PENGENDALIAN ROBOT ARM OPENMANIPULATOR UNTUK PENYORTIRAN BUAH JERUK MENGGUNAKAN *MACHINE LEARNING*

TUGAS AKHIR

SHELLY MARGARETH

NIM : 312210024

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI DAN DESAIN

UNIVERSITAS MA CHUNG

MALANG

2025

LEMBAR PENGESAHAN

TUGAS AKHIR

PENGENDALIAN ROBOT ARM OPENMANIPULATOR UNTUK PENYORTIRAN BUAH JERUK MENGGUNAKAN *MACHINE LEARNING*

Oleh:

SHELLY MARGARETH

NIM. 312210024

Dari:

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI DAN DESAIN

UNIVERSITAS MA CHUNG

Dosen Pembimbing I,

Prof. Dr.Eng. Romy Budhi, ST., MT., M.Pd.

NIP. 20070035

Dosen Pembimbing II,

Mochamad Subianto, S.Kom., M.Cs.

NIP. 20100002

Dekan Fakultas Teknologi dan Desain,

Prof. Dr.Eng. Romy Budhi, ST., MT., M.Pd.

NIP. 20070035

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat dan kasih karunia-Nya selama 4 tahun masa perkuliahan sehingga penulis dapat menyelesaikan laporan Tugas Akhir dengan dengan judul **" PENGENDALIAN ROBOT ARM OPENMANIPULATOR UNTUK PENYORTIRAN BUAH JERUK MENGGUNAKAN *MACHINE LEARNING* "** sebagai salah satu prasyarat untuk mendapatkan gelar Sarjana Komputer di Universitas Ma Chung. Melalui pengalaman ini, penulis memperoleh banyak pembelajaran berharga serta bantuan dari banyak pihak

Penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Tuhan yang Maha Esa atas berkat dan kasih karunia-Nya.
2. Prof. Dr.Eng. Romy Budhi, ST., MT., M.Pd, dan selaku dosen pembimbing I , sekeligus dekan Fakultas Teknologi dan Desain atas bimbingan, arahan, dan dukungan yang telah diberikan selama pelaksanaan Tugas Akhir hingga penyusunan laporan ini.
3. Bapak Mochamad Subianto, S.Kom., M.Cs. selaku dosen pembimbing II, atas bimbingan, arahan, dan dukungan yang telah diberikan selama pelaksanaan Tugas Akhir hingga penyusunan laporan ini.
4. Bapak/Ibu dosen Program Studi Teknik Informatika Universitas Ma Chung, atas ilmu dan wawasan yang telah dibekalkan selama perkuliahan;
5. Keluarga dan teman-teman yang senantiasa memberikan semangat, doa, dan dukungan selama proses penelitian hingga penyusunan laporan Tugas Akhir ini.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, penulis sangat mengharapkan masukan, saran, dan kritik yang membangun dari para pembaca. Akhir kata, penulis berharap laporan ini bermanfaat dan berguna untuk semua pihak yang membutuhkan.

Malang, 05 Januari 2026

Shelly Margareth

PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan dibawah ini :

Nama : Shelly Margareth

NIM : 312210034

Program Studi : Teknik Informatika

Perguruan Tinggi : Universitas Ma Chung

Dengan ini menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "PENGENDALIAN ROBOT *ARM* OPENMANIPULATOR UNTUK PENYORTIRAN BUAH JERUK MENGGUNAKAN *MACHINE LEARNING*" adalah asli (orisinil) atau tidak plagiat dan benar hasil karya intelektual mandiri, dan belum pernah diterbitkan/dipublikasikan dimanapun dan dalam bentuk apapun.

Surat penyataan ini saya buat dengan sebenar-benarnya dengan kesaran sendiri dan tanpa ada paksaan dari pihak manapun. Apabila dikemudian hari diduga kuat ada ketidaksesuai antara fakta dengan dokumen pernyataan ini, saya bersedia diproses oleh Universitas Ma Chung, dengan sanksi terberat berupa pembatalan kelulusan atau pencabutan sarjana.

Malang, 05 Januari 2026,

Shelly Margareth

NIM 312210024

PENGENDALIAN ROBOT ARM OPENMANIPULATOR UNTUK PENYORTIRAN BUAH JERUK MENGGUNAKAN *MACHINE LEARNING*

Shelly Margareth, Romy Budhi, Mochamad Subianto.

Universitas Ma Chung

Abstrak

Proses penyortiran buah jeruk di tingkat produsen saat ini umumnya masih dilakukan secara manual yang memiliki keterbatasan dari segi efisiensi waktu dan konsistensi, serta rentan terhadap *human error* akibat faktor kelelahan. Penelitian ini bertujuan untuk merancang sistem penyortiran otomatis yang mengintegrasikan metode *Deep Learning* dengan lengan robot *OpenManipulator-X*. Sistem dibangun menggunakan komponen utama berupa lengan robot 4-DOF, kamera sebagai sensor visual, dan NVIDIA Jetson Nano sebagai unit pemroses data.

Pada sisi perangkat lunak, algoritma *Convolutional Neural Network* (CNN) dengan arsitektur MobileNetV2 digunakan untuk mengklasifikasikan kualitas jeruk berdasarkan fitur warna dan tekstur secara *real-time*. Hasil klasifikasi tersebut dikonversi menjadi perintah gerak robot menggunakan prinsip *Inverse Kinematics* untuk memindahkan buah ke wadah yang sesuai.

Berdasarkan hasil pengujian, model ini mampu menghasilkan akurasi klasifikasi sebesar 96%. Secara mekanis, sistem berhasil melakukan penyortiran dengan tingkat keberhasilan 100% dan rata-rata waktu proses 31 detik per buah. Penelitian ini diharapkan dapat menjadi solusi alternatif dalam penerapan teknologi otomatisasi di bidang pertanian.

Kata Kunci: OpenManipulator, Machine Learning, Penyortiran Jeruk, *Convolutional Neural Network*, NVIDIA Jetson Nano.

IMPLEMENTATION OF OPENMANIPULATOR ROBOTIC ARM FOR ORANGE SORTING USING MACHINE LEARNING

Shelly Margareth, Romy Budhi, Mochamad Subianto

Universitas Ma Chung

Abstract

Proses Currently, the orange sorting process at the producer level is largely performed manually. This method has limitations in terms of time efficiency, consistency and is prone to human error due to fatigue. This study aims to design an automated sorting system integrating Deep Learning methods with the OpenManipulator-X robotic arm. The system is built using main components consisting of a 4-DOF robotic arm, a camera as a visual sensor, and an NVIDIA Jetson Nano as the data processing unit.

On the software side, a Convolutional Neural Network (CNN) algorithm with MobileNetV2 architecture is used to classify orange quality based on color and texture features in real-time. The classification results are converted into robot movement commands using Inverse Kinematics principles to move the fruit to the appropriate container.

Based on test results, the model achieved a classification accuracy of 96%. Mechanically, the system successfully performed sorting with a success rate of 100% and an average processing time of 31 seconds per fruit. This study is expected to serve as an alternative solution for the application of automation technology in agriculture.

Keywords: OpenManipulator, Machine Learning, Orange Sorting, Convolutional Neural Network, NVIDIA Jetson Nano.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi robotika dan otomasi telah memberikan dampak signifikan dalam berbagai aspek kehidupan, khususnya pada sektor industri pangan. Salah satu proses penting dalam produksi adalah penyortiran buah, yang hingga kini umumnya masih dilakukan secara manual oleh tenaga kerja manusia. Metode manual memiliki sejumlah keterbatasan, seperti keterlambatan, kelelahan, serta potensi kesalahan dalam menentukan kualitas buah. Kondisi ini berakibat pada menurunnya efisiensi sekaligus kualitas hasil produksi (Inuwa et al., 2025).

Selain itu, kualitas buah jeruk sangat memengaruhi harga jual serta tingkat penerimaan konsumen. Konsumen cenderung memilih jeruk dengan kondisi sehat, dan bebas dari cacat. Oleh karena itu, dalam proses penyortiran sangat penting untuk memastikan hanya jeruk dengan kualitas baik yang didistribusikan ke pasar. Jika jeruk dengan kualitas rendah ikut terjual dapat menurunkan kepercayaan konsumen, memengaruhi citra produk, serta berdampak pada turunnya daya saing industri. Dalam industri pertanian modern, penyortiran buah secara otomatis penting dilakukan untuk menjaga konsistensi dan kualitas produk yang diterima konsumen. Penampilan buah khususnya warna kulit yang cerah dan bebas dari noda sangat menentukan keputusan pembelian jeruk oleh konsumen, karena aspek visual menjadi indikator utama kesegaran dan mutu buah (Hj & Rostiati, 2015).

Untuk mengatasi permasalahan penyortiran jeruk secara manual yang lambat, tidak konsisten, dan rentan kesalahan, penerapan teknologi robotika dengan AI menjadi solusi yang menjanjikan. Contohnya, pada penelitian berjudul *Computer Vision and Machine learning Based Control for a 6 Degree of Freedom Robotic Arm* (De Assis et al., 2025), digunakan pendekatan *supervised learning* dengan metode *Artificial Neural Network* (ANN) untuk mengendalikan lengan robot berdasarkan gerakan manusia. Model ANN dilatih menggunakan data hasil tangkapan kamera melalui *framework* Mediapipe, yang mendeteksi titik-titik

penting pada tubuh operator seperti bahu, jari telunjuk, dan ibu jari. Hasil penelitian menunjukkan bahwa model berhasil mengklasifikasi sudut setiap sendi (*joint angle*) pada keenam servo motor lengan robot dengan tingkat akurasi mencapai 82,03%. Klasifikasi ini memungkinkan lengan robot meniru pergerakan tangan manusia secara *real-time* pada lingkungan simulasi Gazebo menggunakan ROS.

Dalam penelitian ini digunakan lengan robot *Arm* OpenManipulator-X, yaitu robot *Arm open-source* yang fleksibel dan dirancang untuk kebutuhan pengembangan, pendidikan, serta penelitian. Sistem kendali utama menggunakan OpenCR (*Open-source Control Robot*) yang memiliki prosesor kompeten, dukungan komunikasi lengkap, serta kompatibel dengan berbagai sistem robotika. Penggunaan OpenCR pada robot *Arm* telah banyak diterapkan dalam berbagai penelitian, misalnya oleh (Kashani et al., 2025) yang meneliti tentang pengembangan sistem kendali cerdas untuk robot *Arm* OpenManipulator-X, fokus utamanya adalah pada peningkatan presisi Gerakan dan kestabilan sistem dengan cara menggabungkan metode control klasik *Fixed-Time Sliding Mode Control* (FTSM) dan algoritma pembelajaran penguatan *Deep Deterministic Policy Gradient* (DDPG) agar dapat menghadapi kondisi lingkungan yang berubah-ubah. (Automation & Automation, 2024) misalnya, membantu individu yang berhasil bertahan dari stroke biasanya membutuhkan bantuan untuk melakukan tugas-tugas dasar. Alat bantu robotik, seperti *Arm* OpenManipulator melakukan berbagai aktivitas dengan cara mengambil dan menyerahkan benda kepada pasien. Dalam proyek ini, sarung tangan berbasis kendali gerakan (*gesture control*) dirancang untuk meningkatkan proses *handover* dari robot ke pengguna. Sarung tangan dilengkapi dua sensor MPU-6050 dan dua sensor lentur (*flex* sensor) untuk mendeteksi orientasi tangan dan tekukan jari. Data dari sarung tangan dikirim ke *Arm* Manipulator guna menggerakkan manipulator dan mengoperasikan *gripper* secara terpisah. Pengujian dilakukan dengan lima jenis benda sebanyak 20 kali masing-masing. Hasil menunjukkan rata-rata waktu penyerahan 7,76 detik dengan tingkat kesalahan 18,03%. Perbedaan waktu antar benda menunjukkan pengaruh dari lingkungan pengujian, pengguna yang sama, dan keterbatasan perangkat keras.

Penelitian-penelitian terdahulu menunjukkan bahwa penggunaan teknologi visi komputer pada perangkat seperti Jetson Nano dapat dimanfaatkan untuk mendeteksi fitur fisik objek secara *real-time*, demikian juga penggunaan metode *Artificial Neural Network* (ANN) yang terbukti efektif dalam mengklasifikasi gerakan dan mengklasifikasikan kualitas objek dengan akurasi tinggi. Selain itu, implementasi lengan robot OpenManipulator-X dengan kontroler OpenCR telah teruji mampu memberikan stabilitas dan presisi dalam manipulasi benda. Oleh sebab itu, pada penelitian ini dihipotesiskan beberapa hal yang bisa dilakukan, yaitu membangun komunikasi data yang efisien antar mikrokontroler (antara unit pemroses citra dan OpenCR) untuk menerjemahkan hasil deteksi kualitas jeruk menjadi perintah gerak penyortiran yang akurat dan otomatis.

1.2 Identifikasi Masalah

Berdasarkan latar belakang tersebut, dapat diidentifikasi beberapa masalah sebagai berikut:

1. Proses penyortiran jeruk yang masih dilakukan secara manual kurang efisien, memerlukan banyak tenaga kerja, serta rawan terjadi kesalahan.
2. Kualitas jeruk sangat memengaruhi harga jual dan penerimaan konsumen, sehingga diperlukan metode penyortiran yang lebih akurat agar hanya jeruk sehat yang terdistribusi ke pasar.
3. Belum adanya pemanfaatan robot *manipulator* sederhana berbasis mikrokontroler yang terintegrasi dengan kecerdasan buatan untuk melakukan penyortiran buah secara otomatis.
4. Diperlukan sistem kendali yang mampu menghubungkan hasil klasifikasi *machine learning* dengan pergerakan robot *Arm* agar proses penyortiran dapat berjalan cepat, konsisten, presisi, dan otomatis.

1.3 Batasan Masalah

1. Penelitian difokuskan pada proses penyortiran buah jeruk menggunakan robot *Arm* OpenManipulator yang dikendalikan oleh OpenCR.
2. Sistem kecerdasan buatan hanya terbatas pada klasifikasi jeruk sehat dan jeruk tidak sehat menggunakan model *machine learning Convolutional*

neural network (CNN) sederhana dengan 2 fitur utama yaitu warna dan tekstur(bercak / noda).

3. Hasil klasifikasi dikirimkan dalam format Jetson Nano melalui komunikasi UART sebagai dasar pergerakan robot *Arm*.
4. Penelitian tidak mencakup pengolahan citra lanjutan seperti deteksi ukuran, warna kompleks, maupun tingkat kematangan buah.
5. Ruang lingkup penelitian hanya berupa prototipe sederhana dan tidak membahas implementasi pada skala industri besar.

Batasan ini ditetapkan guna menjaga fokus pembahasan pada peran penulis dalam merancang dan menguji mekanisme pengendalian gerakan robot *Arm* OpenManipulator berbasis board OpenCR serta integrasi dengan modul kamera, sehingga tidak mencakup pembahasan aspek implementasi industri berskala besar atau penggunaan perangkat tambahan di luar lingkup prototipe sederhana yang dirancang.

1.4 Rumusan Masalah

Bagaimana merancang dan mengimplementasikan sistem pengendalian robot *Arm* OpenManipulator berbasis board OpenCR yang terintegrasi dengan modul kamera dan mikrokontroler Jetson Nano untuk menyortir jeruk secara otomatis. Permasalahan yang diangkat mencakup bagaimana proses klasifikasi jeruk sehat dan tidak sehat dilakukan melalui pemrosesan citra berbasis *machine learning* pada mikrokontroler Jetson Nano, bagaimana hasil klasifikasi tersebut dikirimkan melalui komunikasi UART ke OpenCR, serta bagaimana sistem kendali robot *Arm* diatur agar dapat mengeksekusi gerakan penyortiran secara tepat dan efisien.

1.5 Tujuan Penelitian

1. Merancang dan mengimplementasikan sistem pengendalian gerakan robot *Arm* OpenManipulator berbasis *board* OpenCR yang terintegrasi dengan modul kamera dan mikrokontroler Jetson Nano untuk melakukan penyortiran jeruk secara otomatis.

2. Menguji kinerja robot *Arm* dalam melakukan penyortiran jeruk sesuai hasil klasifikasi *machine learning* yang membedakan antara jeruk sehat dan tidak sehat.
3. Menilai efektivitas sistem yang dikembangkan dalam mendukung proses penyortiran jeruk, khususnya dalam hal akurasi, konsistensi, dan efisiensi, sehingga dapat menjadi gambaran awal penerapan otomasi pada industri pangan.

1.6 Luaran

1. Prototipe sistem penyortiran jeruk berbasis robot *Arm* OpenManipulator, board OpenCR, dan mikrokontroler Jetson Nano.

2. Dokumentasi berupa laporan tugas akhir yang memuat rancangan, implementasi, serta hasil pengujian sistem.

3. Publikasi ilmiah berupa artikel yang dapat dijadikan referensi untuk penelitian sejenis di bidang otomasi dan robotika.

1.7 Manfaat Penelitian

a. Bagi industri: memberikan alternatif solusi otomasi penyortiran buah agar lebih efisien, konsisten, dan akurat. Walaupun masih berupa prototipe sederhana, rancangan ini dapat menjadi gambaran awal penerapan sistem robotik skala kecil yang berpotensi dikembangkan ke arah aplikasi industri sebenarnya.

b. Bagi akademik: menjadi referensi sekaligus dasar pengembangan lebih lanjut mengenai implementasi robot manipulator berbasis mikrokontroler. Hasil penelitian ini juga memperkaya literatur tentang integrasi modul kamera dengan sistem kendali robot pada bidang otomasi.

c. Bagi peneliti: memberikan pengalaman praktis dalam merancang, memprogram, dan mengendalikan sistem robotik sederhana yang relevan dengan kebutuhan industri, serta menjadi bekal untuk penelitian lanjutan dengan skala dan kompleksitas yang lebih tinggi.

1.8 Sistematika Penulisan

Berikut merupakan sistematika penulisan penelitian ini:

1. BAB I Pendahuluan

Bab ini berisi latar belakang penelitian, identifikasi masalah, batasan masalah, rumusan masalah, tujuan penelitian, luaran, manfaat penelitian, serta sistematika penulisan.

2. BAB II Tinjauan Pustaka

Bab ini membahas teori-teori dan konsep yang mendasari penelitian, antara lain mengenai robot manipulator, board OpenCR, modul kamera, mikrokontroler Jetson Nano, komunikasi UART, *machine learning* untuk klasifikasi citra, serta penelitian-penelitian terdahulu yang relevan.

3. BAB III Analisis dan Perancangan Sistem

Bab ini menguraikan analisis kebutuhan, desain sistem, arsitektur perangkat keras dan perangkat lunak, perancangan alur komunikasi, serta perancangan logika pengendalian robot *Arm* OpenManipulator untuk penyortiran jeruk.

4. BAB IV Hasil dan Pembahasan

Bab ini menyajikan hasil implementasi sistem, pengujian prototipe, analisis kinerja robot *Arm* dalam melakukan penyortiran, serta pembahasan mengenai efektivitas dan keterbatasan sistem yang dikembangkan.

5. BAB V Simpulan dan Saran

Bab ini memuat simpulan yang diperoleh dari penelitian serta saran yang dapat dijadikan acuan untuk pengembangan penelitian lebih lanjut.

BAB II
TINJAUAN PUSTAKA

2.1 Bahasa C/C++

Bahasa C merupakan bahasa pemrograman prosedural yang dikembangkan oleh Dennis Ritchie pada awal tahun 1970-an. Bahasa ini dirancang untuk efisien dalam pengelolaan memori dan performa, serta dekat dengan bahasa mesin, sehingga sangat cocok digunakan dalam pengembangan sistem operasi dan perangkat keras. Dalam konteks sistem tertanam, seperti pada mikrokontroler OpenCR, bahasa C menjadi standar industri karena kemampuannya dalam mengakses dan mengontrol perangkat keras secara langsung. Selain itu, kompiler untuk bahasa C juga tersedia luas dan mendukung berbagai arsitektur prosesor.

Tipe data adalah klasifikasi atau jenis data yang menentukan nilai yang dapat disimpan dalam suatu variabel dan operasi apa yang dapat dilakukan terhadap data tersebut. Dalam bahasa C menentukan ukuran dan jenis nilai yang dapat disimpan dalam variabel. Berikut adalah beberapa tipe data dasar yang umum digunakan:

- int – menyimpan bilangan bulat (positif atau negatif).
- char – menyimpan karakter ASCII.
- float dan double – menyimpan angka pecahan atau bilangan desimal.
- uint8_t, uint16_t, uint32_t – tipe data integer tak bertanda (unsigned integer) dengan ukuran tetap.

Variabel dalam pemrograman adalah sebuah nama atau identifikasi yang diberikan untuk menyimpan suatu nilai data dalam memori komputer. Nilai ini dapat dapat diubah atau dimanipulasi selama eksekusi program. Setiap variabel memiliki tipe data yang menentukan jenis nilai yang dapat disimpan, seperti angka, teks, atau struktur data yang lebih kompleks. Nama variabel yang dipilih biasanya mendeskripsikan data yang disimpan, memudahkan pemahaman dan pengelolaan kode. Dalam pemrograman,

variabel memungkinkan programmer untuk menyimpan data, mengubahnya, dan mengaksesnya kapan pun selama program berjalan. Dengan demikian, penulisan kode menjadi lebih fleksibel karena nilai yang spesifik bisa disimpan dan diubah hanya di satu tempat, namun dapat digunakan di banyak bagian dalam program.

