasan dan analisis untuk memahami keseluruhan hasil sistem yang dibuat.

5) KESIMPULAN DAN SARAN

Pada bagian ini berisi kesimpulan berdasarkan temuan penelitian, baik temuan positif maupun temuan negatif (kekurangan pada sistem). Selain itu, pada bab 5 ini juga diberikan saran untuk mendukung pengembangan atau penelitian secara lebih lanjut.

II. TINJAUAN PUSTAKA

2.1 Kematangan Buah

Kematangan buah (*Fruit Maturity*) dapat diartikan sebagai tahap perkembangan pada buah-buahan yang sudah mencapai tingkat pertumbuhan yang optimal dan siap untuk dipanen/dikonsumsi. Setiap jenis buah-buahan memiliki ciri khas tersendiri yang menandakan bahwa buah tersebut telah siap dikonsumsi. Untuk menentukan kematangan dari buah-buahan dapat dilihat dari berbagai parameter, salah satunya adalah perubahan warna kulit buah. Perubahan warna buah menjadi ciri yang paling mudah dikenali karena adanya informasi terkait perubahan fisiologi pada buah tersebut. Perubahan warna buah juga dapat digunakan untuk menentukan kualitas serta kelayakan konsumsi dari buah-buahan tersebut.

Pada beberapa tanaman buah, perubahan warna kulit dapat terjadi akibat adanya degradasi klorofil ataupun sintesis dari pigmen warna. Hal ini dikarenakan selama pertumbuhan buah, terjadi proses perubahan biokimia yang memengaruhi fisik dari buah tersebut. Selain perubahan warna kulit buah, terdapat beberapa parameter lain seperti peningkatan berat dan ketebalan buah, kekenyalan dan tekstur buah, ukuran dan bentuk, aroma, rasa, serta pecahan kulit ataupun tingkat kejatuhan buah dari tangkainya [7]. Dalam penelitian ini, penelitian berfokus pada objek buah yang tingkat kematangannya dapat dilihat dari adanya perubahan warna.

2.2 Pengolahan Citra Digital

Citra dapat diartikan sebagai gambaran, kemiripan atau inisiasi suatu objek. Citra digital merupakan gambar dua dimensi yang dapat ditampilkan melalui layar monitor sebagai nilai diskrit digital yang disebut dengan pixel (*picture*

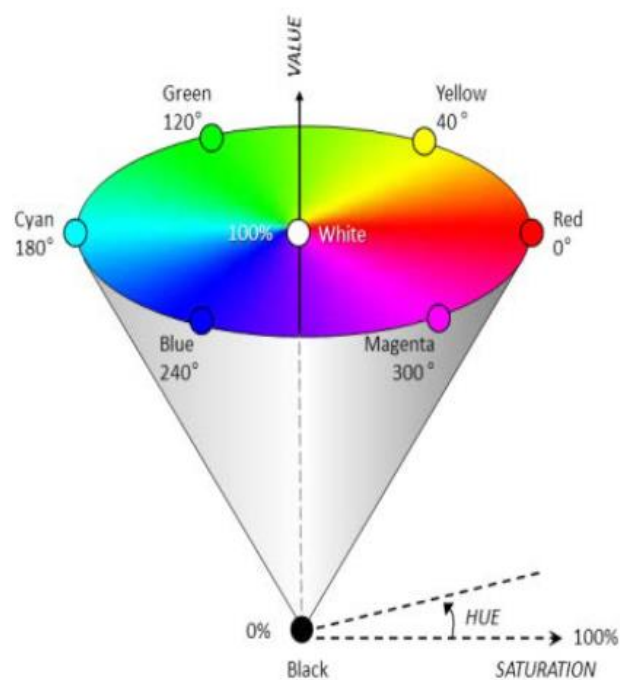
elements). Piksel sendiri dapat diartikan sebagai elemen citra yang menunjukkan intensitas suatu warna [8]. Citra dapat berupa rekaman data optik analog seperti foto, sinyal video seperti gambar pada layar monitor, atau digital yang tersimpan langsung pada media penyimpanan [9]. Untuk mengolah suatu citra digital, umumnya digunakan metode pengolahan citra.

Digital Image Processing (Pengolahan Citra Digital) merupakan proses yang dilakukan pada suatu citra digital untuk memperbaiki kualitas citra atau memperoleh informasi dari citra tersebut. Pada pengolahan citra, umumnya meliputi beberapa proses, seperti akuisisi citra, *preprocessing*, ekstraksi ciri citra, segmentasi citra, dan juga klasifikasi citra [10]. Umumnya, citra digital diperoleh dari penangkapan gambar melalui perangkat kamera atau *scanner*. Pengolahan citra banyak digunakan dalam berbagai bidang, seperti bidang medis, lingkungan, keamanan, bahkan pertanian. Pengolahan citra biasanya dibantu dengan menggunakan bahasa pemrograman Python. Hal ini dikarenakan Python merupakan bahasa pemrograman yang mudah untuk dibaca dan dimengerti oleh manusia, serta umum digunakan untuk pengolahan data.

2.3 Warna Digital

Warna atau *color* dapat dianggap sebagai persepsi yang dirasakan dan dilihat oleh sistem visual manusia terhadap panjang gelombang cahaya suatu objek [9]. Warna yang terlihat merupakan spektrum cahaya yang dipantulkan oleh benda dan ditangkap oleh indra penglihatan (mata) manusia untuk diterjemakan sebagai warna tertentu oleh otak. Panjang gelombang cahaya pada setiap warna berbeda-beda. Warna terdiri dari berbagai model, seperti HSV, RGB, YUV, YCbCr, CMYK, CIELAB, dan lainnya. Ruang warna YUV, YCbCr, CMYK dan CIELAB biasanya masing-masing diterapkan dalam penggunaan televisi analog, video digital, percetakan dan industri otomotif ataupun tekstil [11]. Dalam penelitian ini, ruang warna yang digunakan adalah model warna HSV. Model warna HSV adalah salah satu model warna yang dekat dan representasi penglihatan manusia dalam memandang suatu warna. Artinya, model warna HSV dapat memberikan keleluasaan dalam pemahaman terhadap perubahan warna.

Dibandingkan model warna RGB, model warna HSV dapat mendeteksi dan mengurangi intensitas cahaya dari luar, serta tidak berpengaruh terhadap perangkat [12]. Sementara itu, ruang warna RGB adalah model warna yang dapat bervariasi dan bergantung pada perangkat yang digunakan. Sedangkan ruang warna CMYK kurang cocok apabila diterapkan untuk tampilan digital karena memiliki rentang warna yang lebih sempit (tidak dapat merepresentasikan semua warna yang terlihat oleh mata manusia). Kemudian, ruang warna YUV, CIELAB, dan YCbCr adalah model warna yang memiliki kompleksitas tinggi dan kurang praktis untuk pengolahan citra pada sumber daya rendah [13]. Pada konteks pengolahan citra dalam mikrokontroler, model warna HSV dapat memberikan hasil yang lebih stabil dan konsisten meskipun terdapat perubahan tingkat pencahayaan lingkungan sekitarnya. Gambar 2.1 berikut adalah representasi ruang warna HSV dalam lingkaran warna.



Gambar 2.1 Ruang Warna HSV [14]

HSV terdiri dari 3 warna utama, yaitu *hue*, *saturation* dan *Value*. *Hue* umumnya menunjukkan warna yang sebenarnya. *Saturation* menunjukkan tingkat kemurnian dari warna-warna tersebut. Kemudian *value* atau intensitas yang berarti seberapa besar kecerahan dari warna tersebut [12]. HSV sering juga disebut sebagai model

warna HSB (*Hue, Saturation, Brightness*). Adapun skala *hue* berkisar antara 0-360 derajat. Kemudian untuk *saturation* dan *brightness* memiliki skala 0-100%. Hal ini dapat dilihat dari gambar 2.1 berikut.



Gambar 2.2 Skala HSV [12]

2.4 Segmentasi Warna HSV

Segmentasi citra merupakan suatu proses membagi citra ke dalam beberapa daerah dengan setiap objek. Apabila dalam suatu citra hanya mengandung satu objek, maka objek akan dipisahkan dari latar belakangnya [15]. Segmentasi warna sendiri merupakan salah satu bagian dari proses pengolahan citra. Segmentasi warna dapat dilakukan melalui pendekatan dengan menganalisis nilai warna tiap-tiap piksel pada citra. Salah satu model segmentasi warna adalah dengan model deteksi HSV. Segmentasi dengan deteksi HSV dapat dilakukan dengan menganalisis nilai warna setiap piksel pada setiap dimensi warna HSV [16]. Metode segmentasi warna HSV merupakan salah satu metode yang cukup baik untuk digunakan dalam pendeteksian objek buah dan kulit. Metode ini dapat menangkap informasi warna yang lebih spesifik pada ruang warna HSV.

Informasi warna ini lebih relevan dengan perubahan warna buah yang menandakan tingkat kematangan [6]. Dalam metode segmentasi warna HSV, biasanya melibatkan adanya proses *thresholding* atau konversi citra berwarna ke citra biner. Konversi ini dilakukan dengan mengelompokkan derajat keabuan

setiap piksel ke dalam kategori hitam dan putih. Untuk mendapatkan hasil segmentasi warna HSV, jenis model warna yang perlu dikonversikan adalah model warna RGB. Hal ini dikarenakan warna yang tertangkap oleh kamera umumnya adalah warna R, G dan B. Kemudian, untuk mengurangi efek iluminasi pada gambar atau citra tersebut, maka diperlukan konversi ke dalam *colour space* yang lain seperti HSV. Berikut adalah beberapa persamaan yang digunakan untuk mengonversi model warna RGB ke HSV [17] :

1. Rumus mencari nilai R, G dan B :

$$r = \frac{R}{(R+G+B)}, g = \frac{G}{(R+G+B)}, b = \frac{B}{(R+G+B)} \quad (2.1)$$

2. Rumus mencari nilai *Value* (V) :

$$V = \max(r, g, b) \quad (2.2)$$

3. Rumus mencari nilai *Saturation* (S) :

$$S = \left\{ 1 - \frac{\min(r,g,b)}{v}, V > 0 \right\}$$

0, Jika $V = 0$ (2.3)

4. Rumus mencari nilai *Hue* (H) :

$$H = \begin{cases} 60^\circ \times \frac{(g-b)}{s \times v}, & \text{Jika } V = r \\ 60^\circ \times \left[2 + \frac{b-r}{s \times v} \right], & \text{Jika } V = g \\ 60^\circ \times \left[4 + \frac{b-r}{s \times v} \right], & \text{Jika } V = b \\ 0, & \text{Jika } S = 0 \end{cases}$$

$$H = H + 360 \text{ Jika } H < 0 \quad (2.4)$$

Di mana :

R = Nilai kanal warna merah

G = Nilai kanal warna hijau

B = Nilai kanal warna biru

r,g,b = Proporsi komponen warna R,G,B

H = Nilai *Hue*

S = Nilai *Saturation*

V = Nilai *Value*

Min = Nilai minimum normalisasi RGB

Max = Nilai maksimum normalisasi RGB

Adapun tahapan untuk melakukan segmentasi warna ruang HSV secara umum adalah :

1. Mengambil sampel citra yang akan diolah.
2. Mengonversi citra RGB ke ruang warna HSV.
3. Menentukan rentang warna menggunakan *thresholding* untuk komponen HSV (batas atas dan batas bawah model warna HSV)
4. Membuat *masking* warna untuk setiap rentang warna yang ditentukan.

2.5 Thresholding

Thresholding atau sering disebut sebagai metode batas ambang merupakan salah satu metode yang digunakan dalam proses segmentasi citra. *Thresholding* diawali dengan pengubahan citra warna ke warna abu (*grayscale*). Metode ini digunakan untuk membedakan antara *background* objek dengan objek yang digunakan. Pemisahan ini dilakukan dengan membedakan tingkat kecerahan citra melalui nilai intensitas 0 (hitam sempurna) dan 1 (putih sempurna). Oleh sebab itu, hasil dari proses *thresholding* adalah berupa citra biner dengan piksel 0 atau 1. Dalam penelitian ini, *threshold* yang dicari adalah pada kanal hue. Sementara untuk *threshold* s dan v menerapkan pendekatan sederhana yang sama dan sering digunakan, yakni 100 dan 255. Hal ini dikarenakan nilai v dalam HSV berkisar antara 0-255. Apabila menggunakan 0 sebagai nilai terendah, maka hasil citra tersebut sangat kontras (sangat gelap). Nilai 100 sering digunakan sebagai titik tengah untuk memisahkan objek dengan tingkat kecerahan yang sedang [6]. Adapun proses *thresholding* [18] secara umum terlihat pada persamaan di bawah ini.

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) \geq T \\ 0 & \text{if } f(x, y) < T \end{cases} \quad (2.5)$$

Dengan :

$g(x,y)$ = citra biner

$f(x,y)$ = citra aras keabuan

T = nilai ambang

2.5.1 Histogram

Histogram *threshold* merupakan salah satu teknik segmentasi citra yang paling sederhana dan sering digunakan dalam pengolahan citra digital. Teknik ini bertujuan untuk membagi citra menjadi dua atau lebih daerah berdasarkan nilai intensitas piksel dan memvisualisasikannya. Histogram berbentuk diagram dengan garis-garis yang menunjukkan frekuensi nilai. Prinsip dasar histogram *threshold* adalah dengan menentukan nilai ambang batas yang memisahkan objek dari latar belakang. Pencarian histogram ini dilakukan dengan menghitung berapa banyak piksel dalam gambar yang memiliki nilai *hue* tepat sebesar h . Dalam penelitian ini, histogram digunakan untuk mencari nilai ambang atas dan bawah kanal warna *hue* dari objek buah sebelum diuji kebenarannya menggunakan *combined threshold*. Penerapan histogram menggunakan *library* OpenCV pada bahasa pemrograman Python. Namun, terdapat persamaan yang melandasi pencarian nilai *threshold* pada ruang warna *hue* menggunakan sebaran frekuensi, yaitu [19] :

$$\text{frekuensi}(b) = \sum_{i=1}^N \delta(b, h(i)) \quad (2.6)$$

Dengan :

- b = bin tertentu pada histogram yang merepresentasikan nilai *hue* dalam suatu rentang
- $h(i)$ = nilai *hue* dari piksel ke- i
- N = jumlah total piksel pada gambar
- $\delta(b, h(i))$ = fungsi yang menghitung 1 jika nilai *hue* $h(i)$ berada di bin b , dan 0 jika tidak

Hasil perhitungan frekuensi bin tersebut membentuk histogram agar menampilkan sebaran frekuensi nilai *hue* dari piksel dalam gambar. Frekuensi ini menunjukkan berapa banyak piksel yang memiliki nilai *hue* tertentu. Setelah frekuensi *hue* berhasil ditentukan, hasil yang diharapkan dapat berupa :

$$\text{Threshold } (T) = \begin{cases} T1 \leq \text{hue} \leq T2, \\ T3 \leq \text{hue} \leq T4, \end{cases} \quad (2.7)$$

Dengan :

T = Nilai ambang (*threshold*), dalam hal ini untuk kanal warna *hue*

- T1 = Ambang bawah pertama
 T2 = Ambang atas pertama
 T3 = Ambang bawah kedua
 T4 = Ambang atas kedua
hue = Sebaran nilai distribusi kanal warna *hue* yang berada dalam rentang T

2.5.2 Adaptive Threshold

Adaptive threshold merupakan metode pada segmentasi citra yang digunakan untuk membagi citra menjadi beberapa region/daerah. Metode ini dapat menyesuaikan nilai ambang berdasarkan variasi intensitas cahayanya. Sehingga, metode ini cocok digunakan untuk citra yang memiliki intensitas cahaya yang tidak merata. Algoritma metode ini terbagi menjadi 4, yaitu metode Niblack (1986), Sauvola (2000), Bernsen (1986) dan Gaussian (awal abad ke-19). Adapun persamaan berdasarkan ketiga metode tersebut [20] :

1. Penerapan metode Niblack :

$$T = m + k\sigma \quad (2.8)$$

2. Penerapan Metode Sauvola untuk menghilangkan *noise* pada standar deviasi:

$$T = m[1 + k \left(\frac{\sigma}{R} - 1 \right)] \quad (2.9)$$

3. Penerapan metode Bernsen untuk menghitung nilai T berdasarkan nilai kontras :

$$T(i, j) = 0,5\{max_w[I(i + m, j + n)]min_w[I(i + m, j + n)]\} \quad (2.10)$$

dan,

$$C(i, j) = i_{max}(i, j) - i_{min}(i, j) \geq 15 \quad (2.11)$$

4. Penerapan metode Gaussian untuk menghitung nilai T untuk mengurangi *noise* pada gambar dengan intensitas yang halus :

$$T(x, y) = \mu(x, y) - C \quad (2.12)$$

Di mana :

- T = *Threshold* (Nilai Ambang)
 m = nilai rata-rata
 σ = intensitas citra

k (metode Niblack)	= konstanta (antara 0 sampai 1)
R	= jarak dinamis standar deviasi (128)
k (metode Sauvola)	= konstanta (biasanya 0.5)
$I_{\min}(i,j)$	= nilai rata-rata minimum
$I_{\max}(i,j)$	= nilai maksimum gray
$\mu(x,y)$	= nilai rata-rata Gaussian dari piksel sekitar blok x dan y
C	= konstanta yang dikurangi dari nilai rata-rata untuk menyesuaikan <i>threshold</i> .

Dalam penelitian ini, digunakan *adaptive threshold* dengan menerapkan metode Gaussian. Hal ini dilakukan untuk menyesuaikan implementasi Gaussian filter dan agar mendapatkan hasil *threshold* dengan proses yang sederhana dan cepat. Pemilihan metode Gaussian didasari dengan kinerja dan kecepataannya, *noise handling* yang cukup baik, serta stabilitasnya jika diterapkan dalam gambar dengan kontras yang rendah (pencahayaan tidak merata). Lalu untuk nilai C yang digunakan adalah berkisar pada C=0 hingga C=20.

2.5.3 Otsu Threshold

Otsu thresholding dikenalkan oleh Nobuyuki Otsu pada tahun 1979 untuk mengelompokkan citra biner berdasarkan histogramnya. Tujuan dari adanya metode ini adalah untuk membagi histogram citra keabuan dalam dua daerah berbeda [18]. Metode ini dapat digunakan untuk mencari nilai ambang secara otomatis berdasarkan distribusi intensitas piksel dalam citra. Hal ini memungkinkan penggunaannya pada citra yang memiliki distribusi intensitas piksel yang jelas antara objek dan latar belakang. Cara mencari nilai *threshold* menggunakan metode Otsu adalah dengan meminimalkan varians antara pixel hitam dan putih. Nilai ambang berkisar antara 1 sampai dengan 255. Adapun beberapa persamaan yang mendukung penerapan metode otsu sebagai berikut [21]:

1. Mencari nilai probabilitas piksel pada level 1 :

$$p_i = n_i/N \quad (2.13)$$

2. Mencari nilai *Zeroth Cumulative Moment*, *First Cumulative Moment*, dan total nilai Mean. ZCM merupakan jumlah probabilitas piksel pada level ke-T dan level-level sebelumnya. Sementara itu, FCM dapat diartikan dengan jumlah dari perkalian nilai level piksel dengan probabilitasnya pada level ke-T dan level-level sebelumnya. Adapun persamaannya adalah :

$$\omega(T) = \sum_{i=1}^T p_i \quad (2.14)$$

$$\mu(T) = \sum_{i=1}^T i \cdot p_i \quad (2.15)$$

$$\mu_T = \sum_{i=1}^L i \cdot p_i \quad (2.16)$$

3. Menentukan nilai ambang T :

$$\sigma_B^2(T^*) = \max_{1 \leq T < L} \sigma_B^2(T) \quad (2.17)$$

dengan,

$$\sigma_B^2(T) = \frac{[\mu_T \omega(T) - \mu(T)]^2}{\omega(T)[1 - \omega(T)]} \quad (2.18)$$

Di mana :

n_i = jumlah piksel pada level ke-i

N = total jumlah piksel pada citra

T = nilai ambang yang dicari, antara 1 sampai L (dalam skala keabuan 8-bit,

$L = 255$)

p_i = probabilitas setiap piksel pada level ke-i

$\omega(T)$ = *Zeroth Cumulative Moment*

$\mu(T)$ = *First Cumulative Moment*

μ_T = total nilai mean

$\sigma_B^2(T)$ = fungsi varians objek dan latar belakang

$\sigma_B^2(T^*)$ = nilai maksimum dari fungsi varians objek dan latar belakang

2.5.4 Gaussian Filter

Filter Gaussian merupakan salah satu filter dalam segmentasi citra yang dapat digunakan untuk menghilangkan *noise* atau gangguan yang bersifat sebaran normal. Filter Gaussian yang sering juga disebut dengan *Gaussian Blur* akan

menempatkan warna transisi signifikan dalam suatu citra. Filter ini dikenalkan oleh seseorang matematikawan asal Jerman yang bernama Karl Friedrich Gauss [22]. Filter Gaussian termasuk dalam kelas *low-pass filters* yang didasarkan pada fungsi distribusi gaussian. Prinsip kerjanya adalah dengan mengalikan matriks kernel dengan matriks gambar asli. Berikut merupakan persamaan dalam penggunaan *gaussian filter* [23] :

$$f(x) = e^{\frac{-x^2}{2\sigma^2}} \quad (2.19)$$

Selanjutnya, terdapat persamaan fungsi gauss pada dimensi dua :

$$f(x, y) = e^{\frac{-x^2+y^2}{2\sigma^2}} \quad (2.20)$$

Dengan,

$f(x)$ = filter gaussian pada dimensi satu

e = epselon

x = jarak dari titik asal pada sumbu horizontal

σ = standar deviasi

$f(x,y)$ =filter gaussian dimensi dua

y = jarak dari titik asal pada sumbu vertikal

Dalam penelitian ini, filter gaussian yang digunakan adalah gaussian 2D. Dikarenakan citra yang diproses merupakan citra 2 dimensi yang terdiri dari piksel yang tersebar dalam dua arah (x dan y).

