BAB II
TINJAUAN PUSTAKA

2.1 Sistem Otomasi *Filling Machine*

Sistem otomasi *filling machine* merupakan suatu sistem pengisian cairan otomatis yang dirancang untuk meningkatkan efisiensi, akurasi, dan konsistensi dalam proses produksi. Sistem ini bekerja dengan memanfaatkan sensor, mikrokontroler, serta aktuator yang saling terintegrasi untuk menjalankan proses pengisian cairan tanpa memerlukan campur tangan manusia secara terus menerus. Pada sistem ini, sensor digunakan untuk mendeteksi keberadaan botol pada *conveyor*. Data dari sensor kemudian diproses oleh ESP32-S3 untuk menentukan proses kerja selanjutnya. Setelah botol terdeteksi, mikrokontroler akan memberikan sinyal kepada *driver* motor untuk mengaktifkan pompa peristaltik sehingga cairan dapat dipindahkan ke dalam botol sesuai parameter yang telah ditentukan oleh operator menggunakan *keypad* sebagai media *input parameter* dan LCD sebagai media tampilan informasi proses pengisian.

Selain itu, sistem ini dilengkapi dengan fitur *monitoring* berbasis *Internet of Things* menggunakan ESP32 yang berfungsi menerima data dari ESP32-S3, kemudian mengirimkan data proses pengisian secara *real-time* menuju *server* XAMPP melalui jaringan internet. Data yang dikirim *quantity* produksi botol, jumlah hasil pengisian, status sensor, serta kondisi aktuator pada sistem. Selanjutnya, data tersebut disimpan ke dalam database MySQL dan ditampilkan pada halaman *website monitoring* berbasis PHP melalui *localhost*. Dengan adanya sistem *monitoring* ini, proses produksi dapat dipantau secara lebih mudah, efisien, dan *real-time*

2.2 Mikrokontroller

Mikrokontroler adalah sebuah komputer kecil yang dikemas dalam bentuk chip berupa IC (*Integrated Circuit*) dan dirancang untuk melakukan tugas atau operasi tertentu seperti menerima sinyal *input*, mengolahnya, kemudian memberikan sinyal *output* sesuai dengan program yang telah diisikan ke mikrokontroler tersebut. Pada umumnya, sinyal *input* mikrokontroler dari sensor yang merupakan informasi dari keberadaan botol sedangkan sinyal

output ditujukan kepada aktuator yang dapat melakukan suatu tindakan. Mikrokontroller ini terdiri dari satu atau lebih inti prosesor *CPU*, *memory* (RAM dan ROM), serta perangkat *input* dan *output* yang dapat diprogram. Walaupun mirip dengan komputer namun kecepatan pengolahan data pada mikrokontroler lebih rendah jika dibandingkan dengan komputer atau PC. Kecepatan pengolahan data mikrokontroler umumnya berkisar antara 1 – 240 MHz yang tentu lebih rendah dibandingkan komputer atau PC saat ini yang telah mencapai kecepatan hingga orde GHz. Begitu juga dengan kapasitas *memory* (RAM dan ROM) yang hanya berkisar pada orde Kbytes (MediaCenter, 2025).

2.2.1 **ESP32-S3-WROOM-1U-N16R8**

ESP32-S3 N16R8 merupakan mikrokontroler berbasis *dual-core Xtensa® LX7 32-bit* dengan kecepatan *clock* hingga 240 MHz yang dirancang untuk kebutuhan sistem *embedded* dan otomasi *modern*. Modul ini memiliki kapasitas memori sebesar 16 MB *Flash* dan 8 MB *PSRAM* yang memungkinkan penyimpanan program berukuran besar serta pengolahan data secara lebih optimal. Selain itu, ESP32-S3 N16R8 telah dilengkapi dengan *konektivitas* Wi-Fi 2,4 GHz dan *Bluetooth Low Energy* (BLE) untuk mendukung komunikasi data secara nirkabel.

Gambar 2. 1 Mikrokontroller ESP32-S3

(Sumber : http://wiki.fluidnc.com/en/hardware/ESP32-S3_Pin_Reference)

Keunggulan mikrokontroler ini juga menyediakan hingga 45 GPIO yang mendukung berbagai antarmuka komunikasi seperti UART, SPI, I2C, I2S, PWM, ADC 12-bit, USB OTG, serta mendukung penggunaan LCD dan

sensor eksternal. Dengan spesifikasi tersebut, ESP32-S3 N16R8 memiliki kemampuan pemrosesan data yang cepat, stabil, dan responsif untuk sistem kendali otomatis. (Espressif Systems, 2022).

Pada sistem *filling machine*, Mikrokontroler ini menerima data dari sensor untuk mendeteksi keberadaan botol pada area pengisian, kemudian memproses data tersebut untuk menentukan proses kerja selanjutnya. Setelah botol terdeteksi, ESP32-S3 akan mengendalikan *driver* motor untuk mengatur kecepatan pompa sehingga proses pengisian cairan dapat berjalan sesuai parameter yang telah ditentukan. Selain itu, ESP32-S3 juga mengelola *input* dari *keypad* sebagai media pengaturan parameter pengisian dan menampilkan informasi proses pada LCD secara *real-time*.

2.2.2 ESP32-WROOM-32

ESP32-WROOM-32 merupakan modul mikrokontroler yang dirancang untuk mendukung pengembangan sistem *embedded* maupun *Internet of Things*. Modul ini menggunakan prosesor *dual-core Xtensa® 32-bit LX6* dengan kecepatan *clock* hingga 240 MHz sehingga mampu memproses data dengan cepat dan efisien. ESP32-WROOM-32 dilengkapi dengan memori SRAM sebesar 520 KB dan Flash Memory sebesar 4 MB yang berfungsi untuk penyimpanan program serta pengolahan data sistem. Selain itu, modul ini telah mendukung *konektivitas* Wi-Fi 802.11 b/g/n dan Bluetooth v4.2 BLE yang memungkinkan komunikasi data secara wireless. ESP32 memiliki hingga 34 pin GPIO (*General Purpose Input Output*) yang mendukung berbagai fitur seperti ADC, DAC, PWM, UART, SPI, I2C, dan IoT. (SunFounder, 2026).

Gambar 2. 2 Mikrokontroller ESP32

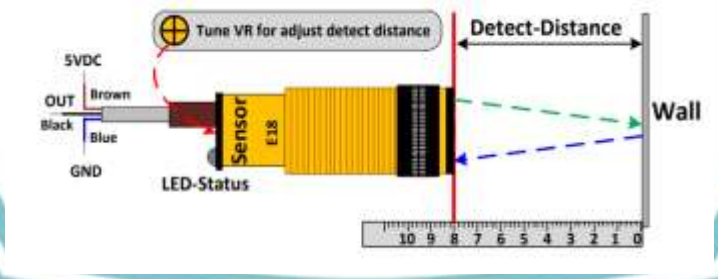
(Sumber : https://www.sunfounder.com/blogs/news/esp32-pinout-guide-gpio-adc-dac-touch-complete-hardware-reference-for-stable-circuit-design)

Dalam penerapan Internet of Things (IoT), ESP32 Dev Module berfungsi sebagai pusat menerima data dari ESP32-S3 N16R8, kemudian memproses dan mengirimkan data tersebut ke server melalui jaringan Wi-Fi. Pada sistem ini, server menggunakan Apache yang berjalan pada aplikasi XAMPP sebagai media komunikasi dan pengelolaan data. Data yang dikirim oleh ESP32 selanjutnya disimpan ke dalam database MySQL dan ditampilkan melalui antarmuka website berbasis PHP dan HTML pada localhost.

2.3 Sensor

2.3.1 Infrared E18-D80NK

Sensor *Infrared* E18-D80NK merupakan sensor proximity berbasis *inframerah* yang digunakan untuk mendeteksi keberadaan objek tanpa kontak langsung. Sensor ini bekerja dengan memanfaatkan pemancar dan penerima sinar *inframerah* untuk mendeteksi pantulan cahaya dari objek yang berada di depan sensor. E18-D80NK memiliki jarak deteksi yang dapat diatur mulai dari 3 cm hingga 80 cm menggunakan potensiometer. (ETC1, 2021)



Gambar 2. 3 Sensor Infrared

(Sumber : https://evelta.com/content/datasheets/083-E-18-D80NK.pdf)

Sensor infrared E18-D80NK digunakan sebagai input pendeteksi keberadaan botol pada area pengisian cairan. Sensor bekerja dengan memancarkan sinar inframerah ke arah botol, kemudian pantulan cahaya tersebut diterima kembali oleh *receiver* sensor. Ketika botol terdeteksi

berada pada posisi pengisian, sensor akan mengirimkan sinyal digital ke mikrokontroler ESP32-S3 N16R8 untuk memulai proses *filling* atau pengisian cairan.

2.2.3 XKC-Y26S-PNP

Sensor XKC-Y26S-PNP merupakan sensor level cairan *non-contact* yang bekerja menggunakan prinsip kapasitif untuk mendeteksi keberadaan cairan tanpa kontak langsung dengan cairan tersebut. Sensor ini mampu mendeteksi cairan melalui dinding wadah berbahan *non-logam* seperti plastik, kaca, dan akrilik. Sensor XKC-Y26S-PNP memiliki ini tegangan kerja (*input voltage*) sebesar DC 5–24V dengan pilihan output signal berupa PNP maupun NPN. Sensor ini memiliki konsumsi arus sebesar 5 mA dengan tegangan output high level sebesar Vin dan low level sebesar 0V. Output current sensor berada pada kisaran 1–100 mA dengan response time sekitar 500 ms. Sensor dapat bekerja pada suhu lingkungan 0–85°C (Naylamp Mechatronics, 2023).

Gambar 2. 4Sensor Water Level

(Sumber : https://naylampmechatronics.com/img/cms/001339/XKC-Y26-V.pdf)

Sensor XKC-Y26S-PNP digunakan untuk mendeteksi level cairan pada tangki penampung atau mendeteksi keberadaan cairan sebelum proses pengisian dilakukan. Sensor ini membantu sistem kontrol agar proses pengisian cairan dapat berjalan otomatis, menjaga kestabilan volume cairan, serta mencegah kondisi tangki kosong.

2.4 Actuator dan Interface

2.4.1 Peristaltic Pump Kamoer KPK 400

Mikro peristaltic pump kamoer KPK 400 adalah jenis pompa persistaltic yang menggunakan mekanisme penekanan selang secara bergantian oleh roller untuk mendorong fluida tanpa kontak langsung antara cairan dan komponen mekanik pompa. Prinsip ini membuatnya sangat cocok untuk aplikasi yang membutuhkan tingkat kebersihan tinggi, presisi dosis, serta minim kontaminasi. KPK400 beroperasi dengan suplai tegangan DC 24 volt, pump ini dapat dikendalikan menggunakan mikrokontroller sebagai pengatur kecepatan melalui sinyal PWM (*Pulse Width Modulation*) untuk mengatur besar kecilnya aliran cairan secara presisi.

Gambar 2. 5 Water Pump

(Sumber : https://pdf.directindustry.com/pdf/kamoer-fluid-tech-shanghai-co-ltd/kpk400-peristaltic-pump/242598-1056365.html)

Pompa ini memiliki kapasitas aliran hingga sekitar 400 mL/menit, sehingga sering digunakan pada sistem filling machine untuk pengisian cairan secara otomatis dan konsisten ke dalam botol atau wadah. Dalam implementasinya, motor pada sistem ini dikendalikan menggunakan *driver* BTS7960 Motor *Driver Module* yang berfungsi sebagai penguat arus sekaligus pengatur arah putaran motor, sehingga mikrokontroler seperti ESP32 dapat mengontrol kecepatan dan arah pompa secara stabil melalui sinyal PWM.

2.4.2 NEMA 17

Motor stepper adalah jenis motor yang berputar dengan langkah-langkah tertentu. Motor ini banyak digunakan dalam industri, seperti pada mesin CNC, lengan robot, pemindai, dan printer. Motor stepper memiliki kumparan pada bagian stator, sedangkan rotornya terdiri dari magnet permanen. Karena konstruksi ini, motor stepper dapat dikendalikan untuk berputar sesuai keinginan, baik searah jarum jam maupun berlawanan arah.

Pada sistem ini digunakan motor stepper tipe Nema 17 sebagai penggerak sistem. Motor stepper Nema 17 memiliki sudut Langkah sebesar 1.8° per step atau 200 langkah dalam satu putaran penuh. Motor ini dipilih karena memiliki tingkat presisi yang baik, mudah dikendalikan menggunakan microcontroller, serta mampu mengatur posisi dan kecepatan putaran dengan akurat.

Gambar 2. 6 Stepper Nema 17

(Sumber : https://digitalelectronics.lk/product/nema17-stepper-motor-17hs4401/#&gid=null&pid=1)

2.4.3 Driver TB6600

Driver TB6600 adalah sebuah *driver* motor yang digunakan untuk mengendalikan motor stepper, terutama motor stepper bipolar. *Driver* ini dapat mengatur aliran arus dan pola pengaktifan kumparan-kumparan motor stepper, sehingga memungkinkan untuk mengontrol posisi, kecepatan, dan arah pergerakan motor dengan presisi. Driver TB6600 mampu mengendalikan motor stepper dari 1/1, 1/2, 1/4, 1/8, 1/16 step, 1/32, dan 1/64.

Gambar 2. 7 Driver TB6600

(Sumber : https://makerhardware.net/knowledge-base/electronics/tb6600-stepper-motor-driver/)

Pada sistem ini, driver TB6600 digunakan untuk mengendalikan motor stepper Nema 17 karena memiliki kestabilan kerja yang baik dan mudah dihubungkan dengan *microcontroller*. Pengaturan *microstepping* pada driver ini membuat pergerakan motor menjadi lebih halus dan presisi sehingga cocok digunakan pada sistem *conveyor*.

2.4.4 Driver BTS7960

Driver BTS7960 merupakan modul motor driver DC berbasis teknologi *H-Bridge* yang dirancang untuk mengendalikan motor DC dengan arus besar secara efisien dan aman. Driver ini bekerja dengan prinsip pengaturan arah putaran motor dan kecepatan melalui sinyal PWM (*Pulse Width Modulation*). BTS7960 memiliki kemampuan untuk menangani tegangan kerja sekitar 5,5 V hingga 27,5 V dengan arus kontinu 42 A, serta mendukung frekuensi PWM hingga 25 kHz. Driver ini juga kompatibel dengan logika 3,3 V maupun 5 V sehingga dapat dihubungkan dengan mikrokontroler.

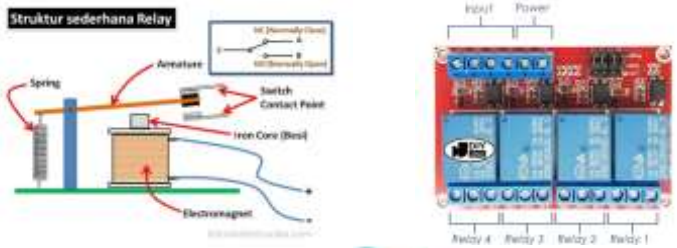
Gambar 2. 8 Driver Water Pump

(Sumber:https://www.handsontec.com/dataspecs/module/BTS7960%20Motor%20Driver.pdf)

Pada sistem ini Driver BTS7960 berfungsi sebagai *interface* penghubung antarmuka daya antara mikrokontroler bertegangan rendah ESP32-S3 melewati sinyal PWM yang dikirim pada driver, Dengan demikian, mikrokontroler hanya bertugas mengirimkan sinyal kontrol seperti PWM dan arah putaran, sementara BTS7960 yang menangani suplai daya utama ke motor. Pada aplikasi Kamoer KPK 400, fungsi *interface* ini sangat penting karena pompa peristaltik membutuhkan kontrol kecepatan yang stabil untuk menjaga akurasi debit aliran cairan.

2.4.5 Relay 4 Channel

Relay 4 channel merupakan modul saklar elektronik yang terdiri dari empat buah relay dalam satu rangkaian sehingga mampu mengendalikan beberapa beban secara bersamaan melalui mikrokontroler. Modul relay ini umumnya menggunakan tegangan kerja relay sebesar 24 VDC dengan tegangan *input* trigger sekitar 3,3 V hingga 5 VDC yang kompatibel dengan keluaran GPIO mikrokontroler. Pada bagian *input* terdapat pin IN1 hingga IN4 sebagai pin kendali, pin VCC, dan GND, sedangkan pada bagian *output* tersedia terminal *Normally Open* (NO), *Normally Close* (NC), dan *Common* (COM) (teknikelektronika.com).



Gambar 2. 9Module Relay 4 Channel

(Sumber : https://arduinogetstarted.com/tutorials/arduino-4-channel-relay-module)

Pada sistem *filling machine*, relay 4 channel digunakan sebagai pengendali pilot lamp 24 volt yang berfungsi sebagai indikator proses kerja mesin. Relay menerima sinyal dari mikrokontroler kemudian menghubungkan tegangan 24 VDC ke pilot lamp sesuai kondisi proses yang sedang berlangsung. Pilot lamp dapat digunakan untuk menunjukkan status mesin seperti proses pengisian berlangsung, mesin standby, proses selesai, maupun kondisi error.

2.4.6 Pilot Lamp AD22B-22D

Pilot lamp 24 volt merupakan komponen indikator visual yang digunakan untuk menampilkan kondisi atau status kerja suatu sistem kelistrikan maupun sistem otomasi industri. Pilot lamp bekerja dengan memanfaatkan sumber tegangan 24 VDC untuk menyalakan lampu indikator berupa LED sehingga operator dapat mengetahui kondisi mesin secara langsung. Pilot lamp tipe AD22B-22D memiliki diameter pemasangan 22 mm dengan tegangan kerja 24 volt DC.

Gambar 2. 10 Pilot Lamp

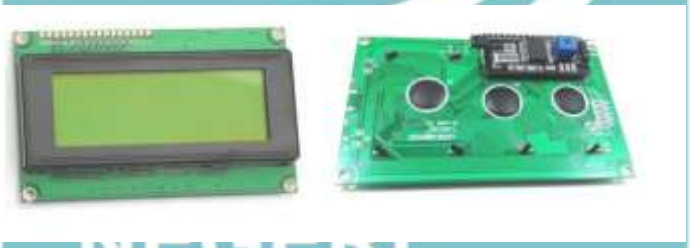
(Sumber : https://dv-electric.id/wp-content/uploads/2023/10/PILOT-LAMP-22MM-AD22B-22D.pdf)

Pada sistem *filling machine*, pilot lamp 24 volt digunakan sebagai indikator proses untuk memberikan informasi visual kepada operator mengenai kondisi kerja mesin. Lampu indikator akan menyala sesuai dengan perintah dari sistem kontrol melalui relay yang terhubung dengan mikrokontroler.

2.5 Human Machine Interface

2.5.1 LCD 20x4 I2C

LCD 20x4 I2C merupakan perangkat *output* yang digunakan sebagai media tampilan pada sistem elektronika dan mikrokontroler. LCD ini mampu menampilkan 20 karakter pada 4 baris sehingga informasi dapat ditampilkan dengan lebih jelas dan terstruktur. Pada penelitian ini, LCD digunakan untuk menampilkan menu, parameter, serta kondisi sistem pada mesin *filling* otomatis.



Gambar 2. 11 LCD 20x4 I2C

(Sumber : https://www.handsontec.com/dataspecs/I2C_2004_LCD.pdf)

LCD 20x4 dilengkapi dengan modul *Inter-Integrated Circuit* (I2C), yaitu metode komunikasi serial yang memungkinkan pertukaran data hanya menggunakan dua jalur, yaitu Serial Data (SDA) dan *Serial Clock* (SCL). Penggunaan modul I2C membuat proses interfacing dengan mikrokontroler menjadi lebih sederhana dan efisien karena hanya membutuhkan 4 pin, yaitu SDA, SCL, VCC, dan GND. Selain menghemat penggunaan pin, komunikasi I2C juga membuat rangkaian lebih rapi dan mudah dalam proses pengembangan sistem, (Handson Technology, 2025).

2.5.2 Keypad Matrix 4x4 I2C

Keypad I2C merupakan perangkat *input* yang digunakan untuk memasukkan data atau perintah ke dalam sistem mikrokontroler dengan menggunakan komunikasi *Inter-Integrated Circuit* (I2C). Penggunaan modul I2C pada keypad memungkinkan pengurangan jumlah pin yang digunakan pada mikrokontroler karena komunikasi hanya memerlukan dua jalur utama yaitu SDA (Serial Data) dan SCL (*Serial Clock*). keypad I2C meliputi konfigurasi tombol 4×4 dengan total 16 tombol, komunikasi berbasis I2C, menggunakan jalur SDA dan SCL, tegangan kerja 3,3V–5V, serta kompatibel dengan mikrokontroler.

Gambar 2. 12 Keypad 4x4 I2C

(Sumber : https://hobbycomponents.com/recently-purchased/1120-mlink-4x4-matrix-keypad)

Pada filling machine, keypad I2C berfungsi sebagai media input untuk mengatur parameter pengisian seperti volume cairan, jumlah pengisian, dan pengoperasian sistem. Penggunaan komunikasi I2C membuat sistem lebih sederhana karena mengurangi penggunaan pin pada ESP32 sehingga mempermudah integrasi dengan sensor dan komponen lainnya.

2.6 Bahasa Pemograman

2.6.1 C++

Bahasa pemrograman C++ merupakan bahasa pemrograman tingkat tinggi yang dikembangkan dari bahasa C dan mendukung konsep pemrograman berorientasi objek *(Object Oriented Programming/OOP)*. Bahasa C++ digunakan untuk membuat berbagai aplikasi karena memiliki kemampuan pengolahan data yang cepat, efisien, serta mampu berinteraksi

langsung dengan perangkat keras. C++ juga dikenal sebagai "*C with Classes*" karena merupakan pengembangan dari bahasa C dengan penambahan fitur class dan object untuk mempermudah pengembangan program yang lebih terstruktur. Dalam bidang teknik dan sistem *embedded*, bahasa C++ banyak digunakan sebagai dasar pemrograman karena mampu memberikan kontrol yang baik terhadap sistem perangkat keras maupun perangkat lunak (Jurnal Majemuk, 2024).