Setelah mengenal tipe data dan variabel ada juga fungsi, dalam bahasa C merupakan blok kode yang dirancang untuk menjalankan tugas tertentu dan dapat dipanggil berkali-kali. Fungsi membantu dalam mengatur kode, membuatnya lebih mudah dibaca, dipahami, dan dikelola, serta memungkinkan penggunaan kode berulang. Contoh penulisan function pada Bahasa C

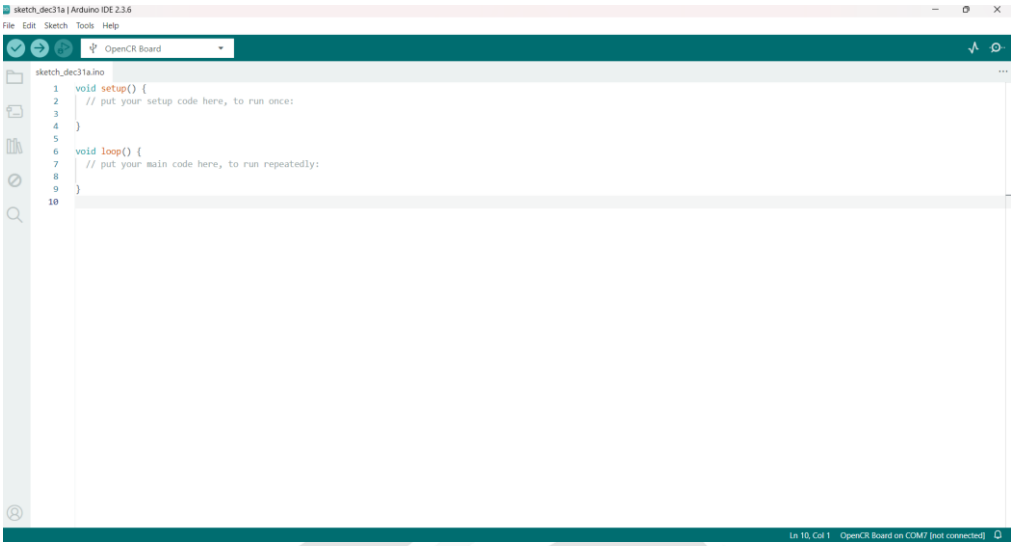
```
void warning()
{
 Serial.println();
 Serial.println("WARNING!!! OpenManipulator-X operates in 5 seconds.");
 delay_ms(1000);
 open_manipulator.receiveAllJointActuatorValue();
 open_manipulator.receiveAllToolActuatorValue();
 open_manipulator.enableAllActuator();
 delay_ms(1000);
}
```

Fungsi warning() adalah sebuah fungsi sederhana dalam bahasa C yang digunakan untuk memberi peringatan dan memunculkan di *serial monitor* pada Arduino IDE. Fungsi ini tidak menerima parameter apa pun dan tidak mengembalikan nilai karena bertipe void.

Bahasa C memiliki relevansi yang sangat tinggi dalam pengembangan sistem embedded karena kedekatannya dengan perangkat keras. Bahasa ini memungkinkan pemrogram untuk mengontrol register mikrokontroler secara langsung, mengakses fitur interrupt dan berbagai peripheral seperti UART, SPI, dan I2C, serta mengatur manajemen memori dengan presisi tinggi. Selain itu, bahasa C juga digunakan untuk menulis driver bagi perangkat-perangkat seperti sensor dan aktuator. Kemampuan tersebut menjadikan C sebagai fondasi utama dalam pengembangan firmware profesional, termasuk pada perangkat seperti OpenCR (Kernighan & Ritchie, 1988).

2.2 Arduino IDE

Arduino Integrated Development Environment (IDE) merupakan perangkat lunak utama yang digunakan dalam proses pengembangan sistem berbasis mikrokontroler Arduino maupun mikrokontroler lain yang mendukung ekosistemnya. Arduino IDE berfungsi sebagai sarana untuk menulis, mengompilasi, dan mengunggah program ke papan mikrokontroler dengan antaemuka yang sederhana dan intuitif. Keunggulan utama dari IDE ini adalah kemampuannya untuk menjembatani pengguna dari berbagai tingkat kemampuan baik pemula yang baru belajar pemrograman maupun peneliti yang sedang mengembangkan sistem robotika atau otomasi tingkat lanjut. Melalui desain yang ramah pengguna, Arduino IDE mempermudah proses pembelajaran pemrograman mikrokontroler tanpa memerlukan pemahaman mendalam terhadap konfigurasi perangkat keras atau bahasa pemrograman tingkat rendah.



Gambar 2. 1 Tampilan Arduino IDE

Gambar 2.1 menunjukan tampilan Arduino IDE saat pertama kali membuka Arduino dan membuat file baru. Arduino memiliki beberapa fitur penting, salah satu fitur penting dari Arduino IDE adalah *serial monitor*, yaitu alat bantu untuk menampilkan data dari mikrokontroler secara *real time*. Fitur ini sangat berguna untuk melakukan proses *debugging*, analisis sensor, serta komunikasi antar perangkat. Arduino IDE juga terintegrasi dengan berbagai pustaka *open-source* yang dapat diunduh langsung dari *library manager*, memudahkan peneliti untuk mengembangkan sistem yang lebih kompleks seperti *Internet of Things* (IoT), sistem kontrol otomatis, hingga robotika. Kemudahan integrasi dengan pustaka eksternal seperti WiFi.h, Servo.h, SoftwareSerial.h, dan DynamixelWorkbench.h menjadikan Arduino IDE pilihan utama bagi banyak *developer*.

Dalam konteks penelitian dan pengembangan robotika, Arduino IDE banyak digunakan untuk memprogram mikrokontroler seperti OpenCR yang digunakan pada robot *Arm* OpenManipulator. Melalui IDE ini, pengguna dapat menulis algoritma kontrol gerak, komunikasi data dengan sensor, serta pengaturan aktuator seperti motor Dynamixel. Tidak hanya terbatas pada perangkat Arduino asli, IDE ini juga mendukung berbagai papan berbasis *ARM*, ESP, dan STM32 yang memiliki kemampuan komputasi lebih tinggi. Keberagaman ini menjadikan Arduino IDE sebagai

alat yang sangat fleksibel untuk berbagai kebutuhan riset, mulai dari eksperimen akademik hingga implementasi industri.

Arduino juga memiliki komunitas yang sangat besar sehingga memberikan peran penting terhadap pengembangan IDE. Arduino memiliki ribuan contoh kode, dokumentasi serta forum diskusi yang membantu penggunanya memecahkan permasalahan teknis yang sedang dihadapi. Pembaruan versi IDE juga dilakukan secara berkala untuk meningkatkan stabilitas, kompatibilitas, dan fitur-fitur baru seperti integrasi dengan *Arduino Cloud* dan dukungan *debugger* bawaan. Dengan dukungan ekosistem yang kuat ini, Arduino IDE menjadi salah satu platform pengembangan paling populer di dunia teknik elektro, mekatronika, dan ilmu komputer terapan (Banzi & Shiloh, 2014).

2.3 Mikrokontroler STM32F746ZGT6

Mikrokontroler merupakan sebuah komputer kecil yang terdapat di dalam satu chip sirkuit terpadu (*Integrated Circuit*/IC) yang berisi CPU (*Central Processing Unit*), memori, serta perangkat *input* dan *output* (I/O) yang dapat diprogram untuk menjalankan fungsi tertentu. Mikrokontroler sering disebut sebagai otak dari berbagai sistem tertanam (*embedded system*) karena memiliki kemampuan untuk memproses data, mengambil keputusan, dan mengendalikan perangkat lain sesuai instruksi yang telah ditanamkan ke dalam programnya. Secara umum, mikrokontroler bekerja dengan menjalankan serangkaian instruksi yang disimpan dalam memori internalnya untuk mengontrol sinyal dari sensor maupun aktuator, sehingga memungkinkan interaksi langsung antara sistem digital dengan lingkungan fisik.

Menurut Valvano (2017), mikrokontroler memiliki peran utama sebagai pusat kendali dalam sistem tertanam yang mengintegrasikan perangkat lunak (*software*) dan perangkat keras (*hardware*) untuk melaksanakan tugas spesifik secara otomatis dan efisien. Berbeda dengan mikroprosesor yang membutuhkan komponen eksternal seperti memori dan perangkat I/O tambahan, mikrokontroler telah menggabungkan seluruh

komponen utama tersebut ke dalam satu *chip* tunggal. Hal ini menjadikan mikrokontroler lebih hemat energi, berbiaya rendah, serta mudah diintegrasikan pada berbagai aplikasi industri maupun non-industri. Dalam sistem otomasi modern, mikrokontroler mampu menjalankan operasi *real-time*, yaitu memproses masukan dan menghasilkan keluaran secara langsung tanpa jeda yang signifikan, sehingga sangat cocok digunakan pada sistem kendali dan monitoring.

Fungsi utama mikrokontroler terletak pada kemampuannya untuk membaca *input* dari lingkungan luar, seperti sinyal dari sensor suhu, cahaya, tekanan, atau arus listrik, kemudian memproses informasi tersebut untuk menghasilkan respon yang sesuai melalui aktuator seperti motor, *relay*, atau modul komunikasi. Dalam konteks ini, mikrokontroler bertindak sebagai jembatan antara dunia analog (lingkungan fisik) dan dunia digital (sistem pemrosesan). Proses ini terjadi secara berulang dan teratur sesuai program yang ditanamkan, sehingga mikrokontroler dapat menjalankan sistem otomatis dengan presisi tinggi.

Efisiensi dan fleksibilitas mikrokontroler menjadikannya sangat populer dalam berbagai bidang penerapan. Mikrokontroler digunakan dalam perangkat rumah tangga seperti mesin cuci, pendingin udara, dan oven, sistem otomasi industri seperti robotik dan kontrol mesin; peralatan medis seperti alat pemantau detak jantung dan pompa infus, serta kendaraan modern untuk mengontrol sistem injeksi bahan bakar, sistem pengereman ABS, hingga sensor parkir. Pada *Internet of Things* (IoT), mikrokontroler menjadi komponen kunci yang memungkinkan perangkat-perangkat cerdas berkomunikasi melalui jaringan internet, mengumpulkan data, serta melakukan analisis sederhana secara lokal sebelum dikirim ke server atau cloud.

Kemajuan teknologi juga mendorong munculnya berbagai jenis mikrokontroler dengan kemampuan yang semakin kompleks, seperti seri AVR, PIC, *ARM* Cortex-M, dan ESP32 yang telah dilengkapi dengan fitur WiFi, Bluetooth, serta kecepatan pemrosesan yang lebih tinggi. Salah satu

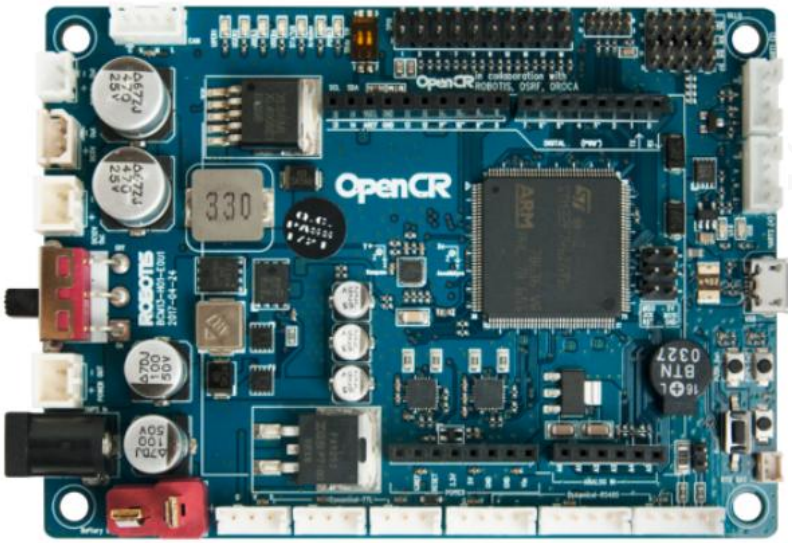
jenis mikrokontroler yang banyak digunakan dalam sistem tertanam modern adalah STM32F746ZGT6, yang merupakan bagian dari keluarga STM32F7 Series buatan STMicroelectronics. Mikrokontroler ini berbasis pada arsitektur *ARM* Cortex-M7 dengan frekuensi clock hingga 216 MHz, yang memberikan kinerja tinggi dan efisiensi daya yang baik untuk aplikasi pemrosesan real-time. STM32F746ZGT6 dilengkapi dengan memori flash sebesar 1 MB, RAM 320 KB, serta berbagai peripheral canggih seperti ADC, DAC, UART, SPI, I2C, CAN, USB, dan Ethernet. Dukungan fitur Floating Point Unit (FPU) dan Digital Signal Processing (DSP) memungkinkan mikrokontroler ini melakukan perhitungan matematis kompleks dengan cepat, sehingga cocok digunakan dalam sistem robotik, kendali motor, pengolahan sinyal, dan aplikasi berbasis visi komputer.

Dalam konteks pengembangan sistem kendali, STM32F746ZGT6 menawarkan fleksibilitas tinggi karena dapat diprogram melalui berbagai lingkungan pengembangan seperti STM32CubeIDE, Keil µVision, atau PlatformIO, dengan dukungan pustaka perangkat keras (HAL dan LL driver) yang disediakan oleh STMicroelectronics. Hal ini memudahkan pengembang untuk melakukan konfigurasi peripheral, manajemen clock, serta komunikasi antar modul melalui antarmuka digital seperti SPI atau UART. Selain itu, mikrokontroler ini juga mendukung fitur *Direct Memory Access* (DMA) dan *Real-Time Operating System* (RTOS) yang memungkinkan pengolahan data lebih cepat dan efisien dalam sistem multitasking. Dalam bidang robotika, mikrokontroler ini digunakan untuk mengendalikan lengan robot atau manipulator melalui komunikasi serial berkecepatan tinggi dengan aktuator dan sensor. Beberapa platform seperti OpenCR (*Open-source Control Robot*) juga menggunakan STM32F746ZGT6 sebagai otak utama untuk mengontrol motor Dynamixel, membaca data sensor

2.4 OpenCR

OpenCR (*Open-source Control Module for ROS*) merupakan papan kendali terbuka yang dikembangkan oleh ROBOTIS untuk mendukung

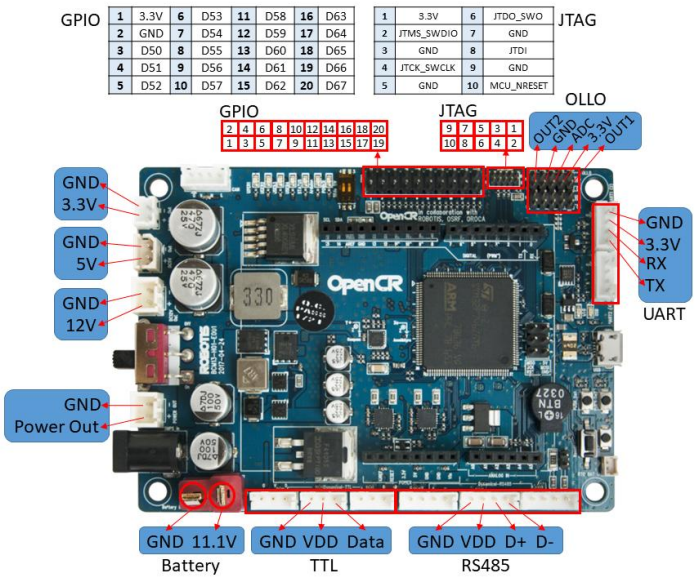
sistem robotik berbasis ROS (*Robot Operating System*). Papan ini berfungsi sebagai pengendali utama yang menghubungkan antara perangkat keras robot, seperti aktuator, sensor, dan modul komunikasi, dengan sistem perangkat lunak pengendali pada komputer atau mikrokontroler lain. Sebagai platform yang bersifat *open source*, seluruh perangkat lunak, skematik rangkaian, dan firmware dari OpenCR dapat diakses, dimodifikasi, serta dikembangkan secara bebas oleh pengguna, sehingga menjadikannya solusi yang fleksibel dan adaptif dalam berbagai aplikasi robotika, baik untuk keperluan pendidikan, penelitian, maupun industri.



Gambar 2. 2 OpenCR 1.0

Gambar 2.2 menunjukan OpenCR 1.0 yang penulis gunakan untuk mengontrol *Arm* OpenManipulator, secara teknis OpenCR menggunakan mikrokontroler STM32F7 Series *ARM* Cortex-M7 dengan kecepatan 216 MHz dan dilengkapi dengan *floating point unit* (FPU) yang memungkinkan pengolahan data numerik kompleks secara efisien. Papan ini juga memiliki berbagai antarmuka komunikasi seperti RS-485 dan TTL untuk koneksi dengan aktuator DYNAMIXEL, UART, I²C, dan SPI untuk komunikasi dengan sensor eksternal, serta port USB dan GPIO untuk *input-output* umum maupun ekspansi sistem. Dukungan komunikasi yang beragam tersebut memungkinkan OpenCR untuk berinteraksi dengan berbagai jenis perangkat keras dalam sistem robotik secara andal dan serbaguna. Selain

itu, OpenCR dapat diprogram menggunakan dua pendekatan utama, yaitu melalui lingkungan pengembangan Arduino IDE untuk sistem mandiri, atau menggunakan ROS untuk sistem terdistribusi yang dijalankan di komputer. Dengan kemampuan ini, OpenCR dapat digunakan baik sebagai pengendali tunggal maupun sebagai bagian dari sistem kontrol yang lebih kompleks.



Gambar 2. 3 OpenCR PinOuts

Gambar 2.3 menunjukan susunan pin (*pinout*) pada papan OpenCR, yang berfungsi untuk menghubungkan berbagai komponen eksternal seperti sensor, aktuator, dan modul komunikasi. Setiap pin memiliki fungsi spesifik, termasuk pin daya, komunikasi serial (UART, I2C, SPI), serta pin *input/output* digital dan analog yang digunakan untuk mengendalikan sistem robot. Dengan memahami konfigurasi *pinout* ini, pengguna dapat melakukan koneksi perangkat keras dengan lebih tepat dan aman.

Dalam konteks sistem robotik, OpenCR berperan sebagai pengendali tingkat rendah (*low-level controller*) yang bertugas mengeksekusi perintah dari pengendali tingkat tinggi (*high-level controller*). Fungsi utamanya meliputi pengendalian aktuator seperti motor DYNAMIXEL, pembacaan data dari berbagai sensor, serta pengelolaan komunikasi dua arah antara perangkat keras dan sistem pengendali eksternal. Selain itu, OpenCR juga mampu menjalankan *loop control* secara

lokal, seperti pengaturan posisi, kecepatan, maupun torsi pada tiap aktuator, sehingga dapat mengurangi beban pemrosesan pada sistem komputer utama. OpenCR juga dilengkapi dengan sistem catu daya yang mendukung tegangan operasi 7–24volt DC, serta fitur proteksi terhadap arus lebih, tegangan berlebih, dan suhu berlebih, yang menjadikannya andal digunakan untuk aplikasi robotika jangka panjang.

Keunggulan utama OpenCR terletak pada sifatnya yang sepenuhnya terbuka, baik dari sisi perangkat keras, perangkat lunak, maupun, sehingga pengguna dapat memodifikasi sistem sesuai kebutuhan riset atau proyek. Selain itu, OpenCR memiliki kompatibilitas tinggi dengan berbagai platform robot ROBOTIS seperti TurtleBot3 dan *Arm* OpenManipulator. Dalam sistem *Arm* OpenManipulator, OpenCR berperan sebagai pengendali utama yang menyalurkan perintah gerak dari sistem kendali eksternal menuju motor DYNAMIXEL pada tiap sendi *manipulator*. Melalui komunikasi berbasis ROS, komputer pengendali mengirimkan perintah lintasan atau posisi (*joint trajectory*) ke OpenCR melalui jalur komunikasi USB atau serial. Selanjutnya, OpenCR mengubah perintah tersebut menjadi sinyal kontrol untuk mengatur profil kecepatan dan posisi setiap motor, sehingga lengan manipulator dapat bergerak sesuai lintasan yang diinginkan secara halus dan presisi.

Penggunaan OpenCR dalam sistem *Arm* OpenManipulator memiliki keunggulan dibandingkan penggunaan antarmuka lain seperti U2D2, karena OpenCR tidak hanya berfungsi sebagai penghubung komunikasi, tetapi juga sebagai pengendali aktif yang mampu menjalankan perintah dan melakukan proses kendali langsung pada tingkat perangkat keras. Hal ini menjadikan sistem lebih efisien dan responsif, serta mengurangi ketergantungan pada komputer utama. OpenCR juga memungkinkan integrasi dengan berbagai sensor tambahan tanpa memerlukan modul konversi eksternal, sehingga lebih fleksibel untuk pengembangan sistem robotika yang kompleks.

Beberapa penelitian telah memanfaatkan OpenCR sebagai pengendali utama dalam sistem robotik berbasis ROS. OpenCR secara

resmi menyatakan bahwa ROS kompatibel dijalankan pada OpenCR contohnya pada penelitian dengan judul "*Navigation and Task Planning of a Mobile Robot under ROS Environment: A Case Study Using AutoRace Challenge"*(Shen et al., 2021) yang menggunakan platform TurtleBot3 pada OpenCR untuk mengikuti tantangan AutoRace, robot harus mengenali jalur dan mencapai tujuan secepat mungkin dengan kesalahan yang minim, dengan memanfaatkan kamera dan sensor LiDAR 2D untuk navigasi dan penerapan algoritma pengolahan citra dengan *deep learning* menghasilkan robot yang mampu menyelesaikan misi dalam waktu 2-3 menit secara stabil.

Secara keseluruhan, OpenCR merupakan komponen penting dalam ekosistem robotika modern. Kombinasi antara kemampuan pengolahan sinyal yang cepat, fleksibilitas pemrograman, dan kompatibilitas tinggi terhadap aktuator DYNAMIXEL menjadikan OpenCR sangat ideal digunakan pada *platform Arm* OpenManipulator. Dukungan penuh terhadap ekosistem *open-source* juga memberikan peluang luas bagi peneliti dan pengembang untuk melakukan inovasi dalam sistem kendali robotika, baik pada tingkat pendidikan, penelitian, maupun industri.