2.6 OpenCV

OpenCV merupakan suatu *library open-source* yang sering digunakan dalam *image processing* dan umumnya diterapkan pada pemrograman Python. OpenCV dikembangkan oleh Intel Corporation dengan berfokus pada fungsi-fungsi *computer vision* dan API untuk pemrosesan citra *low* hingga *high level* [24]. OpenCV (*Open Source Computer Vision Library*) terdiri dari sekitar 2500 algoritma yang termasuk dalam algoritma *computer vision* maupun *machine learning*. Algoritma ini dapat digunakan untuk mendeteksi dan juga mengenali wajah, mengidentifikasikan objek, mengklasifikasikan dan melacak gerakan objek dan kamera, mengekstrak model 3D, bahkan pemrosesan citra lainnya [25].

Meskipun sering digunakan dalam bahasa pemrograman Python, sebenarnya OpenCV juga mendukung antarmuka C++, Java, dan Matlab. OpenCV juga dapat digunakan pada sistem operasi berbasis Windows, Linux, Android, serta Mac OS. Dalam penelitian ini, OpenCV digunakan untuk membangun program perhitungan dalam pencarian nilai *threshold*.

2.7 Metode *Prototype*

Prototype merupakan bentuk dasar atau sebuah model awal dari sistem atau bagian-bagian yang dirancang. *Prototype* digunakan untuk pengembangan suatu sistem agar dapat ditingkatkan terus menerus menjadi sebuah sistem yang utuh. Metode *prototype* selain dapat digunakan dalam pengembangan sistem perangkat lunak, juga sering digunakan dalam mengembangkan sistem yang berbasis pada perangkat keras. Dengan menggunakan metode ini, pengembang dapat lebih menghemat waktu dan dapat meminimalisir terjadinya kesalahan [26]. Hal ini karena, dalam penggunaan metode *prototype*, pengembang dapat melakukan iterasi untuk memperbaiki kesalahan yang ada sebelum sistem disebarluaskan. Adapun tahapan-tahapan dalam pengembangan sistem menggunakan metode *prototype* dijelaskan di bawah ini [27].

1. Identifikasi Kebutuhan

Identifikasi kebutuhan atau *requirements* adalah tahapan yang dilakukan untuk mengetahui kebutuhan apa saja yang diperlukan dalam proyek. Hal ini sangat membantu dalam pengembangan sistem termasuk fitur-fitur yang dibutuhkan.

2. Desain Prototype

Dalam tahap ini, dilakukan desain awal sistem yang dibuat. Tahapan ini dilakukan untuk mempermudah proses pengerjaan dan mendapat gambaran terkait sistem yang dirancang.

3. Implementasi

Pada tahap ini dilakukan implementasi pada sistem. Implementasi ini umumnya berupa membangun *prototype* yang telah didesain sebelumnya menggunakan alat dan bahan yang sudah disediakan.

4. Pengujian

Selanjutnya, pada tahap pengujian, dilakukan evaluasi untuk mengetahui apakah terdapat kesalahan atau ketidaksesuaian pada sistem yang dikembangkan. Hasil dari pengujian ini digunakan untuk memastikan bahwa sistem yang dibangun dapat memenuhi kebutuhan yang diharapkan. Dalam konteks sistem berbasis perangkat keras dan perangkat lunak seperti IoT atau *embedded system*, pengujian sering kali dilakukan untuk mengevaluasi stabilitas sistem dengan mengukur berbagai parameter, salah satunya adalah waktu pemrosesan. Pendekatan dalam pengujian waktu sering melibatkan *customized metrics*, yang sering kali berbasis pada konsep koefisien variasi dalam statistik. Pengujian ini bertujuan untuk mengukur seberapa besar fluktuasi waktu yang terjadi dalam sistem tersebut. Salah satu cara untuk menghitung fluktuasi waktu adalah dengan menggunakan standar deviasi. Semakin besar nilai standar deviasi, semakin besar pula sebaran dan variabilitas waktu, yang menunjukkan bahwa sistem memiliki konsistensi yang lebih rendah. Sebaliknya, standar deviasi yang kecil menunjukkan bahwa waktu pemrosesan sistem lebih konsisten dan stabil. Berikut adalah rumus standar deviasi untuk melihat variabilitas waktu sistem.

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \mu)^2}{n}} \quad (2.21)$$

Dengan :

σ : Standar deviasi waktu

μ : Rata-rata waktu

x_i : Waktu output untuk setiap objek

n : Jumlah objek pengujian total

5. Perbaikan

Apabila ditemukan kesalahan dalam proses pengujian, maka dilakukan iterasi untuk memperbaiki sistem tersebut. Perbaikan ini digunakan untuk meningkatkan kinerja sistem dan meminimalisir kesalahan selanjutnya.

2.8 Protokol HTTP dan WebSocket

HTTP (*HyperText Transfer Protocol*) merupakan protokol yang digunakan untuk mentransfer informasi melalui internet, terutama antara klien (seperti *browser*) dan server. HTTP bekerja dengan model komunikasi *request-response*, di mana klien mengirimkan permintaan ke server, dan server memberikan respons dengan data yang diminta. Setiap transaksi HTTP dianggap independen dan tidak terkait dengan transaksi sebelumnya, menjadikannya protokol *stateless*. Ini berarti setiap permintaan harus menyertakan semua informasi yang diperlukan untuk diproses, seperti *header* HTTP yang berisi metadata tentang permintaan tersebut [28].

Sementara itu, *WebSocket* adalah protokol komunikasi berbasis web yang memungkinkan pertukaran data secara dua arah (*full-duplex*) antara klien dan server melalui satu koneksi TCP. Protokol ini dirancang untuk mengatasi keterbatasan yang ada pada protokol HTTP konvensional, yang umumnya hanya mendukung komunikasi satu arah (*half-duplex*) dengan pengulangan *header* pada setiap permintaan (*request*) dan respons. *WebSocket* memungkinkan klien dan server untuk mempertahankan koneksi terbuka sehingga data dapat dikirim kapan saja, tanpa perlu permintaan berulang dari klien. Hal ini membuat *WebSocket* menjadi pilihan yang lebih efisien dibandingkan HTTP untuk aplikasi yang memerlukan pembaruan data secara *real-time*, seperti permainan daring, chat, atau aplikasi monitoring sensor [28].

Salah satu kelemahan utama HTTP adalah penggunaan *polling*, di mana klien secara terus-menerus mengirimkan permintaan (*request*) ke server untuk memeriksa apakah ada pembaruan data baru. Teknik ini seringkali mengakibatkan peningkatan lalu lintas jaringan yang tidak perlu karena server harus menanggapi setiap permintaan, meskipun tidak ada data baru yang dikirimkan. Alternatif lainnya, *HTTP long polling*, sedikit lebih efisien karena server menahan respons sampai ada data baru yang tersedia, tetapi masih menimbulkan *overhead* komunikasi dan latensi yang signifikan. *WebSocket* mengatasi masalah ini dengan menyediakan kanal komunikasi *full-duplex* melalui satu soket TCP, memungkinkan data dikirim dan diterima secara simultan. Dengan *WebSocket*,

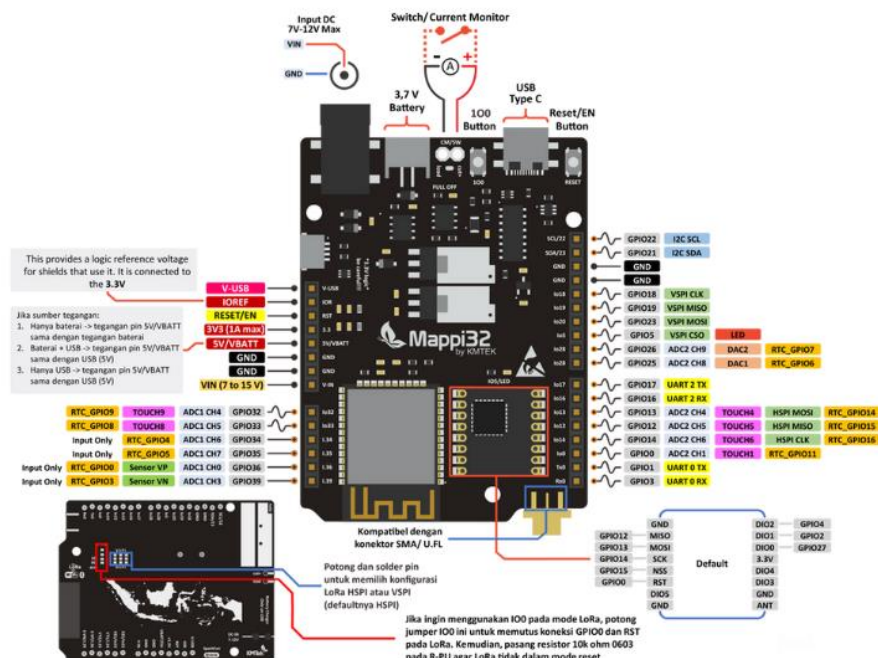
overhead komunikasi berkurang secara signifikan karena tidak ada pengulangan *header* di setiap pesan.

Protokol ini memulai dengan pembukaan koneksi (*handshake*) menggunakan HTTP, dan setelah terhubung, koneksi akan ditingkatkan (*upgrade*) menjadi *WebSocket*. Selanjutnya, komunikasi antara klien dan server dapat terjadi dengan latensi yang lebih rendah dan kecepatan yang lebih tinggi. Dalam implementasinya, *WebSocket* dapat digunakan pada berbagai aplikasi *real-time*, termasuk sistem kontrol jarak jauh, game multiplayer, atau aplikasi sensor yang memerlukan pembaruan konstan. Salah satu keunggulan utama *WebSocket* adalah kemampuannya untuk menjaga stabilitas koneksi dalam waktu yang lama, membuatnya cocok untuk aplikasi yang memerlukan sinkronisasi data terus-menerus antara klien dan server. Dalam penelitian ini, protokol HTTP dan *WebSocket* digunakan bersamaan, baik untuk menampilkan data hasil *counting* secara *real-time*, sekaligus menampilkan *frame-by-frame* citra secara berkelanjutan pada Model II, serta mengontrol gerakan servo.

2.9 Mappi32

Mikrokontroler adalah salah satu bagian elektronik yang memiliki *Integrated Circuit* (IC) dan dapat digunakan sebagai komputer berukuran kecil. Mikrokontroler adalah gabungan dari kata *Micro* yang berarti sangat kecil, dan *Controller* yang artinya adalah pengendali. Mikrokontroler juga dapat diartikan sebagai suatu chip pada komputer yang berfungsi sebagai pengontrol perangkat elektronik. Mikrokontroler terdiri dari RAM, ROM, I/O khusus, *timers* dan pendukung seperti *Analog Digital Converter* [29]. Pada mikrokontroler, umum disebut penyimpanan seperti *flash memory* dan SRAM. *Flash memory* merupakan penyimpanan yang digunakan untuk menyimpan data dalam jangka panjang. Contoh penggunaan *flash memory* adalah untuk menyimpan data *firmware* atau program pembangun sistem. Ini berfungsi agar program tidak perlu diunggah secara terus menerus jika program yang digunakan sama. Sedangkan SRAM (*Static Random Access Memory*) adalah penyimpanan data sementara yang digunakan saat program berjalan. Data yang tidak akan disimpan/berjalan dalam

waktu *real-time* akan memanfaatkan alokasi SRAM. Dalam penelitian ini, *board* mikrokontroler yang digunakan adalah *board* Mappi32. Adapun *pinout* Mappi32 dapat dilihat pada Gambar 2.3 berikut.



Gambar 2.3 Spesifikasi Board Mappi32 [5]

Mappi32 sendiri merupakan *development kit* yang diproduksi oleh KMTek Indonesia. Mappi32 telah berhasil diimplementasikan pada berbagai proyek, seperti kendali dan navigasi, *smart parking*, *smart trash bin*, *water dispenser*, *smart pet care*, hingga sistem dengan tujuan monitoring jarak jauh (menerapkan node pada LoRa). Mappi32 mengambil konsep pengembangan dari mikrokontroler ESP32. Oleh sebab itu, SRAM pada Mappi32 juga mengikuti standar SRAM pada *board* ESP32. ESP32 merupakan salah satu jenis mikrokontroler yang mengembangkan dari mikrokontroler ESP8266. Pada ESP32, tersedia modul WiFi dalam chip yang mendukung untuk pembuatan sistem *embedded system* yang terintegrasi internet (*Internet of Things*) [30]. Mappi32 mendukung penggunaan teknologi jaringan LoRa dan dapat digunakan untuk berbagai sensor serta komponen elektronik seperti sensor ultrasonik, buzzer, AHT-10, LCD, dan lain sebagainya. Mappi32 terhubung dengan 3 konektivitas, yaitu WiFi, Bluetooth dan modul LoRa 920-923 MHz, sehingga

memungkinkannya untuk berkomunikasi dengan perangkat jarak jauh.

Selain dapat diterapkan pada implementasi jarak jauh, Mappi32 juga mendukung berbagai platform pemrograman seperti Arduino IDE, ESP IDF, hingga MicroPython. Hal ini memberikan fleksibilitas pada Mappi32 untuk lebih mudah digunakan. Meskipun harganya cukup terjangkau, tetapi kualitas Mappi32 cukup baik. Mappi32 mendukung CPU sebesar 240MHz. Ukuran CPU ini masih cukup besar dan layak digunakan pada berbagai implementasi seperti pengolahan data sensor, koneksi jaringan, pemrosesan gambar dan video dengan resolusi rendah hingga menengah, *edge detection*, hingga model *machine learning* sederhana. Selain itu, Mappi32 memiliki prosesor berbasis ESP32-WROOM-32E yang mendukung kecepatan *clock* lebih tinggi dan fitur-fitur yang dapat bersaing dengan NodeMCU hingga Arduino. Penggunaan Mappi32 juga dinilai lebih sederhana dan ringan dibandingkan mikrokontroler ataupun mikrokomputer seperti Raspberry Pi [31]. Untuk lebih jelasnya, berikut adalah spesifikasi pada Mappi32 [5] yang disajikan dalam Tabel 2.1.

Tabel 2.1 Spesifikasi Mappi32

<i>Processor</i>	ESP WROOM – 32E core 2
<i>Architecture</i>	32 bit
CPU Frekuensi	240 MHz
<i>Flash Memory</i>	16 Mb <i>allocate</i> SRAM 520Kb
<i>Connectivity on Board</i>	WiFi, Bluetooth, LoRa
<i>Port Input</i>	USB <i>Type C</i> , <i>Power Jack</i> DC, dan JST PH 2.0 mm
<i>Input Voltage</i> (DC)	1-15V
<i>Operating</i>	5V

2.10 Modul Kamera OV2640



Gambar 2.4 Modul OV2640 (Sumber: Dokumen Pribadi)

Modul kamera OV2640 adalah salah satu modul kamera dengan daya rendah atau *low operation voltage*. Modul OV2640 memiliki resolusi kamera maksimum 1600x1200px dan minimum 160x120px(normal) serta 40x30px (diskalakan) [32]. OV2640 merupakan sensor 2 MP dengan ukuran ¼ inci dan merupakan chip kamera terintegrasi. Modul kamera ini memanfaatkan arsitektur OmniPixel2 baru yang memiliki spesifikasi warna yang lebih hidup, struktur lensa tanpa celah dan pemrosesan gambar berkualitas tinggi. Modul ini banyak digunakan dalam berbagai aplikasi, seperti aplikasi pada kamera ponsel, mainan anak-anak, kamera digital, dll [33]. Namun, modul kamera OV2640 umumnya tidak dapat berdiri sendiri, melainkan memerlukan board terintegrasi yang kompatibel untuk penggunaannya. Dalam penelitian ini, penggunaan modul kamera OV2640 dibantu dengan board ESP32S berbasis model AI Thinker. Integrasi modul kamera dan ESP32S ini umum disebut sebagai ESP32-Cam. Adapun *pinout* beserta spesifikasi ESP32-Cam [34] dapat dilihat pada Gambar 2.4 dan Tabel 2.2 di bawah ini.



Gambar 2.5 ESP32-Cam [34]

Tabel 2.2 Spesifikasi ESP32-Cam

<i>Connectivity</i>	WiFi 802.11b/g/n, Bluetooth 4.2 BLE
<i>Pin</i>	UART, SPI, I2C, PWM, 9 pin GPIO
Frekuensi dan Daya Komputasi	+160 MHz, +600 DMIPS
<i>Memory</i>	520 Kb SRAM, 4 Mb PSRAM, Slot Kartu SD

Tabel 2.2 (lanjutan)

Fitu-fitur lain	<i>Sleep mode</i> , upgrade firmware OTA, dan <i>flash internal LED</i>
Spesifikasi Kamera	Kamera OV2640 yang memiliki sensor 2 MP; ukuran UXGA 1600x1200px; Output YUV422, YUV420, RGB565, RGB555 dan kompresi data 8-bit; transfer gambar 15-60 FPS

2.11 Motor Servo

Motor Servo adalah perangkat keras berupa motor listrik yang bekerja menggunakan sistem *closed loop*. Motor servo terdiri dari 2 komponen, yaitu komponen pertama adalah motor untuk penggerak roda gigi agar potensiometer dan poros *output* dapat berputar secara bersamaan. Komponen kedua yakni potensiometer/*encoder* untuk sensor yang memberikan sinyal ke sistem kontrol untuk menentukan posisi dari objeknya [8]. Dalam penelitian ini, tipe motor servo yang digunakan adalah Micro Servo SG90.



Gambar 2.6 Motor Servo SG90 [35]

Motor servo SG90 memiliki tegangan +5V, dengan torsi 2,5kg/cm dan kecepatan 0,1s/60 derajat. Motor servo ini berbahan *gear* dari plastik dengan berat motor 9gm dan tingkat rotasi 0-180 derajat. Umumnya, motor servo digunakan sebagai aktuator yang dapat bergerak hingga mencapai sudut tertentu [36]. Berikut adalah Tabel 2.3 yang berisi spesifikasi motor servo tipe Micro Servo SG90 [35].

Tabel 2.3 Spesifikasi Motor Servo SG90

<i>Load Capacity</i>	Antara 1.2-1.6 Kg
Torsi (4.8V)	1.2 Kg/cm
<i>Operating voltage</i>	4-7.2V
<i>Rotary Speed</i> (4.8V)	0.12s/60 derajat
<i>Rotary Angle</i>	120 derajat
<i>Operating Temperature Range</i>	-30 sampai +60 derajat celsius
Ukuran	22x11.5x27 mm
Berat	9g atau 10.6 g

2.12 Modul LM2596



Gambar 2.7 Modul LM2596 [37]

Modul *stepdown* LM2596 *adjustable* DC-DC adalah salah satu perangkat elektronik yang digunakan untuk menurunkan tegangan DC maksimal hingga 3A dalam range DC 3.2V-46V. Untuk menurunkan tegangan menggunakan modul *stepdown* ini adalah melalui multiturun potensiometer. Meskipun besar tegangan input naik turun, output dari modul LM2596 akan tetap stabil [38]. Modul *stepdown* ini memiliki berbagai fitur, di antaranya tegangan output yang dapat disesuaikan, osilator internal frekuensi tetap 150 kHz, dapat mematikan TTL, dan lain sebagainya. Dalam penggunaan modul *stepdown*, umumnya tidak perlu dilakukan perhitungan manual tambahan. Hal ini dikarenakan untuk menurunkan tegangan dapat dilakukan dengan potensiometer dan sirkuit internal secara

otomatis [37]. Dalam penelitian ini, modul *stepdown* digunakan untuk menurunkan tegangan input dari baterai eksternal menjadi output 5V agar dapat menjalankan motor DC.

2.13 Motor DC

Motor DC (*Direct Current*) adalah salah satu perangkat elektronik yang bekerja dengan mengubah energi listrik menjadi energi gerak (kinetik). Motor DC memiliki dua buah terminal dan membutuhkan tegangan arus yang searah sesuai namanya. Motor ini menghasilkan beberapa putaran dalam waktu per menit atau sering disebut sebagai RPM (*Revolutions per Minute*). Motor DC juga dapat berputar searah jarum jam ataupun berlawanan sesuai dengan kebutuhan [39]. Seiring perkembangannya, motor DC sering dikombinasikan dengan *gearbox*. *Gearbox* digunakan untuk mengubah kecepatan dan torsi sumber daya yang masuk. Adapun spesifikasi dari Motor DC *Gearbox* secara umum seperti terlihat pada Gambar 2.8 dan tabel 2.4 berikut [39].