Gambar 2. 13 Bahasa Pemograman

(Sumber : https://www.logique.co.id/blog/2021/04/26/bahasa-pemrograman-c-2/)

Bahasa pemrograman C++ digunakan pada mikrokontroler ESP32-S3 sebagai algoritma kendali pada sistem *filling machine*. Program C++ berfungsi untuk mengatur proses kerja mesin seperti membaca *input* sensor, mengendalikan motor pompa, mengatur relay, menampilkan data pada LCD, serta melakukan komunikasi data dengan sistem monitoring. Penggunaan bahasa C++ pada ESP32-S3 memungkinkan proses pengendalian mesin berjalan secara cepat, stabil, dan terstruktur.

2.6.2 Javascript

JavaScript adalah bahasa pemrograman yang digunakan untuk membuat halaman web menjadi lebih interaktif dan dinamis. JavaScript memungkinkan pengembang menambahkan berbagai fungsi pada website seperti tombol interaktif, validasi formulir, animasi, menu *navigasi*, hingga pembaruan data secara *real-time* tanpa harus memuat ulang halaman. Selain berjalan di sisi browser (*client-side*), JavaScript juga dapat digunakan pada

sisi server melalui teknologi seperti Node.js. Fungsi JavaScript dalam sistem yaitu untuk mengatur interaksi pengguna dengan website. JavaScript digunakan untuk membuat *dashboard monitoring*, komunikasi data dengan server menggunakan API, validasi *input* data, menampilkan data sensor secara *real-time*, serta menghubungkan tampilan HTML dan CSS agar sistem menjadi responsif dan interaktif. Pada sistem berbasis *dashboard monitoring filling machine* menggunakan mikrokontroller. JavaScript dapat digunakan untuk menampilkan data sensor, status mesin, grafik *monitoring*, dan pembaruan data otomatis pada dashboard web (CORAL, 2023).

Gambar 2. 14 Bahasa Pemograman JavaScript

(Sumber : https://kd-cibiru.upi.edu/index.php/daftar-kategori/prodi-rpl/jacascript)

2.6.3 HTML

HTML (*HyperText Markup Language*) adalah bahasa markup dasar yang digunakan dalam pembuatan dan pengembangan halaman web. HTML berfungsi untuk menyusun struktur suatu halaman website, seperti pengaturan teks, gambar, tabel, tautan, dan berbagai elemen lainnya agar dapat ditampilkan oleh browser. Dalam pengembangan web, HTML menjadi fondasi utama karena digunakan untuk membangun kerangka tampilan website sebelum dikombinasikan dengan CSS dan JavaScript untuk mempercantik desain serta menambahkan fungsi interaktif. (Jurnal Pengabdian Kepada Masyarakat, 2024)

Gambar 2. 15 Bahasa Pemograman HTML

(Sumber : https://course-net.com/blog/apa-itu-html/)

2.6.4 CSS

CSS adalah singkatan dari *Cascading Style* Sheet yaitu dokumen web yang berfungsi mengatur elemen HTML dengan berbagai *property* yang tersedia sehingga dapat tampil dengan berbagai gaya yang diinginkan. Sebagian orang menganggap CSS bukan termasuk salah satu bahasa pemrograman karena memang strukturnya yang sederhana, hanya berupa kumpulan-kumpulan aturan yang mengatur style elemen HTML. Cara kerja CSS dalam memodifikasi HTML dengan memilih elemen HTML yang akan diatur kemudian memberikan *property* yang sesuai dengan tampilan yang diinginkan. Dalam memberikan aturan pada elemen HTML, skrip CSS terdiri atas 3 bagian yaitu Selector untuk memilih elemen yang akan diberi aturan, *property* yang merupakan aturan yang diberikan dan value sebagai nilai dari aturan yang diberikan (Jurnal Informatika Terpadu, 2020).

Gambar 2. 16 Bahasa Pemograman CSS

(Sumber : https://course-net.com/blog/apa-itu-css/)

2.6.5 PHP

PHP (*Hypertext Preprocessor*) adalah bahasa pemrograman open-source yang digunakan untuk membangun aplikasi web dinamis dan interaktif. PHP dijalankan pada server web dan dapat dikombinasikan dengan HTML, CSS, serta JavaScript untuk membuat tampilan website yang dapat berubah sesuai kebutuhan pengguna. Selain itu, PHP juga mendukung berbagai jenis database seperti MySQL, PostgreSQL, dan *Oracle* sehingga banyak digunakan dalam pengembangan sistem berbasis web. Fungsi PHP adalah untuk membuat dan mengembangkan website dinamis maupun aplikasi web. PHP dapat digunakan untuk memproses data formulir, menyimpan data ke *database*, mengubah tampilan halaman sesuai input pengguna, serta mengatur pengolahan data pada server. Dalam sistem berbasis web, PHP sering digunakan sebagai penghubung antara halaman *website* dengan database agar informasi dapat ditampilkan secara otomatis dan interaktif (Raharja.ac.id, 2023)

Gambar 2. 17 Bahasa Pemograman PHP

(Sumber : https://raharja.ac.id/apa-itu-php-pengertian-sejarah-keunggula-dan-fungsinya/)

2.7 Server dan Database

2.7.1 XAMPP

XAMPP adalah perangkat lunak server lokal (*localhost*) yang digunakan untuk menjalankan dan mengembangkan website secara *offline*. XAMPP terdiri dari beberapa komponen Fungsi XAMPP adalah untuk membantu *developer* atau *programmer* dalam melakukan pengembangan dan pengujian website secara lokal. XAMPP dapat digunakan untuk

menjalankan *framework* PHP seperti mengelola *database* melalui PhpMyAdmin, menginstal WordPress secara *offline*, serta melakukan pengujian fitur website sebelum dipublikasikan ke internet. utama yaitu Apache sebagai web server, MySQL sebagai database, PHP sebagai bahasa pemrograman, dan PhpMyAdmin sebagai pengelola *database* berbasis web. (Biznetgio, 2025).

Gambar 2.18 Software XAMPP

(Sumber : https://www.biznetgio.com/blog/apa-itu-xampp/)

Dalam proyek berbasis IoT atau sistem *monitoring*, XAMPP digunakan sebagai server lokal untuk menyimpan data dari mikrokontroler seperti ESP32 ke database MySQL melalui Apache dan PHP.

2.7.2 MYSQL

MySQL adalah sistem manajemen *database* berbasis *relational database* management system *Relational Database Management System* (RDBMS) yang digunakan untuk menyimpan, mengelola, dan mengatur data secara terstruktur menggunakan bahasa SQL (*Structured Query Language*). MySQL digunakan pada aplikasi berbasis web karena memiliki performa yang cepat, mudah digunakan, serta dapat terhubung dengan berbagai bahasa pemrograman seperti PHP. Dalam sistem berbasis IoT, MySQL berfungsi untuk menyimpan data yang dikirim dari mikrokontroler melalui program PHP sehingga data dapat tersimpan ke dalam database dan ditampilkan kembali pada sistem *monitoring* (Sekawan Media, 2025).

Gambar 2.19 Software MySql

(Sumber : https://www.sekawanmedia.co.id/blog/pengertian-mysql/)

2.8 Arduino Ide

Arduino IDE (*Integrated Development Environment*) adalah perangkat lunak yang digunakan untuk menulis, mengedit, mengompilasi, dan mengunggah program ke papan Arduino maupun mikrokontroler lainnya yang kompatibel. Arduino IDE menggunakan bahasa pemrograman C/C++ dan dilengkapi dengan berbagai fitur yang memudahkan pengguna dalam mengembangkan proyek elektronika dan sistem kendali. Beberapa fitur utama yang tersedia pada Arduino IDE antara lain editor kode dengan penyorotan sintaks, pustaka dan contoh program yang membantu proses pemrograman, *compiler* dan *uploader* untuk mengunggah program ke board melalui koneksi USB, serta fitur Monitor Serial yang digunakan untuk memantau komunikasi data secara serial. Selain itu, Arduino IDE juga mendukung berbagai jenis *board* arduino dan menyediakan pesan kesalahan untuk membantu proses debugging atau pemecahan masalah pada program (Arduino Indonesia, 2021).

Gambar 2.20 Software Arduino Ide

(Sumber : https://www.arduinoindonesia.id/2018/07/software-arduino-ide.html)

2.9 Visual Studio Code

Visual Studio Code adalah aplikasi code editor yang dikembangkan oleh Microsoft untuk menulis dan mengedit source code dari berbagai bahasa pemrograman seperti C++, Python, JavaScript, dan PHP. VS Code dapat digunakan pada sistem operasi Windows, Linux, maupun macOS sehingga banyak dimanfaatkan oleh programmer dan *developer* dalam pengembangan aplikasi maupun website. Selain memiliki tampilan yang ringan dan mudah digunakan, Visual Studio Code juga dilengkapi berbagai fitur seperti *IntelliSense*, *debugging*, *extension marketplace*, *auto-completion*, dan *integrasi GitHub* yang membantu proses pemrograman menjadi lebih cepat, mudah, dan efisien. Kelebihan lainnya yaitu dapat digunakan secara gratis, mendukung banyak bahasa pemrograman, serta memungkinkan pengguna menambahkan berbagai *extension* sesuai kebutuhan pengembangan program (Domainesia, 2023).

Gambar 2.21 Software Visual Studio Code

(Sumber : https://www.domainesia.com/berita/visual-studio-code/)

BAB III
PERANCANGAN DAN REALISASI

3.1 Perancangan Alat

Perancangan pemrograman algoritma kendali dan integrasi *internet of things* dilakukan untuk membangun sistem *monitoring* pada alat produksi *filling machine*. Sistem ini menggunakan dua mikrokontroler, yaitu ESP32-S3 sebagai pengendali utama proses kerja *filling machine* dan ESP32 sebagai media komunikasi data antara sistem kendali dengan server XAMPP. Data yang diterima server kemudian diolah dan ditampilkan dalam bentuk *dashboard monitoring* untuk mengamati status kerja mesin serta jumlah produksi secara *real-time.*

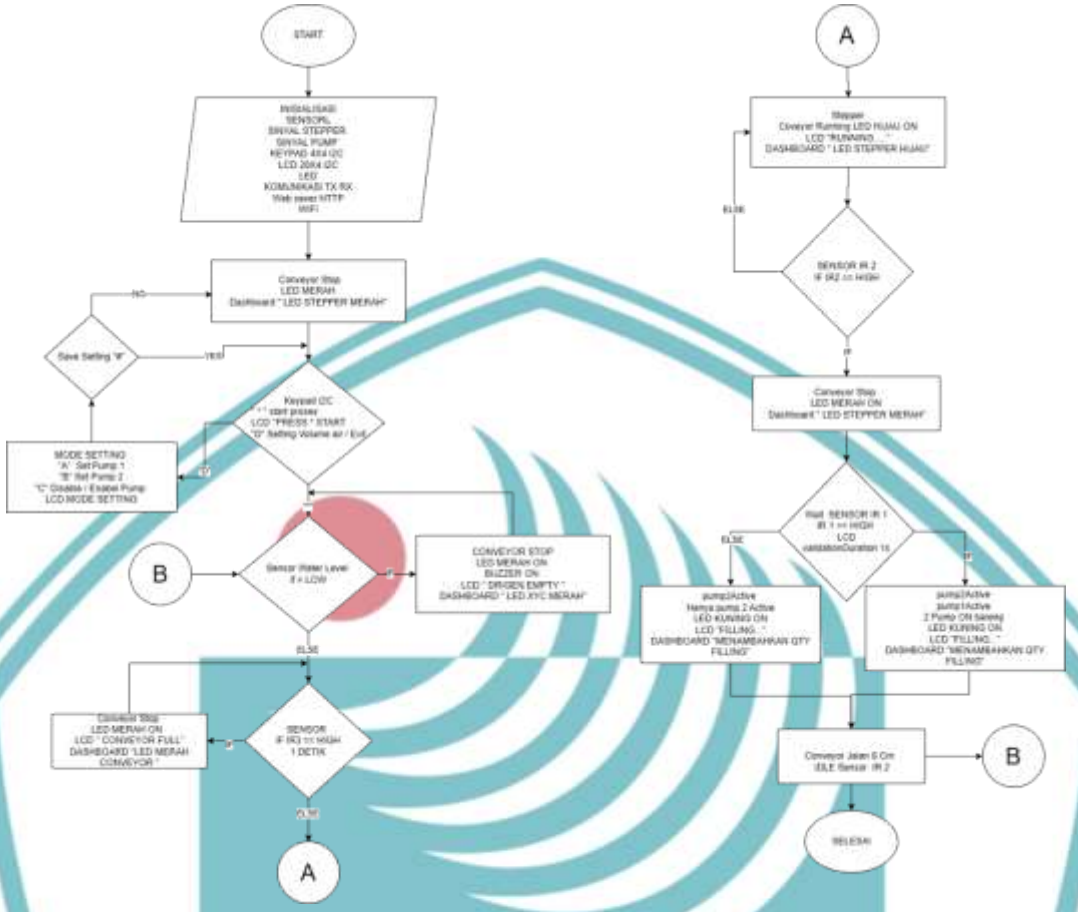
3.1.1 Deskripsi Alat

Nama Alat	:	Perancangan dan Implementasi *Filling Machine* Botol Deodorant Otomatis Berbasis ESP32 Dengan Integrasi IoT
Fungsi Alat	:	Alat berfungsi untuk membantu kerja pada usaha masyarakat pada bagian pengisian botol secara akurat dan efesien.
Nama Subsistem	:	Perancangan Algotima Kendali dan Integrasi IoT Dengan Server XAMPP pada Sistem *Filling Machine*
Fungsi Subsistem	:	Subsistem ini merancang algoritma kendali yang terintegrasi dengan sistem IoT sebagai media monitoring pada filling machine. Perancangan ini bertujuan untuk menjalankan sistem kerja mesin filling machine secara keseluruhan secara terstruktur. Pada

bagian kendali, mikrokontroler ESP32-S3 berfungsi untuk mengendalikan sensor, aktuator, keypad, dan tampilan LCD dalam proses pengisian botol sesuai parameter volume yang dimasukkan oleh operator. Seluruh proses pengisian dilakukan secara otomatis berdasarkan algoritma kendali yang telah dirancang. Pada bagian integrasi IoT, ESP32 Dev Module berfungsi sebagai media komunikasi data menuju server XAMPP menggunakan koneksi WiFi dengan protokol TCP/IP. Data proses *filling machine* yang diterima dari ESP32-S3 akan diproses oleh Apache dan PHP pada server XAMPP, kemudian disimpan ke dalam *database* MySQL untuk kebutuhan *monitoring* dan penyimpanan data. Data tersebut selanjutnya dapat ditampilkan pada *dashboard monitoring* berbasis web.

3.1.2 Cara Kerja Alat

Cara kerja alat *filling machine* otomatis ini menggunakan dua mikrokontroler, yaitu ESP32-S3 sebagai sistem kendali utama proses kerja alat dan ESP32 Dev Module sebagai media penerima data proses dari ESP32-S3 serta integrasi IoT. Sistem IoT tersebut digunakan untuk *monitoring* jumlah produksi dan proses produksi secara *realtime*. Alur kerja sistem filling machine otomatis dapat dilihat pada gambar di bawah ini.



Gambar 3.1 Flowchart Cara Kerja

Berikut cara kerja alat *filling machine* otomatis yang ditunjukan pada gambar :

1. ESP32-S3 akan menerima *input* dari sensor *infrared* 3 jika *infrared high* atau bernilai 1, sistem tidak akan jalan dan akan menampilkan *coveyor full.*
2. ESP32-S3 akan menerima *input* dari sensor XYC jika sensor ini bernilai *low* atau bernilai 0 sistem tidak akan berjalan dan akan menampilkan LCD "*drigen empty*".
3. Jika ESP32-S3 menerima *input* sensor infrared 3 dan sensor XYC bernilai *infrared* 3 bernilai 0 dan sensor XYC bernilai 1, LCD akan menampilkan "Press * Start" untuk mulai alat *filling machine.*
4. Operator menginput nilai "*" akan menjalankan *conveyor* dan akan menyalakan LED hijau sebagai indikator.

5. Selanjutnya, apabila operator menekan tombol “D” pada keypad, sistem *filling machine* akan menghentikan seluruh proses *conveyor* dan *water pump*. Setelah itu, sistem akan masuk ke mode pengaturan parameter volume air pada water pump serta pengaturan fungsi *disable* atau *enable water pump*, mode pengaturan akan ditampilkan pada LCD.
6. Jika operator sudah selesai melakukan pengaturan water pump bisa menekan tombol “D” untuk keluar dari mode pengaturan *water pump* untuk mulai sistem *filling* dan menjalan *conveyor*.
7. Selanjutnya operator bisa melakukan *input* botol pada *conveyor* yang berjalan.
8. Sensor *infrared* 2 akan berada pada kondisi *stand by* untuk mendeteksi keberadaan botol. Ketika botol terdeteksi, sensor akan mengirimkan sinyal *high* ke ESP32-S3 sehingga motor *stepper* berhenti beroperasi.
9. Selanjutnya, ESP32-S3 akan menunggu kondisi sensor *infrared* 1 bernilai *high* selama 1 detik. Apabila kondisi tersebut terpenuhi, *water pump* 1 dan *water pump* 2 akan aktif secara bersamaan hingga volume air mencapai target parameter yang telah ditentukan oleh operator. Jika kondisi *infrared* 1 tidak *high* selama 1 detik maka hanya *water pump* 2 yang aktif.
10. Selanjutnya, ketika *water pump* 1 dan *water pump* 2 aktif, motor *stepper* akan bergerak sejauh 4703 step atau setara dengan perpindahan sejauh 8 cm. Selama proses perpindahan 600 step berlangsung, sensor *infrared* 2 akan masuk ke dalam mode *idle* sehingga ESP32-S3 akan mengabaikan kondisi sensor *infrared* 2 yang bernilai *high*. Namun, apabila hanya *water pump* 2 yang aktif, sensor *infrared* 2 tidak akan masuk ke mode idle. Setelah proses tersebut selesai, algoritma sistem akan kembali melakukan perulangan mulai dari poin “H”.
11. ESP32-S3 akan mengirim data semua proses sensor, *water pump* dan motor stepper kepada ESP32 Dev Module menggunakan

jalur UART pada PIN TX dan RX. Selanjutnya, ESP32 Dev Module mengolah data tersebut dalam format JSON dan mengirimkannya ke server XAMPP melalui jaringan WiFi dengan komunikasi TCP/IP. Data kemudian diterima oleh script PHP untuk disimpan ke dalam *database* MySQL dan ditampilkan pada *dashboard monitoring* berbasis JavaScript yang menampilkan seluruh status sistem, seperti *conveyor* full, kondisi tangki *empty*, proses *filling*, status motor *stepper*, serta jumlah produksi botol secara *realtime*.