2.5 OpenManipulator-X

OpenManipulator-X merupakan salah satu platform robot manipulator *open-source* yang dikembangkan oleh robotis dengan tujuan mendukung kegiatan riset, pendidikan, serta pengembangan teknologi di bidang robotika. Robot ini dirancang untuk meniru pergerakan lengan manusia dengan beberapa derajat kebebasan (*degree of freedom*, DoF), sehingga mampu melakukan berbagai tugas seperti memindahkan, memegang, dan mengatur posisi objek secara otomatis. Keunggulan utama OpenManipulator terletak pada sifatnya yang modular dan terbuka, baik dari sisi perangkat keras maupun perangkat lunak, yang memungkinkan pengguna untuk memodifikasi, menambah, atau mengganti bagian-bagian tertentu sesuai kebutuhan penelitian.

Gambar 2. 4 OpenManipulator-X

Gambar 2.4 OpenManipulator-X menunjukan salah satu versi dari OpenManipulator yang paling banyak digunakan adalah OpenManipulator-X (RM-X52-TNM). Versi ini memiliki empat sendi utama dan satu *gripper*, sehingga total lima derajat kebebasan yang dapat dikontrol. Masing-masing sendi digerakkan oleh motor DYNAMIXEL X-Series, yaitu aktuator pintar yang memiliki kemampuan feedback posisi, kecepatan, dan torsi melalui komunikasi serial berbasis TTL atau RS-485. Struktur mekaniknya dapat dimodifikasi menggunakan komponen 3D *printing*, menjadikan sistem ini fleksibel untuk berbagai eksperimen kinematika dan kontrol gerak.

Items	Unit	OpenMANIPULATOR-X
Actuator		DYNAMIXEL XM430-W350-T
Input Voltage	V	12
DOF	-	5 (4 DOF + 1 DOF Gripper)
Payload	g	500
Repeatability	mm	< 0.2
Speed(Joint)	RPM	46
Weight	kg (lb)	0.70 (1.54)
Reach	mm (in)	380 (14.9)
Gripper Stroke	mm (in)	20~75 (0.79~2.95)
Communication	-	TTL Level Multidrop BUS
Software	-	ROS, DYNAMIXEL SDK, Arduino, Processing
Main Controller	-	PC, OpenCR

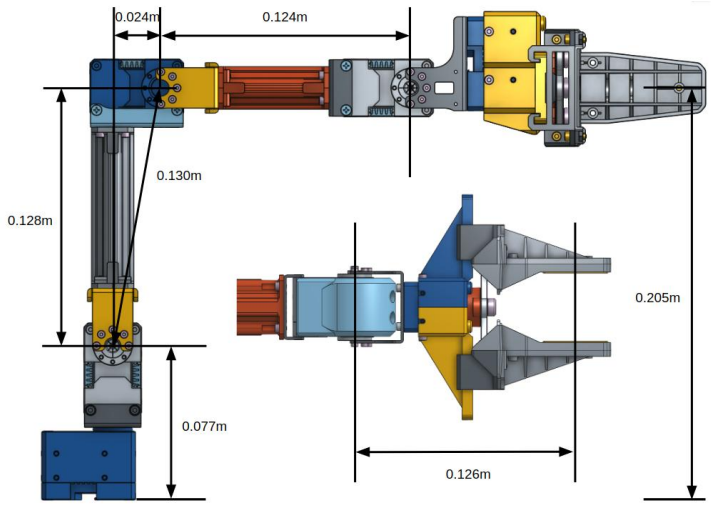
Gambar 2. 5 Spesifikasi OpenManipulator-X

Gambar 2.5 menunjukkan Spesifikasi OpenManipulator-X memperlihatkan rincian teknis dari robot lengan OpenManipulator-X yang digunakan pada penelitian ini. Robot ini menggunakan aktuator Dynamixel XM430-W350-T sebagai penggerak utamanya, dengan sumber tegangan kerja sebesar 12 volt. OpenManipulator-X memiliki 5 derajat kebebasan (4 DOF untuk sendi dan 1 DOF untuk *gripper*), serta mampu mengangkat beban hingga 500 gram. Dari sisi presisi gerak, robot ini memiliki tingkat pengulangan posisi kurang dari 0,2 mm, dengan kecepatan rotasi sendi mencapai 46 RPM. Berat keseluruhan robot sekitar 0,7 kg dengan jangkauan kerja mencapai 380 mm. *Gripper* yang digunakan memiliki rentang bukaan antara 20 hingga 75 mm, sehingga dapat menyesuaikan dengan berbagai ukuran objek. Sistem komunikasinya menggunakan TTL *Level Multidrop* BUS, sementara pengendalian dan pemrograman didukung oleh ROS (*Robot Operating System*), Dynamixel SDK, Arduino, serta Processing. Robot ini dapat dikontrol menggunakan PC maupun

mikrokontroler OpenCR, menjadikannya fleksibel untuk berbagai aplikasi penelitian, pengembangan, maupun integrasi dengan sistem penglihatan komputer.

Contoh nyata penerapan OpenCR pada sistem OpenManipulator dapat ditemukan dalam proyek *Camera-based Augmented Reality Endoscope Auxiliary System* (Wang, 2021), di mana OpenCR digunakan sebagai penghubung antara komputer dan aktuator DYNAMIXEL pada OpenManipulator-X. Pada penelitian tersebut, OpenCR berfungsi sebagai pengendali utama yang menerima perintah gerak dari ROS dan meneruskannya ke motor, sekaligus menyediakan suplai daya serta mengolah data kinematika. Sistem ini berhasil mengendalikan posisi *end-effector* robot dengan ketelitian tinggi dan latensi kendali yang rendah, menunjukkan bahwa integrasi OpenManipulator dan OpenCR efektif untuk aplikasi medis berbasis *real-time control*.

Selain menggunakan OpenCR, sistem OpenManipulator juga dapat dikendalikan menggunakan U2D2, yaitu perangkat konverter USB-to-TTL/RS-485 yang dikembangkan oleh robotis untuk komunikasi langsung antara komputer dan motor DYNAMIXEL tanpa mikrokontroler tambahan. Dalam konfigurasi ini, komputer (atau laptop) menjalankan seluruh proses perencanaan gerak, komputasi kinematika, serta pengiriman perintah melalui ROS. U2D2 hanya berfungsi sebagai antarmuka komunikasi fisik antara PC dan jaringan motor, sehingga cocok untuk penelitian atau pembelajaran yang berfokus pada pemrograman dan simulasi berbasis komputer. Beberapa penelitian, seperti yang dilakukan oleh (Winarta et al., 2024) dalam "Rancang Bangun Pengontrol Gerakan Robot OpenManipulator dengan MATLAB", menggunakan U2D2 untuk menghubungkan OpenManipulator dengan MATLAB dan berhasil menunjukkan bahwa sistem mampu mengikuti posisi target dengan akurasi tinggi.



Gambar 2. 6 Desain dan Dimensi OpenManipulator-X

Gambar 2.6 menunjukkan desain dan dimensi struktural dari robot *Arm* OpenManipulator-X. Setiap bagian pada gambar merepresentasikan panjang segmen lengan (*link*) dan posisi sambungan (*joint*) yang membentuk keseluruhan konfigurasi robot. Ukuran setiap link ditunjukkan dalam satuan meter, seperti 0.077 m, 0.128 m, 0.130 m, 0.124 m, dan 0.205 m, yang menggambarkan jarak antar sumbu rotasi maupun ukuran komponen utama. Informasi ini sangat penting dalam analisis kinematika dan dinamika, karena menjadi dasar dalam menentukan jangkauan gerak (*workspace*) serta posisi *end-effector*. Selain itu, tampak pula bagian *gripper* pada ujung lengan yang berfungsi untuk meng*gripper* atau memegang objek, yang juga memiliki ukuran spesifik agar sesuai dengan tugas manipulasi yang diinginkan.

Secara keseluruhan, OpenManipulator-X memiliki keunggulan dalam hal fleksibilitas, modularitas, dan dukungan sistem terbuka. Kombinasi antara aktuator DYNAMIXEL, komunikasi berbasis ROS, serta kemampuan integrasi dengan OpenCR maupun U2D2 menjadikan OpenManipulator-X ini sangat cocok untuk berbagai keperluan riset dan pendidikan. Dalam konteks penelitian modern, OpenManipulator-X tidak hanya digunakan untuk studi kontrol dan kinematika, tetapi juga untuk pengembangan sistem berbasis kecerdasan buatan, pengenalan visual, dan

otomasi cerdas termasuk aplikasi seperti penyortiran objek berdasarkan citra atau robot kolaboratif yang berinteraksi langsung dengan manusia.

2.6 Dynamixel XM430-W350-T

Dynamixel XM430-W350-T merupakan salah satu jenis servo motor cerdas (*smart actuator*) yang dikembangkan oleh Robotis, dan banyak digunakan dalam berbagai aplikasi robotika, termasuk pada sistem OpenManipulator-X. Modul ini dirancang untuk memberikan performa tinggi dengan presisi gerak yang baik serta fleksibilitas dalam pengendalian posisi, kecepatan, dan torsi. Berbeda dari motor servo konvensional, Dynamixel XM430 dilengkapi dengan mikrokontroler internal, sensor posisi absolut berbasis encoder, dan sistem komunikasi digital, sehingga memungkinkan proses kendali dan monitoring dilakukan secara langsung melalui jaringan komunikasi TTL atau RS-485, tergantung pada variannya.

Gambar 2. 7 DYNAMIXEL-X

Gambar 2.7 diatas merupakan DYNAMIXEL-X, Dynamixel XM430-W350-T memiliki torsi maksimum mencapai sekitar 4,1 N·m pada tegangan 12 V dan kecepatan hingga 46 rpm, menjadikannya cocok untuk digunakan pada aplikasi yang membutuhkan kekuatan dan ketepatan gerak, seperti lengan robot atau sistem aktuasi berderajat kebebasan tinggi. Selain itu, motor ini mendukung pengendalian multi-turn, yang memungkinkan rotasi lebih dari satu putaran penuh dengan akurasi tinggi, sehingga memperluas fleksibilitas gerak manipulator.

Kelebihan utama dari XM430 terletak pada fitur PID control terintegrasi, kompensasi suhu, perlindungan arus lebih, serta kemampuan untuk mengirimkan data umpan balik seperti suhu, tegangan, posisi, dan kecepatan secara real-time. Hal ini membuat aktuator ini sangat efisien untuk sistem yang memerlukan kontrol tertutup (closed-loop control). Casing motor yang terbuat dari aluminium alloy juga membantu menjaga kestabilan termal dan kekuatan mekanis selama operasi berulang dalam waktu lama.

2.7 Mikrokomputer

Mikrokomputer merupakan perangkat komputasi berukuran kecil yang memiliki prosesor, memori, dan sistem *input output* yang memungkinkan perangkat ini bekerja secara mandiri. Perangkat ini pada dasarnya memiliki fungsi sama seperti komputer pada umumnya, hanya saja dengan ukuran yang lebih ringkas dan sumber daya yang lebih efisien. Mikrokomputer banyak digunakan untuk keperluan sistem tertanam (*embedded system*), kontrol otomatis, serta penelitian di bidang robotika karena kemampuannya dalam menjalankan program secara stabil dan hemat daya.

Dalam sistem robotika, mikrokomputer sering berfungsi sebagai otak utama yang memproses data dari berbagai sensor dan mengirimkan perintah ke aktuator atau mikrokontroler. Beberapa jenis mikrokomputer yang cukup populer digunakan di bidang ini adalah Raspberry Pi, NVIDIA Jetson, dan LattePanda. Perangkat tersebut umumnya mendukung sistem operasi Linux dan kompatibel dengan *Robot Operating System* (ROS), sehingga mudah diintegrasikan dengan berbagai modul dan platform robotik seperti *Arm* OpenManipulator.

Selain digunakan secara mandiri, mikrokomputer juga sering dipasangkan dengan mikrokontroler, seperti OpenCR, untuk membentuk sistem kendali dua tingkat sehingga mikrokomputer menangani tugas-tugas berat seperti perhitungan kinematika, pemrosesan data kamera, atau algoritma kecerdasan buatan, sementara mikrokontroler fokus pada kendali

motor dan sensor secara langsung. Dengan pembagian tugas seperti ini, sistem menjadi lebih efisien dan responsif dalam mengendalikan gerakan robot.

Salah satu contoh penerapan mikrokomputer dalam penelitian robotika adalah penelitian oleh (Xu, 2018) berjudul *Remote Control and Monitoring System of Robotic Arm Using Raspberry Pi*. Dalam penelitian tersebut, Raspberry Pi sebagai inti pengendali untuk melakukan perintah secara *remote*. Hasil pengujian menunjukkan bahwa sistem ini mampu mengirimkan perintah kendali dan mentransmisikan gambar dengan andal, serta memungkinkan robot melakukan berbagai tugas kompleks dengan stabil dan akurat. Sehingga dapat disimpulkan mikrokomputer memiliki peran yang sangat penting dalam pengembangan robot modern. Kemampuannya dalam melakukan pemrosesan data secara cepat, mendukung berbagai bahasa pemrograman, serta kemudahan integrasi dengan sistem komunikasi membuat perangkat ini menjadi pilihan utama dalam riset dan implementasi sistem robotik berbasis ROS.

2.8 Jetson Nano

Jetson Nano merupakan salah satu mikrokomputer yang dikembangkan oleh NVIDIA dan dirancang khusus untuk aplikasi berbasis kecerdasan buatan (AI) serta komputasi visual. Perangkat ini dilengkapi dengan prosesor quad-core *ARM* Cortex-A57 dan GPU NVIDIA Maxwell dengan 128 CUDA cores, yang menjadikannya mampu menjalankan pemrosesan paralel secara efisien. Dengan spesifikasi tersebut, Jetson Nano tidak hanya mampu menjalankan sistem operasi linux, tetapi juga mendukung berbagai *framework* pembelajaran mesin seperti TensorFlow, PyTorch, dan OpenCV, sehingga sangat cocok digunakan untuk penelitian di bidang robotika, visi komputer, dan sistem otonom.

Gambar 2. 8 Jetson Nano

Gambar 2.8 menunjukan mikrokomputer Jetson Nano, dalam konteks robotika, Jetson Nano berperan sebagai pengendali utama yang menangani proses komputasi berat, seperti deteksi objek, pengenalan wajah, navigasi visual, dan perencanaan lintasan. Kemampuan GPU-nya membuat Jetson Nano mampu memproses data gambar atau video secara *real-time*, sesuatu yang sulit dicapai jika hanya menggunakan mikrokontroler konvensional. Selain itu, perangkat ini juga kompatibel dengan *Robot Operating System* (ROS), yang memungkinkan integrasi langsung dengan berbagai perangkat robotik seperti *Arm* OpenManipulator, TurtleBot3, maupun sistem sensor eksternal.

Dalam beberapa penelitian, Jetson Nano sering digunakan sebagai bagian dari sistem kendali dan bertugas menjalankan algoritma pengolahan data dan pengambilan keputusan, dan mengirimkan hasil ke robot *Arm*. Salah satu penelitian Jetson Nano yang berhubungan dengan robotika berjudul *Robot intelligent grasping experimental platform combining Jetson Nano and machine vision* (Bao et al., 2022). Pada penelitian tersebut, Jetson Nano digunakan untuk memproses machine vision untuk memindahkan objek secara otomatis, Penelitian ini berhasil membangun sistem robot manipulator berbasis Jetson Nano dengan kemampuan deteksi warna dan pengambilan objek otomatis menggunakan *machine vision*

berbasis dengan metode pengolahan citra dasar. Hasil uji menunjukkan sistem sangat akurat dan efisien (100% deteksi warna dan 96% efisiensi *grasping*), dengan waktu deteksi di bawah 5 detik.

2.9 *Machine Learning*

Machine learning atau pembelajaran mesin adalah cabang dari kecerdasan buatan (*Artificial Intelligence*/AI) yang berfokus pada kemampuan sistem untuk belajar dari data dan membuat keputusan tanpa harus diprogram secara eksplisit. Konsep dasarnya adalah bagaimana komputer dapat mengenali pola dari sejumlah data, kemudian menggunakan pola tersebut untuk mengklasifikasi atau mengambil keputusan pada data baru. Dalam konteks teknologi moderen, *machine learning* banyak digunakan pada berbagai bidang seperti pengenalan wajah, klasifikasi citra, analisis suara, serta sistem rekomendasi.

Dalam bidang robotika, *machine learning* berperan penting untuk meningkatkan kemampuan adaptasi dan kecerdasan robot. Melalui penerapan algoritma pembelajaran, robot dapat memahami lingkungan sekitarnya, mengenali objek, serta menyesuaikan tindakan berdasarkan pengalaman atau data yang telah dikumpulkan. Salah satu contoh penerapan yang umum adalah pada sistem visi komputer (*computer vision*), biasanya robot dilatih untuk membedakan bentuk, warna, atau kondisi suatu objek, misalnya dalam proses penyortiran buah atau deteksi cacat produk di industri manufaktur.

Machine learning terbagi menjadi tiga jenis utama, yaitu *supervised learning, unsupervised learning, dan reinforcement learning*. *Supervised learning* digunakan ketika data yang digunakan memiliki label atau kategori, misalnya dalam klasifikasi gambar sehat dan tidak sehat. *Unsupervised learning* digunakan untuk menemukan pola atau kelompok dalam data tanpa label, sedangkan *reinforcement learning* berfokus pada proses pembelajaran berbasis umpan balik dari lingkungan, misalnya robot yang belajar mengambil keputusan untuk mencapai tujuan tertentu. Beberapa algoritma yang sering digunakan dalam *machine learning* antara

lain *Convolutional Neural Network* (CNN), *Decision Tree, Support Vector Machine* (SVM), dan *K-Nearest Neighbor* (KNN). Masing-masing algoritma memiliki kelebihan tersendiri tergantung pada jenis data dan tujuan analisis. Contohnya, penelitian dengan judul "Penerapan Metode Decision Tree Untuk Mengklasifikasikan Mutu Buah Jeruk Berdasarkan Fitur Warna Dan Ukuran" (Robianto et al., 2021) menunjukkan bahwa metode *Decision Tree* mampu mengklasifikasikan kondisi buah dengan tingkat akurasi lebih dari 92%. Hasil tersebut membuktikan bahwa penerapan *machine learning* sangat membantu dalam proses otomasi berbasis visual, khususnya dalam sistem penyortiran otomatis.

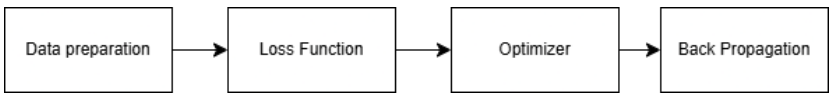
Convolutional Neural Network (CNN) adalah salah satu metode dalam *deep learning* yang banyak digunakan untuk mengenali dan mengolah gambar. CNN bekerja dengan mengenali pola visual, seperti bentuk, warna, dan tekstur. Metode ini terdiri dari beberapa lapisan yang bertugas mengenali ciri-ciri penting dari gambar hingga menghasilkan hasil akhir berupa klasifikasi atau pengenalan objek. Proses pelatihannya dilakukan dengan memberikan banyak contoh gambar yang sudah diberi label agar sistem dapat belajar mengenali perbedaan antar objek. CNN dikenal memiliki akurasi tinggi dalam mendeteksi dan mengklasifikasi gambar, namun membutuhkan data dan sumber daya komputasi yang besar.

Memiliki empat lapisan utama, pertama ada *Convolutional Layers* merupakan lapisan yang menerapkan operasi konvolusi pada citra input menggunakan filter atau kernel untuk mendeteksi fitur tepi, tekstur dan pola yang lebih kompleks. *Convolutional Layers* ini membantu mempertahankan hubungan spasial antar piksel. Selanjutnya ada *Pooling Layers* merupakan lapisan yang melakukan downsampling terhadap dimensi spasial dari masukan sehingga mengurangi kompleksitas komputasi dan jumlah parameter dalam jaringan. *Max pooling* merupakan operasi pooling yang paling umum digunakan (nilai maksimum dari sekelompok piksel tetangga dipilih). Layer selanjutnya adalah *Activation Functions* fungsi ini memeperkenalkan non-linearitas ke dalam model, sehingga memungkinkan jaringan untuk mempelajari hubungan yang lebih kompleks dalam data.

Lapisan terakhir adalah *Fully Connected Layers* merupakan lapisan yang bertanggung jawab dalam membuat klasifikasi berdasarkan fitur Tingkat tinggi yang telah dipelajari oleh lapisan-lapisan sebelumnya, setiap neuron pada lapisan ini terhubung dengan setiap neuron pada lapisan berikutnya.

Cara dasar *Convolutional Neural Network* (CNN) bekerja umumnya dibagi menjadi 5 tahap, tahap pertama merupakan *Input Image* disini *Convolutional Neural Network* (CNN) menerima sebuah gambar sebagai masukan, biasanya ukuran pada gambar diubah menjadi 224x224 piksel dan dikonversi formatnya (misal dari RGB ke bentuk numerik) agar model bisa memprosesnya secara konsisten. Tahap kedua merupakan *Convolutional Layers*, lapisan ini menggunakan filter(kernel) untuk memindai gambar dan mengekstraksi ciri-ciri penting seperti tepi, bentuk, atau tekstur, setiap filternya bisa mendeteksi pola tertentu di gambar misalnya satu filter mendeteksi tepi horizontal filter yang lain mendeteksi warna, dll sehingga hasilnya disebut *feature map*.

Tahap ketiga merupakan *Pooling Layers*, lapisan ini mengecilkan ukuran peta fitur (*downsampling*) untuk mengurangi jumlah data yang harus diproses, resiko *overtfitting* dan mempertahankan fitur yang paling penting biasanya menggunakan *max pooling* (hanya mengambil nilai maksimum dari area kecil misalnya 2x2 piksel). Tahap keempat merupakan *Fully Connected Layers*, setelah fitur penting berhasil diekstraksi dan diperkecil, hasilnya diratakan menjadi satu vector panjang dan dikirim ke lapisan terhubung penuh, setiap neuron terhubung dengan neuron di lapisan berikutnya, jaringan mempelajari hubungan antar fitur dan membuat keputusan akhir (mengenali objek dalam gambar). Tahap terakhir merupakan *Output*, lapisan ini menghasilkan klasifikasi akhir misalnya label kelas seperti jeruk sehat dan tidak sehat atau nilai probabilitas untuk setiap kelasnya.