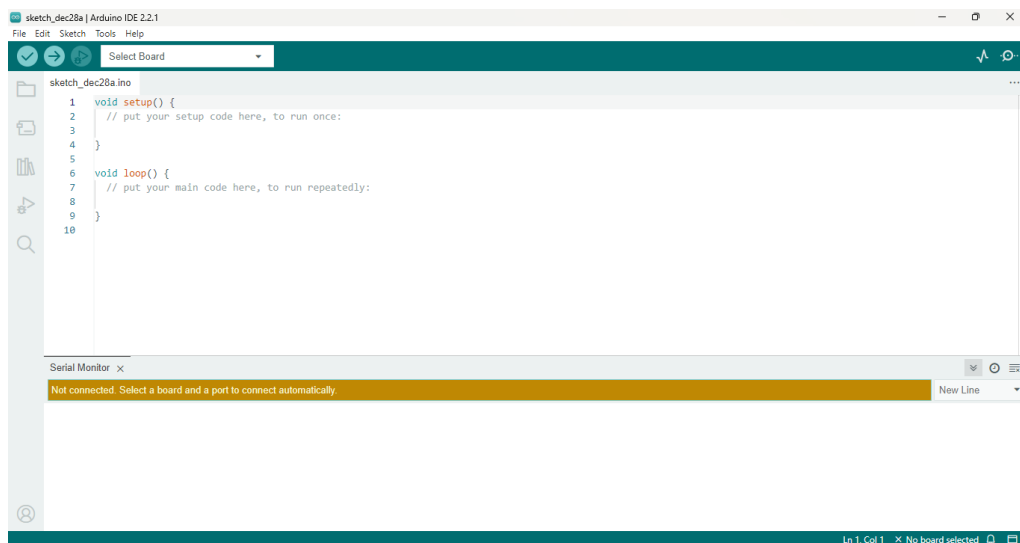
Gambar 2.8 Motor DC *Gearbox* [39]

Tabel 2.4 Spesifikasi Motor DC *Gearbox*

Torsi	1 Kgf.cm pada 3V; 1.5 kgf.cm pada 6V; dan 1.8 kgf.cm pada 7.2V
Kecepatan tanpa beban	130 rpm (3V), 290 rpm (6V), dan 330 rpm (7.2V)
Dimensi	7cm x .7 cm x 2.2 cm
Tegangan tanpa beban	170mA (3V), 240Ma (6V) dan 260 Ma (7.2V)

2.14 Arduino IDE

Arduino IDE (*Integrated Development Environment*) merupakan *tools* berupa *software* yang mencakup editor, *compiler*, dan *uploader sketch* untuk menulis program ke dalam perangkat elektronik seperti arduino. Dalam pengembangannya, arduino IDE dapat digunakan untuk memasukkan program pada perangkat *board* atau mikrokontroler jenis lain seperti esp, dan lain sebagainya. Bahasa pemrograman pada Arduino IDE adalah bahasa C [40].



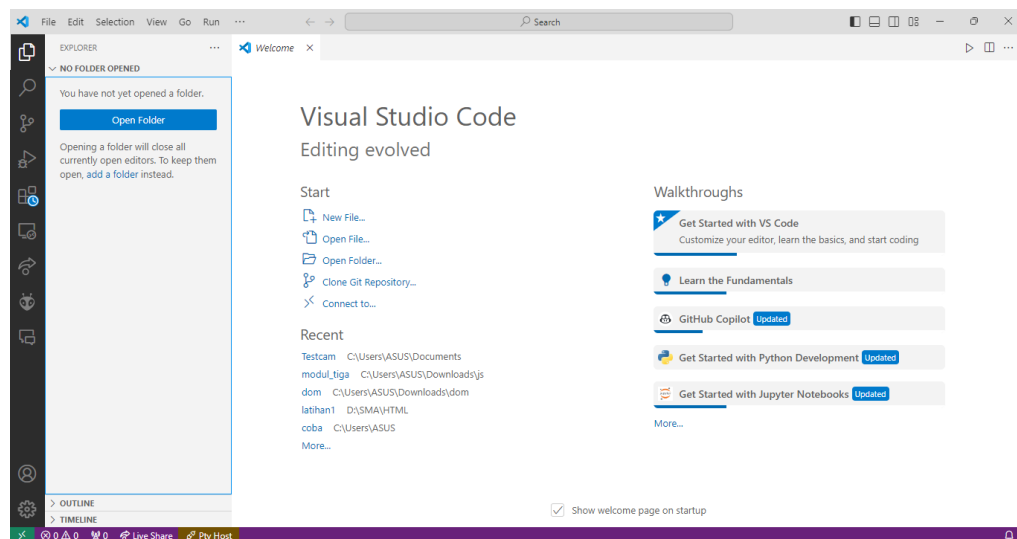
Gambar 2.9 *Dashboard* Arduino IDE (Sumber: Dokumen Pribadi)

Berdasarkan *dashboard* Arduino IDE tersebut, dapat dilihat bahwa Arduino IDE memiliki struktur dasar penulisan program, yaitu *void setup()* dan *void loop()*. *Void setup()* digunakan untuk menyiapkan atau menginisialisasi kebutuhan pada program. Sedangkan *void loop()* digunakan untuk membuka blok program utama yang akan dibuat (isi dari program). Selain itu, Arduino IDE juga mendukung tampilan *output realtime* melalui serial monitor.

2.15 Visual Studio Code

Visual Studio Code atau yang kerap disebut VSCode merupakan sebuah perangkat lunak yang digunakan sebagai teks editor. Teks editor ini dikembangkan oleh Microsoft untuk memudahkan dalam melakukan

pemrograman multi-bahasa. *Software* ini mendukung penggunaan pada berbagai sistem operasi seperti Windows, Linux hingga Mac. Adapun bahasa pemrograman yang didukung oleh Visual Studio Code antara lain seperti JavaScript, Python, bahasa C, Go, Java, dan masih banyak lagi [41]. VSCode juga mendukung penggunaan ekstensi langsung, dan penginstalan ekstensi serta pustaka langsung di dalamnya. Selain itu, pengguna juga dapat langsung menjalankan program secara Live maupun mengontrol menggunakan terminal di bash VSCode.



Gambar 2.10 *Dashboard* Visual Studio Code (Sumber: Dokumen Pribadi)

Dalam penelitian ini, VSCode membantu dalam pemrograman pengolahan warna yang dilakukan. Selain pengolahan citra yang berbasis pada bahasa pemrograman Python, VSCode juga digunakan untuk membuat *dashboard* sederhana menggunakan HTML, CSS dan Javascript.

2.16 Jeruk

Buah jeruk (*Citrus*) merupakan salah satu komoditas buah tropis yang dapat tumbuh dan dibudidayakan di Indonesia. Buah jeruk termasuk komoditas buah tahunan yang memiliki kontribusi besar terhadap pertumbuhan produksi hortikultura di Indonesia. Di Indonesia sendiri, sebenarnya terdapat berbagai jenis buah jeruk. Dalam penelitian ini sendiri, buah jeruk yang digunakan adalah buah jeruk yang pengidentifikasiannya dapat dilihat dari adanya perubahan




warna.







Gambar 2.11 Buah Jeruk (Sumber: Dokumen Pribadi)

Menurut data dari Badan Pusat Statistik dan Direktorat Jenderal Hortikultura tahun 2022 [1], jumlah produksi buah jeruk cenderung meningkat dari tahun 2017, yaitu 2.165.184 ton menjadi 2.401.064 ton pada tahun 2021. Selain itu, luas lahan panen dari buah jeruk pada tahun 2021 juga mencapai sekitar 62.830 Ha. Sedangkan pada tahun 2021-2023, jumlah produksi buah jeruk juga meningkat dari 2.401.064 ton menjadi 2.831.099 ton dengan produktivitas tanaman 21.936.727 rumpun/pohon [42]. Dalam penelitian ini, buah jeruk dijadikan salah satu objek penelitian untuk menguji tingkat kematangan berdasarkan perubahan warna. Semakin matang buah jeruk, maka rasanya juga menjadi lebih manis. Adapun beberapa kategori tingkat kematangan buah jeruk berdasarkan perubahan warna berdasarkan hasil observasi dan wawancara oleh pengelola PKK Agropark Sabah Balau terlihat pada Tabel 2.5 berikut.

Tabel 2.5 Perubahan Warna Buah Jeruk

Gambar	Keterangan
	Hijau tua, buah jeruk masih mentah dan belum bisa dipanen. Rasa buah masih sangat asam dan bergetah.
	Hijau lebih terang, buah jeruk masih belum matang dan belum siap panen. Rasa masih asam.
	Hijau kekuningan, buah belum matang sepenuhnya. Belum bisa dipanen, tetapi sudah bisa dimakan dengan rasa masih asam.

Tabel 2.5 (lanjutan)

Gambar	Keterangan
	Hijau kekuningan lebih terang, buah belum matang sepenuhnya. Belum bisa dipanen tetapi sudah bisa dimakan. Rasa masih cukup asam.
	Kuning muda, buah belum matang sepenuhnya namun sudah bisa dimakan. Rasa asam manis.
	Kuning terang, buah cukup matang dan siap panen. Rasa buah asam manis, cenderung manis.
	Oranye, buah matang dan siap panen. Rasa manis.

Berdasarkan Tabel 2.5 Perubahan Warna Jeruk, dapat dilihat bahwa buah jeruk memiliki beberapa kategori tingkat kematangan. Buah jeruk sudah dapat dipanen pada tingkat kematangan indeks kelima-ketujuh (warna kuning). Hal ini dikarenakan pada tingkat kematangan tersebut, buah jeruk dapat langsung dipasarkan ataupun diolah.

2.17 Stroberi








Gambar 2.12 Buah Stroberi (Sumber: Dokumen Pribadi)

Stroberi (*Fragaria chiopendis L.*) adalah salah satu buah yang juga dapat tumbuh di Indonesia. Di Indonesia sendiri, produksi buah stroberi pada tahun 2021 sebanyak 9.860 ton dengan luas lahan panen yaitu 682 Ha [1]. Kemudian pada tahun 2021 ke 2023 juga terdapat peningkatan dari 682 Ha ke 810 Ha, dengan jumlah produksi 98.596 kuintal menjadi 277.208 kuintal [42]. Meskipun nilai

produktivitas dan luas lahan perkebunan stroberi tidak sebesar jenis buah lainnya, tetapi buah stroberi cukup diminati oleh konsumen serta bernilai jual tinggi. Hal ini dikarenakan selain dapat dikonsumsi secara langsung, buah stroberi juga dapat diolah menjadi berbagai jenis olahan, contohnya selai, dodol, sirup, manisan atau tambahan pada *dessert* seperti kue. Oleh sebab itu, untuk mendapatkan buah stroberi yang berkualitas dapat ditentukan oleh beberapa parameter, salah satunya adalah tingkat kematangan [16]. Buah stroberi juga merupakan salah satu buah yang tingkat kematangannya dapat dilihat dari adanya perubahan warna. Oleh sebab itu, buah stroberi juga dijadikan objek penelitian. Adapun tingkat kematangan buah stroberi berdasarkan perubahan warna berdasarkan hasil observasi dan wawancara oleh pengelola PKK Agropark Sabah Balau dapat dilihat pada Tabel 2.6 berikut.

Tabel 2.6 Perubahan Warna Buah Stroberi

Gambar	Keterangan
	Hijau muda, buah stroberi masih muda dan belum bisa dipanen.
	Hijau lebih terang, buah stroberi masih muda dan belum matang, sehingga belum siap panen.
	Hijau muda terang dengan bercak-bercak merah muda, buah mendekati matang tapi belum siap panen, rasa asam.
	Merah muda, buah hampir matang tapi belum siap panen, rasa buah asam.
	Merah, buah sudah matang dan siap panen. Rasa buah umumnya manis.

2.18 Penelitian Terkait

Dalam penyusunan skripsi ini, terdapat beberapa penelitian terdahulu/terkait yang dapat dijadikan sebagai rujukan dalam penelitian. Adapun rujukan pertama adalah

penelitian yang dilakukan oleh Mutia Maulida dan Nurul Fathanah Mustamin [43] yang bertujuan untuk mengembangkan keramba dan jaring apung oleh masyarakat Kalimantan Selatan menjadi memiliki sistem penyebar alat pakan berbasis IoT. Sistem ini dibuat menggunakan mikrokontroler Mappi32 dengan modul LoRa dan sensor ultrasonik. Hasil dari penelitian ini berupa sistem kontrol pakan berbasis Android dan dapat memberikan pakan ikan secara akurat dan terjadwal. Berdasarkan hasil pengujian, dapat disimpulkan bahwa sistem ini berjalan dengan baik meskipun masih belum menerapkan fitur takaran pakan sesuai jenis ikan.

Selanjutnya adalah penelitian lain yang dilakukan oleh Ali Basrah Pulungan dan Zhafranul Nafis [8] dari Universitas Negeri Padang. Dalam penelitian ini, alat yang dibuat adalah alat untuk mendeteksi benda berdasarkan warna, bentuk dan ukuran. Dalam perancangan alat, digunakan komponen elektronik berupa mikrokontroler arduino uno, *webcam* dan sensor infrared (IR). Penentuan warna, bentuk dan ukuran dilakukan melalui proses pengolahan citra menggunakan bahasa pemrograman Python. Berdasarkan penelitian ini, didapatkan hasil berupa alat yang mampu mendeteksi benda berdasarkan parameter tersebut, serta mampu memisahkan benda-benda tersebut.

Serupa dengan penelitian sebelumnya, pada artikel penelitian yang dilakukan oleh Syahri Muharom, Saiful Asnawi, dan Affan Bachri [44] juga membahas terkait pendeteksi parameter bentuk dan warna menggunakan pengolahan citra. Akan tetapi, dalam penelitian ini hasil yang didapatkan adalah berupa robot (*Assistant Robot*) yang dapat mengikuti target berdasarkan bentuk dan warnanya. Robot pendeteksi ini dirancang menggunakan mikrokontroler dan kamera. Dalam penelitiannya, Syahri dan rekan-rekannya melakukan pengujian pada objek benda berbentuk bola berwarna dan didapatkan keberhasilan alat sebesar 87,6%. Kegagalan ini disebabkan oleh adanya beberapa kesalahan pengujian terhadap parameter jarak yang terlalu jauh atau dekat.

Berbeda dari tiga penelitian sebelumnya, proyek penelitian yang dilakukan oleh Muhammat Andri, dkk [45] membahas perancangan dan pembangunan *prototype* sistem sortir otomatis. Sistem sortir ini digunakan untuk memilah buah kelapa sawit berdasarkan tingkat kematangannya. Sistem berhasil dikembangkan

menggunakan mikrokontroler arduino uno dan sensor warna. Berdasarkan penelitian ini, didapatkan hasil berupa sistem sortir yang dapat memisahkan butir buah kelapa sawit berdasarkan kategori mentah dan matang.

Penelitian kelima yang menjadi salah satu rujukan adalah penelitian oleh Yanri Bili Eliezer Purba, dkk [30]. Hampir sama dengan penelitian sebelumnya, pada projek ini juga membahas terkait pendeteksi kematangan pada objek buah. Akan tetapi, pada perancangan alat deteksi ini, objek yang digunakan adalah buah nanas. Selain menerapkan konsep prototype alat berbasis mikrokontroler ESP32, peneliti juga melakukan perancangan menggunakan metode CNN. Hasil dari penelitian ini berupa alat pendeteksi kematangan buah nanas menjadi kategori mentah, matang dan ranum. Berdasarkan pengujian terhadap klasifikasi warna, didapatkan tingkat akurasi tertinggi sebesar 86% dan terendah 80%. Hal ini dikarenakan dalam beberapa kali pengujian, masih terdapat hasil prediksi yang salah.

Selain beberapa penelitian berskala nasional di atas, terdapat beberapa penelitian berskala internasional yang juga membahas terkait perancangan sistem pendeteksi kematangan maupun mesin sortir otomatis. Salah satu di antaranya adalah penelitian oleh Ganesh Khekara, dkk [46] yang mengembangkan sistem pemisahan dan pengemasan buah lemon berbasis pengolahan citra. Sistem ini dibuat menggunakan Rasperry Pi yang dikombinasikan dengan sensor inframerah. Hasil dari penelitian ini adalah sistem yang dapat mendeteksi objek berdasarkan warna dan melakukan pemisahan secara otomatis.

Selanjutnya adalah penelitian oleh Gregorio Imanuel, dkk [47] yang melakukan perancangan sistem sortir serta penghitungan barang berbentuk lengan robot berdasarkan warna berbasis IoT. Sistem ini dikembangkan menggunakan mikrokontroler NodeMCU ESP8266 dan sensor warna. Selain merancang sistem IoT, peneliti juga melakukan perancangan antarmuka web untuk pemrosesan barang yang akan ditampilkan. Hasil monitoring barang via web akan menampilkan persentase barang berdasarkan warna untuk kategori barang siap kirim dan barang di gudang.

Deepak Devasagayam, dkk [48] juga merancang sistem sortir otomatis berbasis mikrokontroler arduino dan modul kamera OV2640. Dalam penelitiannya, Devasagayam dan rekan-rekannya merancang sistem sortir berbentuk *hopper* untuk digunakan dalam industri pertanian. Tak hanya digunakan untuk melakukan sortir saja, sistem yang dihasilkan juga dapat digunakan untuk pengemasan dan pelabelan melalui pendekatan pengolahan citra. Meskipun tidak dijelaskan secara rinci, sistem ini telah berhasil diujikan untuk 5 jenis buah yang berbeda.

Penelitian lain oleh Ioan Lita, dkk [49]; serta Abdallah, dkk [50] juga membahas hal yang serupa. Pada proyek yang dirancang oleh Lita dan rekannya [49], dihasilkan sistem sortir otomatis berdasarkan warna untuk digunakan dalam sektor pertanian. Sistem ini dibuat menggunakan mikrokontroler arduino uno dan sensor fotodiode. Dalam penelitian ini, pengujian berhasil dilakukan pada objek berupa benda yang memiliki warna berbeda-beda sebagai representasi kematangan buah. Sementara itu, proyek oleh Abdallah dan rekannya [50] dirancang menggunakan mikrokontroler arduino uno dan sensor warna serta diterapkan dalam bentuk *belt conveyor*. Pada sensor ini, didapatkan hasil uji citra dalam bentuk citra RGB yang dikalibrasikan pada beberapa parameter cahaya.

Selanjutnya, untuk mempermudah dalam memahami beberapa sumber literatur di atas, maka dibuatlah sebuah tabel yang berisi ringkasan terkait hal penting yang dapat digunakan sebagai rujukan pada penelitian ini. Adapun Tabel 2.7 tersebut sebagai berikut.

Tabel 2.7 Penelitian Terkait

No	Peneliti/ Tahun	Mikro kontroler	Objek	Metode	Hasil
1.	Mutia Maulida, dkk (2022) [43]	Mappi32	Pakan ikan pada keramba jaring apung	Perancangan sistem <i>hardware</i> dan <i>software</i>	Sistem kontrol pakan berbasis Android dan dapat memberikan pakan ikan secara akurat dan terjadwal
2.	Ali Basrah, dkk (2021) [8]	Arduino Uno	Benda berwarna	Segmentasi warna HSV, <i>thresholding</i> , perancangan	Alat pendeteksi benda berdasarkan ukuran, bentuk dan warna serta mampu

Tabel 2.7 (lanjutan)

No	Peneliti/ Tahun	Mikro kontroler	Objek	Metode	Hasil
				sistem	memisahkan benda-benda tersebut. Bentuk benda yang bisa dikenali hanya persegi
3.	Syahri Muharom, dkk (2021) [44]	Arduino Uno	Benda berwarna	Pengolahan citra dengan metode HSV	Robot pengikut target berdasarkan bentuk dan warna dengan hasil akurasi 86,6% berdasarkan warna, dan 87,6% berdasarkan jarak
4.	Muhammat Andri, dkk (2023) [45]	Arduino Uno	Butiran buah kelapa sawit	Perancangan sistem dengan pengenalan rentang RGB sensor warna	Sistem sortir yang dapat memisahkan butir buah kelapa sawit berdasarkan kategori mentah dan matang
5.	Yanri Bili, dkk (2022) [30]	ESP32	Buah nanas	<i>Convolutional Neural Network</i>	Alat pendeteksi kematangan buah nanas berdasarkan kategori mentah, matang dan ranum dengan akurasi sebesar 86%
6.	Ganesh Khekare, dkk (2020) [46]	Raspberry Pi	Buah Lemon	Segmentasi warna HSV	Pendeteksi kematangan buah lemon berdasarkan warna untuk pemilahan dan pengemasan
7.	Gregorio Immanuel, dkk (2018) [47]	NodeMCU ESP8266	Benda berwarna	Perancangan sistem dengan deteksi warna RGB oleh sensor warna	Lengan robot pemisah barang serta <i>dashboard</i> monitoring barang
8.	Deepak Devasagaya m, dkk (2019) [48]	Arduino Uno	Buah- buahan	Perancangan sistem dan pembacaan sinyal RGB modul kamera	<i>Hooper</i> untuk sortir, pengemasan dan pelabelan buah

Tabel 2.7 (lanjutan)

No	Peneliti/ Tahun	Mikro kontroler	Objek	Metode	Hasil
9.	Ioan Lita, dkk (2019) [49]	Arduino Uno	Benda berwarna	Perancangan sistem dengan deteksi warna RGB oleh sensor warna	Sistem sortir buah- buahan berdasarkan warna
10.	Abdallah, dkk (2023) [50]	Arduino Uno	Benda berwarna	Perancangan sistem dengan pemrosesan citra digital menggunakan teknologi <i>machine vision</i>	<i>Belt conveyor</i> untuk sistem sortir buah- buahan berdasarkan warna

Berdasarkan berbagai penelitian pada Tabel 2.7, dapat diketahui bahwa sebelumnya telah dilakukan penelitian untuk membangun atau merancang sistem pendeteksi warna baik pada objek benda ataupun buah-buahan. Sistem pendeteksi yang dibuat umumnya berbentuk mesin sortir berupa *conveyor* ataupun *hopper*. Selain itu juga, terdapat sistem yang menerapkan penggunaan board Mappi32 yang juga menjadi topik pada penelitian ini. Akan tetapi, dalam penelitian tersebut *board* Mappi32 digunakan untuk merancang sistem pakan otomatis. Sedangkan dalam penelitian ini, Mappi32 digunakan untuk diterapkan dalam perancangan sistem sortir kematangan buah berdasarkan warna.

Dalam mendeteksi warna tersebut, terdapat beberapa sensor yang digunakan, seperti sensor warna TCS3200, fotodiode, sensor IR, modul kamera hingga webcam. Umumnya, penelitian-penelitian tersebut banyak menggunakan sensor warna TCS3200 dalam pengimplementasian deteksi warna. Hal itu tentu saja lebih praktis untuk mendapatkan hasil citra yang diinginkan. Namun, pada beberapa penelitian telah digunakan sensor kamera ataupun webcam untuk mendeteksi warna menggunakan pengolahan citra. Dalam penggunaan kamera, gambar biasanya ditangkap untuk disimpan pada memori sehingga dapat secara terpisah diproses datanya sehingga menghasilkan klasifikasi warna. Sementara itu, dalam penelitian ini dibuat sistem pendeteksi warna menggunakan modul kamera OV2640 agar secara langsung menghasilkan deteksi warna.