3.1.3 Speksifikasi Alat

Perancangan algoritma kendali yang terintegrasi IoT sebagai *monitoring* kerja alat serta mengamati jumlah produksi pada mesin pengisian otomatis dalam tugas akhir ini memiliki spesifikasi softwaew dan hardware pada tabel 3.1 dan 3.2 sebagai berikut :

3.1.3.1 Speksifikasi Hardware

Tabel 3. 1 Speksifikasi Komponen

NO	Nama Komponen	Speksfikasi
1.	ESP32-S3-WROOM-1U-N16R8	Dual-Core Xtensa LX7 32-bit Memori Flash : 16 MB Kecepatan Clock : Hingga 240 MHz PSRAM : 8 MB ROM : 384 KB SRAM Internal : 512 KB Tegangan Operasi : 3,0 – 3,6 VDC Tegangan Nominal : 3,3 VDC Jumlah GPIO : Hingga 45 pin Komunikasi : UART, SPI, I2C, I2S, SDIO

NO	Nama Komponen	Speksfikasi
2.	ESP32-WROOM-32	Dual-Core Xtensa LX6 32-bit Kecepatan Clock : Hingga 240 MHz Memori Flash : 4 MB ROM : 448 KB SRAM Internal : 520 KB RTC SRAM : 8 KB Tegangan Operasi : 3,0 – 3,6 VDC Tegangan Nominal : 3,3 VDC Wi-Fi : 2,4 GHz Jumlah GPIO : Hingga 32 GPIO Komunikasi : UART, SPI, I2C, I2S, SDIO
3.	Sensor Infrared E18-D80NK	Tegangan Operasi : 5 VDC Arus Kerja : ≤ 25 mA Arus Output Maksimum : 100 mA Jarak Deteksi : 3 cm – 80 cm Tipe Output : Digital NPN Waktu Respon : < 2 ms
4.	Sensoe Water Level XKC-Y26S-PNP	Tegangan Operasi : DC 5–12 V Tipe Output : PNP Arus Konsumsi : ≤ 5 mA Output High : DC 5–12 V

NO	Nama Komponen	Speksfikasi
		Output Low **:** 0 V Waktu Respon : 500 ms Ketebalan Dinding Maksimum : 20 mm Suhu Operasi : −20°C hingga +105°C
5.	Peristaltik kamoer KPK 400	Jenis Pompa : Peristaltic Pump Tegangan Kerja 24 VDC Arus : 0.4A Daya : 10 Watt Tipe Motor : DC Brushed Debit Maksimum : 400 mL/menit Jumlah Roller : 3 Roller Material Selang : BPT (Bioprene Tubing) Umur Penggunaan : : ±800 jam Tingkat Kebisingan : 68 dB
6.	Motor Stepper Nema 17	Jenis Motor : Hybrid Bipolar Stepper Motor Jumlah Fasa : 2 Fasa Sudut Langkah (Step Angle) : 1,8°/step Langkah per Putaran : 200 step/revolusi Jumlah Kabel : 4 kabel Arus per Fasa : 0,4 – 1,7 A Holding Torque : 5Kg

NO	Nama Komponen	Speksfikasi
7.	Driver TB6600	Tegangan Input :9–40 VDC Arus Output : 0,5–4,0 A Frekuensi Pulsa Maksimum : 20 kHz Resolusi Microstep : Full, 1/2, 1/4, 1/8, 1/16, 1/32 Level Sinyal Kontrol : 3,3–24 V
8.	Driver BTS7960	Tegangan Operasi : 5,5 – 27,5 VDC Tipe Driver : H-Bridge Arus Maksimum : 43 A Frekuensi PWM : Hingga 25 kHz Tegangan Logika : 3,3 – 5 V Resistansi Internal : 16 mΩ
9.	LCD 20x4 I2C	Resolusi Karakter : 20 × 4 Tegangan Operasi : 5 VDC Kontroler : HD44780 Antarmuka : I2C (PCF8574) Alamat I2C : 0x27 Pin Komunikasi : SDA, SCL
10.	Keypad 4x4 I2C	Jumlah Tombol : 16 Tombol Konfigurasi : 4×4 Tegangan Operasi : 5 VDC Komunikasi : I2C (PCF8574)

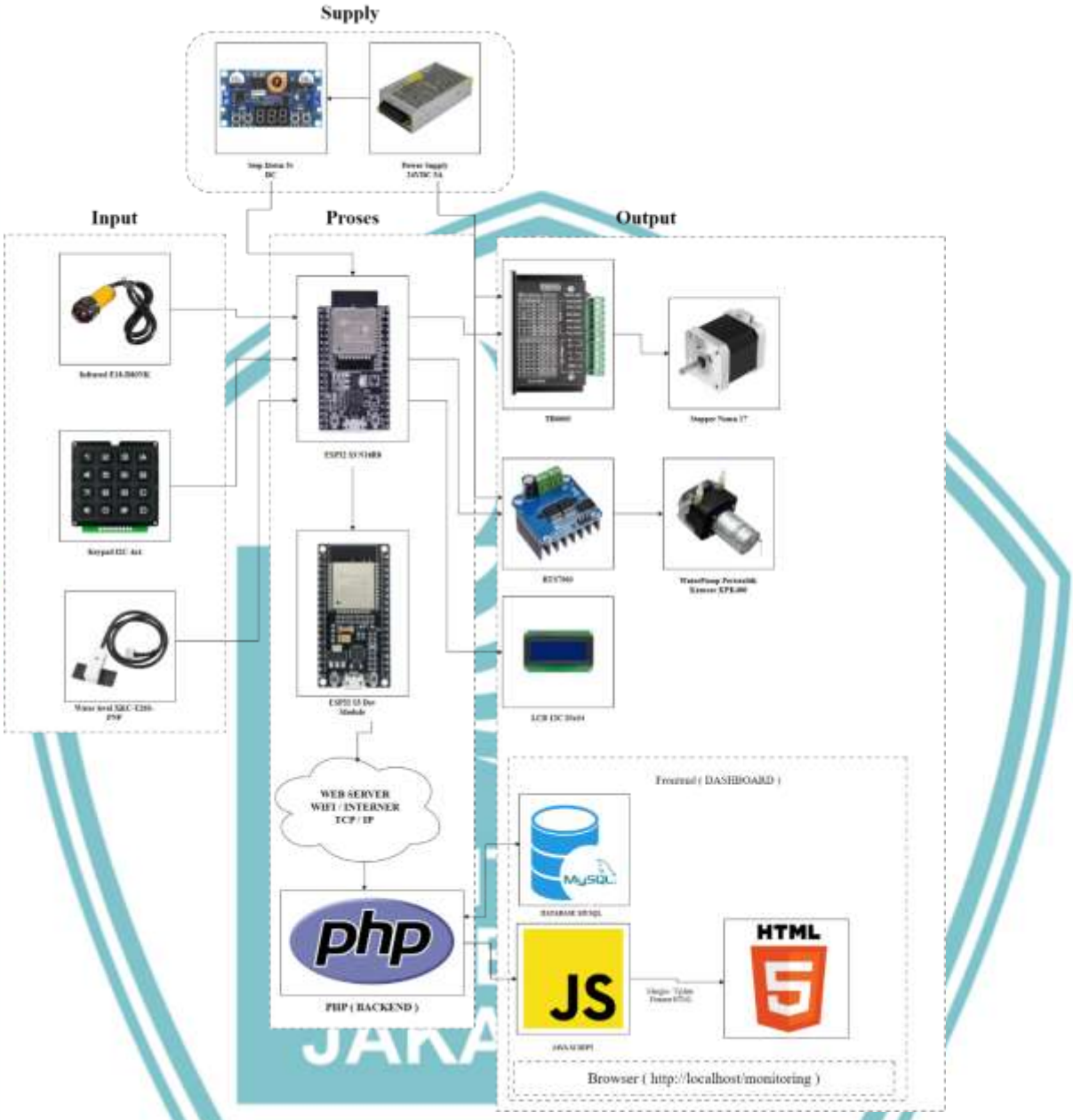
		Alamat I2C : 0x21 Pin Komunikasi : SDA, SCL
NO	Nama Komponen	Speksfikasi
11.	Relay 4 Channel	Tegangan Kerja : 5 VDC Tegangan Trigger : 3,3–5 V Arus Kontak Maksimum 10 A Tegangan Kontak AC : 250 VAC Tegangan Kontak DC : 30 VDC Jenis Kontak : SPDT (NO, NC, COM) Waktu Operasi : ≤ 10 ms

3.1.3.2 Speksifikasi Software

Tabel 3. 2 Speksifikasi Aplikasi

No	SoftWare	Vesion	Menu
1.	Arduino Ide	2.3.6	*Sketch, Serial Monitor, Board Manager, Library Manager, Upload Program, Tools, Edit, Help.*
2.	Visual Studio Code	1.123	*Explorer, HTML, PHP, CSS, JavaScript, Terminal, Extensions.*
3.	XAMPP	3.3.0	*Apache, MySQL, phpMyAdmin*
4.	MySql	3.3.0	*Database, Table, Query SQL, Import/Export Data*

3.1.4 Blok Diagram



Gambar 3. 2 Blok Diagram

Blok diagram pada gambar 3.2 sebagai perancangan sistem *filling machine* yang terintegrasi *internet of things* yang terdiri dari bagian *input*, proses dan *output*. Berikut penjelasan proses blok diagram tersebut :

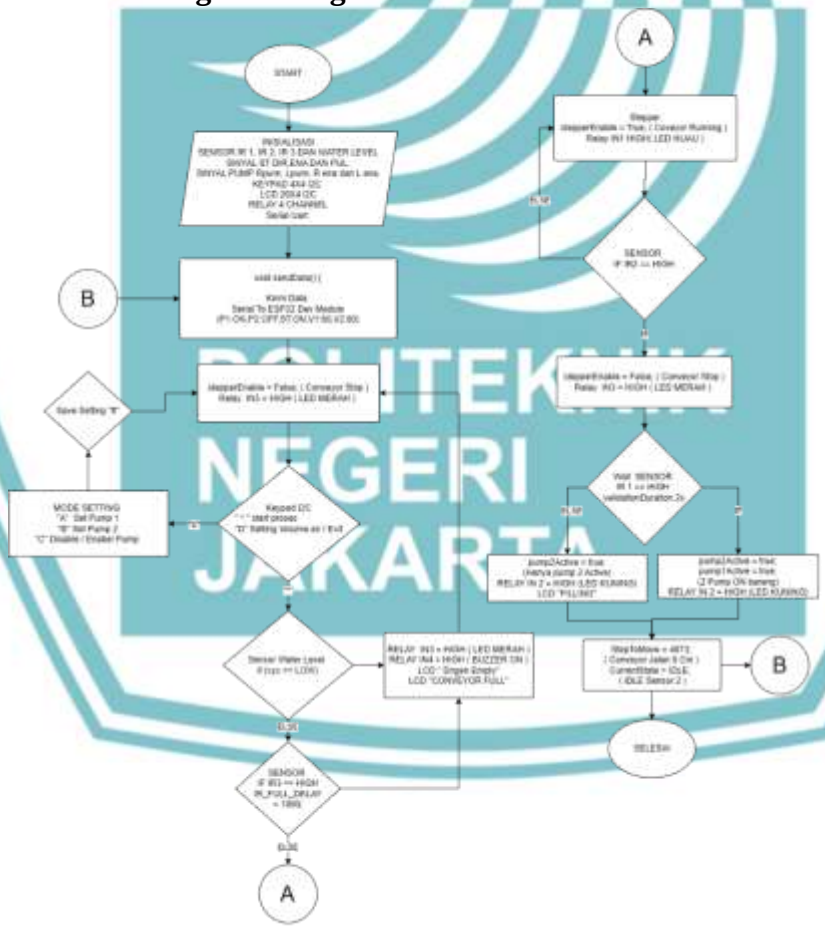
1. Sensor *infarared* digunakan untuk mendeteksi keberadaan botol pada area pengisian

2. Sensor XKC-Y26S-PNP digunakan sebagai mendeteksi ketersediaan cairan di dalam tangki
3. Keypad I2C 4×4 sebagai media komunikasi operator dengan alat untuk menentukan parameter pengisian.
4. ESP32-S3 N16R8 yang berfungsi menerima *input* sebagai pengendali utama sistem filling machine.
5. Driver TB6600 yang digunakan untuk menggerakkan motor stepper NEMA 17 sebagai penggerak konveyor yang dikendalikan oleh ESP32-S3.
6. Driver BTS7960 yang digunakan untuk mengatur kerja water pump peristaltik KPK400 sebagai aktuator pengisian cairan ke dalam botol yang dikendalikan oleh ESP32-S3.
7. Power supply 24VDC 5A yang digunakan untuk mensuplai aktuator, sedangkan modul step-down 5 VDC digunakan untuk menurunkan tegangan sehingga sesuai dengan kebutuhan mikrokontroler dan komponen yang membutuhkan tegangan 5VDC.
8. ESP32 berfungsi sebagai media komunikasi yang menghubungkan sistem kendali dengan server *monitoring* menggunakan jaringan WiFi melalui protokol TCP/IP.
9. PHP befungsi menerima data yang telah dikirim oleh ESP32 melewati komunikasi wifi, serta mengolah data tersebut dan menyimpan kedalam database MySql dan mengirim kembali Javascript
10. MySql berfungsi sebagai penyimpanan data yang telah diproses oleh PHP
11. JavaScript befungsi sebagai kendali untuk menampilkan data dan kondisi sistem *filling machine* yang akan ditampilkan oleh HTML
12. HTML befungsi sebagai menampilkan informasi *monitoring* secara *real-time.*

3.1.5 Perancangan Algoritma kendali dan Intergrasi IoT *Monitoring*

Pada perancangan algoritma kendali yang terintegrasi dengan sistem IoT *monitoring* ini digunakan dua mikrokontroler, yaitu ESP32-S3 dan ESP32. ESP32-S3 berfungsi sebagai pengendali utama proses kerja sistem alat, sedangkan ESP32 berfungsi untuk menerima data dari proses kerja alat serta mengirimkan data tersebut ke server melalui komunikasi wifi menggunakan protokol TCP/IP. Data yang dikirim digunakan untuk sistem IoT *monitoring* sehingga proses kerja alat dan jumlah produksi dapat ditampilkan pada dashboard web secara *realtime*. Berikut proses *flowchart* pemograman kendali yang terinterasi IoT sebagai *dashboard monitoring* :

3.1.5.1 Perancangan Pemograman ESP32-S3



Gambar 3.3 Flowchart Pemograman ESP32-S3

Gambar *flowchart* yang ditampilkan merupakan perancangan pemrograman mikrokontroler ESP32-S3 sebagai algoritma kendali sistem pada alat. Flowchart tersebut menggambarkan proses kerja sistem dalam mengolah *input* dari sensor serta mengendalikan output aktuator. Berikut merupakan alur pemrograman kendali pada ESP32-S3 :

1. ESP32-S3 melakukan inisialisasi seluruh komponen, meliputi sensor IR1, IR2, IR3, sensor water level, PWM driver, relay, LCD I2C, keypad, driver stepper, serta komunikasi serial UART.
2. Kondisi awal sistem diawali dengan motor *stepper* atau *conveyor* dalam keadaan berhenti. Serta dengan kondisi relay pin 3 *high* yang menandakan indikator LED merah menyala.
3. ESP32-S3 akan memastikan sensor *Infrared* 1 memberikan sinyal *low*, Jika sensor memberikan nilai *high* maka ESP32-S3 akan melakukan *stepperEnable = False* memberhentikan motor *stepper* dan memberikan sinyal relay pin 3 *high* dan ESP32-S3 update LCD menampilkan "*Conveyor full*"
4. Selanjutnya, ESP32-S3 akan memastikan sensor XKC memberikan sinyal *high* ke ESP32-S3. Jika sensor berada pada kondisi *low*, ESP32-S3 akan mengaktifkan relay pin 4, sehingga *buzzer* akan menyala dan LCD menampilkan informasi bahwa drigen dalam keadaan kosong, serta ESP32-S3 akan melakukan *stepperEnable = False* yang memberhentikan motor stepper.
5. Selanjutnya jika sensor XKC ini sesuai kondisi. ESP32-S3 akan *stand by* untuk menerima *input* "*" dari keypad untuk memulai proses kerja alat.
6. Ketika ESP32-S3 menerima *input* "D", motor *stepper* akan berhenti beroperasi dan sistem masuk ke dalam mode pengaturan yang ditampilkan melalui LCD. Mode pengaturan ini digunakan untuk mengatur parameter volume air *pada water pump* 1 maupun *water pump* 2, serta untuk melakukan

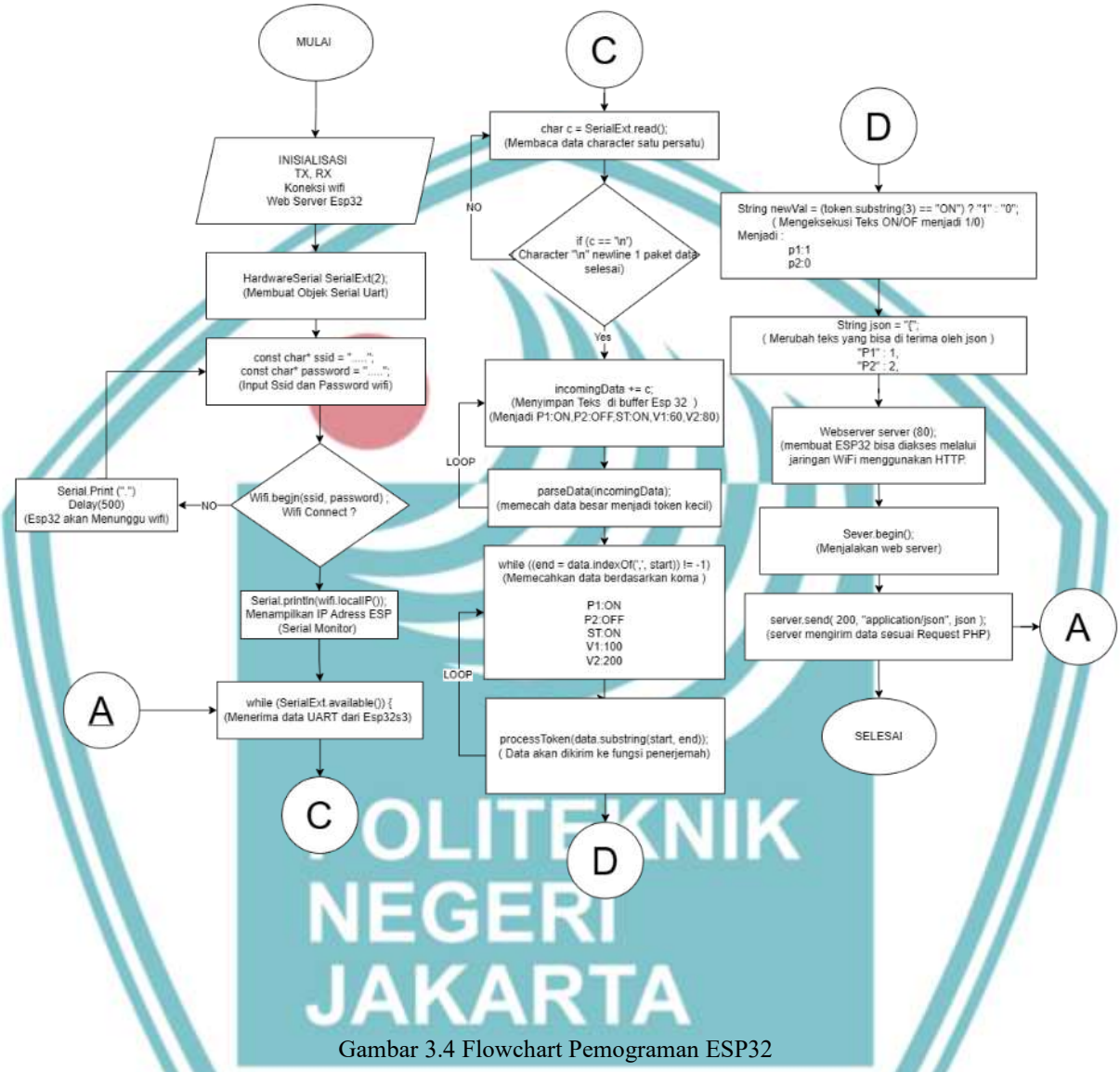
pengaktifan *enable* atau penonaktifan *disable* pada masing-masing *water pump.*

7. Ketika ESP32-S3 menerima *input* "#", sistem akan menyimpan *save* seluruh *parameter* pengaturan yang telah dikonfigurasi sebelumnya.
8. Selanjutnya, selama sistem masih berada dalam mode pengaturan, ESP32-S3 akan menunggu *input* "D" untuk keluar dari mode pengaturan dan melanjutkan proses kerja sistem kendali.
9. Selanjutnya, ESP32-S3 akan memberikan sinyal aktif pada pin driver ENA untuk mengaktifkan motor *stepper* sehingga *conveyor* dapat berjalan. Selain itu, ESP32-S3 juga memberikan sinyal *high* pada pin relay IN1 untuk mengaktifkan indikator hijau, serta memperbarui tampilan LCD dengan menampilkan status "*Running*…".
10. Selanjutnya, ESP32-S3 akan menunggu sinyal dari sensor *infrared* 2 bernilai *high*. Jika sensor *infrared* 2 belum memberikan sinyal high kepada ESP32-S3, maka sistem akan melakukan pengulangan proses kembali ke poin "8".
11. Jika ESP32-S3 menerima sinyal *high* dari sensor infrared 2, maka ESP32-S3 akan memberikan sinyal *low* pada pin driver ENA untuk menghentikan motor stepper, sehingga *conveyor* berhenti beroperasi.
12. Setelah *conveyor* berhenti, ESP32-S3 akan menunggu validasi sinyal dari sensor *infrared* 1 bernilai *high* selama 1 detik. Selain itu ESP32-S3 akan *update* LCD " menunggu sensor *irfarared* 1
13. Jika sensor *infrared* 1 tidak memberikan sinyal *high* sesuai waktu validasi, maka ESP32-S3 akan mengaktifkan *pump2Active = true*, sehingga hanya *water pump* 2 yang bekerja. *Water pump* 2 akan bekerja hingga volume air mencapai parameter yang telah ditentukan. Selain itu, relay IN2

akan diberikan sinyal *high* untuk mengaktifkan indikator LED kuning dan ESP32-S3 akan update LCD " *Filling*…..".

14. Namun, jika sensor *infrared* 1 memberikan sinyal *high* selama 1 detik, maka ESP32-S3 akan mengaktifkan *pump1Active* = *true* dan *pump2Active* = *true* sehingga kedua *water pump* bekerja secara bersamaan. *Water pump* 1 dan *water pump* 2 akan tetap menyala hingga volume air mencapai parameter yang telah ditentukan. Pada kondisi ini, relay IN2 juga akan diberikan sinyal *high* untuk mengaktifkan indikator LED kuning dan ESP32-S3 akan update LCD " *Filling*…..".
15. Selanjutnya, ESP32-S3 akan memberikan perintah StepToMove = 4703 pada motor *stepper* sehingga *conveyor* bergerak sejauh 8 cm. Selama proses *StepToMove* berlangsung, ESP32-S3 akan mengabaikan sinyal dari sensor *infrared* 2 hingga proses stepmove mencapai target *step*.
16. Selanjutnya, ketika *stepmove* ini sudah mencapai target maka ESP32-S3 tidak mengabaikan *infrared* 2 setelah itu pada proses ini akan looping ke poin "2".
17. Selanjutnya, ESP32-S3 akan menjalankan fungsi *sendData*() untuk mengirimkan data proses ke ESP32 Dev Module melalui komunikasi serial. Data yang dikirim meliputi status *water pump* 1 (P1), *water pump* 2 (P2), *status sistem* (ST), serta parameter volume air (V1 dan V2). Contoh data yang dikirim yaitu P1:ON, P2:OFF, ST:ON, V1:60, V2:80.