Gambar 2. 9 Proses pelatihan CNN

Gambar 2.9 Proses pelatihan CNN menunjukan 4 tahap proses pelatihan dari metode CNN. Dimulai dari Data preparation pada tahap awal seluruh gambar dipersiapkan agar seragam misalnya diubah ke ukuran dan format yang sama, kadang dilakukan normalisasi nilai piksel (dari 0-255 menjadi 0-1), atau dilakukan data augmentation misalnya rotasi, flipping, dll agar model tidak mudah overfitiing. Tahap selanjutnya merupakan Loss Function yang digunakan untuk mengukur seberapa baik CNN mengklasifikasi hasil yang benar, jika klasifikasi model berbeda jauh dari label aslinya, maka nilai loss akan besar, tujuan pelatihan ini adalah meminimalkan nilai loss tersebut. Tahapan ketiga optimizer merupakan tahapan yang bertugas untuk memperbarui bobot darti jaringan agar nilai loss semakin kecil, menentukan arah dan seberapa besar perubahan bobot tiap iterasi berdasarkan nilai loss yang dihitung sebelumnya contoh optimizer yang umum digunakan adalah Stochastic Gradient Descent (SGD), Adam dan RMSprop. Tahapan terakhir yaitu Backpropagation merupakan teknik yang digunakan untuk menghitung gradien dari fungsi kerugian (loss function) terhadap bobot-bobot (weights) pada CNN. Nilai gradien tersebut kemudian digunakan oleh optimizer untuk memperbarui bobot-bobot CNN agar hasil klasifikasi menjadi semakin akurat.

Efisiensi dari CNN dapat dievaluasi menjadi beberapa kriteria. Pertama, akurasi adalah presentase seberapa tepat CNN mengklasifikasi gambar uji dengan rumus

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

Kedua , presisi adalah presentase gambar uji yang diklasifikasi oleh CNN sebagai kelas tertentu dan benar termasuk dalam kelas tersebut dengan rumus

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

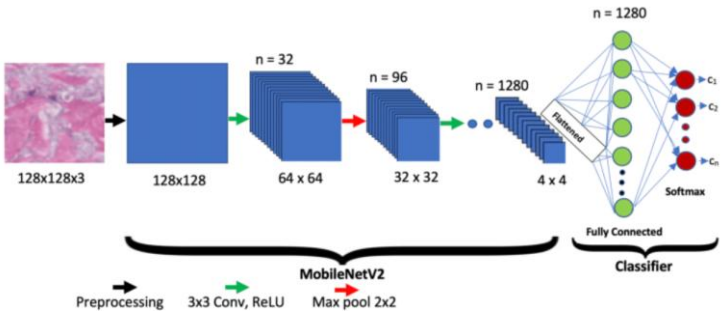
Ketiga, recall adalah presentase gambar uji yang sebenarnya, termasuk dalam suatu kelas dan berhasil diklasifikasi oleh CNN dengan rumus

$$\frac{TP}{TP + FN} \tag{3}$$

Terakhir, f1score adalah rata-rata h*Arm*onik dari presisi dan recall. Metrik ini baik digunakan untuk mengevaluasi kerja CNN terutama ketika jumlah data antar kelas tidak seimbang dengan rumus

$$F1\ score = 2 \cdot \frac{precision * recall}{precision + recall} \tag{4}$$

Fitur yang sering digunakan untuk evaluasi adalah tabel *confusion matrix,* tabel ini membandingkan hasil klasifikasi model dengan label sebenarnya dari data uji. Dalam pengembangan sistem robotik modern, *machine learning* sering digabungkan dengan platform seperti Jetson Nano atau Raspberry Pi untuk melakukan pemrosesan data secara langsung. Kombinasi antara kemampuan komputasi dari perangkat tersebut dengan algoritma pembelajaran mesin memungkinkan robot bekerja secara cerdas dan mandiri, misalnya mengenali objek yang akan diambil oleh *Arm* OpenManipulator-X atau menentukan tindakan berdasarkan kondisi lingkungan sekitar. Perkembangan ini membuat *machine learning* menjadi salah satu komponen penting dalam pengembangan sistem robotik berbasis kecerdasan buatan.



Gambar 2. 10 Arsitektur MobileNetV2

Gambar 2.10 menunjukan Arsitektur MobileNetV2 yang digunakan pada penelitian ini dirancang untuk melakukan proses ekstraksi fitur dan klasifikasi citra jeruk berdasarkan kondisi sehat dan tidak sehat. Struktur jaringan terdiri dari beberapa lapisan utama yang saling terhubung, di mana setiap lapisan memiliki fungsi tertentu dalam proses pembelajaran fitur visual. Rancangan arsitektur CNN yang digunakan dalam penelitian ini ditunjukkan pada tabel 3.1

Tabel 3. 1 Jenis dan Fungsi Layer pada MobilenetV2

Jenis Layer	**Fungsi**
Prepocessing	Normalisasi nilai piksel agar berada dalam rentang tertentu (biasanya 0-1) sehingga proses training lebih stabil dan cepat konvergen
Convolution Layer (3x3 Conv)	Ekstraksi fitur dan citra dengan mengalikan kernel/filter berukuran 3x3 terhadap area lokal pada citra
ReLU Activation	Menambahkan non-linearitas pada jaringan dengan memotong nilai negatif menjadi 0 agar model dapat mempelajari fitur kompleks
Max Pooling	Mengurangi dimensi fitur dengan memilih nilai maksimum pada setiap jendela 2x2 untuk mempertahankan fitur dominan
Fully Connected Layer	Menghubungkan seluruh neuron dari lapisan sebelumnya unuk melakukan pengambilan keputusan berdasarkan fitur yang telah diekstraksi
Softmax Layer (Output)	Mengubah skor keluaran menjadi nilai probabilitas antar kelas, di mana jumlah total probabilitas sama dengan 1.

Tabel 3.1 menunjukan jenis layer yang digunakan pada arsitektur MobileNetV2 yang rangkanya terletak pada gambar 2.10 Setiap layer memiliki fungsi dan rumus matematis yang menggambarkan proses pengolahan citra dari tahap awal hingga menghasilkan output klasifikasi. Proses dimulai dari tahap preprocessing, yaitu normalisasi nilai piksel agar jaringan dapat belajar secara lebih stabil. Selanjutnya, convolution layer (3×3 Conv) berfungsi untuk mengekstraksi fitur penting dari citra melalui operasi konvolusi antara kernel dan citra masukan. Hasil konvolusi

kemudian diproses oleh fungsi aktivasi ReLU untuk menambahkan non-linearitas dan menghilangkan nilai negatif yang tidak relevan lalu dilakukan max pooling 2×2, yang bertujuan untuk mengurangi ukuran citra (*downsampling*) dengan tetap mempertahankan fitur paling dominan. Setelah fitur-fitur diekstraksi dan direduksi, hasilnya diratakan (*flatten*) dan masuk ke dalam fully connected layer, di mana seluruh neuron saling terhubung untuk menggabungkan informasi fitur menjadi representasi yang lebih bermakna. Terakhir, softmax layer digunakan sebagai lapisan keluaran untuk mengubah nilai aktivasi menjadi probabilitas antar kelas, sehingga jaringan dapat menentukan kelas dengan probabilitas tertinggi sebagai hasil klasifikasi.

$$I'(x,y,c) = \frac{I(x,y,c)}{255} \quad (5)$$

a. I(x,y,c) = Intensitas piksel pada posisi (x,y) dan kanal warna c
b. I'(x,y,c) = Hasil normalisasi

Rumus (5) merupakan rumus normalisasi pada MobileNetV**2**. Proses mengubah nilai intensitas piksel gambar dari rentang asli 0–255 menjadi rentang 0–1

$$Zi,j,k = \sum_{m=0}^{2}\sum_{n=0}^{2}\sum_{c=0}^{C-1} Wm,n,c,k \cdot Xi+m,j+n,c + b_k \quad (6)$$

a. X = Input feature map
b. W = Bobot filter ukuran 3x3
c. Bk = Bias untuk filter ke-k
d. Zi,j,k = Output ke-k pada posisi (i,j)

Rumus (6) merupakan proses ekstraksi fitur gambar. Rumus ini menghitung jumlah perkalian antara nilai piksel input (X) dengan bobot filter (W) ditambah bias (b) untuk menghasilkan *feature map* baru.

$$f(x) = \max(0,x) \quad (7)$$

Rumus (7) berfungsi untuk menambahkan non-linearitas. Rumus ini menyaring nilai negatif menjadi 0 dan membiarkan nilai positif tetap, sehingga mempercepat komputasi tanpa menghilangkan informasi penting.

$$Z_j = \sum_{i=1}^{n} WijXi + b_j \quad (8)$$

Rumus (8) merupakan tahap klasifikasi di mana data fitur yang sudah didatarkan dikalikan dengan bobot (W) dan ditambah bias (b). Ini adalah proses menyusun keputusan sebelum masuk ke *output* akhir.

$$P(y = j|x) = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}} \quad (9)$$

Rumus (9) merupakan lapisan output terakhir yang mengubah hasil perhitungan angka menjadi nilai probabilitas. Nilai probabilitas tertinggi menentukan hasil prediksi akhir (apakah Jeruk Sehat atau Tidak Sehat).

2.10 Penelitian Terdahulu

Bagian ini membahas beberapa hasil penelitian terdahulu yang relevan dan dapat dijadikan sebagai dasar acuan dalam topik penelitian ini. Penelitian-penelitian tersebut telah dipilih secara selektif agar sesuai dengan fokus kajian yang diangkat, sehingga diharapkan dapat memberikan gambaran yang lebih jelas serta menjadi referensi pendukung dalam penyusunan dan penyelesaian penelitian ini. Berikut disajikan uraian mengenai beberapa penelitian terdahulu yang menjadi rujukan.

Penelitian pertama, oleh Oki Saputra (2024) berjudul Uji Kinerja Sistem Kontrol *Gripper* pada Robot Lengan untuk Pemetikan Tomat. Penelitian ini meyakini bahwa inovasi dibutuhkan untuk peningkatan efisiensi dalam pertanian salah satunya berupa pengenalan teknologi robotiika dalam pertanian yang memudahkan pekerjaan di lapangan. Metode yang dilakukan adalah metode eksperimental sehingga langkah-langkah eksperimen melibatkan peran lengan robot yang dilengkapi dengan *gripper*. Hasilnya *gripper* pada lengan dapat beroprasi secara efektif mencapai presentase keberhasilan 100% dan dapat diandalkan untuk pengembangan robotika bidang pertanian.

Penelitian kedua, oleh Muhammad Alif (2025) yang berjudul Pengendalian Gerakan Robot OpenManipulator Untuk Operasi Pemindahan Barang Berbasis MATLAB. Penelitian ini bertujuan untuk mengontrol kecepatan dan mengatur sinkronasi gerakan servo agar bisa memindahkan objek (tabung kimia) dari suatu titik ke titik lain sesuai gerakan robot industri sehingga jika terdapat bahan kimia yang berbahaya bagi manusia dapat digantikan oleh robot. Hasil dari penilitian ini menyatakan bahwa prototipe robot *Arm* sudah berhasil memindahkan objek ke satu titik ke titik lain tetapi masih pada dua titik terdekat dan pergerakannya tidak teratur sehingga dapat membuat objek jatuh.

Penelitian ketiga, berjudul *Automated fruit sorting system integrating image processing and support vector machine techniques* (Oyefeso et al., 2025). Penelitian ini membahas sistem sortir buah otomatis (jeruk, tomat, dan mangga) menggunakan kamera ESP32-CAM, pengolahan citra, dan algoritma SVM untuk klasifikasi. Hasilnya menunjukkan tingkat akurasi hingga 100%, dengan sistem mampu menyortir buah secara fisik ke wadah berbeda sesuai hasil klasifikasi. Penelitian ini menunjukkan bahwa *kombinasi machine learning* dan *image processing* efektif meningkatkan efisiensi dan akurasi penyortiran buah.

Penelitian keempat, dengan judul *CNN-MLP-Based Configurable Robotic Arm for Smart Agriculture* (Li et al., 2024) menjelaskan bahwa di tengah meningkatnya populasi global dan berkurangnya lahan pertanian yang dapat digarap, peningkatan produktivitas dan keberlanjutan di bidang pertanian menjadi sangat penting. Untuk mengatasi ketidakefisienan sistem pertanian tradisional yang kesulitan memenuhi kebutuhan produksi skala besar, penelitian tersebut memperkenalkan sistem *Configurable Agricultural Robotic Arm* (CARA), yang dirancang menggunakan *Convolutional Neural Network* (CNN) dan *Multilayer Perceptron* (MLP). Sistem ini mengintegrasikan lengan robot fleksibel, modul akuisisi citra, dan pusat pemrosesan berbasis kecerdasan buatan, guna mendukung berbagai tugas pertanian dengan presisi tinggi seperti pemanenan, penyemprotan pestisida, dan inspeksi tanaman. Hasil pengujian menunjukkan sistem mampu

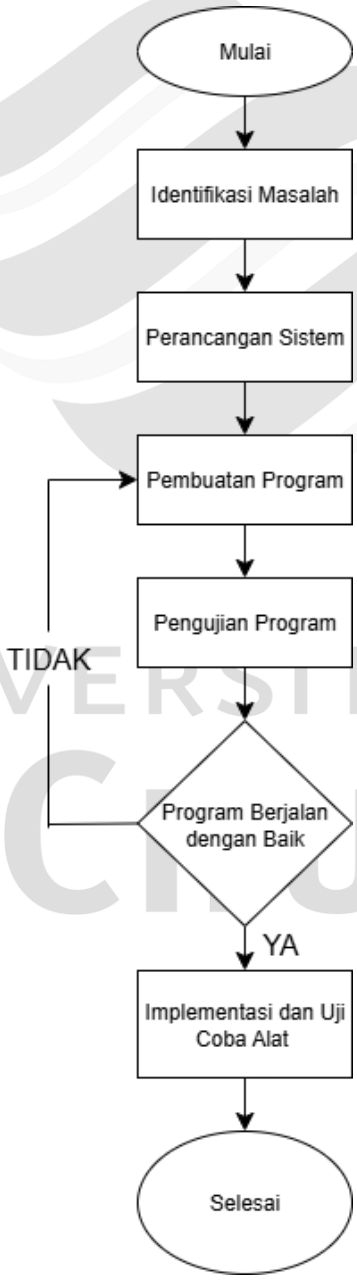
meningkatkan efisiensi operasional, beradaptasi dengan baik terhadap berbagai kondisi pertanian, serta memperkuat presisi dan keberlanjutan praktik pertanian.

BAB III

ANALISIS DAN PERANCANGAN SISTEM

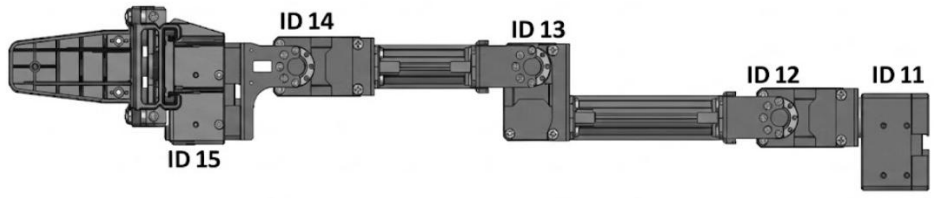
3.1 Tahapan Penelitian

Pada bagian ini dijelaskan tahapan-tahapan penelitian yang dilakukan untuk mencapai tujuan penelitian. Setiap tahapan disusun agar proses penelitian dapat berjalan dengan baik dan menghasilkan data yang valid sesuai dengan permasalahan yang diteliti.



Gambar 3. 1 Diagram alir tahapan penelitian

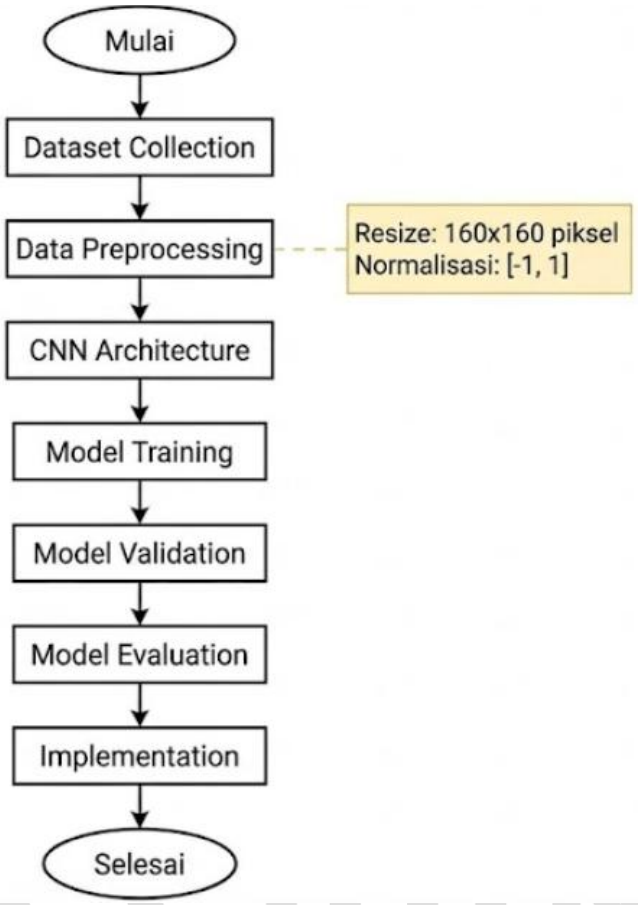
Gambar 3.1 memperlihatkan tahapan penelitian yang dilakukan dalam perancangan dan implementasi sistem penyortiran buah jeruk menggunakan robot *Arm* OpenManipulator berbasis Jetson Nano dan OpenCR. Pada tahap identifikasi masalah, diketahui bahwa proses penyortiran buah jeruk secara manual masih kurang efisien dan rentan terhadap kesalahan penilaian kualitas. Oleh karena itu, diperlukan sebuah sistem otomatis yang mampu membantu proses tersebut agar lebih cepat dan konsisten. Berdasarkan hasil identifikasi ini, dilakukan analisis kebutuhan untuk menentukan komponen perangkat keras dan perangkat lunak yang diperlukan. Proses ini diawali dengan pencarian berbagai sumber dan hasil penelitian sebelumnya yang membahas penerapan *machine learning* dan penggunaan robot *Arm* dalam penyortiran objek. Selanjutnya, dilakukan perancangan serta pembuatan program yang meliputi pelatihan model klasifikasi kualitas buah pada Jetson Nano dan pengendalian gerakan *robot Arm* melalui OpenCR. Dilakukan uji awal terhadap program guna memastikan bahwa pergerakan robot berfungsi sebagaimana mestinya. Jika hasil uji belum sesuai, dilakukan penyesuaian hingga sistem bekerja dengan baik. Setelah tahap tersebut berhasil, dilakukan tahapan untuk implementasi *machine learning*-nya. Hasil uji ini digunakan untuk menilai akurasi dan kinerja sistem serta menjadi dasar dalam penarikan kesimpulan penelitian.



Gambar 3. 2 ID Servo

Gambar 3.2 ID Servo menunjukan posisi serta penomoran masing-masing motor servo yang digunakan pada robot *Arm* OpenManipulator. Setiap servo memiliki ID yang berbeda-beda, dimulai dari bagian pangkal hingga ke ujung *gripper,* sehingga memudahkan proses identifikasi dan pengendalian setiap sendi secara terpisah. Pemberian ID ini sangat penting karena sistem pengendalian pada *Arm* OpenManipulator berbasis komunikasi digital melalui protokol Dynamixel, di mana setiap perintah yang dikirim dari mikrokontroler akan diterima hanya oleh

servo dengan ID yang sesuai. Dengan demikian, koordinasi antar-sumbu gerak dapat dilakukan secara sinkron dan presisi sesuai perintah yang diberikan melalui perangkat lunak pengendali. Selain itu, penomoran ID juga membantu dalam proses kalibrasi dan pemrograman gerakan, karena pengguna dapat dengan mudah mengetahui servo mana yang bertanggung jawab terhadap pergerakan tertentu, misalnya rotasi dasar, pergerakan lengan bawah, lengan atas, maupun pen*gripper* (*gripper*).