Kemudian, dalam penelitian ini juga dibuat algoritma untuk mendeteksi kematangan 2 jenis buah. Objek sampel yang digunakan dalam penelitian ini adalah buah jeruk dan buah stroberi. Selain itu, pada berbagai penelitian dan literatur, umumnya ditetapkan rentang *threshold hue* hanya berdasarkan representasi teoritis lingkaran warna tanpa pembuktian secara kuantitatif. Sedangkan pada topik ini, untuk mendapatkan rentang *hue* yang sesuai, dilakukan dengan mengombinasikan metode visualisasi histogram, beserta verifikasi melalui metode *adaptive threshold* dengan *otsu threshold* dan *gaussian filter*. Hasil dari deteksi kematangan yang diharapkan tidak hanya berbentuk respon dari perangkat *hardware*, melainkan juga ditampilkan berupa hasil *counting* pada *dashboard* visualisasi berbasis web. Namun, penelitian ini tidak berfokus menghasilkan *software* yang di-*hosting* khusus, melainkan masih berbasis pada *localhost*.

III. METODOLOGI PENELITIAN

3.1 Waktu dan Tempat

Dalam menyelesaikan penelitian ini, telah dibuat jadwal pengerjaan serta tempat berlangsungnya kegiatan penelitian. Adapun rinciannya seperti :

Tempat : Laboratorium Teknik Komputer Jurusan Teknik Elektro,
 Universitas Lampung dan Agropark Sabah Balau
 Waktu : Mei 2024 – September 2024

Tabel 3.1 Jadwal Pengerjaan Skripsi

No	Aktivitas	Waktu Penelitian																	
		2024																	
		Mei			Juni				Juli				Agustus				September		
		2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3
1	Studi literatur	■																	
2	Analisis dan persiapan kebutuhan		■	■															
3	Desain Sistem				■	■													
4	Pengolahan Sampel Warna						■	■	■										
5	Perancangan sistem deteksi kematangan Model I									■	■	■	■						
6	Pengujian <i>Prototype</i> Model I									■	■	■	■						
7	Perancangan sistem deteksi kematangan Model II													■	■	■	■		
8	Pengujian Prototype Model II													■	■	■	■		
9	Penyusunan Laporan											■	■	■	■	■	■	■	

3.2 Alat dan Bahan Penelitian

3.2.1 Alat

Adapun beberapa alat yang dibutuhkan dalam penelitian ini termasuk *software* dan *hardware* adalah seperti pada Tabel 3.2 berikut.

Tabel 3.2 Alat Penelitian

No	Perangkat	Keterangan Penggunaan
1.	Laptop ASUS VivoBook X441UAR	Media pemrograman dan pengolahan data.
2.	Arduino IDE versi 2.2.1	<i>Software</i> untuk melakukan pemrograman pada mikrokontroler.
3.	OpenCV	Pustaka <i>open-source</i> untuk melakukan pemrosesan citra (dalam hal ini adalah proses <i>thresholding</i>) menggunakan bahasa pemrograman Python.
4.	Visual Studio Code	<i>Software</i> untuk melakukan pemrograman Python dan pembuatan <i>dashboard</i> hasil hitung.
5.	Mappi32	<i>Board</i> Mikrokontroler sebagai pusat kontrol pengiriman data.
6.	Modul OV2640	Modul kamera yang digunakan untuk menangkap dan mendeteksi gambar.
7.	Motor Servo (Micro Servo 9g SG90)	Sebagai aktuator gerak yang bergerak mendekati buah saat buah terdeteksi matang.
8.	Kabel Jumper	Penghubung antar komponen elektronik (tanpa PCB).
9.	Modul LM2596	Modul <i>step-down</i> untuk menurunkan tegangan DC pada motor DC dari baterai eksternal.
10.	Baterai Eksternal 7,4V Li-Po	Sebagai sumber daya untuk memutar Motor DC.
11.	Motor DC	Sebagai pemutar mesin sortir.
12.	Saklar ON/OFF	Sebagai tombol untuk menyalakan mesin <i>conveyor</i> .

Untuk mendukung berjalannya proses pengolahan pada beberapa *software* dan *hardware* di Tabel 3.2, terdapat beberapa *library* yang digunakan. *Library* tersebut diterapkan pada proses pemrograman, baik menggunakan Arduino IDE, ataupun Visual Studio Code. Adapun *library* atau pustaka yang digunakan ditampilkan pada Tabel 3.3 berikut.

Tabel 3.3 Daftar *Library* dan Modul

No	Pustaka/ <i>Library</i> /Modul	Keterangan Penggunaan
1.	WiFi.h	Bagian <i>library</i> WiFi untuk ESP32. Digunakan untuk konektivitas Wi-Fi.
2.	WebServer.h	Modul yang digunakan untuk menjalankan web server sederhana di ESP32.
3.	WiFiClient.h	Modul bagian dari pustaka WiFi yang digunakan untuk membuat koneksi klien HTTP.
4.	OV2640.h	Modul driver untuk kamera OV2640.
5.	camera_pins.h	Modul yang berisi definisi pin untuk modul kamera pada ESP32-Cam.
6.	esp_camera.h	Modul yang digunakan untuk mengontrol kamera ESP32-Cam
7.	ArduinoJson.h	Modul bagian dari <i>library</i> ArduinoJson yang digunakan untuk memanipulasi data JSON di platform mikrokontroler.
8.	ESP32Servo.h	Modul untuk mengontrol servo motor dengan <i>board</i> ESP32.
9.	os	Modul standar Python untuk operasi terkait sistem operasi atau penarikan file dari sistem operasi.
10.	OpenCV	<i>Library</i> untuk pemrosesan gambar dan visi komputer dengan modul utama cv2.
11.	NumPy	Digunakan untuk komputasi numerik, terutama array multidimensi.
12.	asyncio	Modul yang digunakan untuk pemrograman asinkron.
13.	Matplotlib	Digunakan untuk membuat grafik plot data dengan modul utama <i>pyplot</i> .
14.	websockets	<i>Library</i> untuk mengimplementasikan WebSocket di Python dengan modul utama websockets.
15.	json	Digunakan untuk bekerja dengan format data JSON.
16.	urllib dengan modul request	Digunakan untuk melakukan request HTTP.
17.	base64	Digunakan untuk <i>encoding</i> dan <i>decoding</i> data dalam format Base64.
18.	time	Digunakan untuk operasi terkait waktu.

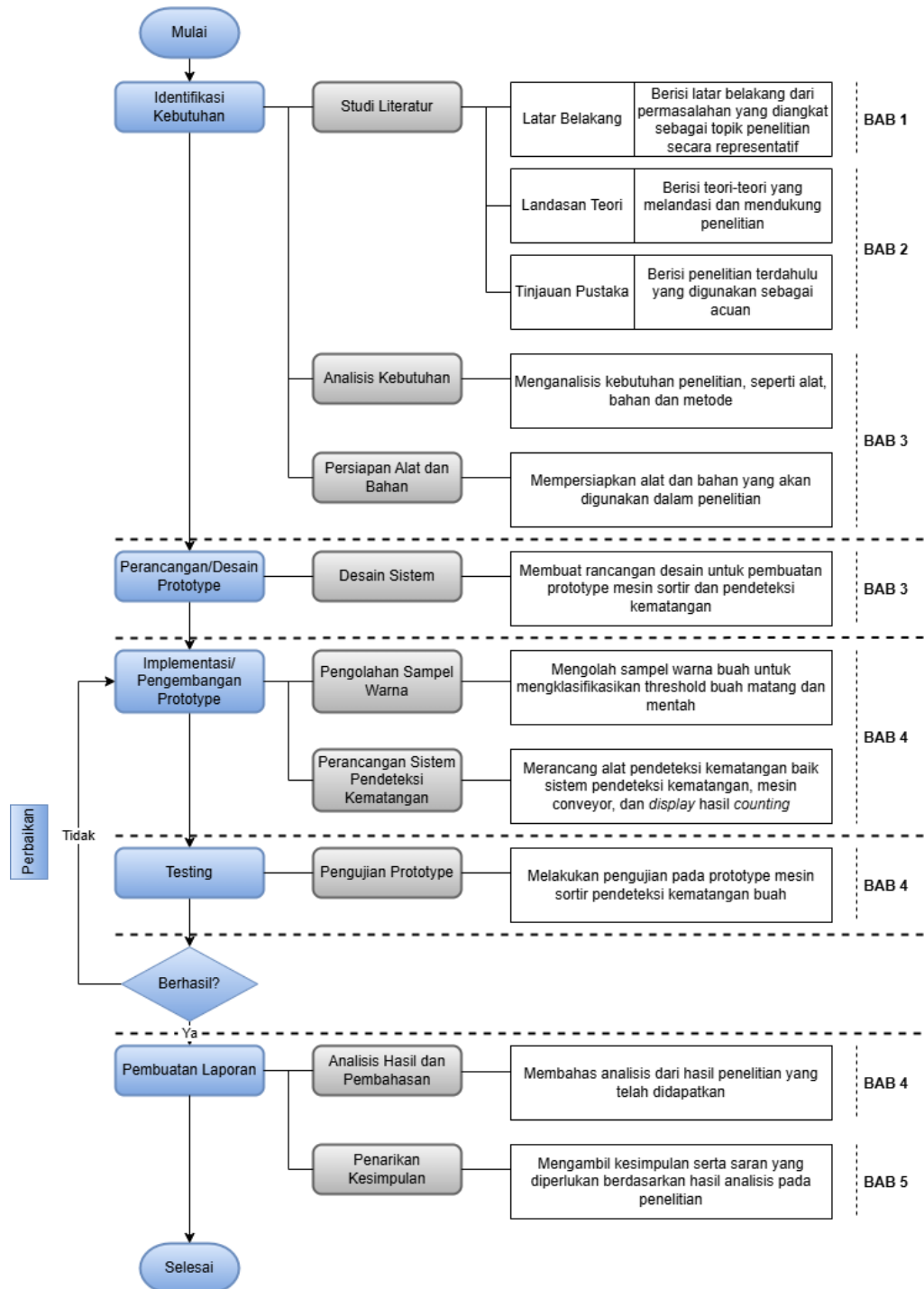
3.2.2 Bahan

Dalam penelitian ini, bahan yang digunakan sebagai objek adalah buah jeruk dan buah stroberi. Buah-buahan ini dapat dijumpai di pasaran dan juga di lahan perkebunan PKK Agropark Lampung (sekitar lokasi penelitian di Bandar Lampung). Kedua buah itu termasuk dalam jenis buah-buahan yang sebagian besar kematangannya dapat dilihat dari perubahan warna. Selain itu, kedua buah itu adalah buah yang memerlukan proses penyortiran terutama jika akan dikemas dan dipasarkan. Dari buah tersebut, didapatkan data warna untuk mendeteksi tingkat kematangannya. Selain itu, buah-buahan ini juga digunakan secara langsung sebagai objek pengujian *prototype* sistem.

3.3 Tahap Penelitian

Tahapan penelitian yang dilakukan mengadaptasi alur kerja dari model *prototype*. Berdasarkan metode tersebut, dalam proses pengerjaan penelitian ini terdiri dari beberapa tahapan. Tahapan tersebut dimulai dengan mengidentifikasi kebutuhan. Setelah didapatkan analisis kebutuhan, selanjutnya dibuat sebuah desain dari *prototype*. Dari desain tersebut, berikutnya dapat dilakukan implementasi *prototype*. Dalam pengimplementasian tersebut, dipastikan apakah sistem berjalan sesuai fungsionalitasnya melalui tahap *testing* atau pengujian. Apabila sistem belum berjalan dengan baik, artinya perlu dilakukan perbaikan. Setelah didapatkan hasil penelitian, dibuatlah laporan akhir penelitian. Setiap tahapan dalam metode tersebut terbagi menjadi beberapa bagian. Untuk mempermudah memahami alur dari penelitian ini, dibuat sebuah diagram alir penelitian. Dalam alur penelitian ini juga dijelaskan keseluruhan isi beserta proses yang ada dalam setiap Bab laporan penelitian (skripsi).

Adapun diagram alir penelitian tersebut dapat dilihat pada Gambar 3.1 berikut.



Gambar 3.1 Diagram Alir Penelitian

Berdasarkan Gambar 3.1 Diagram Alir Penelitian, dapat dilihat bahwa tahapan penelitian tersebut meliputi beberapa blok metode *prototype*, yaitu :

1. Identifikasi Kebutuhan

Dalam blok identifikasi kebutuhan terdapat beberapa tahap, yaitu studi literatur,

analisis kebutuhan, dan persiapan alat dan bahan. Studi literatur di sini adalah mengumpulkan dan mempelajari berbagai sumber referensi/literatur dari penelitian terkait. Hal ini dilakukan untuk mendapatkan pemahaman terkait penelitian yang dikerjakan. Literatur yang digunakan dapat tersedia dari berbagai sumber, seperti jurnal, prosiding, buku, ataupun situs web terkait. Dalam penelitian ini, literatur yang digunakan adalah literatur yang berisi teori-teori yang berkaitan dengan perancangan pendeteksi warna ataupun pemanfaatan komponen serupa.

Setelah memahami berbagai literatur yang digunakan, maka dapat dilanjutkan dengan menganalisis alat dan bahan apa saja yang diperlukan berdasarkan karakteristik dari penelitian. Analisis kebutuhan di sini mencakup analisis alat dan bahan yang digunakan, serta metode dan tahapan penelitian. Hasil analisis kebutuhan dapat digunakan untuk mempersiapkan alat-alat apa saja dibutuhkan dalam penelitian. Alat di sini meliputi perangkat keras dan perangkat lunak yang diperlukan selama penelitian. Selain itu, dilakukan juga observasi untuk memastikan ketersediaan objek penelitian yang diambil.

2. Perancangan/Desain *Prototype*

Pada tahapan ini dilakukan desain sistem untuk membuat rancangan atau gambaran dari sistem yang dihasilkan. Desain sistem ini meliputi desain mesin sortir dan juga desain sistem pendeteksi kematangan. Desain dibuat dalam bentuk gambar 2 dimensi dan blok diagram sistem. Desain sistem pendeteksi kematangan mempertimbangkan 2 model. Model pertama adalah model dengan pendekatan berbasis mikrokontroler. Lalu model kedua adalah model dengan membagi beban kerja antara mikrokontroler dan pemrograman Python pada komputer. Model II digunakan setelah mendapatkan hasil pengujian Model I yang butuh perbaikan.

3. Implementasi/Pengembangan *Prototype*

Dalam tahapan ini, terdiri dari pengolahan sampel warna dan perancangan sistem pendeteksi kematangan. Pengolahan sampel warna buah dilakukan pada buah jeruk dan buah stroberi berdasarkan tingkat kematangannya. Pengolahan ini dilakukan menggunakan metode pengolahan citra berbasis model segmentasi warna HSV untuk proses *thresholding*. Hasil batas ambang warna HSV dari

pengolahan warna diimplementasikan pada program yang membangun sistem pendeteksi warna. Selain itu, dilakukan juga perancangan sistem yang terdiri dari komponen elektronik seperti mikrokontroler, sensor dan aktuator. Kemudian, sistem tersebut dikoneksikan menuju *dashboard* untuk mendukung visualisasi data. Dalam tahapan ini juga, dikembangkan sistem dalam bentuk Model I dan Model II.

4. Pengujian

Setelah sistem selesai dirancang, selanjutnya adalah menguji kinerja sistem pada objek terkait. Apabila hasilnya sesuai dengan yang diharapkan, maka dapat dilanjutkan pada langkah selanjutnya hingga fungsionalitas sistem berjalan. Namun, jika belum sesuai artinya diperlukan penyesuaian ulang pada sistem ataupun kode program yang digunakan. Apabila terdapat perbaikan, artinya tahapan tersebut akan masuk dalam tahap revisi/perbaikan pada metode *prototype*. Sesuai penjelasan sebelumnya, Model II digunakan ketika Model I kurang sesuai.

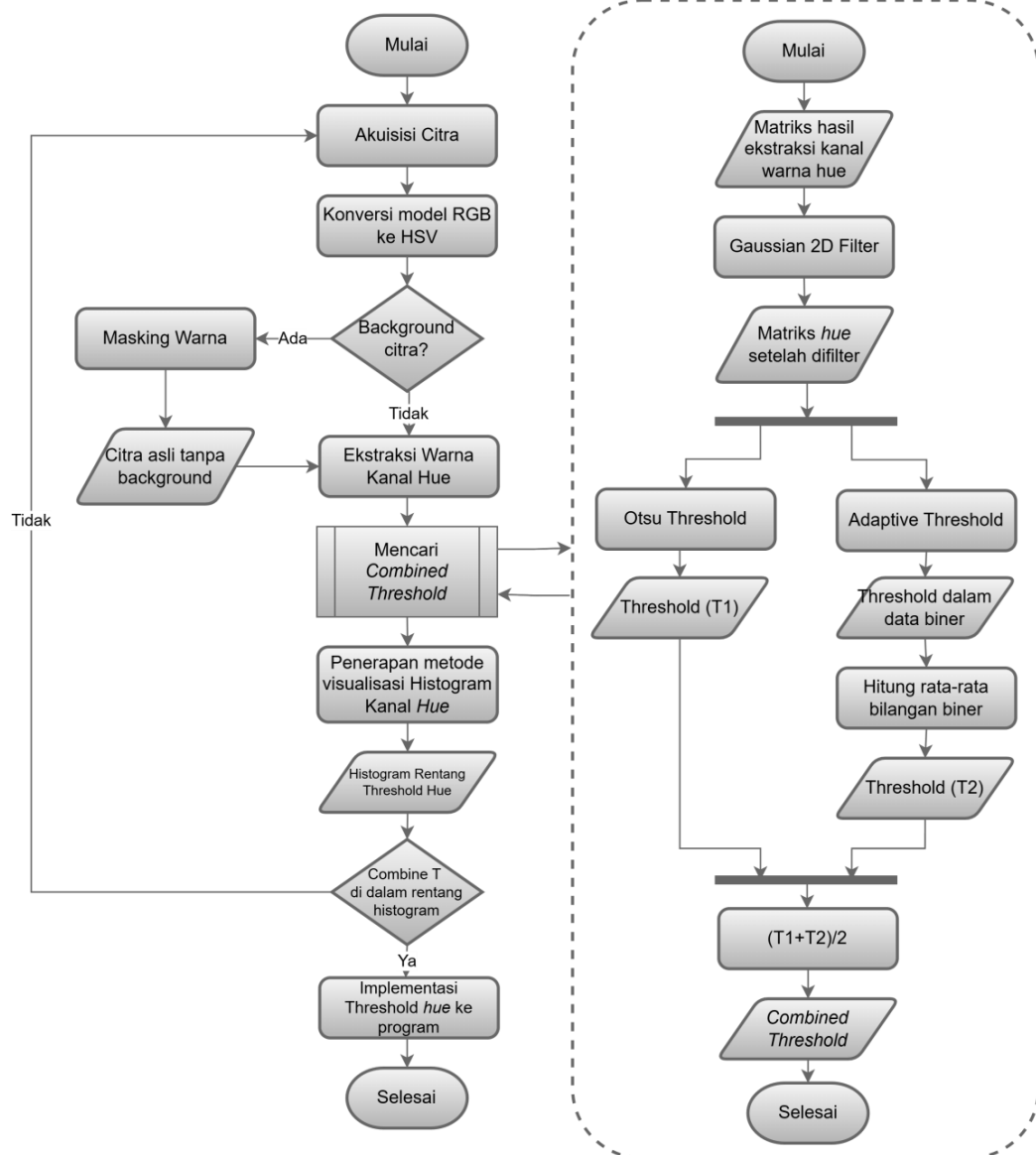
5. Pembuatan Laporan

Setelah menyelesaikan tahapan uji coba, dapat dilakukan penyusunan laporan. Laporan berisi hasil analisis dan pembahasan dari penelitian yang telah dilakukan. Selain itu, akan dilakukan juga penarikan kesimpulan dan saran yang diperlukan dari hasil penelitian yang didapatkan.

3.4 Rancangan Sistem

3.4.1 Algoritma Segmentasi Warna

Dalam penelitian ini, warna adalah parameter terpenting yang perlu diproses lebih lanjut. Pemrosesan warna dilakukan menggunakan metode pengolahan citra berbasis segmentasi warna HSV. Pemrosesan warna terbagi menjadi dua, yaitu pengolahan sampel warna dan implementasi pada komponen. Tahap pertama adalah pengolahan sampel warna. Adapun proses pengolahan sampel warna yaitu seperti pada Gambar 3.2 berikut.



Gambar 3.2 Alur Pencarian *Threshold*

Berdasarkan Gambar 3.2 Alur Pencarian *Threshold*, terdapat beberapa langkah untuk dapat menentukan nilai ambang pada citra buah. Langkah tersebut terbagi menjadi 2, seperti dilambangkan dengan penggunaan simbol *predefined process*. *Predefined process* dilakukan untuk mencari nilai *combined threshold*. Adapun penjelasan langkah-langkah tersebut adalah :

1. Akuisisi Citra

Akuisisi citra merupakan tahapan untuk mendapatkan citra referensi/sampel citra yang digunakan. Citra yang dianalisis merupakan sampel citra dari berbagai

tingkat kematangan buah yang diuji (jeruk dan stroberi). Kategori kematangan yang digunakan pada penelitian ini adalah kategori mentah (hijau) dan matang (merah, oranye dan kuning). Pada skenario buah jeruk, warna yang dianggap sudah masuk pada kategori matang adalah warna kuning-oranye. Sedangkan dalam kasus buah stroberi, warna yang masuk dalam kategori matang adalah warna merah. Proses pencarian nilai *threshold* dilakukan menggunakan program Python. Untuk itu, diimplementasikan beberapa *library* dan modul seperti dijelaskan sebelumnya.