3.1.5.2 Perancangan Pemograman ESP32 Komunikasi Data



Gambar 3.4 Flowchart Pemograman ESP32

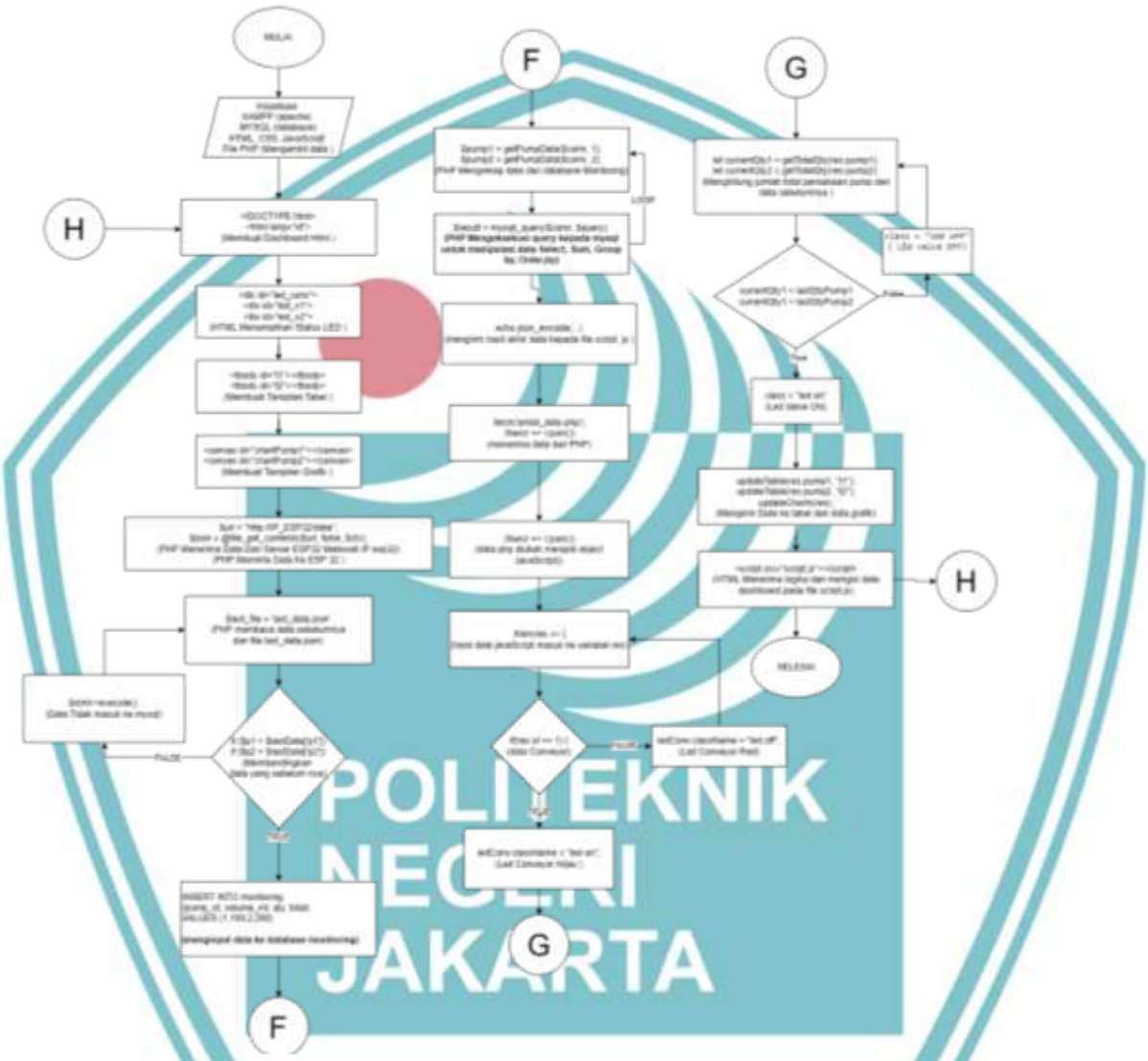
Flowchart pada Gambar 3.4 menjelaskan proses kerja sistem monitoring berbasis IoT menggunakan ESP32 Dev Module sebagai penerima data dari ESP32-S3 serta pengirim data menuju web server melalui jaringan WiFi. Sistem ini dirancang untuk melakukan komunikasi data secara *realtime* sehingga kondisi proses alat dapat dipantau melalui *dashboard* web. Berikut merupakan alur pemrograman kendali pada ESP32 Dev Module :

1. ESP32 diawal dengan inisialisasi komunikasi TX dan RX, konfigurasi koneksi WiFi, serta pembuatan web server pada ESP32 Dev Module.
2. Selanjutnya, ESP32 Dev Module ini akan melakukan konfigurasi objek *HardwareSerial SerialExt2* sebagai media komunikasi UART antara ESP32-S3 dan ESP32 Dev Module
3. Selanjutnya, ESP32 Dev Module akan melakukan pengaturan *SSID* dan *password* WiFi yang digunakan untuk menghubungkan perangkat ke jaringan internet. Sistem kemudian melakukan proses koneksi WiFi menggunakan fungsi *WiFi.begin*(*ssid, password*). Jika koneksi belum berhasil, maka ESP32 akan terus melakukan percobaan koneksi hingga berhasil terhubung ke jaringan WiFi.
4. Setelah koneksi berhasil, alamat IP ESP32 akan ditampilkan pada serial monitor sebagai indikator bahwa perangkat telah terkoneksi dengan jaringan dan menampilkan IP ESP32 Dev Module
5. ESP32 Dev Module menunggu data UART yang dikirimkan oleh ESP32-S3. Data diterima dalam bentuk karakter per karakter melalui komunikasi serial menggunakan fungsi *SerialExt.read*(). Karakter yang diterima kemudian disimpan sementara ke dalam variabel *incomingData* hingga ditemukan karakter *newline* (\n) sebagai penanda bahwa satu paket data telah selesai diterima.
6. Data yang telah diterima oleh ESP32 Dev Module, Selanjutnya diproses menggunakan fungsi *parseData(incomingData*) untuk memisahkan data berdasarkan tanda koma. Data tersebut terdiri dari informasi status *water pump* 1 (P1), *water pump* 2 (P2), *status stepper* (ST), serta parameter volume air (V1 dan V2).
7. ESP32 Dev Module menjalankan fungsi *processToken()* untuk menerjemahkan setiap token data yang diterima. Pada tahap ini, teks *ON*/*OFF* akan dikonversi menjadi logika digital 1 atau 0

agar data dapat diproses lebih lanjut oleh sistem dan dikirimkan dalam format JSON.

8. Selanjutnya, ESP32 Dev Module akan menjalankan web server menggunakan fungsi *server.begin().* Web server ini berfungsi sebagai media pengambilan data berbasis HTTP sehingga data proses alat *filling machine* dapat diakses melalui *dashboard* web secara *realtime*.
9. Ketika terdapat permintaan data dari *dashboard* web, server akan mengirimkan data dalam format JSON menggunakan fungsi *server.send(200, "application/json", json)* setelah itu program akan looping point “5”.

3.1.5.3 Perancangan Pemograman Dashboard Monitoring



Gambar 3.5 Flowchart Pemograman Dashboard Monitoring

Flowchart pada Gambar 3.5 menjelaskan alur kerja sistem *dashboard monitoring* berbasis IoT yang digunakan untuk menampilkan data proses alat secara *realtime* melalui web *monitoring*. Sistem ini memanfaatkan kombinasi teknologi HTML, CSS, JavaScript, PHP, dan database MySQL untuk mengolah, menyimpan, serta menampilkan data yang dikirim dari ESP32. Pada sistem ini, ESP32 berfungsi sebagai pengirim data *monitoring*,

sedangkan web server berfungsi untuk menerima, memproses, dan menampilkan data pada *dashboard monitoring*. Berikut alur kerja sistem *dashboard monitoring* dijelaskan sebagai berikut.

1. Proses dimulai dengan melakukan inisialisasi perangkat lunak yang digunakan pada sistem *monitoring*, yaitu XAMPP sebagai web server Apache, MySQL sebagai *database*, serta HTML, CSS, JavaScript, dan file PHP sebagai pendukung tampilan dan pengolahan data pada dashboard *monitoring*.
2. Setelah proses inisialisasi selesai, sistem akan membuat struktur halaman *dashboard* menggunakan kode HTML *(<!DOCTYPE html>).* Pada tahap ini, dashboard dirancang untuk menampilkan informasi *monitoring* sistem secara *realtime*.
3. Selanjutnya, HTML membuat elemen tampilan LED indikator menggunakan tag * * untuk menampilkan status *conveyor* dan *water pump* pada *dashboard*. Indikator ini digunakan sebagai indikator kondisi sistem apakah sedang aktif atau tidak aktif.
4. HTML kemudian membuat tampilan tabel *monitoring* menggunakan tag *<tbody id="T1">* dan *<tbody id="T2">*. Tabel ini berfungsi untuk menampilkan data hasil monitoring *water pump* 1 dan *water pump* 2 yang tersimpan pada *database*. Selain itu HTML membuat tampilan grafik menggunakan elemen *<canvas>* untuk menampilkan data monitoring dalam bentuk grafik. Grafik ini digunakan untuk mempermudah pengguna dalam melihat perubahan data volume dan jumlah pemakaian *water pump* secara *visual*.
5. Selanjutnya, file PHP akan menerima data dari ESP32 melalui metode HTTP menggunakan format JSON. Data yang diterima kemudian dibandingkan dengan data sebelumnya yang tersimpan pada file *last_data.json*

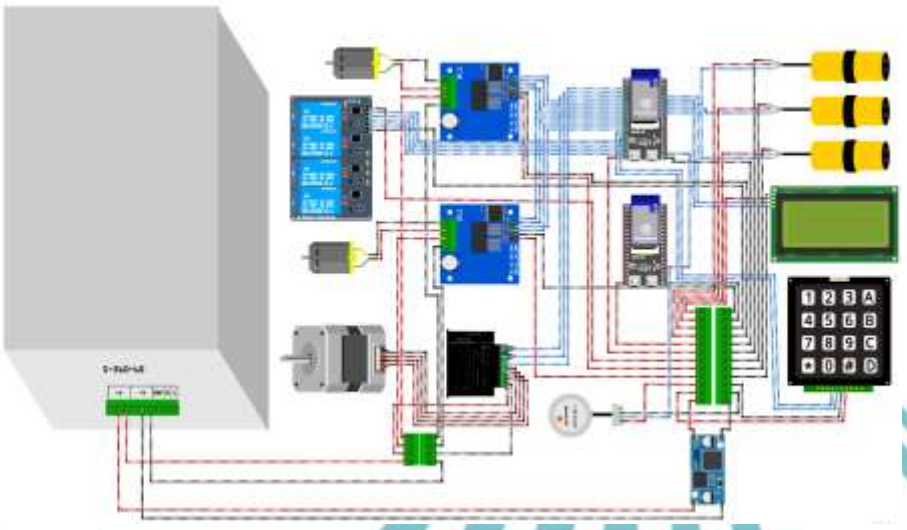
6. Jika data terbaru lebih besar dibandingkan data sebelumnya, maka sistem akan menyimpan data tersebut ke dalam *database* MySQL menggunakan perintah *INSERT INTO monitoring*. Namun, jika data tidak berubah, maka sistem tidak akan melakukan proses penyimpanan data.
7. Setelah data tersimpan di database, file PHP akan mengambil data monitoring menggunakan query MySQL seperti *SELECT, SUM, GROUP BY*, dan *ORDER BY*. Data hasil *query* kemudian dikirimkan ke file JavaScript dalam format JSON menggunakan fungsi echo json_encode().
8. Selanjutnya, Javascript akan menerima data tersebut menggunakan fungsi *fetch*(), kemudian mengubah data JSON menjadi objek JavaScript agar dapat diproses dan ditampilkan pada *dashboard*.
9. JavaScript akan melakukan pembaruan tampilan *dashboard* secara *realtime*. Jika status *conveyor* aktif (*res.st == 1*), maka indikator LED *conveyor* akan berubah menjadi hijau (*led on*). Sebaliknya, jika *conveyor* tidak aktif, maka indikator akan berubah menjadi merah (*led off*). Selain itu, JavaSctript juga akan menghitung total penggunaan *water pump* berdasarkan data sebelumnya. Jika jumlah pemakaian bertambah, maka indikator valve *water pump* akan berubah menjadi aktif (*led on*). Selanjutnya, data *monitoring* akan diperbarui pada tabel dan grafik *dashboard* sehingga pengguna dapat memantau kondisi sistem secara *realtime* melalui halaman web.

3.2 Realisasi Alat

3.2.1 Realisasi Hardware

Realisasi hardware pada sistem pengisian otomatis yang terintegrasi IoT ini difokuskan pada algortima kendali dan pemrograman sensor, *water pump*, dan motor *stepper* serta komunikasi antara ESP32-S3 dengan ESP32 yang akan terintegrasi dengan IoT dengan komunikasi wifi menggunakan protocol TCP/IP yang hasilnya akan ditampilkan

pada *dashboard monitoring*. Gambar 3.6 ini merupakan *wiring diagram* sistem pengisian otomatis.



Gambar 3. 6 Wiring Diagram

3.2.2 Realisasi Pemrograman Kendali ESP32-S3

3.2.2.1 Deklarasi Mode dan State Machine

Tabel 3. 3 Program State Machine

```
// ======== MODE & STATES =========
enum Mode {
  MODE_RUN,      // Mode menjalankan sistem stepper
  MODE_SETTING   // Mode pengaturan parameter volume
};
// State untuk logika sensor IR2
enum SystemState {
  IDLE,                 // Sistem standby menunggu botol
  WAITING_VALIDATION,   // Menunggu validasi sensor IR 1
  PUMPING,              // Proses Filling cairan
  STEPPER_MOVING_AWAY    // Stepper bergerak 8Cm
};
```

Realisasi deklarasi Mode dan Statein digunakan untuk mengatur alur kerja sistem pengisian otomatis in agar proses kerja pada alat dapat berjalan secara teratur. Berikut penjelasan fungsi sistem *Mode* dan *Systemstate* :

1. Pada deklarasi *enum Mode* digunakan untuk menentukan *mode* operasi sistem yang terdiri dua kondisi, yaitu *MODE_RUN* berfungsi sebagai menjalakan sistem otomatis *filling* dan sistem konveyor. Sedangkan, *MODE_SETTING*

digunakan untuk melakukan pengaturan parameter volume pada water pump yang akan di input melalui keypad.

2. Pada deklarasi *enum SystemState* digunakan untuk sebagai tahapan proses kerja sistem mesin ini yang terdiri dari 4 kondisi, yaitu *IDLE* ini merupakan kondisi awal sistem saat konveyor berjalan dan pada sensor *infrared* 2 *stand by* mendeteksi objek. Setelah itu, proses kendali ini masuk kedalam kondisi *WAITING_VALIDATION* pada kondisi ini sistem menunggu pembacaan sensor dari *infrared* 1 apakah ada objek. Selanjutnya, mesin tersebut akan lanjut kedalam kondisi *PUMPING* ini mulai melakukan proses pengisian cairan sesuai *volume* yang telah ditentukan. Setelah selesai pada proses *filling* ini sistem akan berpindah ke state *STEPPER_MOVING_AWAY* pada bagian ini motor stepper melakukan pergerakan 8 cm untuk dan kondisi *infrared* 2 diabaikan.

3.2.2.2 Loop Sistem

Tabel 3. 4Pemograman Loop Sistem

```
void loop() {
    handleKeypad();
// Mengambil waktu saat ini
unsigned long now = millis();
 // Jika sistem belum dijalankan
  if (!systemStarted) {
// Sistem standby menunggu tombol start
   }
// Jika mode sistem RUN
     else if (currentMode == MODE_RUN) {
   // Menjalankan logika utama filling machine
logicProcess(now);}
// Jika mode setting
      else {
 // Menjalankan menu pengaturan parameter
 settingsProcess();  }

static unsigned long lastSend = 0;
// Mengirim data setiap 500 ms
if (now - lastSend > 500) {// Mengirim data sensor dan status
sistem
```

```
 sendData();
// Reset timer pengiriman
lastSend = now; }
 }
```

Realiasi fungsi *loop* sistem sebagai program yang berjalan secara terus menerus pada mikrokontroler ESP32-S3. Bagian ini befungsi sebagai ketika sistem baru dinyalakan oleh operator, maka sistem akan menunggu *input* "*" dari keypad sebagai masuk kedalam *systemStarted = false*. Jika *systemStarted = true* bernilai benar atau operator memberikan nilai "*" maka sistem menjalankan *logicProcess(now);* logika utama *filling machine* seperti pembacaan sensor *infrared*, proses pengisian cairan, pengendalian motor *pump*, penggerak konveyor *stepper*, serta sistem proteksi dan alarm.

Selain itu, jika sistem menerima input keypad "D" sistem akan menjalankan fungsi *settingsProcess();* untuk melakukan menu pengaturan parameter volume *filling* pada LCD.

Pada bagian akhir ini terdapat proses pengiriman data telemetri ke ESP32 kedua menggunakan *sendData();* yang berfungsi untuk mengerim data status sistem yang dilakukan setiap 500 ms menggunakan metode *timer non-blocking* dengan bantuan variabel *lastSend*. Data telemetri yang dikirim meliputi status *pump*, status konveyor, *volume filling*, sensor level cairan, dan kondisi konveyor penuh.

3.2.2.3 Pemograman Sistem State Mesin

Tabel 3. 5 Pemograman Sistem State Meisn

```
void logicProcess(unsigned long now) {
  // Membaca sensor infrared
  int ir1 = digitalRead(IR1_PIN);
  int ir2 = digitalRead(IR2_PIN);

  // ========================
  // PRIORITAS 1 : CEK DRIGEN EMPTY
  // ========================
  // Membaca status sensor level cairan
  bool isDryingEmpty = checkXYCFiltered();
  // Jika cairan habis
```

```
  if (isDryingEmpty) {
    // Mengunci alarm error
    xycAlarmLock = true;
  }
  // Jika alarm aktif
  if (xycAlarmLock) {
    //============================
    // STOP SEMUA AKTUATOR
    //============================
    // Stop stepper
    stepperEnable = false;
    // Disable driver stepper
    digitalWrite(EN_PIN, HIGH);
    // Reset langkah stepper
    stepsToMove = 0;
    // Stop pump
    ledcWrite(RPWM1, 0);
    ledcWrite(RPWM2, 0);
    pump1Active = false;
    pump2Active = false;
    //============================
    // INDIKATOR RELAY ERROR
    //============================
    digitalWrite(RELAY_GREEN, LOW);
    digitalWrite(RELAY_YELLOW, LOW);
    digitalWrite(RELAY_RED, HIGH);
    //============================
    // BUZZER BERKEDIP NON BLOCKING
    //============================
    if (millis() - buzzerTimer >= (buzzerState ? buzzerOnTime :
buzzerOffTime)) {
      // Toggle status buzzer
      buzzerState = !buzzerState;
      // Reset timer buzzer
      buzzerTimer = millis();
      // Aktifkan relay buzzer
      digitalWrite(RELAY_BUZZER, buzzerState ? HIGH :
LOW);}
    //============================
    // TAMPILAN LCD EMPTR
    //=========================
    // Jika tampilan belum pernah muncul
    if (!lastXYCEmptyState) {
      // Bersihkan LCD
      lcd.clear();
      // Tampilkan pesan EMPTY
      lcd.setCursor(3, 1);
      lcd.print("DRIGEN EMPTY!");
```

```
    lcd.setCursor(7, 2);
    lcd.print("REFILL");
    // Status tampilan sudah muncul
    lastXYCEmptyState = true;
    // Reset cache LCD
    for (int i = 0; i < 4; i++) {
      lastLine[i] = "";
    }
  }
  //============================
  // RECOVERY SISTEM
  //============================
  // Jika cairan kembali terdeteksi
  if (digitalRead(XYC_SENSOR_PIN) == HIGH) {
    // Jika timer recovery belum dimulai
    if (xycHighStart == 0) {
      xycHighStart = millis();
    }
    // Jika HIGH stabil sesuai delay
    if (millis() - xycHighStart >= XYC_HIGH_DELAY) {
      // Reset seluruh alarm
      xycAlarmLock = false;
      lastXYCEmptyState = false;
      xycHighStart = 0;
      // Matikan buzzer
      digitalWrite(RELAY_BUZZER, LOW);
      // Matikan indikator merah
      digitalWrite(RELAY_RED, LOW);
      // Bersihkan LCD
      lcd.clear();
    }
      } else {
        // Reset timer recovery
          xycHighStart = 0;
        }
  // Kembali ke state idle
  currentState = IDLE;
  // Keluar dari fungsi
  return;
}
// ========================
// PRIORITAS 2 : CEK CONVEYOR FULL
// ========================
// Jika conveyor penuh
if (checkIRFullFiltered()) {
  // Stop stepper
  stepperEnable = false;
  // Disable driver stepper
```

```
  digitalWrite(EN_PIN, HIGH);
  // Stop seluruh pump
  ledcWrite(RPWM1, 0);
  ledcWrite(RPWM2, 0);
  pump1Active = false;
  pump2Active = false;
  // Relay emergency
  digitalWrite(RELAY_GREEN, LOW);
  digitalWrite(RELAY_YELLOW, LOW);
  digitalWrite(RELAY_RED, HIGH);
  digitalWrite(RELAY_BUZZER, LOW);
  // Kembali ke state idle
  currentState = IDLE;
  // Tampilkan status conveyor penuh
  updateLCD(3, "CONVEYOR FULL!");
  return;
 }
 //======================
 // STATE MACHINE SISTEM
 // ======================
 switch (currentState) {
  //======================
  // STATE IDLE
  //======================
  case IDLE:
   // Aktifkan stepper conveyor
   stepperEnable = true;
   // Enable driver stepper
   digitalWrite(EN_PIN, LOW);
   // Tampilkan status running
   updateLCD(3, "RUNNING...    ");
   // Jika IR2 mendeteksi botol
   if (ir2 == LOW) {
    // Stop conveyor
    stepperEnable = false;
    // Disable driver stepper
    digitalWrite(EN_PIN, HIGH);
    // Simpan waktu validasi
    validationStart = now;
    // Pindah ke state validasi
    currentState = WAITING_VALIDATION;
   }
   break;
  //======================
  // STATE VALIDASI SENSOR
  //======================
  case WAITING_VALIDATION:
   // Tampilkan status validasi
```

```
    updateLCD(3, "WAITING IR 1");
    // Jika waktu validasi selesai
    if (now - validationStart >= validationDuration) {
      // Jika IR1 aktif dan pump1 enable
      if (digitalRead(IR1_PIN) == LOW && pump1Enable) {
        // Aktifkan pump 1
        pump1Active = true;
        pump1Start = now;
        ledcWrite(RPWM1, SPEED);
        // Aktifkan pump 2
        pump2Active = true;
        pump2Start = now;
        ledcWrite(RPWM2, SPEED);
      }
      // Jika tidak maka hanya pump 2 aktif
      else if (pump2Enable) {
        // Aktifkan pump 2
        pump2Active = true;
        pump2Start = now;
        ledcWrite(RPWM2, SPEED);
      }
      // Pindah ke state pumping
      currentState = PUMPING;
    }
    break;
  //======================
  // STATE PENGISIAN
  //======================
  case PUMPING:
    // Tampilkan status filling
    updateLCD(3, "FILLING...   ");
    // Jika durasi pump 1 selesai
    if (pump1Active && now - pump1Start >=
pump1Duration) {
      // Matikan pump 1
      ledcWrite(RPWM1, 0);
      pump1Active = false;
    }
    // Jika durasi pump 2 selesai
    if (pump2Active && now - pump2Start >=
pump2Duration) {
      // Matikan pump 2
      ledcWrite(RPWM2, 0);
      pump2Active = false;
    }
    // Jika seluruh pump selesai
    if (!pump1Active && !pump2Active) {
      // Jumlah langkah conveyor
```

```
      stepsToMove = 600;
      // Aktifkan stepper
      stepperEnable = true;
      // Enable driver stepper
      digitalWrite(EN_PIN, LOW);
      // Pindah ke state stepper moving
      currentState = STEPPER_MOVING_AWAY;
    }
    break;

  //=========================
  // STATE CONVEYOR BERGERAK
  //=========================
  case STEPPER_MOVING_AWAY:
    // Tampilkan status conveyor bergerak
    updateLCD(3, "MOVING 600 STEPS");
    // Jika langkah selesai
    if (stepsToMove <= 0) {
      // Kembali ke state idle
      currentState = IDLE;
    }
    break;
// ==========================
// LOGIKA RELAY
// ==========================
// LED hijau aktif saat conveyor berjalan
if (stepperEnable) {
  digitalWrite(RELAY_GREEN, HIGH);
} else {
  digitalWrite(RELAY_GREEN, LOW);
}
// LED kuning aktif saat filling
if (pump1Active || pump2Active) {
  digitalWrite(RELAY_YELLOW, HIGH);
} else {
  digitalWrite(RELAY_YELLOW, LOW);
}
// Matikan LED merah jika normal
digitalWrite(RELAY_RED, LOW);

// ==========================
// TAMPILAN LCD INFORMASI SISTEM
// ==========================
// Judul sistem
updateLCD(0, "FILLING MACHINE");
// Informasi volume pump
updateLCD(
  1,
```

```
    "V1:" + String(pump1Volume) +
    "ml V2:" + String(pump2Volume) + "ml");
  // Informasi status pump
  updateLCD(
    2,
    "P1:" + String(pump1Active ? "ON " : "OFF") +
    " P2:" + String(pump2Active ? "ON" : "OFF"));
}
```