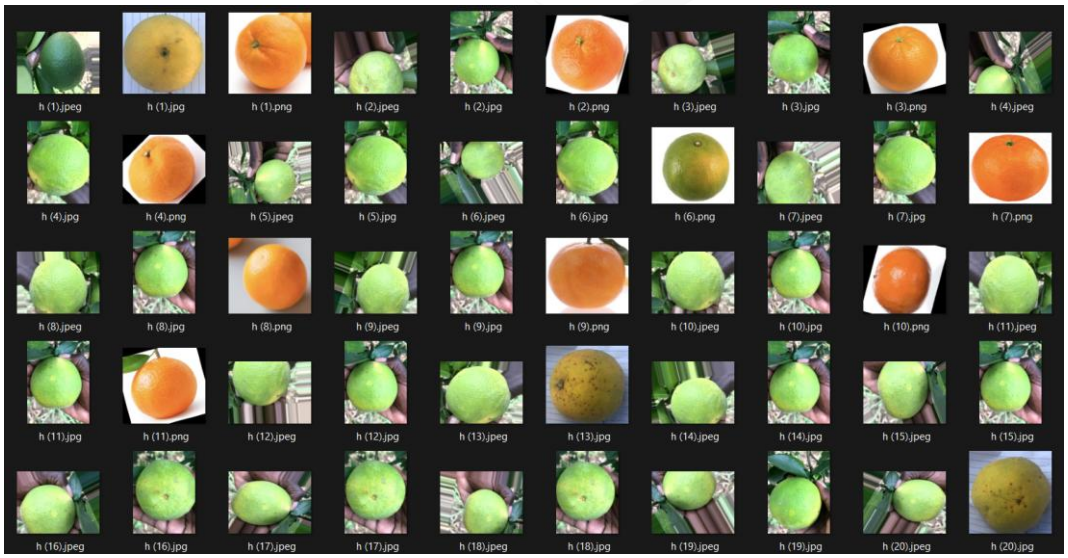
Gambar 3. 3 Diagram alur penerapan *Machine learning*

Gambar 3.3 menunjukan Diagram alur penerapan *Machine learning*. Tahap awal dilakukan pengumpulan data set berupa citra buah jeruk yang digunakan sebagai bahan penelitian model lalu dilakukan data prepocessing untuk mempersiapkan data agar siap diproses, seperti pengubahan ukuran gambar, normalisasi, dan augmentasi data, tahap berikutnya dilakukan perancangan CNN Architecture disini penulis membuat arsitektur jaringan CNN yang digunakan untuk mengenali pola pada citra jeruk. Setelah arsitektur terbentuk dilakukan Model

Training guna melatih model menggunakan data training diikuti oleh Model Validation untuk mengevaluasi kinerja model terhadap data validasi, tahap ini digunakan untuk mengukur performa model seperti akurasi, precision, recall, dan F1-score. Jika model dinyatakan memiliki performa terbaik maka dilakukan Implementation, yaitu penerapan model ke dalam sistem yang telah dirancang. Seluruh proses diakhiri dengan tahap Selesai yang menandakan model siap digunakan untuk melakukan klasifikasi citra jeruk secara otomatis.

3.1.1 DataSet

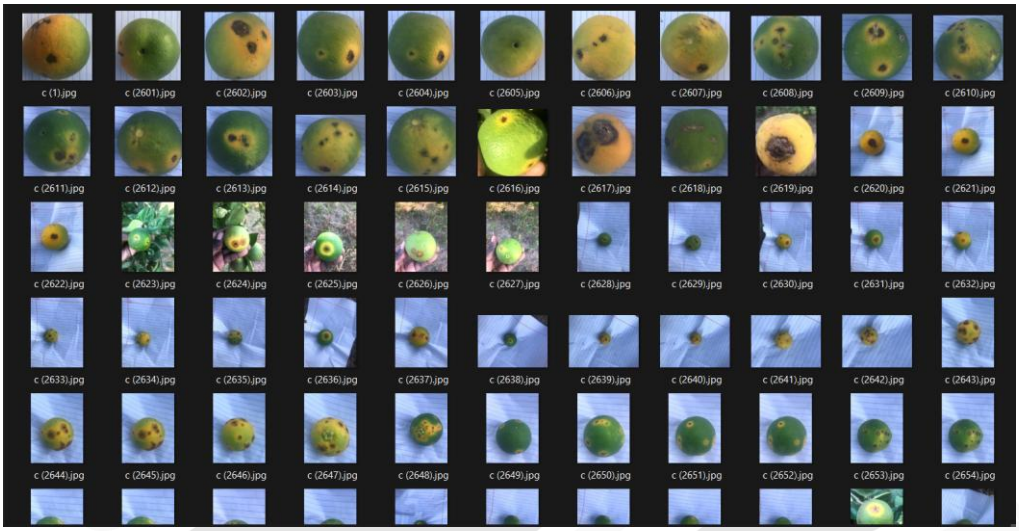
Dataset didapatkan melalui Mendeley Data, memiliki jumlah 3000 dataset per kategori nya (jeruk sehat, kanker jeruk, dan penyakit melanose) tujuan dataset ini dikumpulkan adalah untuk mengetahui jenis penyakit jeruk apa yang paling dominan ditemukan di wilayah timur Uganda. Dataset buah jeruk dikumpulkan menggunakan kamera ponsel dalam format *Highly Enhanced Image Container* (HEIC) dan kemudian dikonversi ke format JPEG. Penulis hanya menggunakan 2 kategori yaitu jeruk sehat, dan tidak sehat diambil dari 1500 dataset kanker jeruk dan 1500 dataset penyakit melanose.



Gambar 3. 4 Dataset jeruk sehat

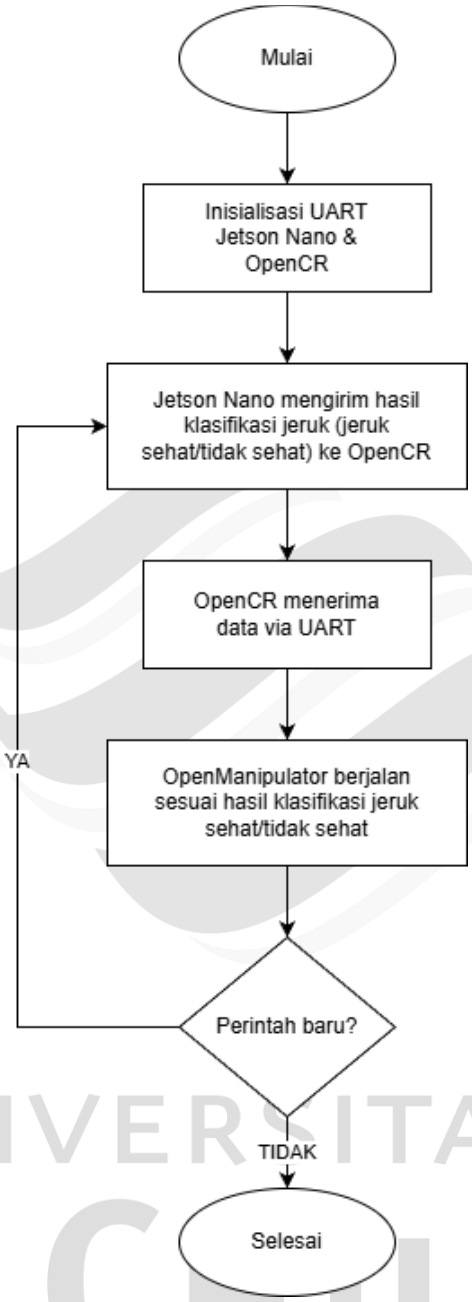
Gambar 3.4 menujukan dataset untuk kategori/label jeruk sehat yang digunakan dalam penelitian ini. Citra-citra pada kategori ini menampilkan jeruk dengan kondisi kulit yang baik, warna merata, dan tidak memiliki cacat fisik seperti bintik hitam, bercak busuk, atau perubahan warna mencolok. Data tersebut

digunakan sebagai representasi visual dari jeruk dengan kualitas normal yang akan menjadi pembanding terhadap kategori jeruk tidak sehat dalam proses pelatihan model *Machine learning*.



Gambar 3. 5 Dataset jeruk tidak sehat

Gambar 3.5 menunjukan dataset untuk kategori jeruk tidak sehat yang digunakan dalam penelitian ini. Kategori ini menampilkan kondisi kulit yang memiliki bercak hitam dari ukuran besar, kecil, banyak bercak maupun sedikit bercak, ada pula jeruk yang seluruhnya berwarna hitam karena terkena penyakit melanose.

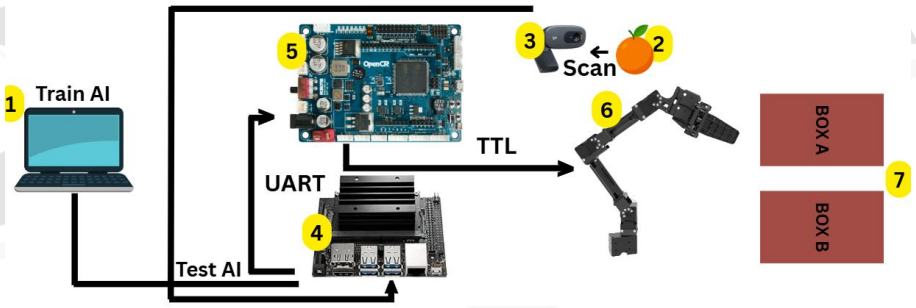


Gambar 3. 6 Diagram Alur Komunikasi Jetson Nano dengan OpenCR

Gambar 3.6 menunjukan Diagram alur komunikasi Jetson Nano dengan OpenCR, alur proses komunikasi antara Jetson Nano sebagai pengendali utama dan OpenCR sebagai penggerak robot *Arm* melalui protokol UART. Jetson Nano mengirimkan hasil klasifikasi dari model *machine learning* dalam bentuk data serial ke OpenCR. Setelah data diterima, OpenCR menginterpretasikan perintah tersebut untuk menggerakkan motor servo sesuai hasil klasifikasi, misalnya untuk mengarahkan atau memindahkan buah ke wadah tertentu.

3.2 Proses Pengerjaan

Proses pengerjaan penelitian ini dilakukan secara bertahap dan terstruktur agar setiap bagian dari sistem dapat berfungsi sesuai dengan tujuan yang telah ditetapkan. Tahapan dimulai dari tahap perancangan konsep sistem secara keseluruhan, yaitu menentukan kebutuhan perangkat keras dan perangkat lunak yang akan digunakan. Perangkat utama yang digunakan meliputi Jetson Nano sebagai pusat pemrosesan data, OpenCR sebagai pengendali robot *Arm* OpenManipulator, serta kamera sebagai alat pengambil citra buah jeruk. Pada tahap ini juga dirancang alur kerja sistem mulai dari pengambilan gambar, penerapan *machine learning* untuk analisis kualitas buah, hingga pengiriman hasil pengenalan ke OpenCR agar robot dapat melakukan penyortiran secara otomatis.



Gambar 3. 7 Topologi Perangkat

Gambar 3.7 merupakan topologi perangkat sehingga proses berjalannya prototipe dimulai dengan memberikan tegangan 10 volt melalui adaptor pada Jetson Nano dan OpenCR agar dapat *standalone*. Tahap awal (nomor 1) berupa komputer untuk melatih data, model yang telah jadi ditanam dan dijalankan oleh Jetson Nano (nomor 2) berupa citra jeruk yang tertangkap kamera (nomor 3) membuat Jetson Nano (nomor 4) menjalankan model AI untuk klasifikasi jeruk sesuai dengan model *machine learning* yang digunakan, *output* dari model dikirim (nomor 5) dari Jetson Nano ke OpenCR via UART. OpenCR kemudian memproses data tersebut untuk menggerakkan (nomor 6) robot *Arm* OpenManipulator sesuai perintah yang diterima, misalnya untuk memindahkan (nomor 7) jeruk sehat ke *Box* B dan jeruk tidak sehat ke *Box* A berdasarkan hasil analisis. Setelah berhasil memindahkan jeruk sesuai hasil klasifikasi robot *Arm* akan kembali ke posisi awal yaitu berada di *center*. Tahap akhir berupa tahap pengujian untuk memastikan seluruh komponen dapat

berfungsi dengan baik. Pengujian dilakukan terhadap proses komunikasi data, keakuratan pengenalan oleh *machine learning*, serta respon robot *Arm* terhadap perintah yang dikirim. Hasil pengujian kemudian dianalisis untuk menilai kinerja sistem secara keseluruhan. Jika ditemukan ketidaksesuaian, dilakukan perbaikan pada bagian terkait hingga sistem dapat bekerja sesuai tujuan penelitian.

Keterangan:

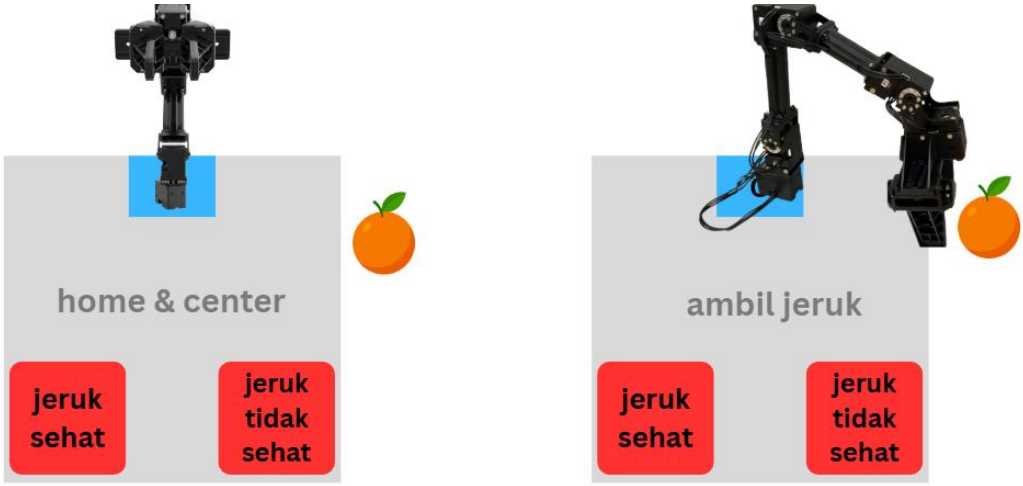
1. Objek/ buah jeruk
2. Kamera webcam logitech C270 HD
3. Jetson Nano
4. OpenCR 1.0
5. Robot *Arm* OpenManipulator-X
6. Wadah (*Box* A/*Box* B) untuk sortir

3.3 Metode Evaluasi

Metode evaluasi digunakan untuk menilai kinerja sistem yang telah dibuat, baik dari sisi penerapan *machine learning* maupun dari integrasi komunikasi antara Jetson Nano dan OpenCR dalam mengendalikan robot *Arm*. Dua jenis evaluasi dilakukan agar dapat diketahui sejauh mana sistem mampu bekerja sesuai dengan tujuan penelitian, yaitu melakukan penyortiran buah jeruk berdasarkan kualitas secara otomatis dan tepat.

Evaluasi model dilakukan untuk mengetahui seberapa akurat metode *machine learning* yang diterapkan, evaluasi difokuskan pada tingkat akurasi model dalam mengklasifikasikan citra buah jeruk. Hal ini dapat dinilai berdasarkan perbandingan antara hasil klasifikasi yang diberikan oleh sistem dengan kondisi aktual dari data uji. Sementara itu, pada evaluasi komunikasi serial, dilakukan pengujian ketepatan transmisi data antara Jetson Nano dan OpenCR menggunakan protokol UART yang dimonitoring melalui *serial monitor* untuk memastikan tidak terjadi keterlambatan atau kehilangan data selama proses pengiriman perintah, yang terakhir evaluasi akurasi pemindahan jeruk untuk mengetahui apakah pemindahan dan deteksi pada buah jeruk sudah berjalan dengan baik sesuai topologi perangkat.

Dengan demikian, hasil evaluasi dapat menjadi dasar untuk melakukan perbaikan dan penyempurnaan sistem agar dapat bekerja lebih optimal.



Gambar 3. 8 Ilustrasi Posisi OpenManipulator *Arm*

Gambar 3.8 merupakan ilustrasi posisi *arm* OpenManipulator karena memiliki keterbatasan jangkauan maka jarak antara *Arm* OpenManipulator ke *box* A dan B tidak jauh. Proses sortir berlangsung secara otomatis setelah sistem melakukan analisis terhadap citra jeruk seperti pada gambar 3.5. Dengan demikian, gambar ini menggambarkan tahapan akhir dari proses pengambilan keputusan berbasis *machine learning*, di mana hasil klasifikasi langsung berpengaruh terhadap tindakan fisik yang dilakukan oleh sistem, yaitu memindahkan objek sesuai dengan kategori yang telah ditentukan.

Keterangan:

1. Posisi awal terlihat pada gambar 3.8 (kiri), posisi mengambil terlihat pada gambar 3.8 (kanan)

2. Kotak sebelah kanan OpenManipulator merupakan tempat jeruk tidak sehat disortir

3. Kotak sebelah kiri OpenManipulator merupakan tempat jeruk sehat disortir

Tabel 3. 2 Tabel Pengujian

Jeruk	Hasil Deteksi	Benar / Salah	Akurasi (0/100 %)
Rata-rata akurasi			

Tabel 3.2 Hasil Pengujian menunjukan hasil dari penelitian yang diperoleh dari penelitian yang telah dilakukan, berdasarkan hasil tersebut, dapat diketahui bahwa sistem mampu bekerja sesuai dengan rancangan yang telah dibuat. Nilai-nilai yang ditunjukkan pada tabel menggambarkan kinerja serta tingkat keakuratan sistem dalam menjalankan fungsi utamanya.

Tabel 3. 3 Tabel Pengujian Pergerakan

Klasifikasi	Posisi pengambilan	Hasil Penempatan	Keterangan

Tabel 3.3 dirancang untuk mendokumentasikan hasil pengujian respon gerak dari robot *Arm* OpenManipulator. Pengujian ini bertujuan untuk memverifikasi akurasi penempatan objek berdasarkan klasifikasi yang diterima dari Jetson Nano, serta memastikan konsistensi posisi pengambilan dan peletakan.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Rincian Penelitian

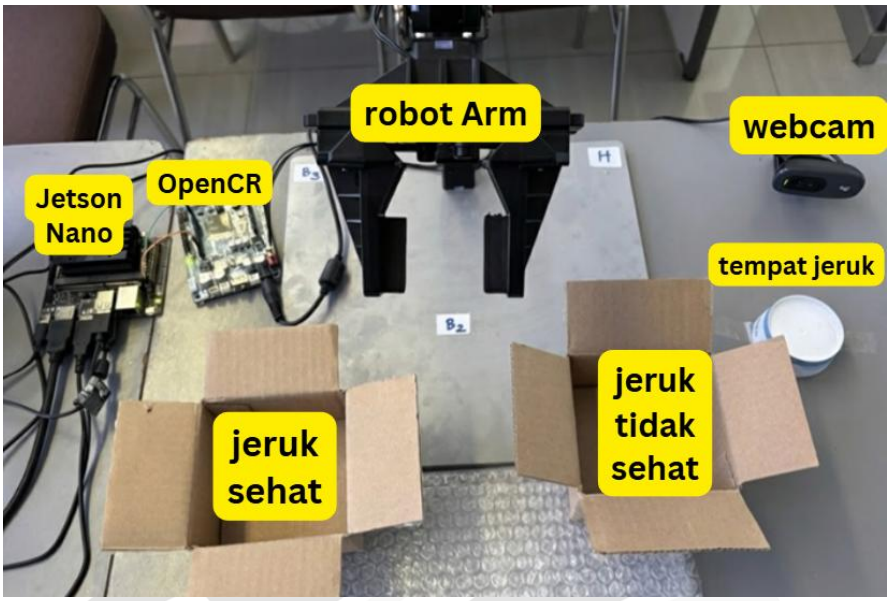
Pada subbab ini, penulis menguraikan rincian pelaksanaan penelitian yang mencakup aspek operasional dan teknis. Pembahasan diawali dengan deskripsi lokasi dan rentang waktu pelaksanaan penelitian guna memberikan gambaran konteks pengambilan data dan pengujian. Selanjutnya, dipaparkan rancangan topologi sistem yang diusulkan untuk menggambarkan alur komunikasi data dan arsitektur sistem secara keseluruhan. Bagian ini diakhiri dengan rincian spesifikasi perangkat keras (hardware) dan perangkat lunak (software) yang digunakan sebagai lingkungan pengembangan dan pengujian sistem.

4.1.1 Tempat dan Waktu

Penelitian dilaksanakan di lab Human Machine Interaction (HMI) gedung *Research and Development* (RnD) lt.6 Universitas Ma Chung Malang dengan Alamat Jalan Villa Puncak Tidar Blok N No. 1 Karangwidoro, Kecamatan Dau, Kabupaten Malang, Jawa Timur. Penelitian dimulai sejak Senin, 01 September 2025 dan diakhiri pada 23 Desember 2025. Hasil akhir penelitian memfokuskan uji sortir buah jeruk menggunakan sistem *standalone* berbasis robot *Arm* OpenManipulator dan board OpenCR, yang terintegrasi dengan modul kamera serta *machine learning* pada Jetson Nano untuk menyortir jeruk secara otomatis.

4.1.2 Alat dan Komponen

Alat dan komponen yang digunakan dalam penelitian ini berupa power adaptor 12 Volt sebagai input daya pada OpenCR dan 5 Volt pada JetsonNano, OpenCR sebagai penerima klasifikasi dan pengendali robot *Arm* OpenManipulator, robot *Arm* OpenManipulator sebagai robot penyortir, Jetson Nano sebagai tempat model *machine learning* dijalankan dan hasil klasifikasi di kirim ke OpenCR, webcam logitech C270 HD sebagai penangkap citra jeruk, kabel jumper sebagai penghubung komunikasi serial UART, monitor, *keyboard* dan *mouse* sebagai alat bantu pengendalian Jetson Nano, terakir berupa 20 buah jeruk dibagi menjadi 10 jeruk sehat dan 10 jeruk tidak sehat sebagai tahap terakir uji coba.



Gambar 4. 1 Rangkaian Prototipe

Gambar 4.1 memperlihatkan rangkaian keseluruhan prototipe yang mengacu pada topologi perangkat di Gambar 3.7. Komunikasi antara Jetson Nano dan OpenCR terjalin melalui protokol UART, di mana pin TX (GPIO_08) pada Jetson Nano dihubungkan ke pin RX (GPIO_0) pada OpenCR, serta menyatukan jalur ground menggunakan kabel jumper. Posisi kamera diarahkan langsung ke area penempatan jeruk, dan tersedia dua kotak penampung untuk hasil akhir penyortiran.

Gambar 4. 2 Input citra jeruk yang digunakan

Gambar 4.2 menunjukan input data berupa 10 buah jeruk yang digunakan penulis, terdiri dari 5 jeruk sehat dan 5 jeruk tidak sehat yang mirip dengan dataset yang telah dilatih.

Gambar 4. 3 Input citra jeruk yang digunakan (2)

Gambar 4.3 memperlihatkan input jeruk 10 buah terakir yang digunakan, terdiri dari 5 jeruk sehat dan 5 jeruk tidak sehat. Coretan spidol ditambahkan untuk menekankan bahwa jeruk tersebut berupa jeruk tidak sehat.