2. Konversi Model RGB ke HSV

Untuk mempermudah analisis warna, dilakukanlah konversi model warna RGB pada citra menjadi model warna HSV. Konversi ini diterapkan menggunakan fungsi pada *library* OpenCV yang mendukung rumus konversi warna. Secara garis besar, penggunaan *library* OpenCV mengadaptasi proses hitung konversi RGB ke HSV pada umumnya (sesuai penjelasan pada Bab 2). Pertama, nilai R, G dan B dinormalisasi ke nilai rentang 0-1 dengan membaginya pada 255. Lalu dicari nilai maksimum dari B, G, R untuk mendapatkan nilai *hue*, *saturation* dan *value*.

3. *Masking* Warna (opsional)

Apabila citra yang diambil masih memiliki latar belakang atau objek lain yang tidak ingin digunakan, maka dilakukan tahap *mask* citra. *Masking* citra dilakukan untuk memisahkan latar belakang dengan objek yang ingin digunakan.

4. Ekstraksi Warna

Tahapan penting selanjutnya adalah tahap ekstraksi warna. Apabila citra yang digunakan adalah citra tanpa latar belakang atau objek lain, maka citra dapat langsung diekstraksi. Namun, jika memerlukan tahapan *masking*, maka ekstraksi warna akan dilakukan pada citra asli hasil *masking* warna. Tahapan ekstraksi warna ini dilakukan dengan memisahkan saluran warna H, S dan V serta berfokus untuk mengambil salinan kanal warna *hue*. Proses ini dilakukan untuk mempermudah dalam menganalisis kanal warna tersebut. Proses ekstraksi warna ini melanjutkan tahapan konversi. Pada bagian ini, nilai kanal warna h, s dan v yang dinormalisasi diambil satu persatu. Namun, untuk proses selanjutnya, fokus

pencarian tersebut menggunakan kanal warna *hue*.

5. *Predefined Process* : Mencari *Combined Threshold*

Tahap selanjutnya adalah pencarian nilai *combined threshold*. Tahap ini dapat dilakukan bersamaan, lebih dahulu ataupun sesudah histogram ditampilkan. Ini karena proses pencarian *combined threshold* tidak mengganggu proses visualisasi histogram. Histogram dibuat untuk menampilkan visualisasi *hue* pada citra original yang belum mengalami pemrosesan tambahan. Hal ini dilakukan untuk menjaga konsistensi dan memberikan distribusi warna yang akurat. Sementara itu, nilai *combined threshold* digunakan untuk memverifikasi rentang tersebut. Kombinasi dari 3 metode *thresholding* akan membentuk 1 *threshold* tunggal yang merepresentasikan ambang dominan dari citra yang digunakan. Berikut adalah proses penentuan nilai ambang kombinasi.

a) Menerapkan *Gaussian Filter 2D* pada kanal *hue* yang sudah diekstraksi

Penerapan *gaussian filter* dilakukan untuk mendapatkan kanal *hue* citra yang difilter. Citra ini adalah versi citra yang telah diperhalus untuk menghindari *noise*. Untuk mendapatkan hasil *blur_h* (kanal *hue* citra yang difilter), dilakukan dengan menerapkan *library* OpenCV. Secara kode program, dituliskan menjadi *cv2.GaussianBlur*. Cara kerja fungsi ini mengadaptasi perhitungan normal pada metode 2D *Gaussian filter* melalui proses konvolusi pada citra *hue*.

b) Penerapan *Otsu Threshold*

Setelah didapatkan citra *hue* yang di-blur, selanjutnya dicari nilai *threshold* satu persatu. Seperti pada *flowchart*, terdapat pemisahan proses menggunakan simbol *fork*. Untuk cabang pemrosesan *otsu thresholding* juga dilakukan dengan *library* OpenCV. Proses di dalam OpenCV juga menerapkan rumus yang ada sebelumnya. Yaitu untuk mendapatkan hasil *threshold*, maka dicari intensitas gambar untuk menentukan distribusi piksel (p_i). Selanjutnya dihitung setiap nilai ambang T beserta nilai ambang optimal T^* . Hasil *threshold otsu* pada kanal warna *Hue* (h) tersebut memberikan nilai ambang tunggal yang disebut dengan T atau *otsu_th_h* pada program.

c) Penerapan *Adaptive Threshold*

Pencarian nilai *threshold* dalam tahap ini melibatkan proses *adaptive threshold* dengan metode Gaussian. Dalam *library* OpenCV, pencarian nilai ambang ini menerapkan fungsi `cv2.ADAPTIVE_THRESH_GAUSSIAN_C`. Nilai *threshold* untuk setiap piksel dihitung menggunakan rata-rata Gaussian dari nilai-nilai piksel dalam sebuah jendela lokal di sekitar piksel tersebut. Nilai konstanta (C) yang digunakan adalah 2. 2 adalah nilai konstanta yang cocok diterapkan untuk gambar yang memiliki variasi intensitas kecil (sedikit *noise* dan variasi pencahayaan). Karena sudah difilter sebelumnya, *noise* pada citra yang digunakan pun berkurang. Dengan nilai C yang kecil, memungkinkan lebih banyak piksel di bawah rata-rata untuk disegmentasi sebagai *foreground*. Hasil dari metode ini berupa bilangan biner 0 atau 255. Sehingga untuk mendapatkan T tunggal, dihitung rata-rata dari *threshold* binernya.

d) Menentukan *Combined Threshold*

Untuk menentukan nilai ambang kombinasi, hasil *threshold otsu* dan *adaptive* hanya perlu digabungkan dan dibagi dengan 2 (*mean*). Penggunaan metode ini dilakukan untuk melengkapi kekurangan masing-masing metode secara terpisah. Dengan mengombinasikannya, diharapkan akan mendapatkan hasil yang lebih baik untuk nilai ambang yang dipengaruhi pencahayaan yang bervariasi dan juga adanya *noise*. Nilai ini digunakan untuk memverifikasi rentang *hue* pada histogram.

6. Penerapan Metode Visualisasi Histogram Kanal *Hue*

Dalam visualisasi histogram kanal *hue*, rentang *hue* yang digunakan adalah 0-180, bukan 0-360. Karena dalam OpenCV, dikenal adanya representasi 8-bit, sehingga 0-180 adalah bentuk penyederhanaan 360 derajat untuk memudahkan pemrosesan gambar. Dalam kata lain, setiap nilai *hue* mewakili dua derajat dari spektrum *hue* sebenarnya. Sehingga rentang 0-180 membuat histogram telah mencakup seluruh spektrum warna *hue* yang relevan dalam format ini. Lalu pencarian distribusi frekuensi *hue* pada histogram, dilakukan dengan menentukan bins (kelompok-kelompok) yang digunakan (rentang 0-180).

Selanjutnya dihitung frekuensi setiap piksel untuk setiap bin sesuai persamaan *threshold* histogram sebelumnya. Karena diasumsikan bahwa rentang ambang

belum diketahui, maka pengelompokan bin dibuat menjadi rentang 0-1, 1-2, dst hingga 179-180. Hasil distribusi piksel yang didapatkan akan mengisi setiap frekuensi bin sehingga terbentuk histogram dengan sebaran *hue* yang variatif sesuai piksel citranya. Dalam tahap ini, histogram menampilkan rentang *threshold* yang terdiri dari *lower* dan *upper*. Sesuai persamaan histogram, hasil yang didapatkan dapat berupa 1 rentang nilai ambang, atau 2 rentang nilai ambang, bahkan lebih sesuai sebaran frekuensi *hue*.

7. Verifikasi Histogram

Seperti dijelaskan sebelumnya, verifikasi dilakukan dengan memastikan apakah *combined threshold* berada dalam rentang histogram. Jika ada, maka rentang histogram dapat digunakan untuk implementasi pada program. Sedangkan jika tidak, maka kalibrasi harus terus dilakukan dengan berbagai sampel citra. Nilai ambang yang dihasilkan berfokus pada kanal *hue* saja, sedangkan kanal *s* dan *v* akan menyesuaikan. Maksudnya menyesuaikan adalah dengan mengadaptasi standar pada berbagai literatur, yaitu 100-255. Dari pertimbangan tersebut, selanjutnya nilai diujikan pada sampel citra menggunakan *trackbar*. Apabila menghasilkan segmentasi citra yang optimal, maka rentang tersebut langsung digunakan.

Setelah didapatkan nilai ambang untuk citra referensi, proses penting selanjutnya adalah penerapan nilai ambang tersebut pada algoritma pengolahan citra. Penerapan tersebut dilakukan untuk mengoptimalkan proses pada komponen sistem yang digunakan. Karena mempertimbangkan penerapan 2 model, alur yang dibuat pun menyesuaikan penerapan untuk keduanya, baik pada mikrokontroler secara langsung ataupun dengan bantuan pemrograman Python di komputer. Proses tersebut dapat dilihat pada Gambar 3.3 berikut.



Gambar 3.3 Segmentasi HSV pada Sistem

Pada Gambar 3.3 Segmentasi HSV pada Sistem, proses segmentasi citra dilakukan sampai mendapatkan persentase warna yang diujikan dalam skala HSV. Hasil persentase tersebut digunakan untuk menentukan kategori dari buah mentah atau matang. Proses ini digunakan sebagai *predefined process* untuk *flowchart* sistem berikutnya. Untuk memperjelas tahapan-tahapan di atas, adapun penjabarannya seperti di bawah ini :

1. Akuisisi Citra

Dalam penerapan Model I, tahap akuisisi citra ditandai dengan pengambilan citra buah oleh sensor kamera. *Buffer* gambar yang dikirimkan oleh sensor kamera diterima melalui pin serial mikrokontroler untuk diolah. *Buffer* citra yang digunakan hanya berjumlah 1 *buffer*. Ini dilakukan sesuai konsep sistem yang berjalan secara *real-time*. Sehingga, setelah *buffer* dikirim dan diolah, maka langsung dilepas dan *buffer* diisi dengan citra yang baru saat itu. Resolusi citra yang digunakan adalah citra dengan ukuran 160x120px (QQVGA) dengan format RGB565. Ukuran ini adalah ukuran minimum yang umum digunakan dalam pengolahan citra digital dengan kualitas yang masih bisa terbaca.

Alasan pengambilan resolusi ini adalah agar tidak memakan penyimpanan yang besar, tetapi juga citra masih bisa terbaca (tanpa perlu *downscaling*). Format RGB565 digunakan untuk tetap mempertahankan kanal warna agar dapat dikonversi ke model HSV. Dalam program, format RGB565 menghasilkan citra 16-bit (*uint_16*). Sedangkan pin serial hanya menangani 8-bit. Oleh sebab itu, pengiriman *frame* citra dilakukan secara bertahap. Alasan tidak digunakannya format JPEG (8-bit) secara langsung adalah karena dengan format tersebut, perlu adanya proses tambahan. Proses tambahan itu mencakup *decoder* citra JPEG dan mengubahnya ke RGB sebelum ke HSV. Sementara itu, pada pengembangan Model II, akuisisi citra dilakukan melalui *streaming* citra pada *web server* secara langsung.

2. Konversi Citra RGB ke HSV

Sesuai dengan metode yang diambil, yaitu metode segmentasi warna HSV. Maka, citra yang diperoleh harus dikonversi dari model warna umum (RGB) ke dalam model warna HSV untuk mempermudah analisis warna. Sama seperti pada proses pencarian nilai *threshold*, alur konversi RGB ke HSV mengadaptasi beberapa persamaan yang diterapkan dalam bentuk kode program. Dalam Model I, proses konversi citra dilakukan langsung di dalam mikrokontroler Mappi32. Citra yang dikonversi adalah RGB565 ke HSV. Sedangkan dalam Model II, proses konversi dapat dilakukan melalui pemrograman Python yang terhubung dengan *websocket web server streaming* citra objek buah.

3. Implementasi Nilai Ambang

Dalam tahap ini, rentang nilai ambang yang telah didapatkan sebelumnya digunakan untuk proses klasifikasi warna pada objek yang dikonversi. Dengan menerapkan nilai ambang ini, perhitungan nilai piksel warna menjadi lebih mudah tanpa proses tambahan.

4. Hitung Piksel Warna

Selanjutnya dihitung nilai piksel warna tersebut. Piksel warna yang dihitung adalah piksel warna buah matang (merah, kuning, oranye) dan warna buah mentah (hijau). Pada Model I, perhitungan dapat dilakukan menggunakan kode program pada mikrokontroler. Secara umum, program perhitungan pada mikrokontroler

Mappi32 didasari dengan persamaan :

a) Mengambil nilai *Hue*, *Saturation*, dan *Value*

$$h = fb \rightarrow buf[i] \quad (3.1)$$

$$s = fb \rightarrow buf[i + 1] \quad (3.2)$$

$$v = fb \rightarrow buf[i + 2] \quad (3.3)$$

b) Menentukan nilai ambang yang sesuai untuk *saturation* dan *value*

$$(s > Threshold_S) \text{ dan } (v > Threshold_V) \quad (3.4)$$

c) Jika kondisi terpenuhi, maka akan dilakukan *increment* pada piksel warna

$$coloredPixels = coloredPixels + 1 \quad (3.5)$$

Sedangkan pada Model II, perhitungan piksel warna dihitung dengan bantuan fungsi *np.count_nonzero*. Secara istilah, fungsi ini digunakan untuk menghitung berapa banyak piksel yang memiliki nilai warna (selain 0).

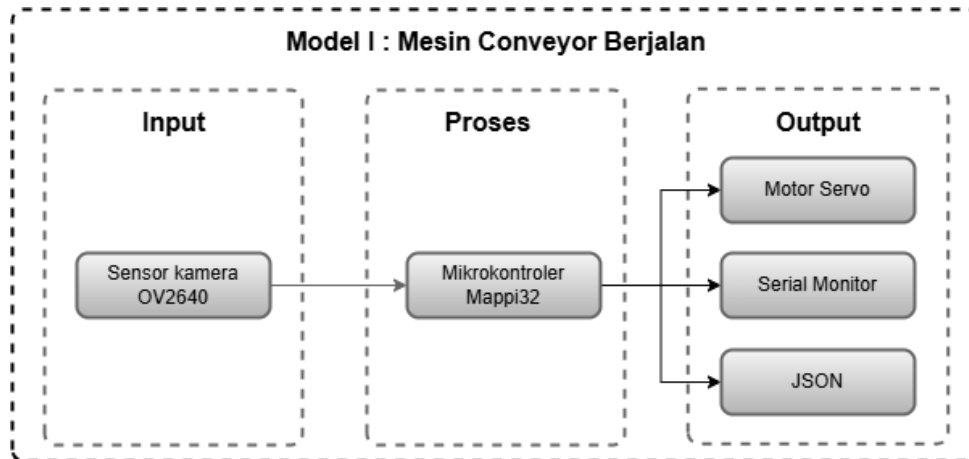
5. Hitung Persentase Warna

Tahap yang terakhir adalah menghitung persentase warna. Persentase warna inilah yang digunakan untuk mengklasifikasikan kategori buah apakah matang atau tidak dalam program mikrokontroler ataupun Python. Adapun rumus persentase warna secara umum adalah:

$$colorPercentage = \frac{Jumlah\ Piksel\ Warna}{Total\ Piksel\ dalam\ Citra} \times 100 \quad (3.6)$$

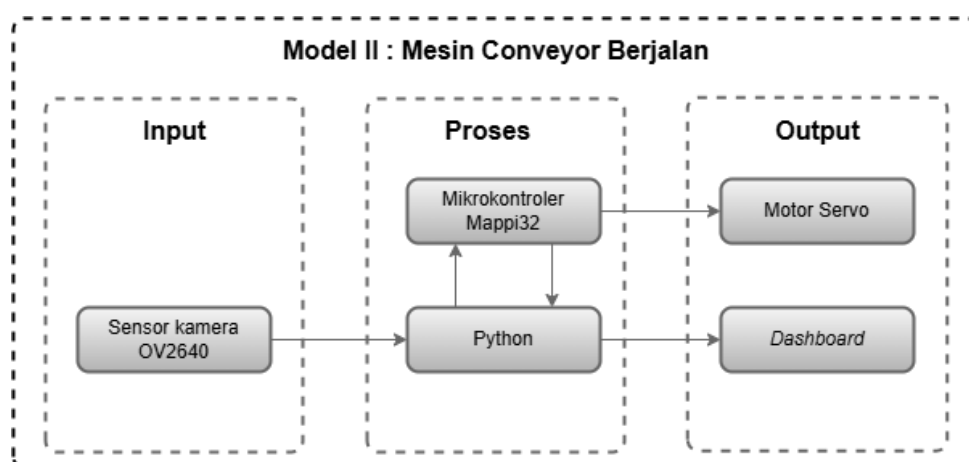
3.4.2 Blok Diagram Sistem

Adapun diagram blok dari sistem pendeteksi kematangan berdasarkan warna dengan pendekatan Model I dan II yang dirancang yaitu seperti pada Gambar 3.4 dan 3.5 berikut.



Gambar 3.4 Blok Diagram Sistem Model I

Berdasarkan Gambar 3.4 Blok Diagram Sistem Model I, terdapat 3 blok yaitu blok input, proses dan output selama mesin *conveyor* berjalan. Pada blok *input*, sensor kamera menangkap citra buah yang diperlukan. Setelah didapatkan citra buah, selanjutnya dilakukan proses pada mikrokontroler Mappi32. Pemrosesan dalam mikrokontroler mengikuti alur segmentasi HSV pada diagram sebelumnya. Dalam blok *output* terdapat 3 komponen, yaitu motor servo sebagai aktuator, serial monitor dan teks JSON sebagai output *display*. Setelah dilakukan proses dan didapatkan hasil deteksi, pada kategori matang atau mentah, sistem menggerakkan motor servo untuk memasukkan buah pada wadah yang disediakan. Selain itu, data hasil *counting* dari buah yang matang dan mentah dikirimkan ke dalam format JSON beserta tampilan matang atau mentah di serial monitor.



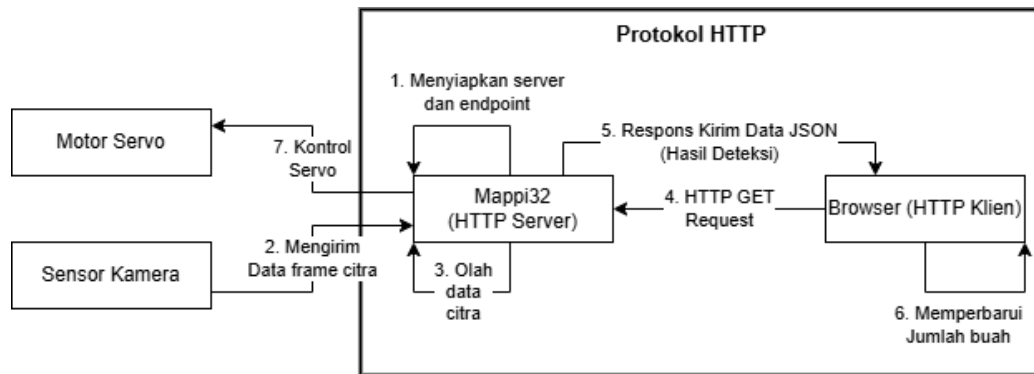
Gambar 3.5 Blok Diagram Sistem Model II

Pada Gambar 3.5 Blok Diagram Model II, terdapat sedikit perbedaan dibandingkan Model I. Pada blok *input* Model II, sensor kamera melakukan *streaming* citra buah. Setelah objek terdeteksi, objek tersebut langsung diolah melalui pemrograman Python pada blok proses. Di blok proses juga terdapat komunikasi antara Mappi32 dengan Python. Komunikasi ini dilakukan untuk memberikan alamat IP Mappi32 ke Python. Komunikasi ini menerapkan penggunaan protokol *WebSocket*. Hal ini diperlukan agar Mappi32 dapat menggerakkan servo di blok *output* berdasarkan hasil klasifikasi dari Python. Pemrosesan dalam Python juga mengikuti alur segmentasi HSV pada diagram sebelumnya. Dalam blok *output* juga ditampilkan hasil *counting* dan hasil klasifikasi ke dalam bentuk *dashboard* sederhana.

3.4.3 Model Komunikasi Sistem

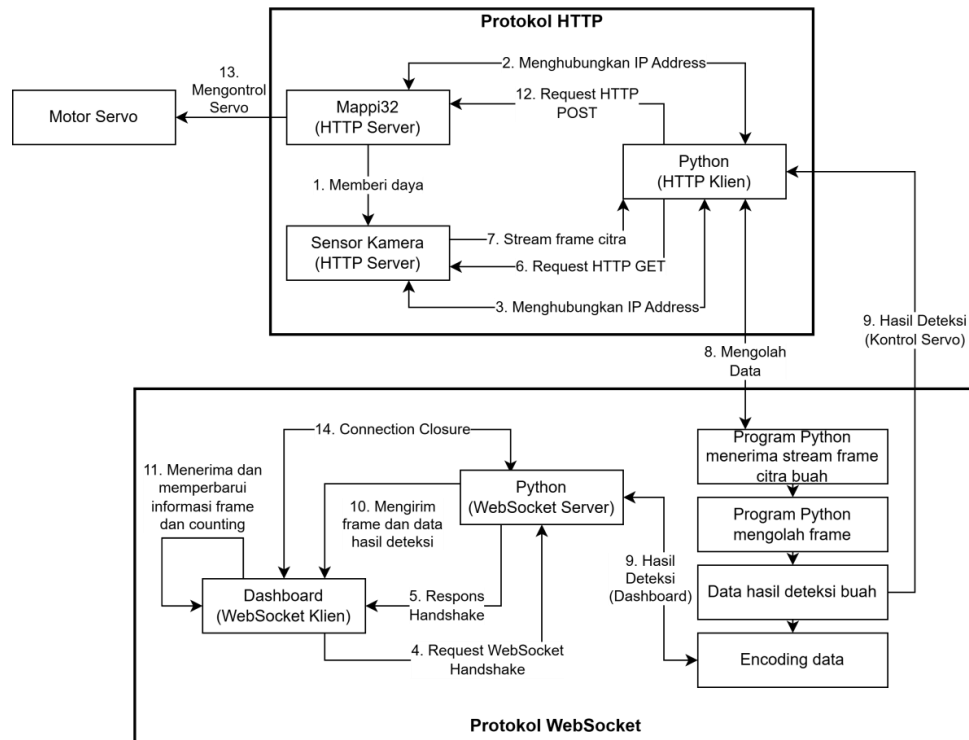
Untuk menghubungkan setiap interaksi pada sistem yang dirancang, terdapat alur komunikasi yang melandasinya. Secara garis besar, pada Model I diterapkan protokol HTTP untuk memperbarui data hasil hitung buah dalam format JSON. Namun, dalam Model II, terdapat penerapan protokol HTTP sekaligus *WebSocket*. Protokol HTTP digunakan untuk *streaming frame video citra* dan mengontrol servo. Sementara itu, *WebSocket* digunakan untuk memperbarui informasi *dashboard* secara *real-time*. Baik dalam Model I ataupun II, Mappi32 tetap diberi peran sebagai server dengan memasang *library* webserver di dalamnya. Hal ini dikarenakan perannya sebagai server dapat digunakan untuk memastikan desain sistem yang sederhana dan efisien, memungkinkan pengelolaan data secara *real-time*, serta *library* webserver yang ringan juga masih sesuai dengan kapasitas Mappi32, sehingga dapat mengoptimalkan fungsinya tanpa membebani sumber daya.