Fungsi *logicProcess()* merupakan fungsi utama yang mengendalikan seluruh proses kerja sistem *fillingg machine* otomatis, fungsi ini dijalankan secara terus menerus yang bertanggung jawab terhadap pembacaan sensor, pengendali actuator, sistem proteksi dan perpindahan state pada proses *filling machine* berikut penjelasan fungsi program fungsi pada tabel 3.9 :

1. Pada bagian awal fungsi dilakukan pembacaan sensor *infrared* 1 dan *infrared* 2 yang sebagai pendeteksi keberadaan botol.
2. Selanjutnya, sebelum masuk kedalam kondisi *state* proses perogaram ini akan melakukan pembacaan sensor *infrared* 3 yang sebagai kondisi *conveyor full* dan sensor XKC yang sebagai pembacaan ketersedian cairan pada tangki penampung.
3. fungsi *bool isDryingEmpty = checkXYCFiltered();* pada bagian ini membaca status sensor XKC, jika *xycEmptyValid* kondisi *true* maka sistem masuk kedalam kondisi *xycAlarmLock* kondisi *true* pada kondisi ini sistem akan melakukan pemberhentian semua aktuator motor stepper dan motor water pump. Selain itu, sistem akan mengaktifkan relay buzzer dan relay led merah, serta menampilkan teks pada lcd *"DRIGEN EMPTY!"* yang sebagai indikator. Sistem hanya akan kembali beroperasi apabila sensor XKC dalam kondisi *high* secara stabil selama waktu validasi yang telah ditentukan. Setelah proses validasi berhasil, alarm akan direset dan sistem kembali ke kondisi normal.
4. Fungsi *if (checkIRFullFiltered())* pada bagian ini befungsi sebagai pembacaan sensor *infrared* 3 pada ujung konveyor yang mendeteksi penumpukan gerbong botol pada konveyor sebagai

proteksi kemacetan produksi. Jadi jika *irFullValid* pada kondisi maka sistem akan melakukan memberhentikan semua proses actuator seperti motor stepper dan water pump dan sistem akan mengaktifkan led merah serta menampilkan teks pada lcd *"CONVEYOR FULL!"* yang sebagai indikator pada sistem bahwa *conveyor full*.

5. Selanjutnya, state machine bagian ini berfungsi untuk logika sistem *filling machine* menggunakan metode *Finite State Machine* yang terdiri dari empat state, yaitu *IDLE, WAITING_VALIDATION, PUMPING,* dan *STEPPER_MOVING_AWAY.*
6. Jadi apabila seluruh sensor proteksi dalam kondisi yang sesuai seperti sensor *infrared* 3 dalam kondisi *low* tidak ada penumpukan botol dan sensor XKC dalam kondisi *high* bahwa cairan pada tangki tidak kosong. Maka sistem akan menjalankan *state machine.*
7. Pada bagian awal state machine adalah *state IDLE* merupakan kondisi mesin yang *standby* dimana sistem akan mengaktifkan konveyor secara terus menerus sampai sensor *infrared* 2 dalam kondisi *high* mendeteksi keberadaan botol, maka pada kondisi *infrared* 2 high sistem akan memberhentikan motor *stepper* untuk memberhentikan konveyor dan sistem berpindah ke *state WAITING_VALIDATION.*
8. Pada bagian *state WAITING_VALIDATION* ini melakukan validasi pada sensor *infrared* 1 yang bertujuan untuk memastikan keberadaan botol pada *nozzel* pertama, jika waktu validasi terpernuhi maka akan memberikan kondisi pada pump sebelum berpindah ke dalam state *PUMPING* yaitu kondisi *pump1Active = true;* dan *pump2Active = true;*. Jika sensor *infrared* 1 dalam kondisi *low* maka state *WAITING_VALIDATION* tidak terpenuhi, maka pada *state* ini

akan memberikan kondisi pada pompa hanya *pump2Active = true;* saja dan kemudian berpindah ke *state PUMPING*.

9. Pada *state PUMPING* ini berfungsi untuk mengaktifkan kondisi *pump* yang telah ditentukan pada *state* sebelumnya dan *water pump* akan aktif sesuai parameter yang telah ditentukan pada *pumpDuration*. Jika *pumpDuration* sudah tercapai pomps akan dimatikan secara otomatis. Setelah seluruh proses pengisian selesai, sistem menyiapkan pergerakan konveyor dan berpindah ke *state STEPPER_MOVING_AWAY.*
10. Pada *state STEPPER_MOVING_AWAY* ini berfungsi untuk memindahkan botol dengan *step* 4.703 serupa dengan sejauh 8 cm. Pada proses pergerakan *step* ini, *infrared* 2 diabaikan oleh sistem supaya pada botol yang di *nozzel* pertama tidak terbaca lagi oleh *infrared* 2 menghindari dua kali pengisian pada botol di *nozzel* pertama. Proses *STEPPER_MOVING_AWAY* langkah yang dijalankan disimpan pada variabel *stepsToMove.* Setelah seluruh langkah selesai dijalankan, sistem akan kembali ke *state IDLE*.

3.2.3 Realisasi Data ESP32 Dev Module dengan Intergrasi IoT

3.2.3.1 Import Library

Tabel 3. 6 Import Library ESP32

```
#include <WiFi.h>
#include <WebServer.h>
```

Pada bagian awal pemograman ini ESP32 yang terintegrasi IoT sebagai *monitoring* sistem, melakukan *import library* yang berfungsi untuk menyediakan berbagai fungsi pendukung yang dibutuhkan selama program berjalan. Berikut penjelasan fungsi *library* pada tabel 3.6 :

1. *Library* WiFi.h digunakan untuk menghubungkan ESP32 ke jaringan WiFi sehingga perangkat dapat berkomunikasi melalui jaringan TCP/IP. Melalui *library* ini, ESP32 dapat melakukan proses koneksi ke *access point*, memperoleh alamat IP, serta

bertukar data melalui jaringan internet maupun jaringan lokal *(local area network).*

2. *library WebServer.h* digunakan untuk membuat *web server* pada ESP32. *Library* ini memungkinkan ESP32 berfungsi sebagai *server* yang dapat menerima permintaan dari *browser* atau *dashboard monitoring.*

3.2.3.2 Inisialisasi UART

Tabel 3. 7 Pemograman Inisialisasi Komunikasi

```
HardwareSerial SerialExt(2);
#define RXD2 16
#define TXD2 17
```

Pada inisialisasi UART *(Universal Asynchronous Receiver Transmitter)* yang sebagai media penerima data dari ESP32-S3 yang diterima oleh ESP32. Pada implementasinya, ESP32 menerima data telemetri yang dikirimkan oleh ESP32-S3 berupa status pump, volume pengisian, status konveyor, kondisi sensor level cairan, serta informasi kondisi sistem. berikut penjelasan inisialisasi UART pada tabel 3.7 :

1. Pada bagian ini *HardwareSerial SerialExt(2);* digunakan digunakan untuk membuat objek komunikasi serial bernama *SerialExt* yang menggunakan UART2 pada ESP32.
2. Selanjutnya dilakukan penentuan pin komunikasi UART *#define RXD2* 16 dan *#define TXD2* 17 digunakan sebagai jalur penerimaan data yang terhubung ke pin TX pada ESP32-S3. Sedangkan pin TXD2 digunakan sebagai jalur pengiriman data yang dapat digunakan apabila diperlukan komunikasi dua arah antar mikrokontroler.

3.2.3.3 Konfigurasi Wifi

Tabel 3. 8 Pemograman Konfigurasi Wifi

```
const char* ssid = "aaa";
const char* password = "arfansyah";
WebServer server(80);
```

Pada sistem pengrimin data *monitoring* ini digunakan koneksi WIFI sebagai media komunikasi data telemetri antara ESP32 dengan *dashboard monitoring*. Konfigurasi jaringan wifi tersebut dilakukan mendefinisikan nama jaringan *ssid* yang digunakan sebagai untuk menyimpan nama jaringan wifi yang akan terhubung oleh ESP32, sedangakan kata sandi *password* yang digunakan ebagai autentikasi agar ESP32 dapat terhubung ke jaringan. Setelah Setelah proses koneksi berhasil, ESP32 akan memperoleh alamat IP yang nantinya digunakan untuk mengakses *dashboard monitoring*. Selain konfigurasi wifi, esp32 melakukan pembuatan *web server* menggunakan perintah *WebServer Server (80) ;* digunakan untuk membuat layanan web server yang berjalan pada port 80. Port 80 merupakan port standar yang digunakan oleh protokol HTTP sehingga pengguna dapat mengakses data monitoring melalui browser tanpa perlu menambahkan nomor port secara khusus pada alamat IP ESP32. Web server ini berfungsi untuk menerima permintaan dari *dashboard monitoring* dan mengirimkan data telemetri dalam format JSON.

3.2.3.4 Fungsi Loop

Tabel 3. 9 Pemrograman Loop ESP32

```
void loop() {

  // Menangani request client secara terus menerus
  // Contoh:
  // Browser
  server.handleClient();
  // ============================
  // MEMBACA DATA DARI UART2
  // ==============================
  // Selama masih ada data masuk dari UART2
  while (SerialExt.available()) {
```

```
    // Membaca 1 karakter
    char c = SerialExt.read();
    // Jika ditemukan newline '\n'
    // artinya 1 paket data sudah selesai
    if (c == '\n') {
  // Menghapus spasi / enter tambahan
  incomingData.trim();
   // Parsing data yang sudah lengkap
   parseData(incomingData);
  // Reset buffer agar siap menerima data baru
        = "";
    }
    // Jika belum newline
    // simpan karakter ke buffer
    else {
      incomingData += c;
    }
  }
}
```

Fungsi *Loop* merupakan program utama yang berjalan secara terus-menerus pada ESP32 *monitoring*. Fungsi ini bertanggung jawab untuk menangani komunikasi antara *dashboard* web dengan ESP32 serta menerima data telemetri yang dikirimkan oleh ESP32-S3 melalui komunikasi UART. Berikut penjelasan fungsi programn tersebut :

1. Pada awal fungsi dilakukan *serverhandleClient()*; Perintah ini digunakan untuk melayani setiap permintaan yang berasal dari browser atau *dashboard monitoring*. Ketika pengguna mengakses halaman *monitoring* melalui alamat IP ESP32, web server akan memproses permintaan tersebut dan mengirimkan data yang diperlukan kepada *client*.
2. Selanjutnya, *while (SerialExt.available())* perintah ini digunakan untuk pemeriksaan data yang masuk melalui UART2 yang dikirim oleh ESP32-S3.
3. Selanjutnya, data tersebut akan dibaca terus menerus secara satu karakter demi satu karakter menggunakan fungsi *SerialExt.read()*.

4. Karakter data yang diterima akan disimpan sementara ke dalam *variabel incomingData*, Proses ini dilakukan hingga sistem menemukan karakter *newline \n* yang menandakan bahwa satu paket data telah diterima secara lengkap.
5. Selanjutnya , ESP32 akan menghilangkan karakter spasi atau karakter enter yang dapat mempengaruhi proses pengolahan data menggunakan fungsi *trim()*.
6. Data telemetri yang telah lengkap selanjutnya dikirim ke fungsi *parseData()* untuk dilakukan proses *parsing* dan pengolahan data.
7. Setelah proses *parsing* selesai dilakukan, *buffer incomingData* akan dikosongkan kembali sehingga siap digunakan untuk menerima paket data berikutnya.

3.2.3.5 Handel data dan Pengiriman Data

Tabel 3. 10 Pemrograman Handel Data dan Pengirimana Data

```
void handleData() {
  // ============================
  // JIKA TIDAK ADA PERUBAHAN DATA
  // ============================
  // Client request /data
  // tetapi data sama seperti sebelumnya
  if (!dataChanged) {
    // Kirim JSON status no_change
    // agar client tahu tidak perlu update
    server.send(
      200,
      "application/json",
      "{\"status\":\"no_change\"}"
    );
    return;
  }
  // ==============================
  // MEMBUAT JSON RESPONSE
  // ==============================
  String json = "{";
  json += "\"p1\":" + p1 + ",";
  json += "\"p2\":" + p2 + ",";
  json += "\"v1\":" + v1 + ",";
  json += "\"v2\":" + v2 + ",";
  json += "\"st\":" + st + ",";
  json += "\"xyc\":\"" + xyc + "\",";
```

```
  json += "\"full\":\"" + full + "\"";
  json += "}";
  // Contoh hasil:
  // {
  //   "p1":1,
  //   "p2":0,
  //   "v1":100,
  //   "v2":200,
  //   "st":1
  // }
  // ================================
  // KIRIM JSON KE CLIENT
  // ================================
  server.send(
    200,
    "application/json",
    json
  );
  // ================================
  // DEBUG SERIAL MONITOR
  // ================================
  Serial.println("KIRIM DATA KE CLIENT:");
  Serial.println(json);
}
```

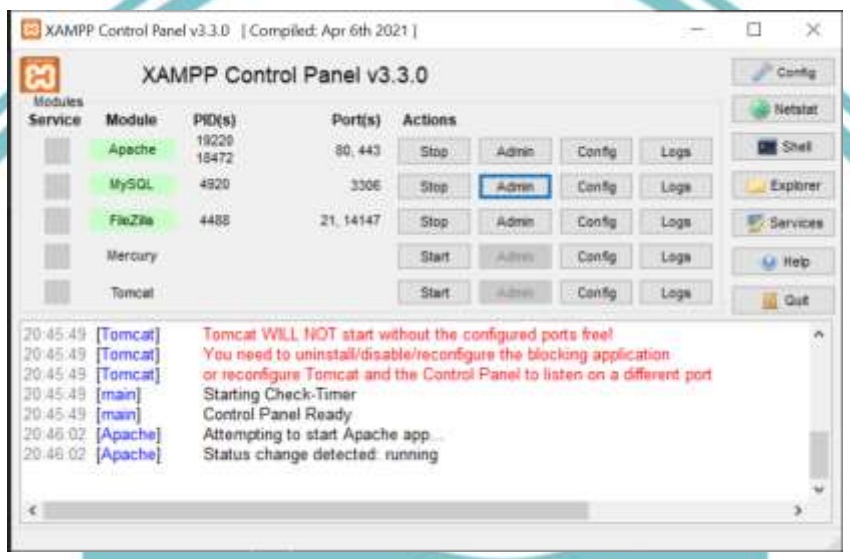
Fungsi handel data dan pengiriman menggunakan fungsi *handleData()* yang berfungsi untuk menangani permintaan data dari *dashboard monitoring*. Fungsi ini akan mengirimkan data telemetri yang telah diterima dari ESP32-S3 dalam format JSON sehingga dapat dibaca dan ditampilkan oleh halaman *dashboard*.

Pada awal fungsi dilakukan pemeriksaan terhadap variabel *dataChanged*. Variabel ini digunakan untuk mengetahui apakah terdapat perubahan data telemetri yang diterima dari ESP32-S3 dibandingkan dengan data sebelumnya. Jika tidak terdapat perubahan data, ESP32 tidak akan mengirimkan data telemetri. Sebagai gantinya, ESP32 hanya mengirimkan respon JSON sederhana dengan status *no_change*. Apabila terdapat perubahan data, maka sistem akan membuat respon JSON yang berisi seluruh parameter monitoring. Proses JSON dilakukan dengan menggabungkan nilai setiap parameter ke dalam sebuah variabel

string. Selanjutnya data JSON dikirimkan ke *client* menggunakan fungsi *server.send()* dengan tipe data *application/json.*

3.2.4 Konfigurasi Server XAMPP

Konfigurasi server XAMPP pada tahap ini implementasi dashboard monitoring, digunakan aplikasi sebagai server lokal XAMPP untuk menjalankan layanan Apache dan MySQL. XAMPP berfungsi sebagai web server dan database server dalam proses sistem monitoring.



Gambar 3.7 Kontrol Panel XAMPP

Berdasarkan Gambar 3.7 layanan *Apache* dan *MySQL* berada dalam kondisi aktif, dengan indikator warna hijau pada kontrol Panel XAMPP. Apache digunakan sebagai web server yang bertugas menjalankan file PHP serta melayani permintaan dari browser. Sedangkan MySQL digunakan sebagai sistem manajemen basis data yang berfungsi menyimpan data *monitoring* yang dikirim dari ESP32.

Pada implementasi sistem ini, Apache berjalan pada port 80 sebagai layanan HTTP, sedangkan MySQL menggunakan port 3306 untuk melayani proses penyimpanan dan pengambilan data dari *database*.