4.2 Pengambilan Data

Dataset yang digunakan dalam penelitian ini merupakan dataset citra buah jeruk yang dikelompokkan ke dalam dua kelas klasifikasi, yaitu kelas sehat (Gambar 3.4) dan tidak sehat (Gambar 3.5). Kumpulan data ini dipersiapkan untuk melatih arsitektur *Convolutional Neural Network* (CNN) berbasis MobileNetV2. Total citra yang dikumpulkan berjumlah 3000 gambar. Sebelum masuk ke tahap pelatihan, keseluruhan data tersebut dipilah melalui proses *spliting* dataset menjadi data latih (*train*) dan data uji (*test)* dengan rasio perbandingan 80:20 sehingga menghasilkan akurasi data seperti pada tabel 4.1 dan 4.2

Tabel 4. 1 Hasil akurasi data train

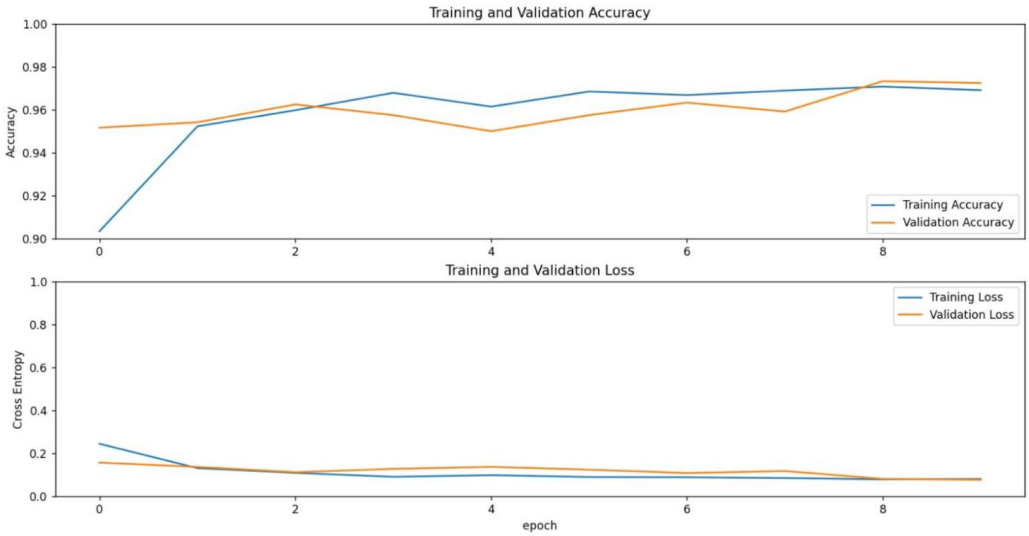
	precision	recall	f1-score	support
jeruk_sehat	0.96	0.98	0.97	2400
jeruk_tidak_sehat	0.98	0.96	0.97	2400
accuracy			0.97	4800
macro avg	0.97	0.97	0.97	4800
weighted avg	0.97	0.97	0.97	4800

Tabel 4.1 menunjukan hasil akurasi, recall, dan f1-score dari data latih (data training) dengan hasil yang memuaskan akurasi 97% , precision 96% untuk jeruk_sehat 98% untuk jeruk_tidak_sehat, recall 98% untuk jeruk_sehat 96% untuk jeruk_tidak_sehat, f1-score 97% untuk kedua kelas.

Tabel 4. 2 Hasil akurasi data test

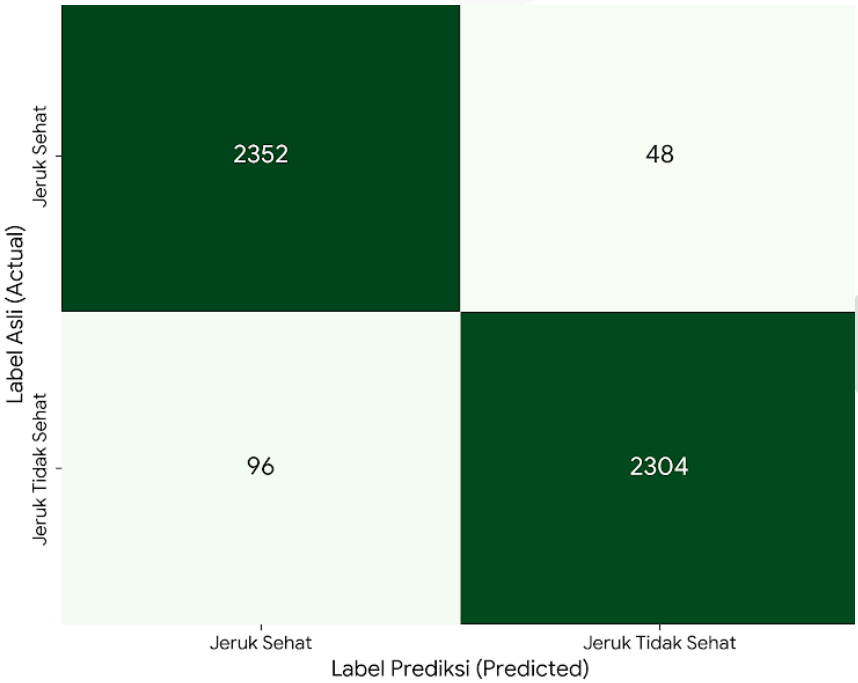
	precision	recall	f1-score	support
jeruk_sehat	0.97	0.98	0.97	600
jeruk_tidak_sehat	0.98	0.96	0.97	600
accuracy			0.97	1200
macro avg	0.97	0.97	0.97	1200
weighted avg	0.97	0.97	0.97	1200

Tabel 4.2 menunjukan hasil akurasi, recall, dan f1-score dari data uji (data test) dengan hasil yang memuaskan hampir sama dengan data latih akurasi 97% , precision 97% untuk jeruk_sehat 98% untuk jeruk_tidak_sehat, recall 98% untuk jeruk_sehat 96% untuk jeruk_tidak_sehat, f1-score 97% untuk kedua kelas menandakan bahwa model dapat mengklasifikasi buah jeruk dengan baik sesuai kelas.



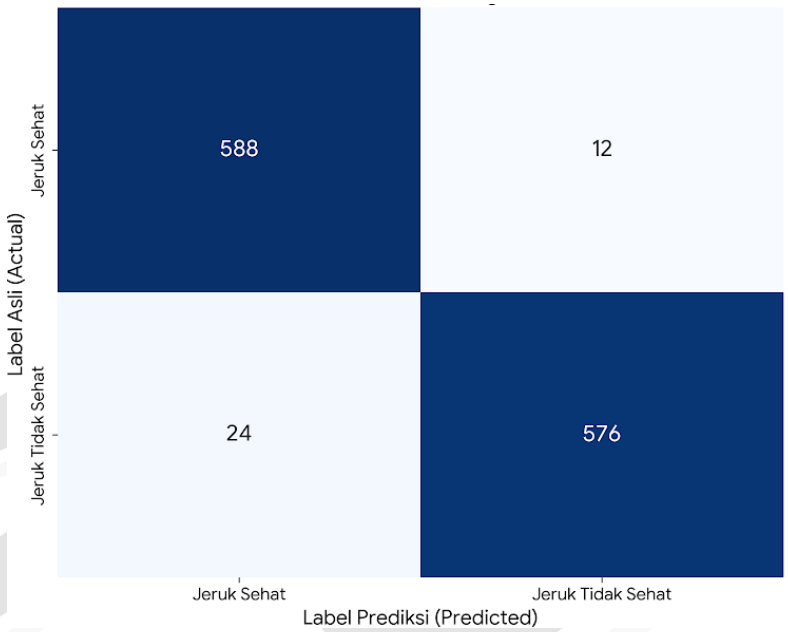
Gambar 4. 4 Grafik training validation

Gambar 4.4 menunjukkan hasil pelatihan selama 10 epoch yang mencapai konvergensi optimal. Grafik memperlihatkan peningkatan akurasi yang stabil hingga 0.97 dan penurunan loss di bawah 0.1. Kurva *training* dan *validation* yang bergerak beriringan menandakan model memiliki kemampuan generalisasi yang baik membuktikan tidak adanya indikasi *overfitting*.



Gambar 4. 5 Confusion matrix train data

Gambar 4.5 memvisualisasikan detail performa klasifikasi model yang merepresentasikan hasil evaluasi pada data *train* dengan total 2400 sampel. Pada tahap ini, model menunjukkan akurasi tinggi dengan berhasil mendeteksi 2352 sampel sebagai *jeruk sehat* dan 2304 sampel sebagai jeruk tidak sehat secara tepat.



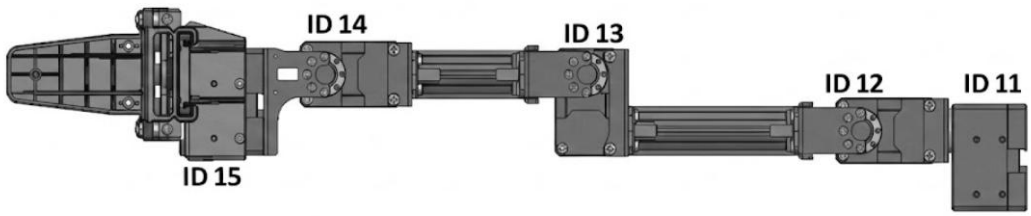
Gambar 4. 6 Confusion matrix test data

Gambar 4.6 memvisualisasikan detail performa klasifikasi model yang merepresentasikan hasil evaluasi pada data *test* dengan total 600 sampel. Pada tahap ini, model menunjukkan akurasi tinggi dengan berhasil mendeteksi 588 sampel sebagai *jeruk sehat* dan 576 sampel sebagai jeruk tidak sehat secara tepat.

Tabel 4. 3 Data posisi robot *Arm*

ID Servo	**Posisi *Home***	**Posisi Ambil**	**Posisi Jeruk Sehat**	**Posisi Jeruk Tidak Sehat**
ID 11	-0.057	1.000	-0.471	0.437
ID 12	-0.202	-0.202	0.753	0.718
ID 13	-0.397	-0.397	-1.002	-0.822
ID 14	1.150	1.150	1.534	1.163
ID 15	0.010	0.005	0.010	0.010

Tabel 4.3 memperlihatkan data *joint* robot *Arm* untuk posisi *home,* posisi mengambil buah jeruk, posisi sortir jeruk sehat dan posisi sortir jeruk tidak sehat. Data ini diambil dengan cara menjalankan program *open manipulator chain teaching pada* Arduino IDE agar bisa mendapat posisi / *joint* yang diinginkan. Sesuai dengan namanya *chain teaching* dilakukan dengan menggerakkan robot ke posisi yang sesuai lalu pada serial monitor akan menampilkan angka joint nya. Visualisasi id bisa dilihat pada gambar 4.4



Gambar 4. 7 ID robot *Arm*

Gambar 4.7 memperlihatkan visualisasi ID pada robot *Arm* dimana robot *Arm* terdiri dari 5 servo dengan ID 11 s/d ID 15.

4.3 Implementasi Kode Program

4.3.1 Kode Implementasi CNN

```
1. train_paths, val_paths, train_labels, val_labels
   = train_test_split(
2. all_filepaths,
3. all_labels,
4. test_size=0.2,
5. random_state=123,
6. stratify=all_labels
7. )

8. input_shape = IMG_SIZE + (3,)
9. preprocess_input =
   tf.keras.applications.mobilenet_v2.preprocess_in
   put
10.     base_model = MobileNetV2(
11.     input_shape=input_shape,
12.     include_top=False,
13.     weights='imagenet'
14.     )
15.     base_model.trainable = False
16.
17.     x = layers.GlobalAveragePooling2D()(x)
18.     x = layers.Dropout(0.2)(x
19.     outputs = layers.Dense(1,
   activation='sigmoid')(x)
```

Gambar 4. 8 Implementasi *machine learning* CNN

Gambar 4.8 menunjukan potongan kode untuk implementasi *machine learning* CNN. Pada baris 1-9 dataset dibagi menjadi 80% data latih dan 20% data validasi menggunakan metode *stratified* sampling pada variabel stratify . Metode ini menjamin bahwa model dilatih dan diuji dengan tingkat kesulitan yang seimbang untuk setiap kategori, sehingga menghindari bias pada hasil akurasi. Pada baris 8-15 bertujuan menginisialisasi arsitektur MobileNetV2 sebagai pengekstraksi fitur untuk metode *transfer learning*. Input diatur agar menerima citra berwarna (RGB) yang telah dinormalisasi sesuai standar model. Model dimuat menggunakan bobot pre-trained ImageNet tanpa lapisan klasifikasi bawaan (include_top=False), lalu seluruh parameternya dibekukan (trainable=False) agar kemampuan dasar model dalam mengenali fitur visual tidak rusak saat dilatih dengan dataset jeruk. Pada baris 17-19 menunjukan arsitektur klasifikasi tambahan yang dirancang untuk memproses fitur visual hasil ekstraksi MobileNetV2 menjadi

sebuah keputusan akhir. Dimulai dengan mereduksi dimensi data secara efisien menggunakan *global average pooling*, dilanjutkan dengan penerapan teknik regularisasi *dropout* untuk mencegah, dan diakhiri dengan lapisan *dense* tunggal beraktivasis *sigmoid* yang bertugas menghasilkan probabilitas biner (nilai 0 hingga 1) untuk menentukan kategori jeruk (sehat atau tidak). Hasil akhir code ini berupa model dalam format .npz agar sesuai dengan Jetson Nano.

4.3.2 Kode Transmitter

```
1. def build_manual_model():
2. inputs = Input(shape=IMG_SIZE + (3,))
3. x =
   tf.keras.applications.mobilenet_v2.preprocess_in
   put(inputs)
4. base_model = MobileNetV2(input_shape=IMG_SIZE +
   (3,), include_top=False, weights=None)
5. base_model.trainable = False
6. x = base_model(x, training=False)
7. x = layers.GlobalAveragePooling2D()(x)
8. x = layers.Dropout(0.2)(x)
9. x = layers.Dense(128, activation='relu')(x)
10.     outputs = layers.Dense(1,
   activation='sigmoid')(x)
11.     return tf.keras.Model(inputs, outputs)
```

Gambar 4. 9 Kode fungsi untuk menjalankan model

Gambar 4.9 menunjukan kode fungsi untuk menjalankan model .npz yang telah dilatih dengan *machine learning* CNN (baris 1-11) pada Jetson Nano dengan cara merekonstruksi ulang arsitektur model agar sama persis dengan struktur saat pelatihan. Fungsi ini menyusun lapisan input, *base model* MobileNetV2, dan lapisan klasifikasi tambahan agar siap dimuati oleh bobot dari .npz yang telah disimpan.

```
1. def cek_keberadaan_jeruk(frame):
2. hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
3. mask_orange = cv2.inRange(hsv, LOWER_ORANGE,
   UPPER_ORANGE)
4. mask_green = cv2.inRange(hsv, LOWER_GREEN,
   UPPER_GREEN)
5. combined_mask = cv2.bitwise_or(mask_orange,
   mask_green)
6. kernel = np.ones((5,5), np.uint8)
```

```
7. combined_mask = cv2.morphologyEx(combined_mask,
   cv2.MORPH_OPEN, kernel)
8. pixel_count = cv2.countNonZero(combined_mask)
9. return pixel_count > MIN_AREA_PIXEL,
   combined_mask
```

Gambar 4. 10 Kode fungsi untuk keberadaan jeruk

Gambar 4.10 menunjukan kode fungsi untuk mendeteksi keberadaan buah jeruk berdasarkan rentang warna oranye dan hijau dalam format HSV. Jika jumlah piksel warna yang terdeteksi melebihi batas tertentu (MIN_AREA_PIXEL = 5000), sistem akan menganggap ada jeruk dan mengizinkan model AI untuk mulai bekerja. Ini bertujuan agar AI tidak berjalan saat meja kosong. Jadi meskipun kamera menyala pada background apapun, jika tidak ada objek jeruk yang tertangkap pada kamera maka AI tidak akan berjalan (tidak adanya klasifikasi yang muncul). Setelah kamera dipastikan menangkap objek jeruk maka jeruk akan menjadi input dan diproses dengan kode pada gambar 4.11

```
1. img_input = cv2.resize(rgb, IMG_SIZE)
2. img_array = np.expand_dims(np.array(img_input),
   axis=0)


3. klasifikasi = model.predict(img_array)
4. score = klasifikasi[0][0]


5. if score < 0.5:
6. current_label = "JERUK SEHAT"
7. data_to_send = CMD_SEHAT
8. else:
9. current_label = "JERUK TIDAK SEHAT"
10.     data_to_send = CMD_TIDAK_SEHAT
```

Gambar 4. 11 Kode klasifikasi jeruk berdasarkan model *machine learning*

Gambar 4.11 merupakan proses klasifikasi, singkatnya gambar jeruk yang terdeteksi kamera akan menjadi input untuk menjalankan model, gambar tersebut akan diperkecil ke 160x160 agar sesuai dengan model MobileNetV2. Selanjutnya, model melakukan *predict* untuk menghasilkan nilai antara 0 atau 1. Penentuan keputusan dilakukan menggunakan nilai *threshold* sebesar 0.5. Jika skor klasifikasi

bernilai di bawah 0.5, sistem akan mengklasifikasikan objek sebagai jeruk sehat dan menyiapkan perintah CMD_SEHAT. Sebaliknya, jika skor berada di atas 0.5, objek diklasifikasikan sebagai jeruk tidak sehat dengan perintah CMD_TIDAK_SEHAT yang siap dikirimkan ke mikrokontroler.

```
1. if current_label == last_prediction:
2.                         consistent_frames += 1
3.                     else:
4.                         consistent_frames = 0
5.                     last_prediction =
   current_label
6.
7.                     if consistent_frames >=
   FRAMES_TO_VALIDATE:
8.                         print(f" MENGIRIM:
   {current_label}")
9.
10.
11.         if ser is not None: ser.close()
12.         cap.release()
13.         cv2.destroyAllWindows()
```

Gambar 4. 12 Kode transmitter dari Jetson Nano ke OpenCR

Gambar 4.12 menunjukan potongan kode untuk mengirim data yang telah berhasil di klasifikasi oleh CNN ke OpenCR melalui UART. Pada baris 1-7 menunjukan agar klasifikasi CNN lebih akurat, kamera harus menangkap sebanyak 15 frame dahulu (FRAMES_TO_VALIDATE = 15) jika dalam 15 frame hasilnya sama semua secara beruntun maka hasil tersebut akan dikirim ke OpenCR melalui kode baris 11-13. Serial port yang digunakan pada Jetson Nano agar bisa berkomunikasi dengan OpenCR dengan kabel jumper Bernama ttyTHS1.

4.3.3 Kode Receiver

```
1. double G_BUKA  = 0.010;
2. double G_TUTUP = 0.005;

3. void setup() {
4. Serial.begin(115200);
5. JETSON_SERIAL.begin(115200);

6. else if (step == 6) {
7. if (tujuan_sehat == false) {
```

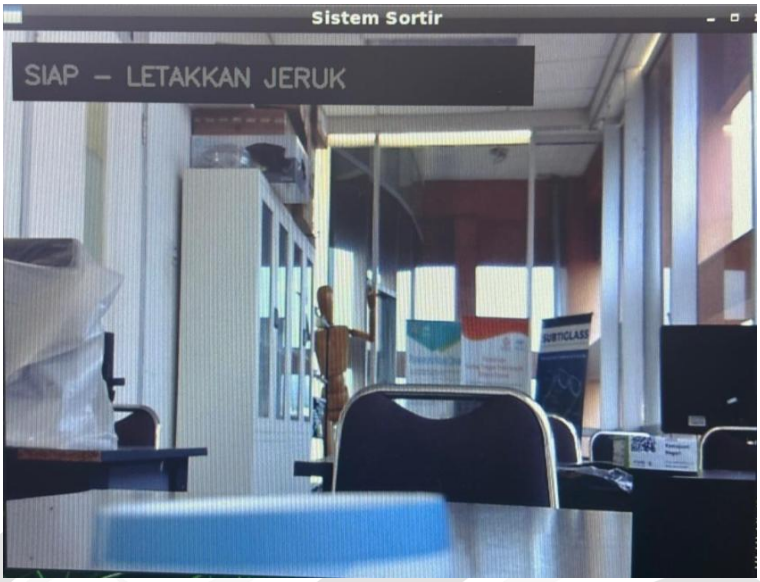
```
8. target.push_back(0.437);
   target.push_back(0.718); target.push_back(-
   0.822); target.push_back(1.163);
9. } else {
10.     target.push_back(-0.471);
   target.push_back(0.753); target.push_back(-
   1.002); target.push_back(1.534);
11.     }
12.     open_manipulator.makeJointTrajectory(target
   , 4.0);
13.     step++; step_timer = millis() + 5000;
14.     }

15.     }
```

Gambar 4. 13 Kode receiver dan menjalankan robot *Arm*

Gambar 4.13 memperlihatkan kode pada OpenCR yang menerima data dari Jetson Nano dan menjalankan robot *Arm* untuk klasifikasi ke posisi jeruk sehat dan jeruk tidak sehat. Baris 1 dan 2 mendefinisikan *joint gripper* untuk buka dan tutup, lalu baris 4 dan 5 menunjukan baudrate yang sama antara Jetson Nano dengan OpenCR agar data yang disampaikan bersih. Pada baris 6-15 menunjukan *joint* pergerakan untuk sortir ke jeruk sehat dan jeruk tidak sehat. Pada masa percobaan penulis melihat melalui serial monitor Arduino IDE apakah data klasifikasi dari Jetson Nano ke OpenCR berhasil terkirim, setelah berhasil dan di program sedemikian rupa sesuai dengan posisi klasifikasi baru usb antar laptop dan OpenCR dilepas agar 100% hardware (*standalone*).