Berikut adalah diagram arsitektur komunikasi untuk kedua model perancangan yang disajikan dalam gambar 3.6 dan 3.7.



Gambar 3.6 Diagram Komunikasi Model I

Berdasarkan Gambar 3.6, dapat dilihat alur dari komunikasi pada Model I. Pada Model I, Mappi32 berperan sebagai pusat kontrol komponen sistem tertanam, sekaligus sebagai server HTTP. Kemudian *browser* yang digunakan pengguna bekerja sebagai klien HTTP. Pertama, Mappi32 yang sudah terhubung dengan komponen lainnya dan sudah mendapatkan daya akan menginisialisasikan server pada port 80 dan menyiapkan *endpoint /result* dalam format JSON. Kemudian sensor kamera mengirimkan data *frame* citra melalui pin serial. Setelah Mappi32 menerima data tersebut, data diletakkan pada *buffer* untuk dilakukan pengolahan data hingga didapatkan hasil deteksi. Apabila hasil deteksi sudah didapatkan, *browser* bisa segera menampilkan hasil *counting* buah dengan meminta dan mengakses data melalui HTTP GET ke alamat IP Mappi32 (mengakses URL *http://[ip-address]/result*). Kemudian, motor servo dapat bergerak sesuai dengan hasil deteksi yang didapatkan.



Gambar 3.7 Diagram Komunikasi Model II

Selanjutnya, alur komunikasi pada Model II terlihat seperti Gambar 3.7 tersebut. Pertama, Mappi32 tetap terhubung ke sensor kamera dan servo. Perbedaannya, pada Model II, Mappi32 hanya memberikan daya pada sensor kamera. Pada Model II, dikombinasikan kedua protokol. Penggunaan kedua protokol secara bersamaan disesuaikan dengan karakteristik tugas masing-masing komponen. Dengan mengombinasikannya, protokol HTTP dapat berfokus mengerjakan permintaan sederhana dan lebih ringan untuk kontrol servo dan meminta *stream video frame* dari sensor kamera. Sementara itu, *WebSocket* digunakan untuk mengirim dan menerima data hasil deteksi yang perlu ditampilkan secara *real-time* pada *dashboard*. Dalam protokol HTTP, Mappi32 dan sensor kamera berperan sebagai server HTTP. Sedangkan Python berperan sebagai klien HTTP sekaligus server *WebSocket*. Untuk klien *WebSocket* adalah *dashboard* yang berjalan pada *browser*. Setelah Mappi32 dan sensor kamera berhasil mendapatkan alamat IP masing-masing, keduanya dihubungkan melalui URL IP pada program Python.

Seperti dijelaskan sebelumnya, Model II menerapkan protokol HTTP dan

WebSocket secara beriringan. Dalam protokol *WebSocket* terdapat beberapa unsur seperti *handshake*, *frame data*, *encoding data*, dan *connection closure*. Memasuki unsur pertama, yaitu *handshake* yang ditandai dengan alur komunikasi nomor 4 dan 5. *Handshake* dapat diartikan sebagai proses membuka koneksi *WebSocket* antar server dan klien. *Dashboard* sebagai klien *WebSocket* melakukan *request handshake* untuk membuka koneksi *WebSocket* ke Python melalui alamat `ws://localhost:8765`. Kemudian, program Python sebagai server *WebSocket* menerima *handshake* dan memberikan respons dengan mengirimkan status 101 *Switching Protocols* untuk menyelesaikan *handshake*.

Selanjutnya, pada alur ke-6, Python me-*request* HTTP GET ke sensor kamera pada *endpoint stream video*. Sensor kamera pun menerima permintaan dan mulai mengirimkan *stream video* secara berkelanjutan ke python. *Frame stream video* citra tersebut diterima dan diolah menggunakan program Python hingga didapatkan hasil deteksi buah dan *frame* citra. *Frame* citra hasil deteksi tersebut, dalam *WebSocket* termasuk dalam unsur *frame data*. Dalam diagram tersebut, hasil deteksi ditandai dengan alur nomor 9, secara bersamaan dikirimkan untuk merespon servo dan memperbarui *dashboard*. Akan tetapi, pada data yang digunakan untuk memperbarui *dashboard* perlu dilakukan proses *encoding data*. Data hasil hitung dan deteksi buah dikodekan dalam format JSON. Sedangkan *frame* citra dikodekan dalam format base64 sebelum dikirimkan melalui *WebSocket*. *Frame* ini diubah menjadi format base64 agar dapat dikirim dalam bentuk teks.

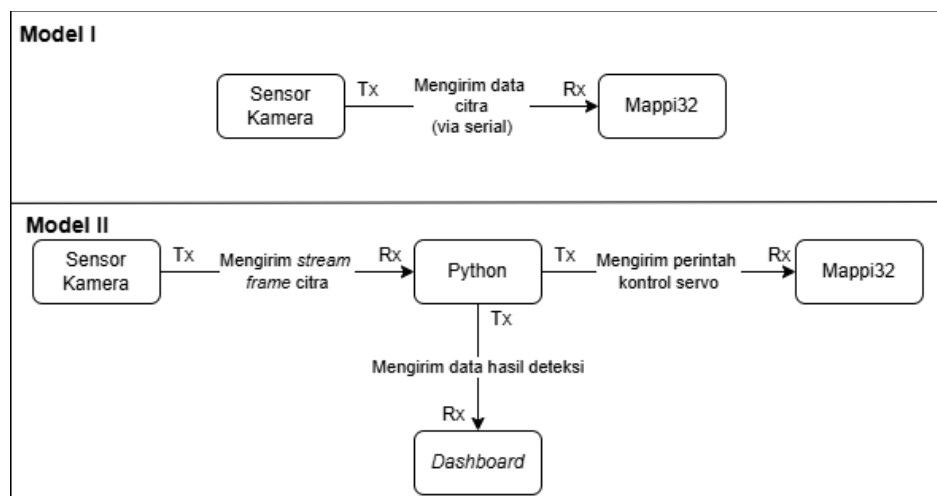
Adapun tahapan *encoding data frame*, adalah dimulai dari akuisisi *frame* berukuran QVGA (320x240 piksel) dengan format MJPEG. Dari format MJPEG, *frame* didekode kembali ke bentuk gambar dalam format JPEG untuk diencode menjadi base64 sebelum dikirimkan ke *dashboard*. Data *frame* inilah yang ditampilkan sebagai visualisasi hasil deteksi yang diperbarui secara terus menerus (*frame by frame*). Setelah data-data tersebut berhasil dikirimkan, *dashboard* dapat memperbarui informasi di dalamnya. Secara bersamaan, di sisi lain Python melakukan *request* HTTP POST ke Mappi32 pada alamat `http://<IP_Mappi32>/control` dengan parameter *command* “open” atau “close”

sesuai hasil deteksi. Berdasarkan kontrol tersebut, servo akan bergerak sesuai dengan hasil yang diberikan.

Dalam program yang dibuat, algoritma yang terjadi adalah menambah jumlah buah terlebih dahulu, baru menggerakkan servo. Oleh sebab itu, meski alurnya dilakukan bersamaan, *dashboard* akan merespon lebih dahulu sebelum servo (dengan perbedaan waktu yang kecil). Proses ini dapat terjadi secara terus menerus selama koneksi masih terbuka dan sistem masih berjalan. Apabila terjadi pemutusan koneksi pada salah satu jalur komunikasi, maka sistem akan berhenti berjalan. Dalam diagram ini, pemutusan koneksi diibaratkan dengan pemutusan koneksi *WebSocket* pada nomor 14, atau disebut sebagai *connection closure* oleh klien ataupun server.

3.4.4 Model Algoritma Pemrograman Sistem

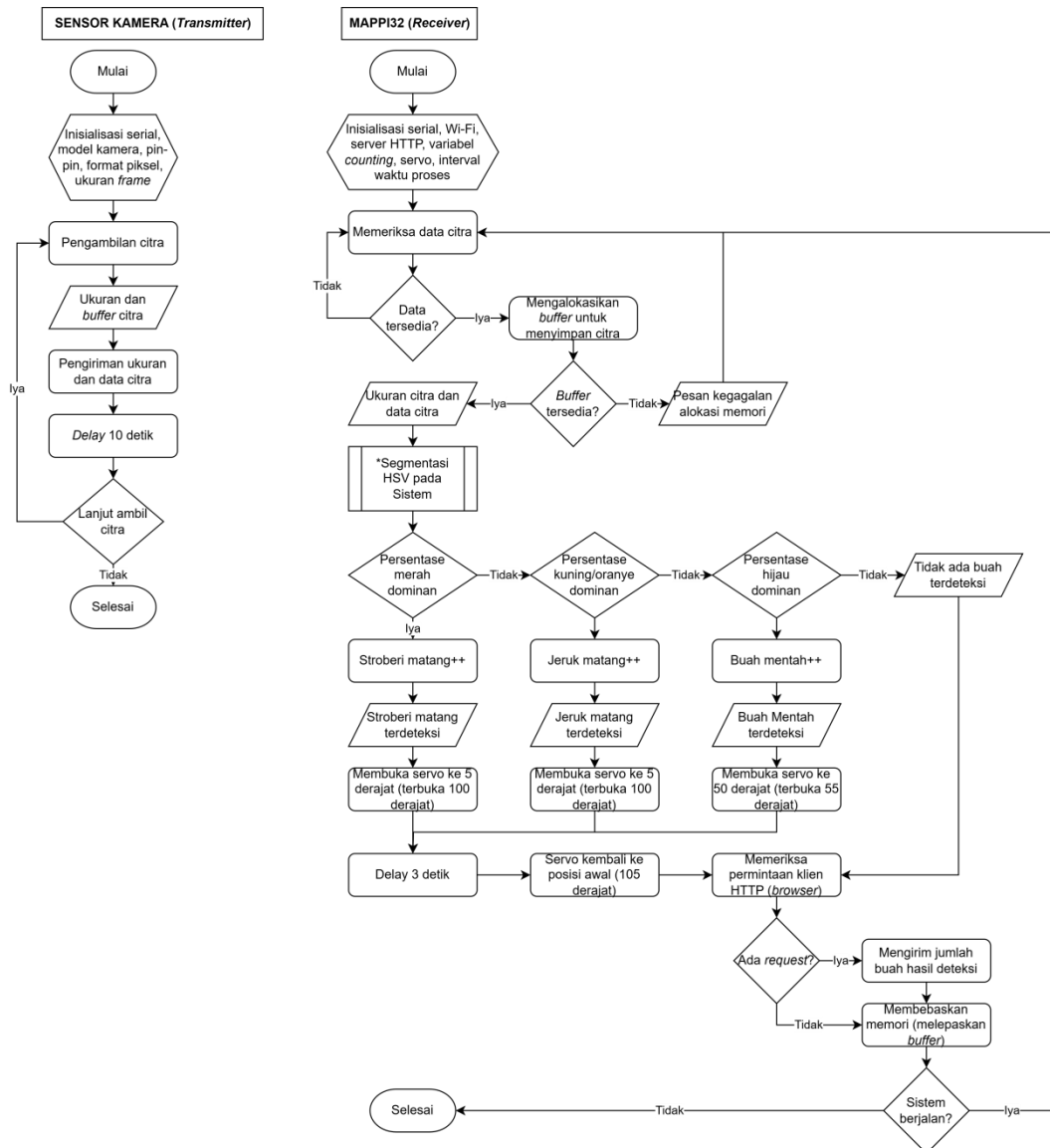
Untuk merancang sistem pendeteksi kematangan, penting untuk membuat algoritma pemrograman terlebih dahulu. Alur ini memudahkan dalam pengimplementasian dan penyusunan kode program pada sistem. Model algoritma pemrograman dibuat menggunakan *flowchart* untuk setiap komponen yang memerlukan implementasi kode program, seperti program Mappi32, sensor kamera, program Python dan juga *dashboard*. Adapun model algoritma pemrograman pada sistem dapat dilihat pada Gambar 3.8 hingga 3.11 berikut.



Gambar 3.8 Skema Peran Komponen pada Sistem

Sebelum membahas *flowchart* pemrograman, terdapat skema peranan setiap komponennya (*transmitter* atau *receiver*). Dalam Gambar 3.8 tersebut, diagram terdiri dari dua model sistem deteksi kematangan. Pada Model I, sensor kamera langsung mengirimkan data citra ke Mappi32 melalui komunikasi serial. Sensor kamera bertindak sebagai *transmitter* (TX) yang mengirimkan data citra, dan Mappi32 bertindak sebagai *receiver* (RX) yang menerima data tersebut. Sementara itu, dalam Model II, sensor kamera mengirimkan *stream frame* citra ke Python (dari TX ke RX), di mana Python melakukan pemrosesan data lebih lanjut. Python kemudian bertindak sebagai *transmitter* (TX) yang mengirimkan perintah kontrol servo ke Mappi32 (RX), untuk mengatur tindakan berdasarkan hasil deteksi dari kamera. Selain itu, Python juga bertugas mengirimkan data hasil deteksi ke *dashboard* untuk ditampilkan secara *real-time*. Dalam hal ini, dashboard bertindak sebagai *receiver* (RX) yang menerima data hasil deteksi dari Python (TX).

Selanjutnya adalah model alur pemrograman pada Model I dengan sensor kamera sebagai *transmitter*, dan Mappi32 sebagai *receiver*. Alur program dapat dilihat pada Gambar 3.9 berikut.



Gambar 3.9 Algoritma Program Model I

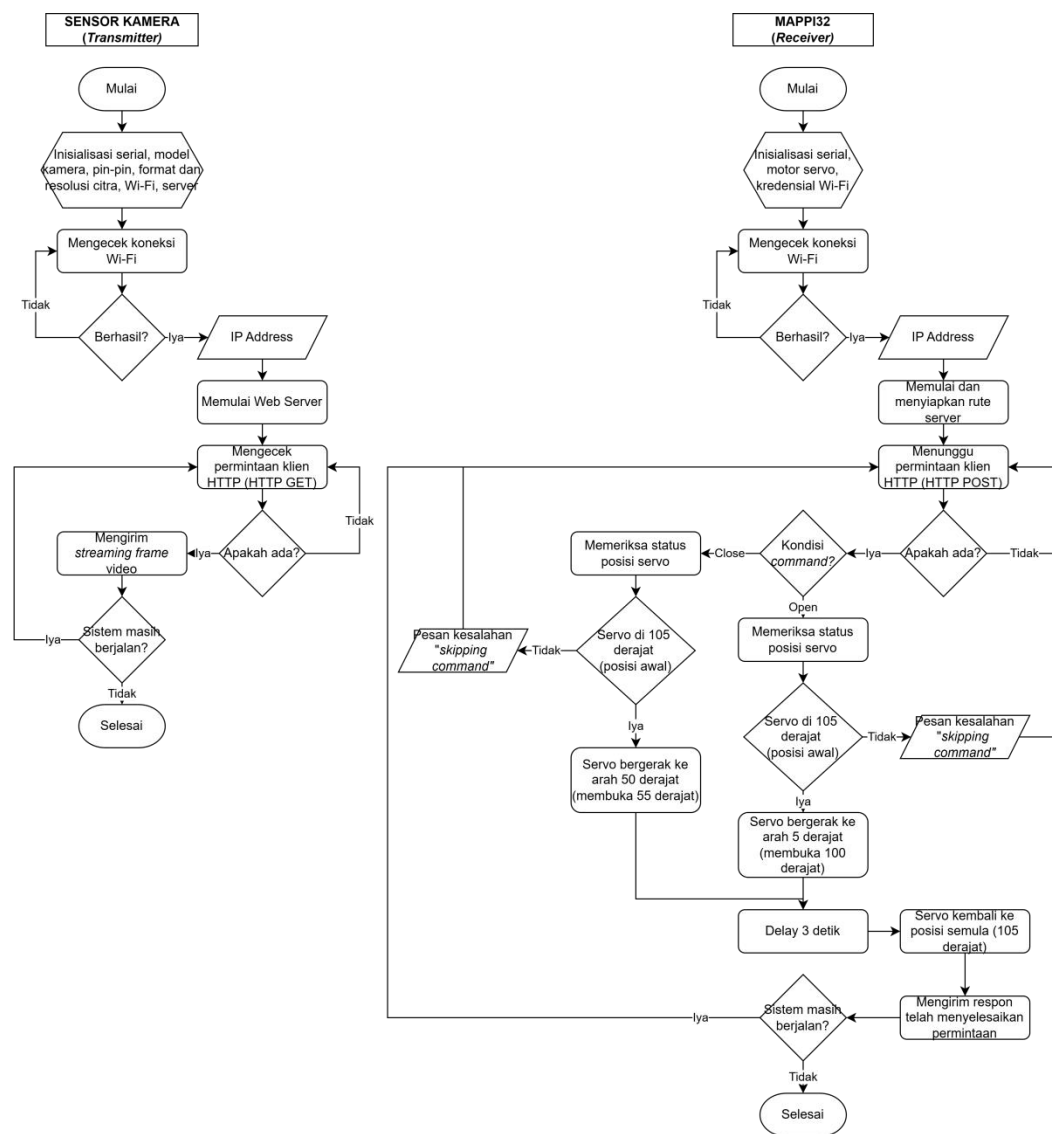
Flowchart pada Gambar 3.9 di atas menggambarkan alur algoritma pemrograman untuk proses pada sensor kamera dan Mappi32. Untuk algoritma pada sensor kamera merupakan alur untuk pengambilan citra dan pengiriman data dari kamera menuju Mappi32. Program dimulai dengan inisialisasi perangkat, termasuk pengaturan serial (115200), model kamera (AI Thinker), konfigurasi pin-pin yang terhubung, format piksel (RGB565), dan ukuran *frame* (QQVGA). Setelah inisialisasi selesai, sistem langsung masuk ke proses utama, yaitu pengambilan citra. Kamera mengambil gambar dan menyimpannya dalam *buffer*, serta menghitung ukuran citra tersebut. Setelah citra berhasil diambil dan disimpan,

sistem memulai pengiriman ukuran dan data citra ke Mappi32. Setelah data selesai dikirim, sistem menunggu selama 10 detik untuk memberikan jeda sebelum memulai proses pengambilan gambar berikutnya. Program ini menggunakan mekanisme *looping*, di mana setelah jeda, sistem memeriksa apakah akan melanjutkan pengambilan citra berikutnya atau tidak. Jika kondisinya terpenuhi, proses pengambilan citra diulang kembali dari awal. Namun, jika tidak, proses akan dihentikan dan program selesai. Dengan kata lain, sensor kamera akan terus menangkap dan mengirim citra selama sistem masih berjalan.

Sementara itu, untuk alur pada Mappi32 merupakan algoritma untuk menerima mengolah dan menampilkan *output*. Algoritma pemrograman sistem ini dimulai dengan inisialisasi perangkat, mencakup konfigurasi komunikasi serial, koneksi Wi-Fi, dan server HTTP yang akan menangani permintaan dari klien. Setelah itu, servo diinisialisasi pada posisi awal (105 derajat), serta mengatur variabel yang digunakan untuk menghitung jumlah buah matang dan mentah. Sistem kemudian masuk ke dalam *loop* utama, yang berfungsi memeriksa secara terus-menerus apakah ada data citra yang diterima dari ESP32-CAM. Ketika data citra tersedia, sistem akan mengalokasikan *buffer* memori untuk menyimpan citra tersebut. Jika *buffer* berhasil dialokasikan, gambar diproses untuk mendeteksi buah yang ada di dalamnya. Jika alokasi *buffer* gagal, sistem akan menampilkan pesan kegagalan alokasi memori dan kembali ke tahap memeriksa data citra. Setelah *buffer* tersedia dan gambar diterima, gambar kemudian diolah menggunakan segmentasi HSV untuk memisahkan warna dan menentukan jenis serta kematangan buah yang terdeteksi berdasarkan persentasenya (alur *predefined process* yang berisi alur Segmentasi HSV pada Sistem sebelumnya).

Berdasarkan hasil segmentasi, sistem melakukan percabangan untuk menentukan persentase warna mana yang lebih dominan. Jika persentase warna merah paling besar, maka sistem menambahkan hitungan stroberi matang dan menggerakkan servo ke sudut 5 derajat untuk membuka pintu bagi stroberi matang. Setelah servo bergerak, sistem memberikan jeda selama 3 detik sebelum servo kembali ke posisi awal di sudut 105 derajat. Hal yang sama dilakukan jika persentase warna kuning/oranye atau warna hijau lebih besar, di mana masing-masing buah memicu

gerakan servo yang berbeda sesuai jenis buahnya. Setelah proses deteksi dan gerakan servo selesai, sistem memeriksa apakah ada permintaan dari klien HTTP. Jika ada permintaan, server akan mengirimkan data jumlah stroberi matang, jeruk matang, dan buah mentah kepada klien dalam format JSON. Setelah permintaan klien ditangani, sistem melepaskan *buffer* yang telah digunakan untuk menyimpan data citra sebelumnya agar memori dapat digunakan kembali untuk gambar berikutnya. Proses ini terus berulang selama sistem masih berjalan, memastikan bahwa setiap gambar diproses secara *real-time*, dan permintaan klien selalu direspon dengan informasi terbaru yang dimiliki sistem.