3.2.5 Realisasi Pembuatan Database Mysql

Tabel 3. 11 Pemrograman Pembuatan Database MySql

```
-- Membuat database
CREATE DATABASE monitoring_db;

-- Menggunakan database
USE monitoring_db;

-- Membuat tabel data_produksi
CREATE TABLE monitoring  (
id INT AUTO_INCREMENT PRIMARY KEY,
pump_id INT NOT NULL,
volume_ml INT NOT NULL,
qty INT NOT NULL,
total INT NOT NULL,
tanggal TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Pada sistem *monitoring filling machine* menggunakan *database* MySQL digunakan sebagai media penyimpanan data hasil proses produksi yang diterima dari perangkat ESP32. Menggunaan *database* bertujuan agar data produksi dapat disimpan secara terstruktur, mudah dikelola, serta dapat ditampilkan kembali pada *dashboard monitoring* untuk memantau jumlah produksi. Proses pembuatan database diawali dengan membuat sebuah database bernama *monitoring_db* menggunakan perintah *CREATE DATABASE monitoring_db;*. Setelah database berhasil dibuat, sistem akan mengaktifkan database tersebut menggunakan perintah *USE monitoring_db;*. Berikutnya membuat tabel monitoring yang digunakan untuk menyimpan seluruh data produksi yang diterima dari sistem *filling machine*. Struktur tabel yang digunakan ditunjukkan pada gambar 3.8.

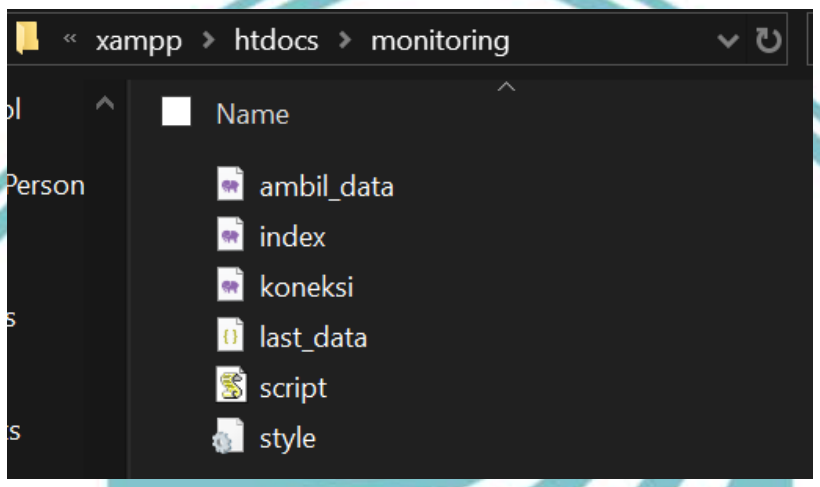
	#	Name	Type
☐	1	id	int(11)
☐	2	pump_id	int(11)
☐	3	volume_ml	int(11)
☐	4	qty	int(11)
☐	5	total	int(11)
☐	6	tanggal	timestamp

Gambar 3. 8 Struktur Tabel

Field id digunakan sebagai identitas unik setiap data yang tersimpan pada *database*. Nilai *field* ini akan bertambah secara otomatis menggunakan fitur *AUTO_INCREMENT* sehingga tidak perlu diinput secara manual. Field *pump_id* digunakan untuk membedakan data yang berasal dari pump 1 maupun pump 2. Informasi ini diperlukan untuk mengetahui *aktivitas* masing-masing pump selama proses produksi berlangsung. *Field volume_ml* digunakan untuk menyimpan nilai volume pengisian yang telah diatur pada sistem. Data ini digunakan untuk memantau parameter pengisian yang diterapkan pada setiap proses *filling*. *Field qty* berfungsi untuk menyimpan jumlah botol yang berhasil diproduksi pada setiap proses pengisian, sedangkan *field* total digunakan untuk menyimpan total akumulasi hasil produksi yang telah dicapai oleh sistem. Selain itu, *field* tanggal bertipe *TIMESTAMP* digunakan untuk mencatat waktu penyimpanan data secara otomatis berdasarkan waktu server. Informasi ini memudahkan proses pelacakan histori produksi berdasarkan tanggal dan waktu kejadian.

3.2.6 Realisasi Penempatan File Program

Setelah konfigurasi server XAMPP selesai dilakukan, seluruh file program *dashboard monitoring* ditempatkan pada folder *monitoring* yang berada di dalam *direktori* htdocs. *Direktori* htdocs merupakan direktori utama web server Apache yang digunakan untuk menyimpan file website agar dapat diakses melalui browser.



Gambar 3. 9 Struktur File *Monitoring*

Berdasarkan Gambar 3.9, folder *monitoring* berisi beberapa file yang digunakan untuk membangun sistem *monitoring filling machine* berbasis web. Masing-masing file memiliki fungsi yang berbeda sesuai dengan kebutuhan sistem. File index.php berfungsi sebagai halaman utama dashboard monitoring yang menampilkan informasi kondisi sistem *filling machine* secara *realtime* kepada operator. File *koneksi.php* digunakan untuk koneksi antara program PHP dengan database MySQL sehingga proses penyimpanan maupun pengambilan data dapat dilakukan. File *ambil_data.php* berfungsi untuk mengambil data dari esp32, mengambil data yang tersimpan pada database dan mengirimkannya ke *dashboard* dalam format yang dapat diproses oleh JavaScript. File *script.js* berisi kode JavaScript yang bertugas melakukan pengambilan data secara *realtime* dari server serta memperbarui tampilan *dashboard* tanpa perlu melakukan refresh halaman. Sedangkan file *style.css* digunakan untuk mengatur tampilan antarmuka *dashboard*, seperti tata letak, ukuran komponen, warna, dan

elemen visual lainnya sehingga informasi *monitoring* dapat ditampilkan dengan lebih menarik dan mudah dipahami oleh operator.

3.2.7 Realisasi Koneksi Database

Pada sistem *monitoring filling machine*, koneksi database digunakan sebagai media penghubung antara program PHP dengan *database* MySQL. Koneksi ini diperlukan agar program dapat melakukan proses penyimpanan, pembacaan, maupun pengelolaan data produksi yang tersimpan pada *database monitoring_db*. Realisasi koneksi database dilakukan menggunakan fungsi *mysqli_connect()* yang disediakan oleh PHP sebagai tabel berikut:

Tabel 3. 12 Pemrograman Koneksi Database MySql

```
<?php
$conn = mysqli_connect("localhost", "root", "",
"monitoring_db");

if (!$conn) {
   die("Koneksi gagal: " . mysqli_connect_error());
}
?>
```

Pada program tersebut, parameter localhost menunjukkan bahwa database MySQL berada pada komputer yang sama dengan web server Apache. Parameter root merupakan nama pengguna (*username*) yang digunakan untuk mengakses database, sedangkan parameter kosong ("") menunjukkan bahwa akun database tidak menggunakan kata sandi. Parameter terakhir yaitu monitoring_db merupakan nama database yang akan diakses oleh sistem monitoring. Hasil koneksi tersebut akan disimpan kedalam variabel $conn, variable ini digunakan oleh file PHP lainnya untuk menjalankan berbagai operasi database, seperti penyimpanan data produksi, pengambilan data monitoring, maupun menampilkan data pada dashboard web. Selanjutnya, program juga melakukan pemeriksaan terhadap status koneksi menggunakan perintah if (!$conn). Apabila koneksi gagal dilakukan, sistem akan menampilkan pesan kesalahan beserta informasi penyebab kegagalan koneksi yang diperoleh dari fungsi mysqli_connect_error().

3.2.8 Realisasi Pengolahan Data dan Integrasi Database

Pada sistem *monitoring filling machine*, proses pengolahan data dan *integrasi database* bagian ini berfungsi sebagai penghubung antara perangkat ESP32, *database* MySQL, dan *dashboard* web yang digunakan oleh operator. Pengolahan data ini dan intergrasi database bertugas mengambil data yang dikirim oleh ESP32 melalui jaringan WiFi, kemudian melakukan proses pengolahan data sebelum disimpan ke dalam *database* MySQL. Selain itu, program juga untuk mengambil data produksi yang telah tersimpan pada *database* dan mengirimkannya kembali ke javascript dalam format JSON

3.2.8.1 Koneksi File

Tabel 3. 13 Koneksi File dengan Koneksi.php

```
<?php
// Menghubungkan program dengan file koneksi database
include 'koneksi.php';
// Nama file yang digunakan untuk menyimpan data terakhir
$last_file = 'last_data.json';
// Inisialisasi nilai awal data pompa
// p1 = status pompa 1
// p2 = status pompa 2
$lastData = [
    "p1" => 0,
    "p2" => 0
];
// Memeriksa apakah file penyimpanan data terakhir tersedia
if (file_exists($last_file)) {

// Membaca isi file JSON dan mengubahnya menjadi array PHP
// Data ini digunakan sebagai referensi status terakhir sistem
    $lastData = json_decode(file_get_contents($last_file), true);
}
```

Pada tahap awal melakukan inisialisasi koneksi database serta mempersiapkan data *monitoring* sebelumnya yang digunakan sebagai acuan untuk mendeteksi perubahan data dari perangkat ESP32. Program diawali dengan pemanggilan file koneksi.php menggunakan perintah *include*. File tersebut berisi konfigurasi koneksi ke database MySQL sehingga seluruh

proses pengolahan dan penyimpanan data dapat dilakukan menggunakan koneksi yang telah dibuat.

Selanjutnya sistem mendefinisikan variabel *$last_file* yang menunjuk ke file *last_data.json*. File ini digunakan untuk menyimpan data *monitoring* terakhir yang telah diterima oleh server. Data tersebut berfungsi sebagai pembanding terhadap data baru yang akan diterima dari ESP32. Variabel *$lastData* kemudian diinisialisasi dengan nilai awal berupa status pump 1 dan pump 2 dalam kondisi tidak aktif. Untuk memastikan keberadaan file penyimpanan data sebelumnya, sistem melakukan pemeriksaan menggunakan fungsi *file_exists()*. Jika file ditemukan, program akan membaca seluruh isi file menggunakan fungsi *file_get_contents()*. Data yang masih berbentuk format JSON kemudian dikonversi menjadi array PHP menggunakan fungsi *json_decode()*.

3.2.8.2 Komunikasi Data ESP32

Tabel 3. 14 Pemrograman Komunikasi Data ESP23

```
// Alamat IP ESP32 yang digunakan sebagai sumber data monitoring
$url = "http://10.136.54.111/data";
// Membuat konfigurasi koneksi HTTP
// Timeout 1,5 detik agar program tidak terlalu lama menunggu respons
$ctx = stream_context_create([
    'http' => [
        'timeout' => 1.5
    ]
]);
// Mengambil data dari ESP32 dalam format JSON
// Simbol @ untuk menyembunyikan pesan error jika koneksi gagal
$json = @file_get_contents($url, false, $ctx);
```

Pada tahap ini sistem melakukan proses pengambilan data monitoring dari perangkat ESP32 melalui jaringan WiFi menggunakan protokol HTTP. Data yang diperoleh berisi informasi kondisi sistem *filling machine* yang selanjutnya akan diolah dan disimpan ke dalam database MySQL. Program

diawali dengan mendefinisikan alamat IP perangkat ESP32 pada variabel *$url*. Alamat tersebut mengarah ke *endpoint* atau data yang telah disediakan oleh web server pada ESP32. *Endpoint* ini berfungsi untuk mengirimkan data telemetri dalam format JSON yang berisi informasi kondisi sistem.

Selanjutnya, program membuat konfigurasi koneksi HTTP menggunakan fungsi *stream_context_create()*. Pada konfigurasi tersebut ditetapkan nilai *timeout* sebesar 1,5 detik. Pengaturan ini bertujuan untuk membatasi waktu tunggu ketika server mencoba mengambil data dari ESP32.

Selanjutnya, pengambilan data dilakukan menggunakan fungsi *file_get_contents()*. Fungsi ini akan mengirimkan permintaan HTTP ke alamat ESP32 dan membaca data yang dikirimkan sebagai respon. Hasil respon kemudian disimpan ke dalam variabel *$json* yang masih berbentuk string berformat JSON.

3.2.8.3 Pengolahan Data dan Menyimpan Data Monitoring

Tabel 3. 15Pemrograman Pengolahan Data dan Menyimpan Data

```
// Memastikan data JSON berhasil diterima dari ESP32
if ($json !== FALSE) {

    // Mengubah data JSON menjadi array PHP
    $data = json_decode($json, true);
    // Memastikan data berhasil didekode
    if ($data) {
        // Mengambil nilai counter produksi Pompa 1
        $p1 = $data['p1'] ?? 0;
        // Mengambil nilai counter produksi Pompa 2
        $p2 = $data['p2'] ?? 0;
        // Mengambil volume pengisian Valve 1 (mL)
        $v1 = $data['v1'] ?? 0;
        // Mengambil volume pengisian Valve 2 (mL)
        $v2 = $data['v2'] ?? 0;
        // Mengambil status stepper motor
        $st = $data['st'] ?? 0;
        // Mengambil status sensor level cairan XKC
        $xyc = $data['xyc'] ?? "OK";
        // Mengambil status sensor deteksi botol penuh
        $full = $data['full'] ?? "NO";
        // ==============================
```

```
// Penyimpanan Data Produksi Pompa 1
// ============================
// Memeriksa apakah terjadi penambahan jumlah produksi
if ($p1 > $lastData['p1']) {
    // Menghitung jumlah produk baru yang diproses
    $selisih = $p1 - $lastData['p1'];
    // Menghitung total volume produksi
    $total1 = $v1 * $selisih;
    // Menyiapkan query penyimpanan data ke database
    $stmt = $conn->prepare(
     "INSERT INTO monitoring (pump_id, volume_ml, qty, total)
        VALUES (?, ?, ?, ?)"
    );
    // Menghubungkan variabel dengan parameter query
    $stmt->bind_param("iiii", $pump_id, $vol, $qty, $ttl);
    // Menentukan data yang akan disimpan
    $pump_id = 1;
    $vol = $v1;
    $qty = $selisih;
    $ttl = $total1;
    // Menjalankan query penyimpanan data
    $stmt->execute();
}
// ==================================
// Penyimpanan Data Produksi Pompa 2
// ==================================
// Memeriksa apakah terjadi penambahan jumlah produksi
if ($p2 > $lastData['p2']) {
    // Menghitung jumlah produk baru yang diproses
    $selisih = $p2 - $lastData['p2'];
    // Menghitung total volume produksi
    $total2 = $v2 * $selisih;
    // Menyiapkan query penyimpanan data ke database
    $stmt = $conn->prepare(
    "INSERT INTO monitoring (pump_id, volume_ml, qty, total)
        VALUES (?, ?, ?, ?)"
    );
    // Menghubungkan variabel dengan parameter query
    $stmt->bind_param("iiii", $pump_id, $vol, $qty, $ttl);
    // Menentukan data yang akan disimpan
    $pump_id = 2;
    $vol = $v2;
    $qty = $selisih;
    $ttl = $total2;
    // Menjalankan query penyimpanan data
    $stmt->execute();
}
// Menyimpan nilai counter terbaru sebagai referensi
```

```
        // untuk proses monitoring berikutnya
        file_put_contents(
            $last_file,
            json_encode([
                "p1" => $p1,
                "p2" => $p2
            ])
        );
    }
}
```

Pada bagian ini melakukan proses pengolahan data *monitoring* yang diterima dari ESP32 sebelum disimpan ke dalam *database* MySQL. Data yang diterima masih berbentuk format JSON sehingga perlu dilakukan proses dekode agar setiap parameter dapat diakses oleh program PHP.

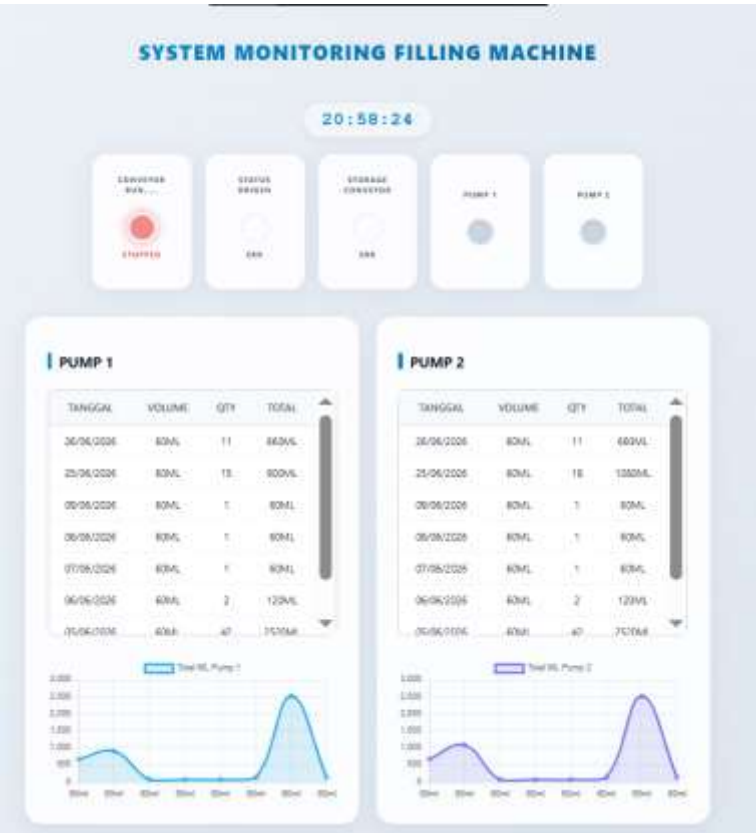
Program di awali memeriksa apakah data berhasil diterima dari ESP32. Jika tidak tersedia, maka fungsi *json_decode()* digunakan, untuk mengubah format JSON menjadi array PHP setiap parameter *monitoring* dapat diproses secara terpisah.

Selanjutnya mengambil nilai status produksi, volume pengisian, status motor stepper, kondisi sensor level cairan, dan status conveyor penuh dari data yang diterima. Jika parameter tidak tersedia, maka menggunakan tanda tanya yang akan memberikan nilai *default*. Jika data berhasil dibaca akan melakukan perbandingan dengan data sebelumnya yang tersimpan pada file *last_data.json.* Perbandingan ini digunakan untuk mendeteksi apakah terjadi penambahan jumlah produksi pada pump 1 maupun pump 2.

Selanjutnya, Apabila terjadi peningkatan jumlah produksi, maka sistem akan menghitung selisih jumlah botol yang bertambah menggunakan perintah *$selisih = $p1 - $lastData['p1'];* untuk menghitung total volume produksi yang dihasilkan oleh pump. Hasil perhitungan selanjutnya disimpan ke dalam tabel *monitoring* menggunakan perintah *INSERT INTO*. Setelah proses penyimpanan selesai, nilai *counter* terbaru

disimpan kembali ke dalam file *last_data.json* sebagai data referensi untuk siklus *monitoring* berikutnya.

3.2.9 Realisasi Pembuatan Dashboard Monitoring Web



Gambar 3. 10 Realisaasi Dashboard

Pada gambar 3.10 ini tampilan *dashboard monitoring* pada sistem *filling machine*. Dashboard ini digunakan untuk memantau kondisi sistem secara *real-time* berdasarkan data yang dikirim oleh ESP32 dan ditampilkan melalui program PHP. Informasi yang ditampilkan meliputi status *conveyor*, status *drigen*, status *storage conveyor*, indikator Pompa 1 dan Pompa 2, serta waktu pembaruan data. Selain itu, dashboard juga menampilkan data hasil pengisian setiap pompa yang terdiri dari tanggal, volume, jumlah, dan total volume. Data tersebut disajikan dalam bentuk tabel dan grafik sehingga memudahkan pengguna dalam memantau hasil proses pengisian dan kondisi sistem secara keseluruhan.

3.2.9.1 Konfigurasi Halaman Dashboard

Tabel 3. 16Pemrograman Konfigrasi Halaman Dashboard

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Monitoring Pump System</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <link rel="stylesheet" href="style.css">
</head>

    …….. Tampilan Dashboard dan Monitoring Machine

<script src="script.js"></script>
</body>
</html>
```

Konfigurasi halaman *dashboard* menggunakan bahasa markup HTML yang berfungsi sebagai struktur utama tampilan web. Pada tahap ini dilakukan konfigurasi dokumen HTML, pemanggilan file CSS untuk pengaturan tampilan antarmuka, serta pemanggilan file JavaScript yang digunakan untuk proses monitoring dan pembaruan data secara *real-time.*

Proses ini diawali dengan deklarasi dokumen HTML dan pengaturan bahasa halaman menggunakan atribut *lang="id"* yang menunjukkan bahwa halaman menggunakan Bahasa Indonesia. Selanjutnya pada bagian <*head*> dilakukan konfigurasi karakter UTF-8 supaya pada halaman dapat menampilkan karakter dengan benar serta pengaturan *viewport* supaya tampilan *dashboard* dapat menyesuaikan pada ukuran layar perangkat.

Selanjutnya, Judul halaman dashboard menggunakan tag <*title*> sehingga nama halaman yang ditampilkan pada tab browser menjadi *"Monitoring Pump System"*. Kemudian memanggil *library* chart.js yang befungsi untuk menampilkan grafik pada *dashboard*. Selain itu, file *stylesheet* eksternal

style.css dipanggil untuk mengatur tampilan antarmuka *dashboard* seperti tata letak, warna, tabel *monitoring*, indikator status, dan grafik.

Pada bagian <*body*> ini akan menampilkan komponen-komponen *dashboard* yang terdiri dari indikator status mesin, tabel *monitoring* produksi, serta grafik *monitoring Pump* 1 dan *Pump* 2. Pada akhir proses terdapat *script.js* digunakan untuk memanggil menjalankan fungsi *monitoring*, pengambilan data dari server, pembaruan indikator status, tabel *monitoring*, serta grafik produksi secara *real-time.*

3.2.10 Realisasi Sistem Dashboard Monitoring

3.2.10.1 Menjumlah Total Produksi

Tabel 3. 17 Pemrograman Menjumlah Total Produksi

```
// Fungsi untuk menghitung total jumlah produk (qty)
function getTotalQty(data){
// Menjumlahkan seluruh nilai total_qty yang terdapat pada array data
   return data.reduce((sum, item) =>
      // Menambahkan nilai total_qty ke hasil penjumlahan sebelumnya
      sum + parseInt(item.total_qty),
      // Nilai awal penjumlahan
      0
   );
}
```

Fungsi *getTotalQty()* digunakan untuk untuk menghitung total jumlah produk yang telah diproduksi berdasarkan data yang diterima dari server. Fungsi ini melakukan penjumlahan seluruh nilai *total_qty* menggunakan metode *reduce().* Pada fungsi ini, variabel sum berperan sebagai penampung hasil penjumlahan, sedangkan *item* merepresentasikan setiap data produksi yang sedang diproses. Selain itu, perintah ini *total_* menggunakan fungsi *parseInt qty* digunakan untuk mengkonversi data menjadi tipe numerik *()* sebelum dijumlahkan.