4.3.4 Tampilan Jendela Monitoring



Gambar 4. 14 Jendela untuk memantau hasil klasifikasi pada Jetson Nano

Gambar 4.14 memperlihatkan kamera yang sudah siap untuk merekam citra jeruk, terlihat dari keterangan di atas kiri dengan tulisan siap – letakan jeruk. Lanjut ke tahap berikutnya yaitu meletakkan jeruk, setelah jeruk diletakkan barulah model machine learning akan melakukan klasifikasi apakah jeruk yang tertangkap kamera ini berupa jeruk sehat atau jeruk tidak sehat, untuk contoh jeruk sehat bisa dilihat pada gambar 4.15

Gambar 4. 15 Jendela untuk memantau memperlihatkan hasil jeruk sehat

Gambar 4.15 memperlihatkan keterangan jeruk sehat yang berasal dari klasifikasi machine learning terhadap jeruk sehat, garis putih pada bawah kotak jeruk sehat berjalan dari kiri ke kanan untuk menunjukan bahwa frame ditangkap sebanyak 15 kali dengan hasil yang sama, jika hasil berbeda dan belum sampai 15 kali frame maka garis putih akan mengulang kembali sampai berhasil berjalan 15 kali (garis putih full). Demikian pula hasil pada jeruk tidak sehat gambar 4.16

Gambar 4. 16 Jedela untuk memantau memperlihatkan hasil jeruk tidak sehat

Gambar 4.16 memperlihatkan keterangan jeruk tidak sehat yang berasal dari klasifikasi machine learning terhadap jeruk tidak sehat, dengan indicator warna merah yang memperlihatkan kondisi seakan jeruk tersebut tidak sehat.

Gambar 4. 17 Jendela untuk memantau memperlihatkan kondisi setelah machine learning berhasil mengklasifikasi

Gambar 4.17 memperlihatkan kondisi dimana machine learning telah berhasil mengklasifikasi suatu jeruk, lalu menunggu diambil oleh robot Arm, sehingga untuk sortir jeruk selanjutnya menunggu jeruk yang didepan kamera diambil, setelah berhasil diambil oleh robot arm, jendela pemantau akan kembali pada keterangan siap – letakkan jeruk.

4.4 Hasil Pengujian

Hasil pengujian terdiri dari pengujian pada klasifikasi *machine learning* dan pengujian ketepatan pada pergerakan robot *Arm.*

Tabel 4. 4 Pengujian klasifikasi machine learning

Jeruk	**Hasil Deteksi**	**Benar / Salah**	**Akurasi (0/100 %)**
Sehat	Sehat	Benar	100%
Sehat	Sehat	Benar	100%
Tidak sehat	Tidak sehat	Benar	100%
Tidak sehat	Tidak sehat	Benar	100%
Sehat	Tidak sehat	Salah	0%
Sehat	Sehat	Benar	100%
Tidak sehat	Tidak sehat	Benar	100%
Tidak sehat	Tidak sehat	Benar	100%
Sehat	Sehat	Benar	100%
Tidak sehat	Tidak sehat	Benar	100%
Tidak sehat	Tidak sehat	Benar	100%
Sehat	Sehat	Benar	100%
Sehat	Sehat	Benar	100%
Tidak sehat	Sehat	Salah	0%
Sehat	Sehat	Benar	100%
Sehat	Sehat	Benar	100%
Tidak sehat	Tidak sehat	Benar	100%
Sehat	Sehat	Benar	100%
Tidak sehat	Tidak sehat	Benar	100%

Jeruk	Hasil Deteksi	Benar / Salah	Akurasi (0/100 %)
Tidak sehat	Tidak sehat	Benar	100%
Rata-rata akurasi			90%

Pada tabel 4.4 terlihat bahwa rata-rata akurasi yang didapatkan pada pengujian ini sebesar 90%, dalam percobaan kamera menangkap 20 citra jeruk mendapatkan hasil yang salah dalam mendeteksi 1 jeruk sehat dan 1 jeruk tidak sehat. Membuktikan bahwa *machine learning* tergolong akurat, kesalahan yang terjadi kebanyakan diakibatkan pencahayaan dan *background* saat menangkap citra jeruk, dapat dibuktikan saat jeruk diarahkan lagi di kamera hasilnya sesuai dengan kenyataan.

Tabel 4. 5 Tabel hasil pengujian gerak robot *Arm*

Klasifikasi	Posisi Pengambilan	Hasil Penempatan	Keterangan
Jeruk sehat	✓	✓	Berhasil
Jeruk sehat	✓	✓	Berhasil
Jeruk tidak sehat	✓	✓	Berhasil
Jeruk tidak sehat	✓	✓	Berhasil
Jeruk sehat	✓	✓	Berhasil
Jeruk sehat	✓	✓	Berhasil
Jeruk tidak sehat	✓	✓	Berhasil
Jeruk tidak sehat	✓	✓	Berhasil
Jeruk sehat	✓	✓	Berhasil
Jeruk tidak sehat	✓	✓	Berhasil
Jeruk tidak sehat	✓	✓	Berhasil
Jeruk sehat	✓	✓	Berhasil
Jeruk sehat	✓	✓	Berhasil
Jeruk tidak sehat	✓	✓	Berhasil
Jeruk sehat	✓	✓	Berhasil
Jeruk sehat	✓	✓	Berhasil
Jeruk tidak sehat	✓	✓	Berhasil
Jeruk sehat	✓	✓	Berhasil
Jeruk tidak sehat	✓	✓	Berhasil
Jeruk tidak sehat	✓	✓	Berhasil

Pada tabel 4.5 memperlihatkan hasil pengujian gerak robot *Arm* yang seluruhnya mengalami keberhasilan, membuktikan bahwa komunikasi UART dapat berjalan dengan baik dan tidak ada kesalahan kode pada receiver di OpenCR, simbol ✓ menandakan posisi pengambilan dan penempatan yang benar.

Gambar 4. 18 Waktu satu siklus klasifikasi

Gambar 4.10 secara spesifik memperlihatkan waktu klasifikasi 15 frame pada Jetson Nano adalah 3 (tiga) detik dan waktu tanggap robot setelah menerima data klasifikasi bersifat *real-time* jedanya <1ms sedangkan waktu yang dibutuhkan robot untuk menyelesaikan satu proses pemindahan, dimulai dari posisi tengah, turun mengambil jeruk, hingga meletakkannya ke wadah kategori jeruk sehat / sakit adalah 28 (dua puluh delapan) detik. Durasi ini merupakan hasil penjumlahan dari delay yang ditetapkan pada setiap langkah pergerakan program. Rincian pergerakan meliputi waktu 4 (empat) detik untuk joint saat turun mengambil jeruk dan membawa jeruk keposisi tengah, 2 (dua) detik saat gripping, dan 5 (lima) detik untuk posisi mengangkat jeruk dan menaruh ke wadah. Sehingga untuk satu siklus penuh total waktu yang dihabiskan berjumlah 31(tiga puluh satu) detik.

4.5 Analisis dan Pembahasan

Sistem integrasi antara Jetson Nano dan OpenManipulator terbukti mampu beroperasi secara *standalone* dengan kinerja yang andal. Salah satu keunggulan signifikan dari sistem ini adalah model *MobileNetV2* dalam melakukan klasifikasi pada latar belakang (*background*) yang beragam. Kemampuan ini didasari oleh

prinsip kerja *Convolutional Neural Network* (CNN) yang tidak memproses citra sebagai satu kesatuan gambar statis, melainkan berfokus pada ekstraksi fitur visual (*feature extraction*). Model ini telah terlatih untuk mengenali karakteristik intrinsik yang melekat pada objek jeruk seperti tekstur pori kulit, gradasi warna, dan geometri bentuk, sehingga latar belakang dianggap sebagai *noise* yang tidak mempengaruhi hasil klasifikasi. Oleh karena itu, selama variabel lingkungan seperti jarak pengambilan citra antara kamera dan objek serta kondisi pencahayaan ruangan dijaga konstan, sistem mampu memisahkan objek utama dari latar belakangnya secara efektif tanpa memerlukan penyeragaman background. Selain kemampuan adaptasi visual tersebut, ketepatan pemilahan juga dijamin oleh penerapan algoritma validasi temporal sebanyak 15 *frame* sebelum pengiriman instruksi agar lengan robot hanya menerima perintah eksekusi yang konsisten guna meminimalisir kesalahan gerak pada aktuator.

Meskipun model mampu beradaptasi dengan berbagai latar belakang, instabilitas intensitas cahaya yang ekstrem tetap menjadi kendala yang dapat mendistorsi fitur visual dan menyebabkan kesalahan klasifikasi. Di sisi mekanik, batasan fisik lebar bukaan *gripper* menjadi hambatan utama, di mana proses pengambilan dan pengangkatan seringkali gagal pada jeruk berukuran besar, sehingga sistem saat ini hanya mampu beroperasi secara optimal pada objek uji berupa jeruk varietas Siam berukuran medium. Potensi kegagalan lainnya terletak pada aspek integrasi perangkat keras, seperti kesalahan pendefinisian alamat *port* serial atau kualitas penyambungan kabel *jumper* pada pin GPIO yang kurang presisi, yang dapat memutus transmisi data instruksi dari Jetson Nano ke OpenCR meskipun deteksi visual berhasil dilakukan.

BAB V

SIMPULAN DAN SARAN

5.1 Simpulan

Berdasarkan hasil perancangan dan pengujian yang telah dilakukan, penelitian ini berhasil mengimplementasikan sistem pengendalian gerakan robot *OpenManipulator* berbasis *board* OpenCR yang terintegrasi secara penuh dengan modul kamera dan Jetson Nano untuk melakukan penyortiran otomatis. Sistem terbukti mampu beroperasi secara mandiri (*standalone*), di mana komunikasi data melalui jalur serial UART berjalan stabil dalam menghubungkan hasil pemrosesan citra cerdas dengan instruksi pergerakan robot.

Dalam hal kinerja penyortiran, lengan robot menunjukkan kemampuan yang andal dalam memisahkan jeruk berdasarkan klasifikasi kondisi sehat dan tidak sehat. Keakuratan aksi mekanik ini sangat didukung oleh penerapan logika validasi temporal pada perangkat lunak, yang secara efektif meredam fluktuasi pembacaan data sehingga robot hanya mengeksekusi perintah saat hasil deteksi sudah konsisten. Hal ini memastikan bahwa setiap gerakan penyortiran yang dilakukan robot benar-benar sesuai dengan keputusan sistem *machine learning*.

Secara keseluruhan, sistem ini telah menunjukkan efektivitas yang baik sebagai gambaran awal penerapan otomasi pada industri pangan, khususnya dalam aspek akurasi logika pemilahan. Namun, evaluasi terhadap efisiensi penanganan fisik menunjukkan adanya batasan pada mekanisme *gripper* bawaan yang hanya bekerja optimal pada jeruk varietas Siam berukuran sedang. Kendala teknis masih ditemukan saat menangani jeruk dengan dimensi yang lebih besar, di mana keterbatasan lebar bukaan *gripper* seringkali menyebabkan kegagalan pengangkatan, sehingga aspek fleksibilitas mekanik ini menjadi catatan penting dalam penilaian efisiensi sistem secara menyeluruh.

5.2 Saran

Untuk penyempurnaan sistem di masa mendatang, fokus utama sebaiknya diarahkan pada pembaruan mekanisme robot dan sistem visi komputer. Disarankan untuk beralih menggunakan robot dengan mekanisme vacuum gripper (sistem

hisap) sebagai alternatif penjepit mekanis, guna meminimalkan risiko kerusakan fisik pada buah sehingga robot bisa memilah jeruk jika dalam total yang banyak. Selain itu, sistem deteksi visual perlu ditingkatkan dengan mengimplementasikan konfigurasi multi-kamera atau menggunakan kamera yang dapat berputar 360 derajat. Peningkatan ini bertujuan agar jeruk dapat dianalisis dari berbagai sudut pandang dengan minimal dua sisi, sehingga penilaian kualitas buah menjadi jauh lebih akurat dan menyeluruh.

DAFTAR PUSTAKA

Automation, P., & Automation, P. (2024). *Facilitating Smooth Handovers with a Gesture-Control Glove for Assistive Robotic Manipulators Dilara Yigit*.

Bao, X., Li, L., Ou, W., & Zhou, L. (2022). Robot intelligent grasping experimental platform combining Jetson NANO and machine vision. *Journal of Physics: Conference Series*, *2303*. https://doi.org/10.1088/1742-6596/2303/1/012053

De Assis, J. P. B., Günther, G., Pellenz, M., & Teixeira, M. (2025). Computer Vision and Machine Learning-Based Control for a 6-Degree-of-Freedom Robotic Arm. *2025 IEEE International Conference on Industrial Technology (ICIT)*, 1–6. https://doi.org/10.1109/ICIT63637.2025.10965279

Hj, O., & Rostiati, N. (2015). *WILLINGNESS TO PAY KONSUMEN TERHADAP BUAH JERUK IMPOR (Perspektif Konvensional vs Islam)*. *1*, 23–38.

Inuwa, A. B., Ahmad, A. S., Auwal, S. T., Muhammad, M., Lukuman, A., & Babangida, I. (2025). *JOURNAL OF SUSTAINABLE ENGINEERING AND Advances in Sensor-Based and Machine Learning Techniques for Automated Tomato Sorting and Grading : A Review*. *2*(1), 1–12.

Kashani, M. R., Malekzadeh, M., Mansfield, D., & Montazeri, A. (2025). Robust Control of OpenMANIPULATOR-X Using Reinforcement Learning. *IFAC-PapersOnLine*, *59*(10), 703–708. https://doi.org/https://doi.org/10.1016/j.ifacol.2025.09.120

Li, M., Wu, F., Wang, F., Zou, T., Li, M., & Xiao, X. (2024). CNN-MLP-Based Configurable Robotic Arm for Smart Agriculture. *Agriculture (Switzerland)*, *14*(9). https://doi.org/10.3390/agriculture14091624

Oyefeso, B. O., Oyewande, O. E., & Audu, J. (2025). Automated fruit sorting system integrating image processing and support vector machine techniques. *International Journal of AI for Materials and Design*, *2*(2), 79–90. https://doi.org/https://doi.org/10.36922/IJAMD025150011

Robianto, R., Sitorus, S. H., & Ristian, U. (2021). Penerapan Metode Decision Tree Untuk Mengklasifikasikan Mutu Buah Jeruk Berdasarkan Fitur Warna Dan Ukuran. *Coding Jurnal Komputer Dan Aplikasi*. https://api.semanticscholar.org/CorpusID:270644864

Shen, D.-R., Chin, H.-L., Tu, C.-H., Chih, J.-S., Venglář, V., Chen, K.-S., Krejsa, J., & Vechet, S. (2021). Navigation and Task Planning of a Mobile Robot under ROS Environment: A Case Study Using AutoRace Challenge. *2021 60th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 612–617.

Wang, T. (2021). *Camera-based Augmented Reality Endoscope Auxiliary System By*. *9*.

Winarta, D., Widodo, R., & Subianto, M. (2024). Rancang Bangun Pengontrol Gerakan Robot Openmanipulator Dengan Matlab. *Sainsbertek Jurnal Ilmiah*

Sains & Teknologi, *5*, 83–91. https://doi.org/10.33479/sb.v5i1.343

Xu, L. (2018). Remote Control and Monitoring System of Robotic Arm Using Raspberry Pi. *Advances in Intelligent Systems and Computing*. https://doi.org/10.1007/978-3-030-00214-5_81

Kernighan, B. W., & Ritchie, D. M. (1988). *The C programming language* (2nd ed.). Prentice Hall.

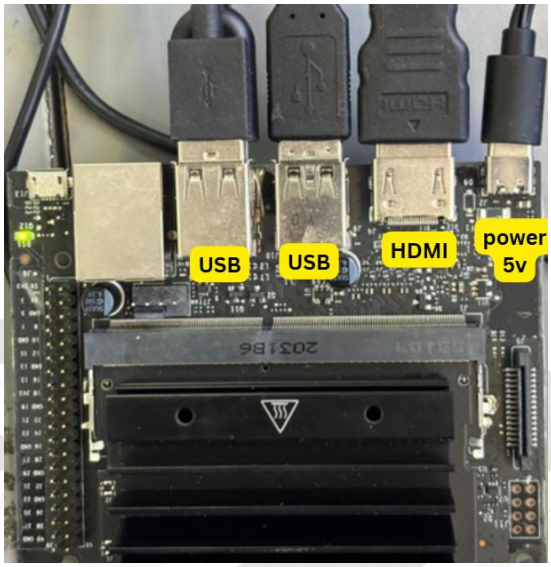
Banzi, M., & Shiloh, M. (2014). Getting started with Arduino: The open source electronics prototyping platform (3rd ed.). Maker Media.

ROBOTIS. (2024). *OpenManipulator-X*. ROBOTIS e-Manual. https://emanual.robotis.com/docs/en/platform/openmanipulator_x/

Lampiran

Lampiran A. Manual Book

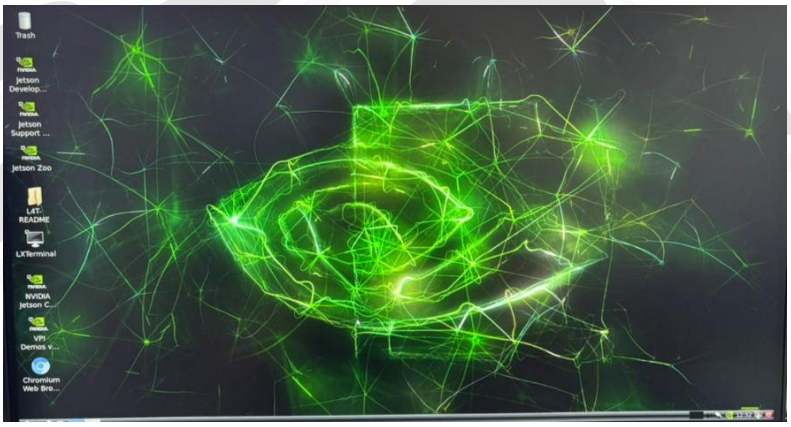
1. Tahap menyalakan Jetson Nano
   a. Hubungkan ke power adaptor (5V), HDMI, Keyboard, Mouse, Webcam portnya untuk USB bebas



   b. Jika tidak muncul gambar seperti ini saat booting, maka SD Card harus di flash ulang.
   Refrensi: https://www.youtube.com/watch?v=IVw1cyanRi0

   c. Selanjutnya jika muncul gambar seperti diatas, maka tampilan selanjutnya adalah anda diminta untuk memasukan password, username: machung, password: machung123

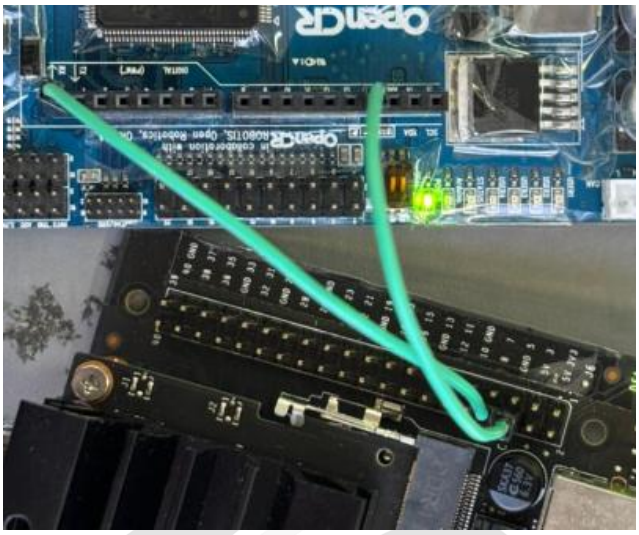
d. Tampilan jika anda sudah berhasil masuk ke Jetson Nano



e. Memindahkan data dari PC ke Jetson Nano saya menggunakan wifi (lewat drive) bisa juga menggunakan flashdisk. Jika melalui drive maka bisa masuk ke chromium web browser bawaan jetson nano (gambar point d pojok kiri bawah)

f. Masuk ke LXTerminal, dan masuk directory tempat filemu, contoh file saya ada di Downloads, maka perintahnya adalah cd Downloads, lalu tinggal jalankan file nya, saya menggunakan bahasa pemrograman python, jadi diawali dengan python3 lalu "nama file.py" ini merupakan proses uploading ke jetson nano

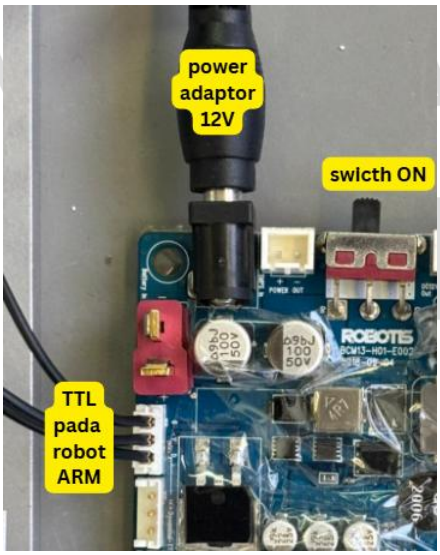
g. Pastikan kabel Jumper pada Jetson Nano dan OpenCR sudah benar, Jetson Nano GPIO_08 terhubung dengan OpenCR GPIO_0, dan GND saling terhubung.