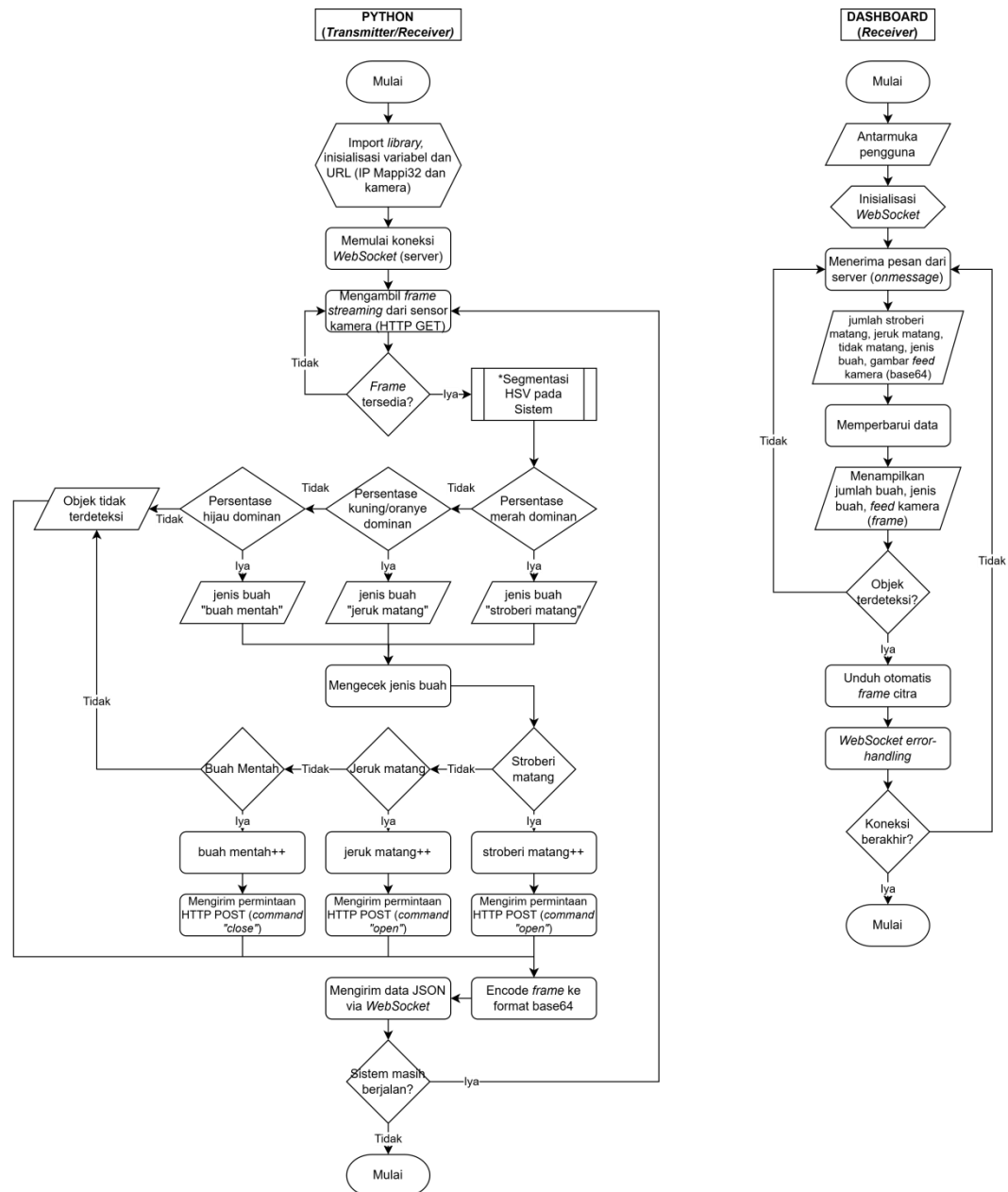


Gambar 3.10 Algoritma Program Kamera dan Mappi32 Model II

Selanjutnya adalah Gambar 3.10 yang berisi *flowchart* untuk algoritma program sensor kamera dan Mappi32 pada Model II. Pada alur sensor kamera menggambarkan proses kerja utama dari program sensor kamera yang terhubung ke jaringan Wi-Fi untuk melayani permintaan HTTP dari klien. Program dimulai dengan inisialisasi komponen-komponen penting, yaitu konfigurasi kamera, pin-pin yang digunakan, format gambar, resolusi citra, serta koneksi Wi-Fi dan server HTTP. Setelah inisialisasi, program akan masuk ke proses pengecekan koneksi Wi-Fi. Jika koneksi gagal, sistem akan terus mencoba untuk terhubung kembali dengan menampilkan status di monitor serial. Ketika koneksi berhasil, sensor kamera akan mendapatkan alamat IP yang kemudian dicetak ke monitor serial sebagai informasi kepada pengguna dan dapat diimplementasikan pada program Python. Selanjutnya, sistem akan memulai Web Server untuk melayani klien yang ingin mengakses *streaming video* secara terus-menerus melalui permintaan HTTP GET. Pada tahap ini, program akan terus-menerus mengecek apakah ada permintaan dari klien. Jika ada permintaan, dan sistem masih berjalan, server akan mengirimkan *frame* video yang diambil dari kamera ke klien dalam bentuk *streaming*. *Streaming* ini dilakukan secara berulang hingga ada sinyal bahwa sistem harus berhenti atau klien memutuskan koneksi. Proses ini terus berulang selama sistem masih berjalan.

Sementara itu, pada Mappi32, alur program difokuskan untuk mengontrol servo. Model algoritma pemrograman Mappi32 dimulai dengan inisialisasi sistem yang melibatkan serial monitor, motor servo, dan kredensial Wi-Fi. Sistem akan memeriksa koneksi Wi-Fi secara berulang hingga berhasil terhubung. Jika koneksi Wi-Fi berhasil, alamat IP dari perangkat akan diperoleh dan server HTTP akan dimulai dengan menyiapkan rute yang akan digunakan untuk menerima perintah dari klien. Namun, jika koneksi Wi-Fi gagal, sistem terus mencoba terhubung tanpa melanjutkan ke proses berikutnya. Setelah server berjalan, sistem memasuki fase di mana ia menunggu permintaan dari klien melalui metode HTTP POST. Permintaan ini berupa perintah "*open*" atau "*close*" yang akan memicu tindakan pada servo. Ketika sebuah perintah diterima, sistem pertama-tama akan memeriksa apakah perintah tersebut adalah "*open*" atau "*close*".

Untuk perintah "*open*", sistem akan memeriksa posisi servo saat ini. Jika servo berada di posisi 105 derajat (posisi awal), servo kemudian akan bergerak ke posisi 5 derajat, membuka jalur sepenuhnya untuk mengelola buah yang matang. Namun, jika servo tidak berada di posisi 105 derajat, sistem akan mengabaikan perintah tersebut dan menampilkan pesan kesalahan "*skipping command*" yang menandakan bahwa perintah tidak bisa dijalankan karena posisi servo tidak sesuai. Demikian pula, untuk perintah "*close*", sistem akan kembali memeriksa posisi servo. Jika servo berada di posisi awal 105 derajat, servo akan bergerak ke posisi 50 derajat untuk menutup sebagian jalur. Jika servo tidak berada di posisi tersebut, perintah juga akan diabaikan, dan pesan kesalahan akan ditampilkan. Setelah servo bergerak sesuai perintah, sistem akan menunggu selama 3 detik, memberi waktu agar pergerakan servo dapat selesai sebelum servo kembali ke posisi semula, yaitu 105 derajat. Setelah servo kembali ke posisi awal, sistem akan mengirimkan respon kepada klien yang menandakan bahwa perintah telah diproses dengan sukses. Seluruh proses ini dilakukan secara berulang dalam *loop* utama, di mana server terus memeriksa apakah ada permintaan baru dari klien selama sistem masih berjalan.



Gambar 3.11 Algoritma Program Python dan Dashboard Model II

Terakhir adalah Gambar 3.11 yang merupakan algoritma pemrograman Python dan *dashboard* pada Model II. Pada gambar sebelah kiri, menunjukkan alur program yang terjadi dalam pemrograman Python. *Flowchart* ini menggambarkan proses deteksi dan klasifikasi buah secara otomatis melalui pemrosesan citra, serta pengiriman data ke sistem lain melalui WebSocket. Proses dimulai dengan inisialisasi beberapa komponen penting, termasuk pustaka yang diperlukan, URL IP untuk mengakses Mappi32 dan sensor kamera, serta variabel yang digunakan

untuk mencatat jumlah stroberi matang, jeruk matang, dan buah mentah. Setelah inisialisasi, server WebSocket diaktifkan dan sistem memulai *loop* utama, di mana sistem terus mengambil *frame streaming* dari sensor kamera yang terhubung melalui HTTP GET. Jika *frame* berhasil diambil, *frame* tersebut kemudian diproses melalui langkah Segmentasi HSV pada Sistem (*predefined process*) sampai mendapatkan persentase warna.

Jika persentase dominasi warna merah lebih besar dari yang lain, buah dianggap sebagai stroberi matang. Jika oranye/kuning lebih dominan, maka itu jeruk matang. Sedangkan jika warna hijau lebih dominan, buah dianggap belum matang. Jika tidak ada warna yang memenuhi kriteria dominasi ini, objek dianggap tidak terdeteksi, dan sistem terus berulang untuk mendeteksi *frame* berikutnya tanpa menghitung jumlah atau mengontrol perangkat servo. Ketika sistem berhasil mendeteksi jenis buah tertentu, langkah selanjutnya adalah memeriksa jenis buah tersebut untuk mengontrol servo. Jika sudah, sistem akan menambah jumlah buah yang terdeteksi berdasarkan jenisnya: stroberi matang, jeruk matang, atau buah mentah. Pada saat yang sama, perintah HTTP POST dikirim ke Mappi32 untuk mengontrol servo, di mana servo akan membuka jika buah matang (stroberi atau jeruk) terdeteksi, dan menutup jika buah mentah terdeteksi.

Setelah itu, *frame* yang sudah diproses diubah ke format JPEG, kemudian di-*encode* ke base64 untuk dikirimkan bersama data JSON yang memuat informasi jumlah masing-masing jenis buah dan gambar *frame*. Data ini dikirim melalui WebSocket ke klien yang terhubung. Setelah proses pengiriman data selesai, sistem akan kembali ke awal *loop* untuk mengambil *frame* berikutnya. Proses ini akan terus berulang selama sistem masih aktif. Jika tidak ada *frame* yang tersedia, sistem akan menunggu sejenak sebelum mencoba mengambil *frame* baru.

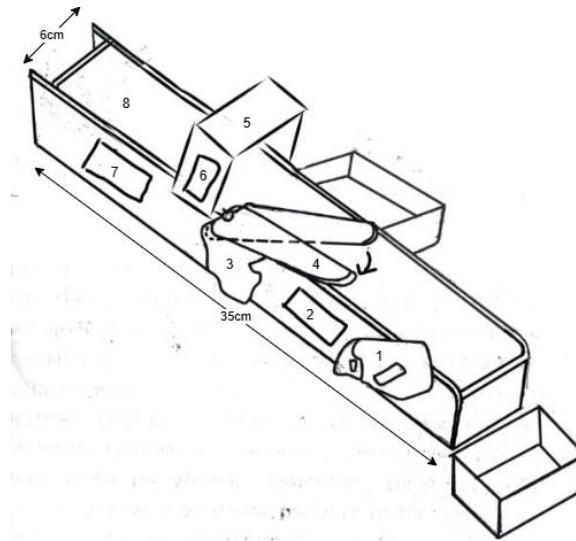
Sementara itu, pada gambar sebelah kanan menunjukkan algoritma pemrograman untuk *dashboard*. *Flowchart* ini menggambarkan alur dari sebuah sistem deteksi dan tampilan data secara *real-time* menggunakan WebSocket. Proses dimulai dengan memuat antarmuka pengguna di mana halaman web akan menampilkan berbagai elemen seperti jumlah stroberi matang, jumlah jeruk matang, jumlah buah yang tidak matang, serta *feed* kamera. Setelah itu, sistem melakukan

inisialisasi koneksi WebSocket yang berfungsi sebagai penghubung antara server dan klien untuk menerima data secara *real-time*. Ketika pesan dari server diterima melalui fungsi *onmessage()*, sistem akan meng-*update* data yang diterima seperti jumlah buah matang, jenis buah yang terdeteksi, dan *feed* gambar yang diambil dari kamera dalam bentuk data base64. Proses ini kemudian dilanjutkan dengan menampilkan data yang telah diperbarui ke dalam elemen-elemen HTML di halaman web, sehingga jumlah buah serta jenis buah yang terdeteksi dapat diperlihatkan kepada pengguna secara *real-time*.

Flowchart tersebut juga menunjukkan percabangan logika ketika memeriksa apakah ada objek (buah) yang terdeteksi atau tidak. Jika ada buah yang terdeteksi, sistem secara otomatis akan mengunduh *frame* gambar dari hasil deteksi tersebut dan menyimpannya sebagai file gambar. Jika tidak ada objek yang terdeteksi, sistem akan terus menunggu pesan berikutnya dari server tanpa menyimpan gambar. Sistem juga dilengkapi dengan mekanisme penanganan kesalahan (*error handling*) yang akan menangani setiap kesalahan yang terjadi dalam komunikasi WebSocket. Jika koneksi WebSocket berakhir, proses akan berhenti dan kembali ke awal. Selama koneksi aktif, sistem terus menerus melakukan siklus menerima data, memperbarui tampilan, dan melakukan pengecekan apakah ada objek yang terdeteksi.

3.4.5 Desain Sistem

Desain sistem berikut adalah desain mesin sortir yang digunakan untuk mengimplementasikan sistem pendeteksi kematangan. Mesin sortir hanya berupa mesin mini *conveyor* berukuran 35x6cm. Mesin sortir dibuat dapat berputar dalam tegangan 5V. Hal ini dilakukan agar mesin *conveyor* memiliki kompatibilitas dengan komponen elektronik yang lain. Selain itu, 5V dapat menjadi pilihan sebagai regulasi tegangan yang stabil dan efisien dalam konsumsi daya. Oleh karenanya, putaran yang dihasilkan tidak akan terlalu cepat ataupun terlalu lambat. Untuk desain mesin sortir secara garis besar dapat dilihat pada Gambar 3.12 di bawah ini.



Gambar 3.12 Sketsa Mesin Sortir (Desain Pribadi)

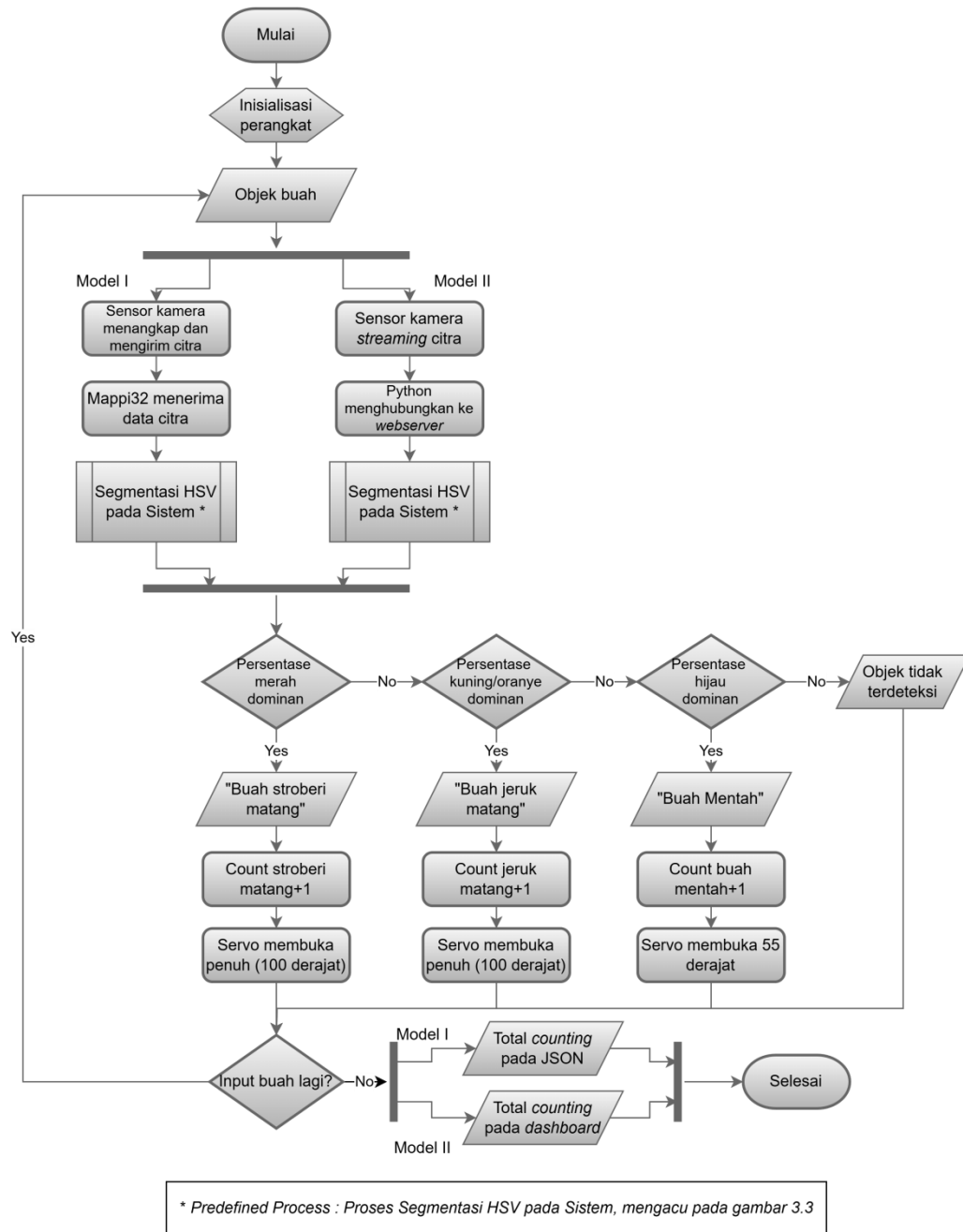
Pada Gambar 3.12 Sketsa Mesin Sortir, mesin sortir berbentuk *belt conveyor* sederhana. Selain itu, terdapat beberapa komponen yang ditandai dengan nomor 1 sampai dengan 8, di mana bagian tersebut dapat dilihat pada Tabel 3.4 berikut.

Tabel 3.4 Letak Komponen Mesin

No	Komponen	No	Komponen
1	Motor DC Gearbox kuning	5	Sensor kamera (di dalam penutup)
2	Modul LM2596	6	Mikrokontroler Mappi32
3	Motor Servo	7	Baterai eksternal dan saklar on-off
4	Lengan Pemisah	8	<i>Belt conveyor</i> dari kain kanvas hitam

3.4.6 *Flowchart* Cara Kerja Sistem

Pada algoritma pemrograman sebelumnya, telah diberikan informasi terkait alur rinci pembangun program pada setiap komponen. Akan tetapi, dalam bagian ini disajikan diagram alur yang merangkum keseluruhan alur program sebelumnya. Oleh sebab itu, *flowchart* ini hanya menyajikan cara kerja sistem deteksi kematangan buah yang diterapkan pada mesin sortir secara keseluruhan. *Flowchart* ini berisi proses baik pada Model I dan juga Model II. Adapun alur sistem dapat dilihat pada Gambar 3.13 *flowchart* berikut.



Gambar 3.13 Cara Kerja Sistem

Gambar 3.13 menunjukkan cara kerja sistem secara umum. Seperti dijelaskan sebelumnya, setelah perangkat berhasil diinisialisasikan dan objek disiapkan, langkah selanjutnya terdapat perbedaan antara Model I dan Model II. Perbedaan ditandai dengan pemisahan alur melalui lambang *fork* dan disatukan kembali dengan *join*. Pada Model I, sensor kamera menangkap citra dan mengirimkannya ke Mappi32 melalui pin serial. Kemudian, Mappi32 menerima data tersebut dan

mengolahnya mengikuti alur proses Segmentasi HSV pada Sistem (*predefined process*) menggunakan bahasa C++. Sementara itu, pada Model II, sensor kamera melakukan *streaming* citra melalui *web server*. Lalu komputer mengolah data tersebut melalui Python yang telah terhubung ke *web server*. Selanjutnya, pengolahan data di komputer juga mengikuti alur proses Segmentasi HSV pada Sistem menggunakan bahasa Python. Hasil akhir pada alur proses segmentasi adalah persentase warna.

Secara sistematis, setelah didapatkan persentase warna, maka diklasifikasikan menurut dominansinya. Jika persentase merah lebih dominan, artinya buah stroberi matang, hasil *counting* stroberi bertambah, dan servo bergerak membuka jalur secara penuh (didefinisikan 100 derajat). Jika persentase kuning/oranye yang lebih dominan, maka buah jeruk matang, hasil *counting* jeruk bertambah, dan servo membuka jalur secara penuh (100 derajat). Tetapi, jika persentase hijau yang lebih dominan, maka buah mentah, hasil *counting* buah mentah bertambah, dan servo membuka jalur separuh (didefinisikan 55 derajat). Hasil *counting* bertambah setiap mendeteksi objek. Apabila tidak ada satupun kondisi tersebut terpenuhi saat interval pengolahan, artinya tidak ada objek yang terdeteksi. Sistem ini terus berjalan selama pengguna masih meletakkan objek buah pada *conveyor*. Jika telah selesai, total keseluruhan hasil *counting* dapat dilihat dalam *endpoint* JSON (Model I) atau pada *dashboard* (Model II).