3.2.10.2 Pengambilan Data dan Kendali Dashboard Monitoring

Tabel 3. 18 Pemrograman Pengambila Data dan Kendali Dashboard

```
// Fungsi utama untuk mengambil data monitoring dari server PHP
function fetchData() {

  // Mengambil data monitoring dalam format JSON
  fetch('ambil_data.php')
  .then(r => r.json())
  .then(res => {
    // ====================================
    // 1. Update Indikator Status Conveyor
    // ====================================
    // Mengambil elemen LED dan teks status conveyor
    let ledConv = document.getElementById("led_conv");
    let txtConv = document.getElementById("txtStatusConv");
    // Jika stepper/conveyor aktif
    if(res.st == 1) {
      ledConv.className = "led on";
      txtConv.innerText = "RUNNING";
      txtConv.style.color = "#10b981";
    } else {
      // Jika conveyor berhenti
      ledConv.className = "led danger";
      txtConv.innerText = "STOPPED";
      txtConv.style.color = "#ef4444";
    }

    // ================================
    // 2. Update Indikator Status Drigen (Sensor XKC)
    // ================================
    // Mengambil elemen LED dan teks status drigen
    let ledXYC = document.getElementById("led_xyc");
    let txtXYC = document.getElementById("txtXYC");
    // Menampilkan status sensor XKC
    txtXYC.innerText = res.xyc;
    // Jika cairan tersedia
    if(res.xyc === "OK") {
      ledXYC.className = "led on";
      txtXYC.style.color = "#10b981";
    // Jika drigen kosong
    } else if(res.xyc === "EMPTY") {
      ledXYC.className = "led danger";
      txtXYC.style.color = "#ef4444";
    // Jika status tidak diketahui
    } else {
      ledXYC.className = "led off";
```

```
    }

    // ====================================
    // 3. Update Indikator Storage Conveyor
    // ====================================
    // Mengambil elemen LED dan teks status storage
    let ledFull = document.getElementById("led_full");
    let txtFull = document.getElementById("txtFull");
    // Menampilkan status storage conveyor
    txtFull.innerText = res.full;
    // Jika storage penuh
    if(res.full === "YES") {
      ledFull.className = "led danger";
      txtFull.style.color = "#ef4444";
    // Jika storage belum penuh
    } else {
      ledFull.className = "led off";
      txtFull.style.color = "#475569";
    }
    // ================================
    // 4. Update Indikator Valve Pump
    // ================================
    // Menghitung total produksi Pump 1
    let currentQty1 = getTotalQty(res.pump1);
    // Menghitung total produksi Pump 2
    let currentQty2 = getTotalQty(res.pump2);
// Menyalakan LED Valve 1 jika terjadi penambahan produksi
    document.getElementById("led_v1").className =
      (currentQty1 > lastQtyPump1) ? "led on" : "led off";
// Menyalakan LED Valve 2 jika terjadi penambahan produksi
    document.getElementById("led_v2").className =
      (currentQty2 > lastQtyPump2) ? "led on" : "led off";
    // Menyimpan jumlah produksi terbaru sebagai referensi
    lastQtyPump1 = currentQty1;
    lastQtyPump2 = currentQty2;
    // ============================
    // 5. Update Tabel Monitoring Produksi
    // ============================
    // Memperbarui tabel data Pump 1
    updateTable(res.pump1, "t1");
    // Memperbarui tabel data Pump 2
    updateTable(res.pump2, "t2");
    // ========================
    // 6. Update Grafik Monitoring Produksi
    // ========================
   // Memperbarui grafik berdasarkan data terbaru
    updateCharts(res);
  })
```

```
    // Menampilkan pesan error jika data gagal dibaca
    .catch(err => console.error("Data tidak terbaca:", err));
}
```

Fungsi pengambilan data menggunakan *fetchData()* yang berfungsi untuk mengambil data monitoring dari server. Data tersebut dari file *ambil_data.php* dalam format JSON, kemudian diolah untuk memperbarui seluruh komponen *dashboard* seperti indikator status sistem, tabel produksi, dan grafik *monitoring*.

Pada awal proses menggunakan *fetch()* untuk mengirim permintaan ke file *ambil_data.php*. Data yang telah diterima dari server kemudian akan dikonversi ke format JSON menggunakan fungsi json(), sehingga dapat di proses oleh javascript dan data tersebut akan disimpan kedalam *variable res*. Berikut logika sistem dashboard monitoring :

1. Pada indikator status koveyor berdasarkan nilai st yang telah diterima dari server, apabila nilai st bernilai "1", maka indikator coveyor akan berwarna hijau dengan status "*RUNNING*". Sebaliknya, apabila nilai st bernilai "0", indikator konveyor akan berubah menjadi merah dan status menampilkan tulisan *"STOPPED"*.
2. Selanjutnya, pada bagian Indikator status tangki jika nilai yang diterima "*OK*" maka indikator led akan berwarna hijau yang menunjukkan ketersediaan cairan masih tercukupi. Sebaliknya, jika status " *EMPTY"*, indikator berubah menjadi merah sebagai tanda bahwa cairan pada drigen telah habis.
3. Selanjutnya, pada bagian Indikator *Storage Conveyor* apabila data yang diterima " *YES* " maka pada *dashboard* led *coveyor* berwana merah. Jika bernilai *"NO"*, indikator berada pada kondisi normal yang menunjukkan masih tersedia pada konveyor tidak ada penumpukan wadah botol.
4. Pembaruan indikator pump pada bagian ini menghitung total produksi Pump 1 dan Pump 2 menggunakan fungsi *getTotalQty()*. Hasil perhitungan kemudian dibandingkan

dengan data produksi sebelumnya yang tersimpan pada variabel *lastQtyPump1* dan *lastQtyPump2*. Jika jumlah produksi bertambah maka indikator pada pump akan aktif berwanra hijau.

5. Selanjutnya pada update tabel produksi pada bagian *updateTable()* yang berfungsi *updateTable()* digunakan untuk menampilkan data produksi *Pump* 1 dan *Pump* 2 yang diperoleh dari *database* MySQL. Informasi yang ditampilkan meliputi tanggal produksi, volume pengisian, jumlah produk, dan total volume produksi.
6. Selanjutnya pada update bagian grafik produksi pada bagian *updateCharts(res);* yang berfungsi untuk memperbarui data grafik berdasarkan informasi produksi terbaru yang diterima dari server.

BAB IV
PEMBAHASAN

4.1 Pengujian Algoritma Kendali

Pada pengujian algoritma kendali dilakukan untuk mengetahui apakah sistem pada mesin mampu menjalankan proses *filling machine* secara otomatis sesuai dengan logika yang telah dirancah dan realisasi pada mikrokontroller ESP32-S3. Tujaun pengujian ini memastikan setiap *state* pada program dapat berjalan sesuai dengan parameter yang telah ditentukan.

4.1.1 Deskripsi Pengujian

LOKASI	:	Rumah Rafi Arfahnsyah
TANGGAL	:	5 June 2026
PELAKSANAAN	:	1. Rafi Arfansyah 2. Muhammad Rafii Nur Rahkim
TUJUAN	:	Memastikan ESP32 menerima data dari ESP32-S3 dan mengirim data dengan format JSON

Berikut merupakan daftar alat dan bahan yang digunakan pada pengujian pada alat ditunjukkan pada Tabel 4.1.

Tabel 4. 1 Daftar alat pengujian algoritma kendali

NO	NAMA ALAT	TIPE	JUMLAH	FUNGSI
1	Mikrokontroller	ESP32-S3	1	Menerima Input dari Sensor dan mengenalikan proses
2	Laptop + Software	Arduino ide	1	Mengupload Program Kendali

NO	NAMA ALAT	TIPE	JUMLAH	FUNGSI
3	Sensor Infrared	E18-D80NK	3	Medeteksi Objek dan Mengirim Input
4	Sensor Water Level	XKC-Y26S-PNP	1	Mendeteksi level cairan pada tangki non-kontak
5	Motor Stepper	Nema 17	1	Menggerkan Conveyor
6	Water Pump	Kamoer KPK 400	2	Mempompa cairan kedalam botol
7	Driver Stepper	TB6600	1	penerjemah sinyal dari mikrokontroler untuk menggerakkan motor
8	Driver Water Pump	BTS7960	2	Mengatur kecepatan motor water pump yang diberikan sinyal pada mikrokontroller
9	LCD	20x4 I2C	1	Indikator Proses
10	Keypad	4x4 I2C	1	Setting Parameter
11	Power Supply	24V 5A	1	Sebagai tegangan dan arus pada motor
12	Kabel Jumper	Male to Male	42	Sebagai penghubung antara komponen
13	Kabel USB	USB to type C	1	koneksi Laptop dengan Mikrokontroller
14	Wadah	-	1	Sebagai tempat botol
15	botol	60ml		sebagai wadah air
16	Selang	ID 7mm	4 meter	sebagai saluran air
17	tangki	10 liter	1	penampungan air

4.1.2 Prosedur Pengujian

Langkah-langkah Pengujian :

1. Menyiapkan alat yang akan digunakan pada tabel 4.1.
2. Hubungkan pin ESP32-S3 dengan pin sensor.
3. Hubungkan pin LCD dengan pin ESP32-S3.
4. Hubungkan pin keypad dengan pin ESP32-S3.
5. Hubungkan pin ESP32-S3 dengan pin sinyal driver.
6. Hubungkan pin driver motor TB6600 dengan pin stepper.
7. Hubungkan pin driver motor BTS7960 dengan pin *water* pump.
8. Hubungkan supply pada driver stepper dan driver *water pump.*
9. Hubungkan ESP32-S3 dengan Laptop dengan kabel USB to type C.
10. Upload program algoritma kendali pada ESP32-S3.
11. Hubungkan selang *input* dengan tangki penampungan air pada water pump.
12. Siapkan air pada tangki yang sebagai objek hasil pengukuran.
13. Siapkan botol dan wadah sebagai tempat penampungan hasil pengisian.
14. Selanjutnya, *Input* "*" pada keypad untuk memulai proses.
15. Lakukan *priming* terlebih dahulu untuk mengeluarkan udara *yang terperangkap pada selang.*
16. Tempatkan wadah yang telah berisi botol pada konveyor yang sedang berjalan.
17. Kemudian catat hasil pengisian *water pump* pada hasil botol.

4.1.3 Data Hasil Pengujian Alogirma Kendali

Tabel 4. 2 Pengujian Water pump 2

NO	WADAH	TIMMING	TARGET VOLUME P2	SINYAL IR 2	STEPPER	Water pump P2	HASIL P2	EROR/ml
1				1	OFF	ON	60	0
2				0	ON	OFF	-	-
3				1	OFF	ON	60	0
4	1	7917mS	60ML	0	ON	OFF	-	-
5				1	OFF	ON	60	0
6				0	ON	OFF	-	-
7				1	OFF	ON	59	-1
8				0	ON	OFF	-	-
9				1	OFF	ON	60	0
10	2	7917mS	60ML	0	ON	OFF	-	-
11				1	OFF	ON	60	0
12				0	ON	OFF	-	-
13				1	OFF	ON	60	0
14				0	ON	OFF		-
15				1	OFF	ON	60	0
16	3	7917mS	60ML	0	ON	OFF		-
17				1	OFF	ON	60	0
18				0	ON	OFF	-	-
19				1	OFF	ON	60	0
20				0	ON	OFF	-	-
21				1	OFF	ON	60	0
22	4	7917mS	60ML	0	ON	OFF	-	-
23				1	OFF	ON	60	0
24				0	ON	OFF	-	-
25				1	OFF	ON	60	0
26				0	ON	OFF	-	-
27				1	OFF	ON	60	0
28	5	7917mS	60ML	0	ON	OFF	-	-
29				1	OFF	ON	60	0
30				0	ON	OFF	-	-
31				1	OFF	ON	60	0
32				0	ON	OFF	-	-
33				1	OFF	ON	60	0
34	6	7917mS	60ML	0	ON	OFF	-	-
35				1	OFF	ON	60	0
36				0	ON	OFF	-	-

Berdasarkan Tabel 4.2 hasil pengujian kendali *Water Pump* 2, pembacaan tabel dilakukan berdasarkan urutan kerja sistem. Ketika sensor IR 2 mendeteksi botol dengan sinyal *input* 1 maka motor *stepper* berhenti (*OFF*) dan *Water Pump* 2 aktif (*ON*) untuk melakukan proses pengisian sesuai target volume 60 mL dengan waktu aktif selama timming yaitu 7917mS. Setelah proses pengisian selesai, sensor IR 2 bernilai 0 sehingga motor stepper kembali aktif (*ON*) untuk menggerakkan konveyor dan *Water Pump* 2 berhenti (*OFF*).

Tabel 4. 3 Pengujian Water pump 1

NO	WADAH	TIMMING	TARGET VOLUME P1	SINYAL IR 1	STEPPER	Water pump P1	HASIL P1	EROR/ml
1				1	OFF	ON	60	0
2				0	ON	OFF	-	-
3	1	8108mS	60ML	1	OFF	ON	59	-1
4				0	ON	OFF	-	-
5				1	OFF	ON	60	0
6				0	ON	OFF	-	-
7				1	OFF	ON	61	1
8				0	ON	OFF	-	-
9	2	8108mS	60ML	1	OFF	ON	60	0
10				0	ON	OFF	-	-
11				1	OFF	ON	60	0
12				0	ON	OFF	-	-
13				1	OFF	ON	60	0
14				0	ON	OFF	-	-
15	3	8108mS	60ML	1	OFF	ON	60	0
16				0	ON	OFF	-	-
17				1	OFF	ON	60	0
18				0	ON	OFF	-	-
19				1	OFF	ON	60	0
20				0	ON	OFF	-	-
21	4	8108mS	60ML	1	OFF	ON	60	0
22				0	ON	OFF	-	-
23				1	OFF	ON	60	0
24				0	ON	OFF	-	-
25	5	8108mS	60ML	1	OFF	ON	60	0

NO	WADAH	TIMMING	TARGET VOLUME P1	SINYAL IR 1	STEPPER	Water pump P1	HASIL P1	EROR/ml
26				0	ON	OFF	-	-
27				1	OFF	ON	60	0
28				0	ON	OFF	-	-
29				1	OFF	ON	61	1
30				0	ON	OFF	-	-
31				1	OFF	ON	60	0
32				0	ON	OFF	-	-
33	6	8108mS	60ML	1	OFF	ON	60	0
34				0	ON	OFF	-	
35				1	OFF	ON	60	0
36				0	ON	OFF	-	-

Berdasarkan Tabel 4.3 hasil pengujian kendali *Water Pump* 1, pembacaan tabel dilakukan berdasarkan urutan kerja sistem. Ketika sensor IR 1 mendeteksi botol dengan sinyal input 1 maka motor *stepper* berhenti (*OFF*) dan *Water Pump* 1 aktif (*ON*) untuk melakukan proses pengisian sesuai target volume 60 mL dengan waktu aktif selama *timming* yaitu 8108mS. Setelah proses pengisian selesai, sensor IR 1 bernilai 0 sehingga motor *stepper* kembali aktif (*ON*) untuk menggerakkan konveyor dan *Water Pump* 1 berhenti (*OFF*).

Tabel 4. 4 Pengujian Indikator LCD Filling Machine

NO	WADAH	STATUS P1	STATUS P2	STATUS MESIN LCD	SINYAL IR 3	SINYAL XKC
1		P1 : ON	P2 : ON	Filling	0	1
2		P1 : OFF	P2OFF	Move Step	0	1
3		P1 : ON	P2 : ON	Filling	0	1
4	1	P1 : OFF	P2OFF	Move Step	0	1
5		P1 : ON	P2 : ON	Filling	0	1
6		P1 : OFF	P2 : OFF	Move Step	0	1
7		P1 : OFF	P2 : OFF	Running	0	1

NO	WADAH	STATUS P1	STATUS P2	STATUS MESIN LCD	SINYAL IR 3	SINYAL XKC
8		P1 : OFF	P2OFF	Conveyor Full	1	1

Berdasarkan Tabel 4.4 hasil pengujian LCD *Filling Machine*, ketika sistem melakukan proses pengisian, LCD akan menampilkan teks **"***Filling***"** sebagai indikator bahwa *Water Pump* sedang aktif melakukan pengisian cairan. Setelah proses *filling* selesai, sistem akan berpindah ke kondisi "*Move Step*" yang ditampilkan pada LCD sebagai tanda bahwa konveyor bergerak untuk memindahkan posisi wadah. Selain itu, ketika wadah botol telah mencapai ujung konveyor, sensor *infrared* 3 akan memberikan sinyal 1 kepada sistem sehingga konveyor berhenti dan LCD menampilkan kondisi "*Conveyor Full*" sebagai indikator bahwa area penyimpanan telah penuh.

Tabel 4. 5 Pengujian Indikator LCD Filling Machine

NO	WADAH	KONDISI MESIN	LED HIJAU	LED KUNING	LER MERAH	SINYAL IR 3	SINYAL XKC
1		Filling	OFF	ON	OFF	0	1
2		Move Step	ON	OFF	OFF	0	1
3		Filling	OFF	ON	OFF	0	1
4		Move Step	ON	OFF	OFF	0	1
5	1	Filling	OFF	ON	OFF	0	1
6		Move Step	ON	OFF	OFF	0	1
7		Running	ON	OFF	OFF	0	1
8		Conveyor Full	OFF	OFF	ON	1	1

Berdasarkan Tabel 4.5 hasil pengujian indikator kondisi mesin, pembacaan tabel dilakukan berdasarkan perubahan kondisi kerja sistem. Pada saat kondisi mesin berada pada proses *Filling*, LED kuning akan aktif sebagai indikator bahwa proses pengisian,

sedangkan berpindah ke kondisi *Move Step*, LED hijau akan aktif sebagai indikator bahwa motor *stepper* bergerak untuk memindahkan posisi wadah. Pada kondisi *Running*, LED hijau tetap aktif yang menunjukkan sistem dalam keadaan berjalan normal. Selain itu, ketika sensor IR 3 mendeteksi kondisi penuh dengan nilai 1 dan sensor XKC bernilai 1, sistem masuk pada kondisi *Conveyor Full* yang ditandai dengan LED merah aktif, indikator LED mampu menunjukkan kondisi mesin sesuai dengan proses kerja aktual sistem.

4.1.4 Analisis dan Evaluasi

Berdasarkan hasil pengujian, algoritma kendali yang menggunakan metode *state machine* mampu mengatur setiap proses kerja sistem secara teratur. Secara keseluruhan, algoritma kendali yang dirancang untuk mengendalikan proses *filling* secara otomatis, urutan kerja sistem, serta mengatur target volume air.

$$x = \frac{TOTAL}{DATA} = \frac{1081}{18} = 60{,}06\text{mL}$$

Berdasarkan tabel 4.3 hasil pengujian *water pump* 1, volume air yang dihasilkan oleh *water Pump* 1 dengan target 60 ml sebanyak 18 kali percobaan mengasilkan 1081mL. memiliki rata-rata 60,06 ml.

$$\%Eror = \frac{60{,}06 - 60}{60} \times 100\% = 0{,}11\%$$

Hasil tersebut menunjukkan bahwa Water Pump 1 memiliki tingkat kesalahan sebesar 0,11% terhadap target volume 60 mL.

$$x = \frac{TOTAL}{DATA} = \frac{1079}{18} = 59{,}94mL$$

Sedangkan, hasil pengujian pada tabel 4.2, volume air yang dihasilkan oleh *water Pump* 2 dengan target 60 ml sebanyak 18 kali percobaan mengasilkan 1079 mL. memiliki rata-rata 59,94 mL.

$$\%Eror = \frac{59{,}94 - 60}{60} \times 100\% = 0{,}11\%$$

Hasil tersebut menunjukkan bahwa Water Pump 2 memiliki tingkat kesalahan sebesar 0,11% terhadap target volume 60 mL.

Dengan demikian water pump 1 dan water pump 2 mampu menghasilkan volume pengisian dalam batas toleransi. Hasil pengujian tersebut menunjukkan bahwa algoritma kendali yang diterapkan mampu mengatur durasi pengisian sesuai parameter volume yang ditentukan dan nilai *eror* yang diperoleh relatif kecil sehingga sistem pengisian botol dapat otomatis dengan tingkat akurasi yang baik.

Selain itu, berdaskan tabel pengujian 4.4 dengan 4.5 dalam algoritma kendali pada LCD berhasil memberikan informasi kondisi proses *filling machine*. Serta, pada idikator LED berhasil aktif dalam kondisi *state* sistem *filling machine* sesuai dengan logika yang telah dirancang.

4.2 Pengujian Penerimaan Data dan Pengirim Data

Pengujian penerimaan dan pengiriman data dilakukan untuk mengetahui kemampuan ESP32 dalam menerima data telemetri dari ESP32-S3 melalui komunikasi UART serta mengirimkan kembali data tersebut kepada *client* melalui jaringan WiFi menggunakan format JSON. Pengujian ini bertujuan untuk memastikan bahwa seluruh data *monitoring* dapat diterima, diproses, dan dikirim tanpa mengalami perubahan nilai sehingga informasi yang ditampilkan pada *dashboard* sesuai dengan kondisi sistem.

4.2.1 Deskripsi Pengujian

LOKASI	:	Bengkel Elektronika Industri
TANGGAL	:	5 June 2026
PELAKSANAAN	:	1. Rafi Arfansyah 2. Muhammad Rafii Nur Rahkim
TUJUAN	:	Memastikan ESP32 menerima data dari ESP32-S3 dan mengirim data dengan format JSON

Berikut merupakan daftar alat dan bahan yang digunakan pada pengujian pada alat ditunjukkan pada Tabel 4.6.

Tabel 4. 6 Daftar alat pengujian proses data

NO	NAMA ALAT	JUMLAH	FUNGSI
1	ESP32-S3	1	Pengirim data sistem
2	ESP32	1	Penerma data sistem dan pengirim data ke client
3	WIFI	1	komunikasi data protocol TCP/IP
4	LAPTOP	1	Digunakan pengujian sistem.
5	software Arduino ide	1	Digunakan untuk menulis program ESP32
6	KABEL USB to Type C	1	Digunakan untuk upload program ESP32 dan pengujian data.
7	KABEL JUMMPER	3	Digunakan sebagai penghubung komunikasi UART.

4.2.2 Prosedur Pengujian

Langkah-langkah pengujian :

1. Aktifkan sistem *filling machine* hingga menghasilkan data produksi.
2. Hubungkan pin TX dan RX antara ESP32 dengan ESP32-S3
3. Upload Program ESP32 yang sudah dirancang dan reallisasikan.
4. Aktifkan sistam kendali pada ESP32-S3 sehingga mengirim data.

5. Amati data yang diterima oleh ESP32 melalui Serial Monitor.
6. Mengakses alamat IP ESP32 Monitoring menggunakan browser.
7. Mengamati data JSON yang dikirimkan ESP32 *Monitoring* kepada *client*.
8. Membandingkan data yang diterima dari ESP32-S3 dengan data yang dikirimkan ke client.
9. Mencatat hasil pengujian untuk setiap parameter yang diuji.