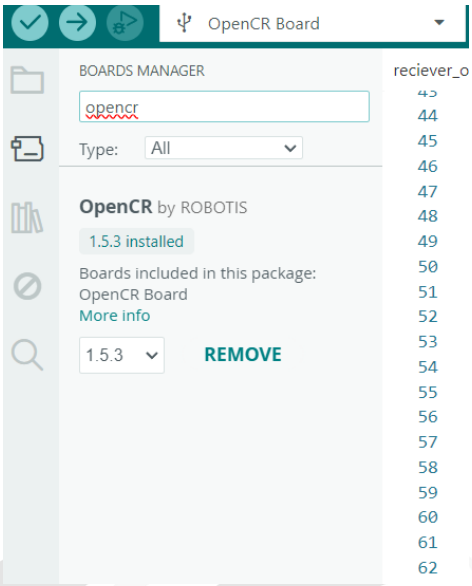
h. Jika UART tidak bisa connect, keluar dengan cara ctrl+z lalu ketik command sudo chmod 666/ dev/ttyTHS1 lalu run ulang programnya

2. Tahap menyalakan dan menggerakan OpenManipulator

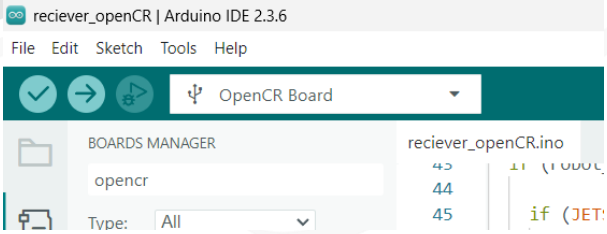
a. Setup OpenCR karena OpenCR sebagai driver, hubungkan openCR dengan power adaptor 12V, sambungkan TTL dari OpenManipulator ke OpenCR, pastikan bahwa switch ON, jika OFF maka robot tidak akan bergerak



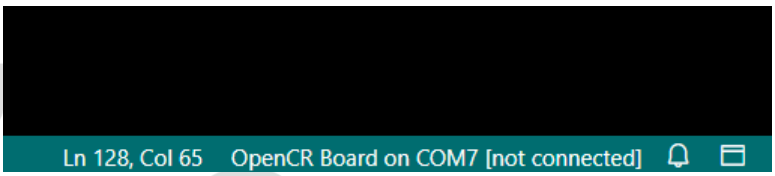
b. Untuk upload program pada OpenCR, anda harus mengunduh board OpenCR pada Arduino UNO.
Refrensi: https://emanual.robotis.com/docs/en/software/arduino_ide/



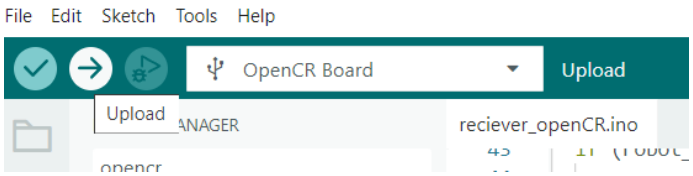
c. Saat board sudah terdownload dibutuhkan usb yang terhubung dari laptop/pc ke OpenCR, dan memilih menu board menjadi OpenCR, sesuaikan dengan COM pada board manager anda



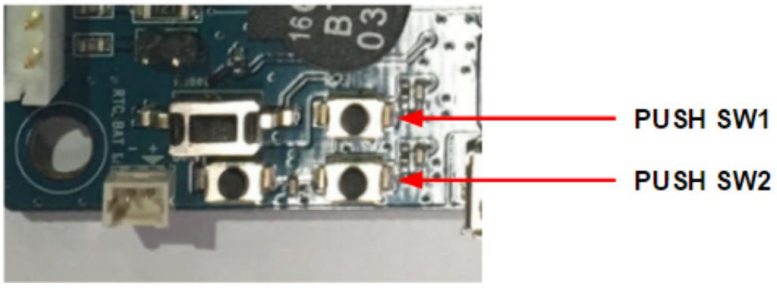
d. Pastikan bahwa indikasi pojok kanan bawah adalah connected, jika not connected bisa menekan tombol reset.



e. Setelah sudah connect, selanjutnya upload dengan cara menekan tombol panah ke kanan, setelah dilakukan uploading maka sudah tersimpan di chip anda bisa melepas usb yang terhubung dengan OpenCR.



f. Jika proses uploading gagal maka anda harus masuk ke bootloader dengan cara menekan Reset + SW2.
Refrensi:
https://emanual.robotis.com/docs/en/parts/controller/opencr10/



Lampiran B. Kode Training CNN

```
1. import tensorflow as tf
2. from tensorflow.keras import layers, models
3. from tensorflow.keras.applications import
   MobileNetV2
4. import matplotlib.pyplot as plt
5. import numpy as np
6. from sklearn.metrics import
   classification_report
7. from sklearn.model_selection import
   train_test_split # <-- KITA PAKAI INI
8. import os
9. import sys

10.     # --- 1. Konfigurasi Awal ---
11.     IMG_SIZE = (160, 160)
12.     BATCH_SIZE = 16
13.     EPOCHS = 10
14.     LEARNING_RATE = 0.0001
15.     DATA_DIR = 'dataset'

16.     print("--- Membaca daftar file dari
   'dataset' ---")

17.     all_filepaths = []
18.     all_labels = []

19.     class_names = sorted(os.listdir(DATA_DIR))
20.     if len(class_names) < 2:
21.     print(f"ERROR: Hanya ditemukan 1 kelas di
   {DATA_DIR}. Periksa folder dataset Anda.")
22.     sys.exit(1)

23.     print(f"Kelas terdeteksi: {class_names}")

24.     for i, class_name in
   enumerate(class_names):
25.     class_dir = os.path.join(DATA_DIR,
   class_name)
```

```
    if not os.path.isdir(class_dir):
    continue

    filepaths = [os.path.join(class_dir, f) for
  f in os.listdir(class_dir) if
  f.endswith(('.jpg', '.jpeg', '.png'))]
    labels = [i] * len(filepaths)

    all_filepaths.extend(filepaths)
    all_labels.extend(labels)

    print(f"Total gambar ditemukan:
  {len(all_filepaths)}")
    if len(all_filepaths) == 0:
    print("ERROR: Tidak ada gambar
  .jpg/.jpeg/.png yang ditemukan di dalam folder
  dataset.")
    sys.exit(1)

    print("--- Membagi data (80% train, 20%
  validation) ---")
    train_paths, val_paths, train_labels,
  val_labels = train_test_split(
    all_filepaths,
    all_labels,
    test_size=0.2,
    random_state=123,
    stratify=all_labels
    )

    print(f"Total data latih:
  {len(train_paths)}")
    print(f"Total data validasi:
  {len(val_paths)}")


    def load_and_preprocess_image(path, label):
    """Fungsi untuk memuat, decode, dan resize
  gambar"""
    image = tf.io.read_file(path)
    image = tf.image.decode_jpeg(image,
  channels=3)
    image = tf.image.resize(image, IMG_SIZE)
    return image, label

    AUTOTUNE = tf.data.AUTOTUNE
```

```
    train_dataset = tf.data.Dataset.from_tensor_slices((train_paths, train_labels))
    train_dataset = train_dataset.map(load_and_preprocess_image, num_parallel_calls=AUTOTUNE)
    train_dataset = train_dataset.cache().shuffle(len(train_paths)).batch(BATCH_SIZE).prefetch(buffer_size=AUTOTUNE)

    val_dataset = tf.data.Dataset.from_tensor_slices((val_paths, val_labels))
    val_dataset = val_dataset.map(load_and_preprocess_image, num_parallel_calls=AUTOTUNE)
    val_dataset = val_dataset.batch(BATCH_SIZE).cache().prefetch(buffer_size=AUTOTUNE)

    data_augmentation = models.Sequential([
    layers.RandomFlip('horizontal'),
    layers.RandomRotation(0.2),
    ], name="augmentation")

    print("--- Membangun model ---")
    input_shape = IMG_SIZE + (3,)
    preprocess_input = tf.keras.applications.mobilenet_v2.preprocess_input
    base_model = MobileNetV2(
    input_shape=input_shape,
    include_top=False,
    weights='imagenet'
    )
    base_model.trainable = False

    inputs = tf.keras.Input(shape=input_shape)

    x = data_augmentation(inputs)

    x = preprocess_input(x)

    x = base_model(x, training=False)
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dropout(0.2)(x)
    x = layers.Dense(128, activation='relu')(x)
    outputs = layers.Dense(1, activation='sigmoid')(x)
```

```
    model = tf.keras.Model(inputs, outputs)

    model.compile(
    optimizer=tf.keras.optimizers.Adam(learning
_rate=LEARNING_RATE),
    loss=tf.keras.losses.BinaryCrossentropy(),
    metrics=['accuracy']
    )
    model.summary()

    print("--- Memulai pelatihan model ---")
    history = model.fit(
    train_dataset,
    epochs=EPOCHS,
    validation_data=val_dataset
    )

    print("--- Menyimpan model ---")
    model.save('model_klasifikasi_jeruk.keras')
    print("Model berhasil disimpan sebagai
'model_klasifikasi_jeruk.keras'")

    print("\n--- Memulai Evaluasi Model (F1-
Score, dll.) ---")

    val_dataset_for_predict =
tf.data.Dataset.from_tensor_slices((val_paths,
val_labels))
    val_dataset_for_predict =
val_dataset_for_predict.map(load_and_preprocess_
image, num_parallel_calls=AUTOTUNE)
    val_dataset_for_predict =
val_dataset_for_predict.batch(BATCH_SIZE)

    print("--- Memprediksi Validation Set ---")
    y_pred_probs_val =
model.predict(val_dataset_for_predict)
    y_pred_val =
np.where(y_pred_probs_val.flatten() > 0.5, 1, 0)
    y_true_val = np.array(val_labels)

    print("\n--- HASIL EVALUASI (VALIDATION /
TEST SET) ---")
    print(classification_report(y_true_val,
y_pred_val, target_names=class_names))

    train_dataset_for_predict =
tf.data.Dataset.from_tensor_slices((train_paths,
train_labels))
```

```
    train_dataset_for_predict =
    train_dataset_for_predict.map(load_and_preproces
    s_image, num_parallel_calls=AUTOTUNE)
    train_dataset_for_predict =
    train_dataset_for_predict.batch(BATCH_SIZE)

    print("--- Memprediksi Training Set ---")
    y_pred_probs_train =
    model.predict(train_dataset_for_predict)
    y_pred_train =
    np.where(y_pred_probs_train.flatten() > 0.5, 1,
    0)
    y_true_train = np.array(train_labels)

    print("\n--- HASIL EVALUASI (TRAINING SET)
    ---")
    print(classification_report(y_true_train,
    y_pred_train, target_names=class_names))

    print("--- Menampilkan plot Akurasi dan
    Loss ---")
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    plt.figure(figsize=(8, 8))
    plt.subplot(2, 1, 1)
    plt.plot(acc, label='Training Accuracy')
    plt.plot(val_acc, label='Validation
    Accuracy')
    plt.legend(loc='lower right')
    plt.ylabel('Accuracy')
    plt.ylim([min(plt.ylim()),1])
    plt.title('Training and Validation
    Accuracy')

    plt.subplot(2, 1, 2)
    plt.plot(loss, label='Training Loss')
    plt.plot(val_loss, label='Validation Loss')
    plt.legend(loc='upper right')
    plt.ylabel('Cross Entropy')
    plt.ylim([0,1.0])
    plt.title('Training and Validation Loss')
    plt.xlabel('epoch')
    plt.show()
```

Lampiran C. Kode testing dan sending

```
import cv2
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models, Input
from tensorflow.keras.applications import MobileNetV2
import os
import serial
import time

PATH_WEIGHTS = 'bobot_jeruk_final.npz'
    IMG_SIZE = (160, 160)

    SERIAL_PORT = '/dev/ttyTHS1'
    BAUD_RATE = 115200

    CMD_SEHAT = b'1'
    CMD_TIDAK_SEHAT = b'0'

    FRAMES_TO_VALIDATE = 15
    MIN_AREA_PIXEL = 5000

    LOWER_ORANGE = np.array([10, 100, 100])
    UPPER_ORANGE = np.array([30, 255, 255])
    LOWER_GREEN = np.array([35, 50, 50])
    UPPER_GREEN = np.array([85, 255, 255])

    def build_manual_model():
    inputs = Input(shape=IMG_SIZE + (3,))
    x = tf.keras.applications.mobilenet_v2.preprocess_input(inputs)
    base_model = MobileNetV2(input_shape=IMG_SIZE + (3,), include_top=False, weights=None)
    base_model.trainable = False
    x = base_model(x, training=False)
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dropout(0.2)(x)
    x = layers.Dense(128, activation='relu')(x)
    outputs = layers.Dense(1, activation='sigmoid')(x)
    return tf.keras.Model(inputs, outputs)

    def cek_keberadaan_jeruk(frame):
```

```
    hsv = cv2.cvtColor(frame,
    cv2.COLOR_BGR2HSV)
    mask_orange = cv2.inRange(hsv,
    LOWER_ORANGE, UPPER_ORANGE)
    mask_green = cv2.inRange(hsv, LOWER_GREEN,
    UPPER_GREEN)
    combined_mask = cv2.bitwise_or(mask_orange,
    mask_green)
    kernel = np.ones((5,5), np.uint8)
    combined_mask =
    cv2.morphologyEx(combined_mask, cv2.MORPH_OPEN,
    kernel)
    pixel_count =
    cv2.countNonZero(combined_mask)
    return pixel_count > MIN_AREA_PIXEL,
    combined_mask

    def main():
    # 1. SETUP UART
    ser = None
    try:
    ser = serial.Serial(SERIAL_PORT, BAUD_RATE,
    timeout=1)
    print(f"Terhubung ke OpenCR di
    {SERIAL_PORT}")
    except Exception as e:
    print(f"WARNING: Gagal konek UART: {e}")

    # 2. SETUP MODEL
    if not os.path.exists(PATH_WEIGHTS):
    print(f"ERROR: File {PATH_WEIGHTS} tidak
    ditemukan!")
    return

    try:
    model = build_manual_model()
    with np.load(PATH_WEIGHTS) as data:
    weight_list = [data[key] for key in
    data.files]
    model.set_weights(weight_list)
    print("MODEL SIAP!")

    dummy_img = np.zeros((1, 160, 160, 3))
    model.predict(dummy_img)
    print("SIAP DETEKSI.")

    except Exception as e:
    print(f"Error Model: {e}")
    return
```

```
    # 3. KAMERA
    cap = cv2.VideoCapture(0)
    if not cap.isOpened():
    print("ERROR: Kamera error.")
    return

    consistent_frames = 0
    last_prediction = None
    data_sudah_dikirim = False

    while True:
    ret, frame = cap.read()
    if not ret: break

    frame_resized = cv2.resize(frame, (640,
  480))
    ada_jeruk, mask_debug =
  cek_keberadaan_jeruk(frame_resized)

    if not ada_jeruk:

    display_text = "SIAP - LETAKKAN JERUK"
    box_color = (100, 100, 100)
    data_sudah_dikirim = False
    consistent_frames = 0
    frame_resized[380:480, 540:640] = [0,0,0]

    else:
    frame_resized[380:480, 540:640] =
  [255,255,255]

    if data_sudah_dikirim:
    display_text = "MENUNGGU DIAMBIL..."
    box_color = (0, 255, 255) # Kuning
    cv2.rectangle(frame_resized, (0,0), (640,
  60), (0,0,0), -1)

    else:

    try:
    rgb = cv2.cvtColor(frame_resized,
  cv2.COLOR_BGR2RGB)
    img_input = cv2.resize(rgb, IMG_SIZE)
    img_array =
  np.expand_dims(np.array(img_input), axis=0)

    prediksi = model.predict(img_array)
    score = prediksi[0][0]
```

```
97.     if score < 0.5:
98.     current_label = "JERUK SEHAT"
99.     color = (0, 255, 0)
100.    data_to_send = CMD_SEHAT
101.    else:
102.    current_label = "JERUK TIDAK SEHAT"
103.    color = (0, 0, 255)
104.    data_to_send = CMD_TIDAK_SEHAT

105.    display_text = f"{current_label}"
106.    box_color = color

107.    if current_label == last_prediction:
108.    consistent_frames += 1
109.    else:
110.    consistent_frames = 0
111.    last_prediction = current_label

112.    if consistent_frames >= FRAMES_TO_VALIDATE:
113.    print(f"MENGIRIM: {current_label}")

114.    if ser is not None:
115.    ser.write(data_to_send)
116.    print(f"Data terkirim ke UART:
  {data_to_send}")

117.    data_sudah_dikirim = True
118.    consistent_frames = 0

119.    except Exception as e:
120.    print(f"Error Coding: {e}")


121.    cv2.rectangle(frame_resized, (10, 10),
  (450, 60), (0, 0, 0), -1)
122.    cv2.putText(frame_resized, display_text,
  (20, 45), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
  box_color, 2)

123.    if consistent_frames > 0 and not
  data_sudah_dikirim:
124.    bar_len = int((consistent_frames /
  FRAMES_TO_VALIDATE) * 340)
125.    cv2.rectangle(frame_resized, (10, 55), (10
  + bar_len, 60), (255, 255, 255), -1)

126.    cv2.imshow('Sistem Sortir', frame_resized)
```

```
127.    if cv2.waitKey(1) & 0xFF == ord('q'):
128.    break

129.    if ser is not None: ser.close()
130.    cap.release()
131.    cv2.destroyAllWindows()

132.    if __name__ == "__main__":
133.    main()
```

Lampiran D. Kode menerima klasfikasi dan menggerkaan robot pada OpenCR

```
1. #include <open_manipulator_libs.h>

2. #define JETSON_SERIAL Serial1

3. OpenManipulator open_manipulator;

4. double control_time = 0.010;
5. double previous_time = 0.0;
6. unsigned long step_timer = 0;
7. int step = 0;

8. bool robot_sedang_kerja = false;
9. bool tujuan_sehat = true;

10.     double G_BUKA  = 0.010;
11.     double G_TUTUP = 0.005;

12.     void setup() {
13.     Serial.begin(115200);
14.     JETSON_SERIAL.begin(115200);

15.     open_manipulator.setOpenManipulatorCustomJo
  intId(11, 12, 13, 14, 15);
16.     open_manipulator.initOpenManipulator(true);
17.     open_manipulator.enableAllActuator();


18.     while(JETSON_SERIAL.available() > 0) {
19.     char sampah = JETSON_SERIAL.read();
20.     }

21.     }

22.     void loop() {
```

```
23.     double present_time = millis()/1000.0;
24.     if(present_time - previous_time >=
  control_time) {
25.     open_manipulator.processOpenManipulator(pre
  sent_time);
26.     previous_time = present_time;
27.     }

28.     if (robot_sedang_kerja == false) {

29.     if (JETSON_SERIAL.available() > 0) {
30.     char data = JETSON_SERIAL.read();

31.     if (data == '1') {
32.     Serial.println("DITERIMA: JERUK SEHAT
  (1)");
33.     tujuan_sehat = true;
34.     robot_sedang_kerja = true;
35.     step = 0;
36.     step_timer = millis();

37.     }
38.     else if (data == '0') {
39.     Serial.println("DITERIMA: JERUK SAKIT
  (0)");
40.     tujuan_sehat = false;
41.     robot_sedang_kerja = true;
42.     step = 0;
43.     step_timer = millis();
44.     }
45.     }
46.     }

47.     if (robot_sedang_kerja == true) {

48.     if (millis() > step_timer) {
49.     std::vector<double> target;
50.     target.clear();

51.     if (step == 0) {
52.     target.push_back(-0.057);
  target.push_back(-0.202); target.push_back(-
  0.397); target.push_back(1.150);
53.     open_manipulator.makeJointTrajectory(target
  , 3.0);
54.     open_manipulator.makeToolTrajectory("grippe
  r", G_BUKA);
55.     step++; step_timer = millis() + 4000;
56.     }
```

```
57.     else if (step == 1) {
58.     target.push_back(1.000); target.push_back(-
  0.202); target.push_back(-0.397);
  target.push_back(1.150);
59.     open_manipulator.makeJointTrajectory(target
  , 3.0);
60.     step++; step_timer = millis() + 4000;
61.     }

62.     else if (step == 2) {
63.     target.push_back(1.000);
  target.push_back(0.718); target.push_back(-
  0.822); target.push_back(1.163);
64.     open_manipulator.makeJointTrajectory(target
  , 3.0);
65.     step++; step_timer = millis() + 4000;
66.     }

67.     else if (step == 3) {
68.     open_manipulator.makeToolTrajectory("grippe
  r", G_TUTUP);
69.     step++; step_timer = millis() + 2000;
70.     }

71.     else if (step == 4) {
72.     Serial.println("5. Angkat Lurus (Pinggang
  Tahan)...");
73.     target.push_back(1.000); target.push_back(-
  0.202); target.push_back(-0.397);
  target.push_back(1.150);
74.     open_manipulator.makeJointTrajectory(target
  , 4.0);
75.     step++; step_timer = millis() + 5000;
76.     }

77.     else if (step == 5) {
78.     Serial.println("6. Ke Center...");
79.     target.push_back(-0.057);
  target.push_back(-0.202); target.push_back(-
  0.397); target.push_back(1.150);
80.     open_manipulator.makeJointTrajectory(target
  , 3.0);
81.     step++; step_timer = millis() + 4000;
82.     }

83.     else if (step == 6) {
84.     if (tujuan_sehat == false) {
```

```
85.     target.push_back(0.437);
   target.push_back(0.718); target.push_back(-
   0.822); target.push_back(1.163);
86.     } else {
87.     target.push_back(-0.471);
   target.push_back(0.753); target.push_back(-
   1.002); target.push_back(1.534);
88.     }
89.     open_manipulator.makeJointTrajectory(target
   , 4.0);
90.     step++; step_timer = millis() + 5000;
91.     }

92.     else if (step == 7) {
93.     open_manipulator.makeToolTrajectory("grippe
   r", G_BUKA);
94.     step++; step_timer = millis() + 2000;
95.     }

96.     else if (step == 8) {
97.     target.push_back(-0.057);
   target.push_back(-0.202); target.push_back(-
   0.397); target.push_back(1.150);
98.     open_manipulator.makeJointTrajectory(target
   , 4.0);
99.     open_manipulator.makeToolTrajectory("grippe
   r", G_BUKA);

100.    step++; step_timer = millis() + 5000;
101.    }

102.    else if (step == 9) {
103.    robot_sedang_kerja = false;
104.    step = 0;
105.    step_timer = millis();
106.    }
107.    }
108.    }
109.    }
```