3.5 Pengujian Sistem

Pengujian *prototype* sistem dilakukan untuk mengetahui tingkat dari keberhasilan sistem yang dirancang. Tahapan pengujian terbagi menjadi dua, yaitu pengujian komponen dan pengujian pada keseluruhan sistem. Adapun kedua pengujian yang dilakukan sebagaimana dijelaskan pada sub bab berikutnya.

3.5.1 Pengujian Komponen

Pengujian komponen digunakan untuk memastikan kinerja dari komponen telah berhasil berfungsi dengan baik. Hal ini dilakukan agar mengetahui eror pada

komponen-komponen pembangun sistem secara keseluruhan.

3.5.2 Pengujian Keseluruhan Sistem

Pengujian ini lebih spesifik dibandingkan pada pengujian pertama. Pada pengujian ini, berbagai parameter yang memengaruhi berjalannya sistem pun menjadi pertimbangan. Adapun parameter yang diujikan adalah jarak kamera ke objek buah, waktu proses, variabilitas waktu dan tingkat keberhasilan sistem. Pada skenario pengujian kalibrasi jarak kamera ke buah, dilihat pada jarak berapakah sensor kamera dapat mengenali warna objek dengan baik. Pengambilan citra oleh kamera dilakukan dengan meminimalisir kemungkinan adanya objek lain di sekitarnya yang tertangkap.

Selanjutnya, untuk pengujian waktu proses, parameter waktu yang dihitung adalah pemrosesan klasifikasi kematangan, pemunculan data hasil hitung, beserta respon lengan pemisah. Pengujian dari sisi waktu dilakukan pada beberapa sampel buah stroberi dan jeruk. Pengujian ini dilakukan untuk mengetahui seberapa lama waktu sistem dalam merespon/memproses sampel buah tersebut. Kemudian sistem juga dihitung variabilitas waktu dan tingkat keberhasilannya berdasarkan pengujian sebelumnya. Ini dilakukan untuk melihat apakah proses penyortiran sudah berhasil atau belum, dan apakah sistem cukup stabil ataukah tidak.

Adapun parameter yang dihitung keberhasilannya adalah kondisi buah, kondisi servo dan hasil *counting*. Apabila kondisi buah matang, lengan pemisah terbuka 100 derajat. Sementara itu, jika buah mentah, maka lengan pemisah terbuka 55 derajat. Hasil *counting* yang dimaksud adalah apakah banyak buah matang dan mentah yang bertambah sesuai dengan kondisi buah atau tidak. Dari beberapa skenario pengujian yang ada, diambil kesimpulan berupa tingkat keberhasilan sistem penyortiran berbasis mikrokontroler tersebut. Apabila masih menghasilkan *error rate* yang besar, dilakukan perbaikan menggunakan Model II. Pengujian Model II juga menggunakan alur pengujian yang sama, tetapi diharapkan hasilnya berbeda. Ini berguna untuk membandingkan kinerja antara Model I dan Model II.

V. SIMPULAN DAN SARAN

5.1 Simpulan

Berdasarkan hasil dari penelitian yang telah dilakukan, adapun kesimpulan yang diperoleh yaitu :

1. Metode visualisasi histogram serta kombinasi *adaptive* dan *Otsu threshold* dengan *Gaussian filter* berhasil digunakan untuk menentukan nilai *threshold hue* pada stroberi matang (rentang T1 = 0 hingga T2 = 10; dan T3 = 155 hingga T4 = 180), jeruk matang (rentang T1 = 10 hingga T2 = 30), dan buah mentah (rentang T1 = 35 hingga T2 = 90), dengan penyesuaian *trackbar* untuk *saturation* dan *value* pada nilai 100 (*lower*) dan 255 (*upper*). Metode ini terbukti mampu menghasilkan segmentasi yang baik dalam mendeteksi kematangan buah berdasarkan warna.
2. Penerapan segmentasi citra HSV pada objek buah dengan resolusi terendah QQVGA (160x120px) dan format RGB565 oleh Mappi32 terbukti dapat bekerja dengan baik selama sebagian waktu pengujian dan menghasilkan *output* yang sesuai. Akan tetapi, karena adanya *overheating*, terjadi kegagalan deteksi pada sebagian waktu lainnya. Penggunaan yang berkepanjangan dapat mengakibatkan *overheating* dan memicu penurunan kinerja, termasuk masalah latensi pada pelepasan *buffer* yang menyebabkan kegagalan pada alokasi memori.
3. Penggunaan pemrograman Python pada Model II untuk membagi beban kerja dari Mappi32 menghasilkan peningkatan yang cukup baik. Rata-rata waktu pemrosesan citra berkurang dari 1,8 detik menjadi 0,78 detik; waktu tampilan hasil *counting* berkurang dari 0,58 detik menjadi 0,33 detik; respons lengan pemisah berkurang dari 0,39 detik menjadi 0,19 detik; dan variabilitas waktu

berkurang dari 0,94 detik menjadi 0,77 detik. Tingkat keberhasilan sistem juga meningkat dari 56,25% menjadi 100%.

4. *Dashboard* hasil *counting* berhasil dibuat menggunakan HTML, CSS dan JavaScript, serta mampu menampilkan data secara *real-time* menggunakan protokol *WebSocket*. *WebSocket* menciptakan sistem yang dapat saling berkomunikasi untuk mengontrol informasi *dashboard* secara *real-time*, seperti jumlah buah matang, jumlah buah mentah, jenis buah, dan *frame* citra buah yang dideteksi.

5.2 Saran

Dari proses dari penelitian yang telah dilakukan, terdapat beberapa kekurangan dari sistem pendeteksi kematangan buah. Oleh sebab itu, terdapat beberapa saran pengembangan sistem lebih lanjut, yaitu:

1. Pengembangan sistem pendinginan dan manajemen daya untuk mengatasi masalah *overheating* pada Mappi32 dan komponen lainnya, seperti dengan pemasangan *heat sink*, optimasi *sleep mode*, ataupun penggunaan bahan pendukung *hardware* yang dapat membantu dalam penyebaran panas keluar dari sistem.
2. Penambahan fitur pengolahan citra lainnya, seperti menggunakan *machine learning* untuk menangani buah dengan pola warna yang sulit dideteksi hanya melalui segmentasi citra HSV. Selain itu, dapat diperluas variasi nilai *threshold* untuk mendeteksi kematangan aneka jenis buah selain yang berwarna merah, kuning, hijau, dan oranye, dengan kondisi lainnya, seperti buah yang sudah busuk, sehingga cakupan deteksi menjadi lebih komprehensif.
3. Penambahan fitur input manual yang memungkinkan pengguna memilih jenis buah tertentu yang ingin disortir sesuai dengan kebutuhannya.

DAFTAR PUSTAKA

- [1] Pusat Data dan Sistem Informasi Pertanian, *Statistik pertanian = Agricultural statistics, 2014*. Ragunan, Jakarta Selatan: Pusat Data dan Sistem Informasi Pertanian, Kementerian Pertanian, 2014, hlm. 93-249
- [2] T. A. Teka, "Analysis Of The Effect Of Maturity Stage On The Postharvest," *IRJPAS*, vol. 3, no. 5, hlm. 180–186, 2013.
- [3] Awanis, M. Syarif, R. Qomariah, S. Lesmayati, dan M. Amin, *Penanganan Pascapanen dan Pemasaran Hasil Pertanian*. Banjarbaru: Balai Pengkajian Teknologi Pertanian Kalimantan Selatan, 2022. hlm. 3-8. Diakses: 3 September 2024. [Daring]. Tersedia pada: <https://repository.pertanian.go.id/>
- [4] Humkoler UNPAR, "Mahasiswa Informatika UNPAR Buat Program Sortir Buah," Universitas Katolik Parahyangan. 2023. Diakses: 12 Desember 2023. [Daring]. Tersedia pada: <https://unpar.ac.id/mahasiswa-informatika-unpar-buat-program-sortir-buah-berdasarkan-warna-atau-ukuran/>
- [5] R. W. Fenia, "Introducing New Version of Mappi32," *KMTek*. 2023. Diakses: 12 April 2024. [Daring]. Tersedia pada: <https://www.kmtech.id/post/introducing-new-version-of-mappi32>
- [6] Irfan, S. Widayati, dan I. P. Wardhani, "Analisa Segmentasi Warna HSV pada Citra Video dengan Metode Threshold," *SeNTIK*, vol. 4, no. 1, hlm. 339–345, Sep 2020.
- [7] P. Chełpiński, I. Ochmian, dan P. Forczmański, "Sweet Cherry Skin Colour Measurement as an Non-Destructive Indicator of Fruit Maturity," *Acta Universitatis Cibiniensis Series E Food Technology*, vol. 23, no. 2, hlm. 157–166, Des 2019, doi: 10.2478/aucft-2019-0019.
- [8] A. B. Pulungan dan Z. Nafis, "Rancangan Alat Pendeteksi Benda dengan Berdasarkan Warna, Bentuk, dan Ukuran dengan Webcam," *Jurnal Teknik Elektro Indonesia*, vol. 2, no. 1, hlm. 49–54, Feb 2021, doi:

- 10.24036/jtein.v2i1.111.
- [9] Muh. R. Kurniawan, "Implementasi Object Tracking Berbasis Filtering Warna Pada Sensor Kamera Pixy CMUcam 5," *Jurnal Fokus Elektroda*, vol. 3, no. 3, Agu 2018, doi: 10.33772/jfe.v3i3.6586.
- [10] A. C. Saputra dan E. D. Oktaviyani, "Rancang Bangun Sistem Deteksi Kematangan Buah Kelapa Sawit Berdasarkan Deteksi Warna Menggunakan Algoritma K-NN," *Jurnal Teknologi Informasi*, vol. 7, no. 2, hlm. 222–229, Agu 2023, doi: 10.47111/jti.v7i2.9232.
- [11] Trivusi, "Mengenal Model Ruang Warna pada Pengolahan Citra," Trivusi. 2022. Diakses: 12 April 2024. [Daring]. Tersedia pada: <https://www.trivusi.web.id/2022/11/model-ruang-warna.html>
- [12] S. Kurniawan Dwi dan T. Junaidi, "Implementasi Algoritma K-Nearest Neighbor Dengan Metode Hue Saturation Value Untuk Pendeteksi Kematangan Buah Jambu," *Smart Comp*, vol. 11, no. 3, Jul 2022, doi: 10.30591/smartcomp.v11i3.3908.
- [13] S. Eko Wahyudi, "Teori Warna (Multimedia #4)," Universitas Ciputra Creating World Class Entrepreneurs, 2020. Diakses: 10 Oktober 2024. [Daring]. Tersedia pada: <https://informatika.ciputra.ac.id/2020/02/teori-warna-multimedia-4/>
- [14] R. Jain dan P. K. Johari, "An improved approach of CBIR using Color based HSV quantization and shape based edge detection algorithm," dalam *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology*, Bangalore: IEEE, Mei 2016, hlm. 1970–1975. doi: 10.1109/RTEICT.2016.7808181.
- [15] A. Kadir dan A. Susanto, *Teori dan Aplikasi Pengolahan Citra*. Yogyakarta: Penerbit Andi, 2013.
- [16] I. S. Areni, I. Amirullah, dan N. Arifin, "Klasifikasi Kematangan Stroberi Berbasis Segmentasi Warna dengan Metode HSV," *Jurnal Penelitian Enjiniring*, vol. 23, no. 2, hlm. 113–116, Nov 2019, doi: 10.25042/jpe.112019.03.
- [17] D. I. Muhammad, E. Ermatita, dan N. Falih, "Penggunaan K-Nearest Neighbor (KNN) untuk Mengklasifikasi Citra Belimbing Berdasarkan Fitur

- Warna,” *Informatik: Jurnal Ilmu Komputer*, vol. 17, no. 1, hlm. 9–16, Apr 2021, doi: 10.52958/iftk.v17i1.2132.
- [18] A. Ambarwati, R. Passarella, dan Sutarno, “Segmentasi Citra Digital Menggunakan Thresholding Otsu untuk Analisa Perbandingan Deteksi Tepi,” *Prosiding Annual Research*, vol. 2, no. 1, hlm. 216-226, Des 2016.
- [19] R. C. Gonzales dan R. E. Woods, *Digital Image Processing*. New Jersey: Pearson Prentice Hall, 2008. Diakses: 10 September 2024. [Daring]. Tersedia pada: <https://archive.org/details/digitalimageproc0003gonz>
- [20] W. Mellyssa, M. Misriana, S. Suryati, dan M. Milawarni, “Perbandingan Penggunaan Metode Otsu Thresholding dan Adaptive Thresholding pada Proses Binerisasi Sistem Dokumentasi Buku Tugas Akhir,” *Proceeding Seminar Nasional Politeknik Negeri Lhokseumawe*, vol. 4, no. 1, hlm. 49–54, Nov 2020.
- [21] W. Mellyssa, M. Misriana, S. Suryati, dan M. Milawarni, “Penerapan Metode Otsu Thresholding dan Koefisien Korelasi pada Pendeteksian Judul Buku Tugas Akhir Judul Artikel,” *Jurnal Elektro dan Telekomunikasi Terapan*, vol. 8, no. 1, hlm. 925–933, Jul 2021, doi: 10.25124/jett.v8i1.3686.
- [22] A. Wedianto, H. Latipa Sari, dan Y. Suzantri H, “Analisa Perbandingan Metode Filter Gaussian, Mean dan Median terhadap Reduksi Noise,” *Jurnal Media Infotama*, vol. 12, no. 1, hlm. 21–30, 2016.
- [23] M. R. Khilmawan dan A. A. Riadi, “Implementasi Pengurangan Noise Pada Citra Tulang Menggunakan Metode Median Filter Dan Gaussian Filter,” *JUPI*, vol. 3, no. 2, hlm. 116–121, Des 2018, doi: 10.29100/jipi.v3i2.865.
- [24] Y. M. Poysancin dan A. N. Utomo, “Rancang Bangun Sistem Deteksi Wajah Dengan Metode Viola-Jones Untuk Mengidentifikasi Identitas Seseorang,” *Incomtech*, vol. 8, no. 2, hlm. 69–76, Desember 2019.
- [25] Team, “About OpenCV,” OpenCV, 2023. Diakses: 28 Desember 2023. [Daring]. Tersedia pada: <https://opencv.org/about/>
- [26] Sutiono, “Metode Prototype: Pengertian, Kekurangan, dan Kelebihan,” DosenIT.com, 2023. Diakses: 1 September 2024. [Daring]. Tersedia pada: <https://dosenit.com/software/metode-prototype>

- [27] F. Ilhami, P. Sokibi, dan A. Amroni, “Perancangan Dan Implementasi Prototype Kontrol Peralatan Elektronik Berbasis Internet Of Things Menggunakan NodeMCU,” *Jurnal Digit*, vol. 9, no. 2, hlm. 143–155, Nov 2019, doi: 10.51920/jd.v9i2.115.
- [28] V. Pimentel dan B. G. Nickerson, “Communicating and Displaying Real-Time Data with WebSocket,” *IEEE Internet Computing*, vol. 16, no. 4, hlm. 45–53, Jul 2012, doi: 10.1109/MIC.2012.64.
- [29] T. F. P. Siallagan dan F. Ramadhan, “Machine Learning Tingkat Kematangan Buah Nanas Subang Berbasis Internet Of Things Menggunakan Metode K-Means Pada Platform Thingspeak,” *Jurnal Teknologi Informasi dan Komunikasi*, vol. 14, no. 2, hlm. 20–30, Oktober 2021.
- [30] Y. B. E. Purba, N. F. Saragih, A. P. Silalahi, S. Sitepu, dan A. Gea, “Perancangan Alat Pendeteksi Kematangan Buah Nanas Dengan Menggunakan Mikrokontroler Dengan Metode Convolutional Neural Network (CNN),” *Methotika*, vol. 2, no. 1, hlm. 13–21, Apr 2022.
- [31] A. K. Permana dan A. Rachmawan, “Studi Komparasi Platform Open-Source Internet of Things,” *Jurnal Teknologi dan Manajemen*, vol. 21, no. 1, hlm. 43–48, Mar 2023, doi: 10.52330/jtm.v21i1.38.
- [32] W. Triyoga, Y. Ageng Suryo, dan R. Puji Astutik, “Rancang Sistem Keamanan Pada Laboratorium Berbasis Internet Of Things Menggunakan Rowl Sebagai Pendeteksi Gerakan,” *Journal of Comprehensive Science*, vol. 2, no. 6, hlm. 1593–1606, Jun 2023, doi: 10.59188/jcs.v2i6.381.
- [33] OmniVision, “OV2640 2 MPixel Product Brief.” DatasheetBank (Online), Mei 2006. Diakses: 3 September 2024. [Daring]. Tersedia pada: www.datasheetbank.com/en/OV2640-ETC
- [34] Isaac, “ESP32-CAM: Apa yang harus anda ketahui tentang modul ini,” Perangkat Keras Pound, 2021. Diakses: 12 Mei 2023. [Daring]. Tersedia pada: <https://www.hwlibre.com/id/kamera-esp32/>
- [35] Isaac, “Servo SG90: Semua yang perlu anda ketahui tentang motor listrik kecil ini,” Perangkat Keras Pound, 2021. Diakses: 12 Mei 2023. [Daring]. Tersedia pada: <https://www.hwlibre.com/id/servo-sg90/>

- [36] A. I. Salim, Y. Saragih, dan R. Hidayat, “Implementasi Motor Servo SG 90 Sebagai Penggerak Mekanik Pada E. I. Helper (Electronics Integration Helmet Wiper),” *Electro Luceat*, vol. 6, no. 2, hlm. 236–244, Nov 2020, doi: 10.32531/jelekn.v6i2.256.
- [37] Onsemi, “3.0 A, Step-Down Switching Regulator L2596.” Semiconductor Components Industries, 2008. Diakses: 3 September 2024. [Daring]. Tersedia pada: www.onsemi.com
- [38] T. Septyawan, “Rancang Bangun Keamanan Ruangan Pribadi dengan Arduino dan SMS Gateway,” *Portal Data*, vol. 2, no. 3, hlm. 1–11, Apr 2022.
- [39] D. Kho, “Pengertian Motor DC dan Prinsip Kerjanya,” *Teknik Elektronika*, 2023. Diakses: 14 Desember 2023. [Daring]. Tersedia pada: <https://teknikelektronika.com/pengertian-motor-dc-prinsip-kerja-dc-motor/>
- [40] A. Feriska dan D. Triyanto, “Rancang Bangun Penjemur Dan Pengering Pakaian Otomatis Berbasis Mikrokontroler,” *Journal Coding Sistem Komputer Untan*, vol. 05, no. 2, hlm. 67–76, 2017
- [41] Y. Susanto, M. Tarigan, dan Yulhendri, “Pengukuran Dan Pendataan Zat Cair Toluene Dengan Akses Rfid Berbasis Nodemcu Esp8266 Yang Termonitor Melalui Web,” *SINTAMA*, vol. 2, no. 3, hlm. 382–395, 2022.
- [42] A. S. Wibowo dkk., *Statistik Hortikultura 2023*, vol. 5. Jakarta: Badan Pusat Statistik Indonesia, 2024. Diakses: 3 September 2024. [Daring]. Tersedia pada: <https://webapi.bps.go.id/>
- [43] M. Maulida dan N. F. Mustamin, “Pengembangan Sistem Pakan Budidaya Ikan Keramba Dan Jaring Apung Dengan Pemanfaatan Sensor Ultrasonik Hcsr04 Dan Modul Komunikasi LoRa,” *Infotech Journal*, vol. 8, no. 2, hlm. 101–105, Desember 2022.
- [44] S. Muharom, S. Asnawi, dan A. Bachri, “Robot Pengikut Target Berdasarkan Bentuk dan Warna Menggunakan Metode HSV Untuk Aplikasi Assistant Robot,” *JE-Unisla*, vol. 6, no. 1, hlm. 415–423, Mar 2021.
- [45] M. Andri, Jasmir, dan W. Riyadi, “Rancang Bangun Prototype Sortir Buah Kelapa Sawit Berdasarkan Tingkat Kematangan Berbasis Arduino Uno,”

- Jurnal Informatika dan Rekayasa Komputer*, vol. 3, no. 1, hlm. 501–510, Apr 2023, doi: 10.33998/jakakom.2023.3.1.812.
- [46] G. Khekare, S. Verma, S. Sur, R. Haldkar, dan P. Moon, “Design of Fruit Segregation and Packaging Machine,” *2020 International Conference on ComPE*, hlm. 709–714, 2020, doi: 10.1109/ComPE49325.2020.9199986.
- [47] G. I. E. Panie dan A. B. Mutiara, “Development of Robotic Arm for Color Based Goods Sorter in Factory Using TCS3200 Sensor with a Web-Based Monitoring System,” dalam *2018 Third International Conference on Informatics and Computing (ICIC)*, Palembang, Indonesia: IEEE, Okt 2018, hlm. 1–6. doi: 10.1109/IAC.2018.8780461.
- [48] D. Devasagayam, A. Shende, A. Gonsalves, K. Padalkar, dan V. Rodrigues, “Design and Development of Fruit Sorting and Packaging Machine,” *International Conference on Industrial Engineering*, vol. 2, hlm. 875–879, 2019.
- [49] I. Lita, D. A. Visan, L. M. Ionescu, dan A. G. Mazare, “Color-Based Sorting System for Agriculture Applications,” dalam *2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, Pitesti, Romania: IEEE, Jun 2019, hlm. 1–4. doi: 10.1109/ECAI46879.2019.9041923.
- [50] A. E. Elwakeel *dkk.*, “Designing, Optimizing, and Validating a Low-Cost, Multi-Purpose, Automatic System-Based RGB Color Sensor for Sorting Fruits,” *Agriculture*, vol. 13, no. 9, hlm. 1824, Sep 2023, doi: 10.3390/agriculture13091824.