4.2.3 Data Hasil Pengujian Menerima Data dan Mengirim Data

Tabel 4. 7 Hasil Pengujian Proses Data

NO	Wadah	WAKTU	Status data	Delay	P1	P2	ST	V1	V2	XKC	FULL
1	1	14:12:19	Terima	1	ON	ON	OFF	60	60	OK	NO
2		14:12:20	Terkirim		1	1	0	60	60	OK	NO
3		14:12:27	Terima	1	OFF	OFF	ON	60	60	OK	NO
4		14:12:28	Terkirm		0	0	1	60	60	OK	NO
5		14:12:29	Terima	1	ON	ON	OFF	60	60	OK	NO
6		14:12:30	Terkirim		1	1	0	60	60	OK	NO
7		14:12:37	Terima	1	OFF	OFF	ON	60	60	OK	NO
8		14:12:38	Terkirm		0	0	1	60	60	OK	NO
9		14:12:39	Terima	1	ON	ON	OFF	60	60	OK	NO
10		14:12:40	Terkirim		1	1	0	60	60	OK	NO
11		14:12:47	Terima	1	OFF	OFF	ON	60	60	OK	NO
12		14:12:48	Terkirm		0	0	1	60	60	OK	NO
13		14:12:51	Terima	1	OFF	OFF	OFF	60	60	OK	YES
14		14:12:52	Terkirim		0	0	0	60	60	OK	YES
15	2	15:13:04	Terima	2	ON	ON	OFF	60	60	OK	NO
16		15:13:06	Terkirim		1	1	0	60	60	OK	NO
17		15:13:13	Terima	1	OFF	OFF	ON	60	60	OK	NO
18		15:13:14	Terkirm		0	0	1	60	60	OK	NO
19		15:13:15	Terima	1	ON	ON	OFF	60	60	OK	NO
20		15:13:16	Terkirim		1	1	0	60	60	OK	NO
21		15:13:23	Terima	1	OFF	OFF	ON	60	60	OK	NO
22		15:13:24	Terkirm		0	0	1	60	60	OK	NO
23		15:13:25	Terima	1	ON	ON	OFF	60	60	OK	NO
24		15:13:26	Terkirim		1	1	0	60	60	OK	NO
25		15:13:33	Terima	1	OFF	OFF	ON	60	60	OK	NO
26		15:13:34	Terkirm		0	0	1	60	60	OK	NO
27		15:13:37	Terima	1	OFF	OFF	OFF	60	60	OK	YES
28		15:13:38	Terkirim		0	0	0	60	60	OK	YES

NO	Wadah	WAKTU	Status data	Delay	P1	P2	ST	V1	V2	XKC	FULL
29		15:14:29	Terima	1	ON	ON	OFF	60	60	OK	NO
30		15:14:30	Terkirim		1	1	0	60	60	OK	NO
31		15:14:37	Terima	1	OFF	OFF	ON	60	60	OK	NO
32		15:14:38	Terkirm		0	0	1	60	60	OK	NO
33		15:14:39	Terima	1	ON	ON	OFF	60	60	OK	NO
34		15:14:40	Terkirim		1	1	0	60	60	OK	NO
35	3	15:14:47	Terima	1	OFF	OFF	ON	60	60	OK	NO
36		15:14:48	Terkirm		0	0	1	60	60	OK	NO
37		15:14:50	Terima	2	ON	ON	OFF	60	60	OK	NO
38		15:14:52	Terkirim		1	1	0	60	60	OK	NO
39		15:14:58	Terima	2	OFF	OFF	ON	60	60	OK	NO
40		15:15:00	Terkirm		0	0	1	60	60	OK	NO
41		15:15:02	Terima	1	OFF	OFF	OFF	60	60	OK	YES
42		15:15:03	Terkirim		0	0	0	60	60	OK	YES
43		15:57:19	Terima	1	ON	ON	OFF	60	60	OK	NO
44		15:57:20	Terkirim		1	1	0	60	60	OK	NO
45		15:57:27	Terima	1	OFF	OFF	ON	60	60	OK	NO
46		15:57:28	Terkirm		0	0	1	60	60	OK	NO
47		15:57:30	Terima	1	ON	ON	OFF	60	60	OK	NO
48		15:57:31	Terkirim		1	1	0	60	60	OK	NO
49	4	15:57:37	Terima	1	OFF	OFF	ON	60	60	OK	NO
50		15:57:39	Terkirm		0	0	1	60	60	OK	NO
51		15:57:40	Terima	1	ON	ON	OFF	60	60	OK	NO
52		15:57:41	Terkirim		1	1	0	60	60	OK	NO
53		15:57:47	Terima	1	OFF	OFF	ON	60	60	OK	NO
54		15:57:48	Terkirm		0	0	1	60	60	OK	NO
55		15:57:51	Terima	3	OFF	OFF	OFF	60	60	OK	YES
56		15:57:54	Terkirim		0	0	0	60	60	OK	YES
57		15:58:37	Terima	1	ON	ON	OFF	60	60	OK	NO
58		15:58:38	Terkirim		1	1	0	60	60	OK	NO
59		15:58:44	Terima	1	OFF	OFF	ON	60	60	OK	NO
60		15:58:45	Terkirm		0	0	1	60	60	OK	NO
61		15:58:46	Terima	2	ON	ON	OFF	60	60	OK	NO
62		15:58:48	Terkirim		1	1	0	60	60	OK	NO
63	5	15:58:54	Terima	2	OFF	OFF	ON	60	60	OK	NO
64		15:58:56	Terkirm		0	0	1	60	60	OK	NO
65		15:58:57	Terima	1	ON	ON	OFF	60	60	OK	NO
66		15:58:58	Terkirim		1	1	0	60	60	OK	NO
67		15:59:05	Terima	1	OFF	OFF	ON	60	60	OK	NO
68		15:59:06	Terkirm		0	0	1	60	60	OK	NO
69		15:59:09	Terima	1	OFF	OFF	OFF	60	60	OK	YES

Berdasarkan hasil pengujian pada tabel 4.7 hasil berhasil menerima data kondisi sistem yang telah dikirim oleh ESP32-S3 melalui komunikasi UART dan melakukan pengiriman data tersebut kepada *client* dengan format JSON melalui komunikasi wifi. Hasil pengujian menenujukan bahwa data yang diterima tidak ditemukan perubahan nilai data selama proses penerimaan maupun pengiriman berlangsung. Hal ini menenjukan bahwa proses pengolahan data terlah berjalan dengan baik. Selain itu, melakukan pengamatan terhadap waktu penerimaan dan pengiriman data berdasarkan hasil pengujian, diperoleh waktu tunda pengiriman antara 1 samapai 3 detik dengan rata-rata 1,2 detik. Dengan total waktu tunda 42 detik dalam jumlah pengujian waktu terima dengan waktu terkirim sebanyak 35 data.

$$t = \frac{Delay}{Pengujian} x = \frac{42}{35} = 1{,}2\ detik$$

Waktu tunda tersebut dipengaruhi oleh proses pengambilan data oleh *dashboard monitoring* dengan komunikasi jaringan antara *client* dan ESP32. Meskipun terdapat waktu tunda pengiriman, seluruh data berhasil diterima dan dikirimkan tanpa kehilangan informasi.

4.3 Pengujian Dashboard Monitoring

Pengujian *dashboard monitoring* dilakukan untuk mengetahui kemampuan sistem dalam menerima data, menyimpan data dan menampilkan data pada dashboard monitoring berdasarkan data yang berasal dari ESP32, pengujian ini meliputi parameter dalam keberhasilan penyimpanan data water pump ke dalam database, ketepatan menampilkan data produksi pada tabel *monitoring*, serta kesesuaian indikator status sistem yang terdiri dari indikator motor *stepper*, status level cairan drigen, status *storage conveyor*, dan status *Water Pump*.

4.3.1 Deskripsi Pengujian

LOKASI : Rumah Rafi Arfahnsyah

TANGGAL : 5 June 2026

PELAKSANAAN : 1. Rafi Arfansyah
2. Muhammad Rafii Nur Rahkim

TUJUAN : memastikan data yang diterima dari ESP32 dapat ditampilkan dengan benar pada dashboard web.

Berikut merupakan daftar alat dan bahan yang digunakan pada pengujian pada alat ditunjukkan pada Tabel 4.8.

Tabel 4. 8 Daftar alat pengujian dashboard monitoring

NO	NAMA ALAT	JUMLAH	FUNGSI
1	ESP32-S3	1	Mengirim data sistem filling machine
2	ESP32	1	Menerima data dari ESP32-S3 dan mengirimkan data ke dashboard.
3	Laptop	1	Menjalankan dashboard monitoring dan melakukan pengujian
4	Wifi	1	Sebagai komunikasi data antara ESP32 dengan dashboard web.
5	XAMPP	1	Menjalankan layanan Apache dan MySQL sebagai server lokal.

NO	NAMA ALAT	JUMLAH	FUNGSI
6	Visual Studio Code	1	menjalankan kode program dashboard monitoring berbasis HTML, CSS, JavaScript, dan PHP.
7	Mysql	1	Menyimpan data produksi dan data monitoring sistem.
8	Web Browser	1	Menampilkan dashboard monitoring.
9	Kabel Jumper	3	Menghubungkan jalur komunikasi UART antar ESP32.

4.3.2 Prosedur Pengujian

Langkah-langkah pengujian :

1. Aktifkan sistem *filling machine* hingga menghasilkan data produksi.
2. Pastikan pada ESP32 menerima data dari ESP32-S3 dan mengirim data dengan format JSON.
3. Pastikan ESP32 dengan laptop terkoneksi dengan wifi yang sama.
4. Selanjutnya, buat folder *monitoring* kemudian letakkan folder tersebut di dalam *direktori* htdocs pada *instalasi* XAMPP.
5. Kemudian buka aplikasi *visual studio code* dan pilih folder *monitoring*.
6. Selanjutnya, buat dan simpan file program PHP, JavaScript dan HTML yang sesuai dengan rancangan sistem *dashboard monitoring* yang telah direalisasikan.
7. Pastikan IP *address* pada ESP32 yang digunakan pada program PHP sesuai.

8. Aktifkan layanan server Apache pada aplikasi XAMPP sebagai server local.
9. Aktifkan layanan MySQL pada aplikasi XAMPP agar *database* dapat diakses oleh program PHP.
10. Selanjutnya, lakukan pembuatan *database* dan tabel pada MySQL yang digunakan untuk menyimpan data produksi.
11. amati perubahan data yang ditampilkan pada tabel produksi *Water Pump 1* dan *Water Pump 2*.
12. Mengamati perubahan indikator status *conveyor*, status drigen, dan indikator pompa.
13. Buka web browser kemudian akses halaman *dashboard monitoring* melalui alamat localhost/monitoring**.**
14. Mencatat hasil pengujian setiap perubahan data *filling machine.*

4.3.3 Data Hasil Pengujian *Dashboard Monitoring* dan *Database*

Tabel 4. 9 Data Hasil Pengujian *Water Pump* 1 Pada *Dashboard Monitoring*

NO	TANGGAL	WAKTU	WADAH BOTOL	VOLUME P1	VOLUME + QTY P1	TOTAL	HASIL
1	05/06/2026	14:12:20	1	60ML	1	60	BERHASIL
2		14:12:28			0	-	-
3		14:12:30			1	120	BERHASIL
4		14:12:38			0	-	-
5		14:12:40			1	180	BERHASIL
6		14:12:48			0	-	-
7		15:13:06	2	60ML	1	60	BERHASIL
8		15:13:14			0	-	-
9		15:13:16			1	120	BERHASIL
10		15:13:24			0	-	-
11		15:13:26			1	180	BERHASIL
12		15:13:34			0	-	-
13		15:14:30	3	60ML	1	240	BERHASIL
14		15:14:38			0	-	-
15		15:14:40			1	300	BERHASIL
16		15:14:48			0	-	-
17		15:14:52			1	360	BERHASIL
18		15:15:00			0	-	-

NO	TANGGAL	WAKTU	WADAH BOTOL	VOLUME P1	VOLUME + QTY P1	TOTAL	HASIL
19		15:57:20			1	420	BERHASIL
20		15:57:28			0	-	-
21		15:57:31	4	60ML	1	480	BERHASIL
22		15:57:39			0	-	-
23		15:57:41			1	540	BERHASIL
24		15:57:48			0	-	-
25		15:58:38			1	600	BERHASIL
26		15:58:44			0	-	-
27		15:58:48	5	60ML	1	660	BERHASIL
28		15:58:56			0	-	-
29		15:58:58			1	720	BERHASIL
30		15:59:06			0	-	-

Berdasarkan Tabel 4.9 hasil pengujian data Water Pump 1 pada dashboard monitoring, proses pembacaan data dilakukan berdasarkan jumlah botol yang berhasil melakukan proses pengisian. Pada pengujian wadah botol 1, Water Pump 1 melakukan pengisian dengan volume 60 mL setiap satu kali proses. Ketika nilai quantity (QTY) bernilai 1, sistem akan melakukan penambahan jumlah produksi dan total volume akan bertambah sebesar 60 mL, seperti pada pengujian pertama dengan total 60 mL, pengujian kedua menjadi 120 mL, hingga pengujian ketiga menjadi 180 mL. Sedangkan ketika nilai QTY bernilai 0, tidak terjadi penambahan volume karena proses pengisian tidak berlangsung.

Tabel 4. 10 Data Hasil Pengujian Tabel *Water Pump* 2 Pada *Dashboard Monitoring*

NO	TANGGAL	WAKTU	WADAH BOTOL	VOLUME P2	VOLUME + QTY P2	TOTAL (ML)	HASIL
1		14:12:20			1	60	BERHASIL
2		14:12:28			0	-	-
3	05/06/2026	14:12:30	1	60ML	1	120	BERHASIL
4		14:12:38			0	-	-
5		14:12:40			1	180	BERHASIL

NO	TANGGAL	WAKTU	WADAH BOTOL	VOLUME P2	VOLUME + QTY P2	TOTAL (ML)	HASIL
6		14:12:48			0	-	-
7		15:13:06			1	60	BERHASIL
8		15:13:14			0	-	-
9		15:13:16	2	60ML	1	120	BERHASIL
10		15:13:24			0	-	-
11		15:13:26			1	180	BERHASIL
12		15:13:34			0	-	-
13		15:14:30			1	240	BERHASIL
14		15:14:38			0	-	-
15		15:14:40	3	60ML	1	300	BERHASIL
16		15:14:48			0	-	-
17	05/06/2026	15:14:52			1	360	BERHASIL
18		15:15:00			0	-	-
19		15:57:20			1	420	BERHASIL
20		15:57:28			0	-	-
21		15:57:31	4	60ML	1	480	BERHASIL
22		15:57:39			0	-	-
23		15:57:41			1	540	BERHASIL
24		15:57:48			0	-	-
25		15:58:38			1	600	BERHASIL
26		15:58:44			0	-	-
27		15:58:48	5	60ML	1	660	BERHASIL
28		15:58:56			0	-	-
29		15:58:58			1	720	BERHASIL
30		15:59:06			0	-	-

Berdasarkan Tabel 4.10 hasil pengujian data Water Pump 2 pada dashboard monitoring, proses pembacaan data dilakukan berdasarkan jumlah botol yang berhasil melakukan proses pengisian. Pada pengujian wadah botol 1, Water Pump 2 melakukan pengisian dengan volume 60 mL setiap satu kali proses. Ketika nilai quantity (QTY) bernilai 1, sistem akan melakukan penambahan jumlah produksi dan total volume akan bertambah sebesar 60 mL, seperti pada pengujian pertama dengan total 60 mL, pengujian kedua menjadi 120 mL, hingga pengujian ketiga menjadi 180 mL.

Sedangkan ketika nilai QTY bernilai 0, tidak terjadi penambahan volume karena proses pengisian tidak berlangsung.

Tabel 4. 11 Data Hasil Pengujian Penyimpanan *Database*

NO	Tanggal	WADAH	Waktu	PUMP ID	VOLUME (mL)	QTY	TOTAL	Data Tersimpan
1	05/06/2026	1	14:12:20	1	60	1	60	Tersimpan
2				2	60	1	60	Tersimpan
3			14:12:30	1	60	1	60	Tersimpan
4				2	60	1	60	Tersimpan
5			14:12:40	1	60	1	60	Tersimpan
6				2	60	1	60	Tersimpan
7		2	15:13:06	1	60	1	60	Tersimpan
8				2	60	1	60	Tersimpan
9			15:13:16	1	60	1	60	Tersimpan
10				2	60	1	60	Tersimpan
11			15:13:26	1	60	1	60	Tersimpan
12				2	60	1	60	Tersimpan
13		3	15:14:30	1	60	1	60	Tersimpan
14				2	60	1	60	Tersimpan
15			15:14:40	1	60	1	60	Tersimpan
16				2	60	1	60	Tersimpan
17			15:14:52	1	60	1	60	Tersimpan
18				2	60	1	60	Tersimpan
19		4	15:57:20	1	60	1	60	Tersimpan
20				2	60	1	60	Tersimpan
21			15:57:31	1	60	1	60	Tersimpan
22				2	60	1	60	Tersimpan
23			15:57:41	1	60	1	60	Tersimpan
24				2	60	1	60	Tersimpan
25		5	15:58:38	1	60	1	60	Tersimpan
26				2	60	1	60	Tersimpan
27			15:58:48	1	60	1	60	Tersimpan
28				2	60	1	60	Tersimpan
29			15:58:58	1	60	1	60	Tersimpan
30				2	60	1	60	Tersimpan

Tabel 4.11 menunjukkan hasil pengujian proses penyimpanan data ke dalam database pada sistem *filling machine*. Pengujian dilakukan pada tanggal 5 Juni 2026. Data yang dikirim oleh ESP32

kemudian diterima dan diproses oleh program PHP untuk disimpan ke dalam database. Waktu penyimpanan data pada database sesuai dengan waktu pengiriman data dari ESP32 ke program PHP. Data yang berhasil disimpan meliputi tanggal, nomor wadah, waktu proses pengisian, ID pompa yang digunakan, volume cairan (mL), jumlah produk (*QTY*), dan total volume.

Tabel 4. 12 Data Hasil Pengujian Indikator Pada *Dashboar Monitoring*

NO	WADAH BOTOL	WAKTU	LED ST	DRIGEN	STATUS CONVEYOR	LED P1	LED P2
1		14:12:20	OFF	ON	NO	ON	ON
2		14:12:28	ON	ON	NO	OFF	OFF
3		14:12:30	OFF	ON	NO	ON	ON
4	1	14:12:38	ON	ON	NO	OFF	OFF
5		14:12:40	OFF	ON	NO	ON	ON
6		14:12:48	ON	ON	NO	OFF	OFF
7		14:12:52	OFF	ON	YES	OFF	OFF
8		15:13:06	OFF	ON	NO	ON	ON
9		15:13:14	ON	ON	NO	OFF	OFF
10		15:13:16	OFF	ON	NO	ON	ON
11	2	15:13:24	ON	ON	NO	OFF	OFF
12		15:13:26	OFF	ON	NO	ON	ON
13		15:13:34	ON	ON	NO	OFF	OFF
14		15:13:38	OFF	ON	YES	OFF	OFF
15		15:14:30	OFF	ON	NO	ON	ON
16		15:14:38	ON	ON	NO	OFF	OFF
17		15:14:40	OFF	ON	NO	ON	ON
18	3	15:14:48	ON	ON	NO	OFF	OFF
19		15:14:52	OFF	ON	NO	ON	ON
20		15:15:00	ON	ON	NO	OFF	OFF
21		15:15:03	OFF	ON	YES	OFF	OFF
22		15:57:20	OFF	ON	NO	ON	ON
23		15:57:28	ON	ON	NO	OFF	OFF
24		15:57:31	OFF	ON	NO	ON	ON
25	4	15:57:39	ON	ON	NO	OFF	OFF
26		15:57:41	OFF	ON	NO	ON	ON
27		15:57:48	ON	ON	NO	OFF	OFF
28		15:57:54	OFF	ON	YES	OFF	OFF
29		15:58:38	OFF	ON	NO	ON	ON

NO	WADAH BOTOL	WAKTU	LED ST	DRIGEN	STATUS CONVEYOR	LED P1	LED P2
30	5	15:58:44	ON	ON	NO	OFF	OFF
31		15:58:48	OFF	ON	NO	ON	ON
32		15:58:56	ON	ON	NO	OFF	OFF
33		15:58:58	OFF	ON	NO	ON	ON
34		15:59:06	ON	ON	NO	OFF	OFF
35		15:59:10	OFF	ON	YES	OFF	OFF

monitoring. Pengujian dilakukan dengan mengirimkan data dari ESP32, kemudian data tersebut diambil oleh program PHP untuk ditampilkan pada *dashboard monitoring*. Waktu yang ditampilkan pada dashboard sesuai dengan waktu pengiriman data dari ESP32. Data yang ditampilkan meliputi nomor wadah, waktu, status LED *start*, status *drigen*, status *conveyor*, serta kondisi LED Pompa 1 dan LED Pompa 2. Berdasarkan hasil pengujian, seluruh indikator pada *dashboard monitoring* berhasil ditampilkan sesuai dengan data yang dikirim oleh ESP32.

4.3.4 Analisis dan Evaluasi

Berdasarkan hasil pengejuan pada tabel 4.9, 4.10 dan 4.11 pada tabel *water pump* 1 dengan *water pump* 2 yang terdapat pada *dashboard monitoring* berhasil menampilkan data produksi *water pump* yang tersimpan pada *database* secara otomatis.

Hasil pengujian bahwa setiap *water pump* melakukan proses pengisian, nilai QTY bertambah satu serta nilai total volume meningkat sesuai hasil dari perkalian antara volume dengan jumlah produksi. Pada pengujian tabel jumlah produksi pada *dashboard monitoring* data berhasil tersimpan pada *database* tanpa ditemukan kehilangan maupun duplikasi data. Hal ini menunjukkan bahwa proses pengiriman data dari ESP32 menuju database serta proses pembacaan data oleh dashboard telah berjalan dengan baik. Sedangkan, hasil pengujian pada tabel 4.12 seluruh indikator yang ditampilkan pada *dashboard monitoring* berhasil menunjukkan kondisi sistem sesuai dengan keadaan sistem *filling machine*.