

BAB IV

PEMBUATAN ALAT

Proses pembuatan alat stempel pneumatik otomatis meliputi tahap realisasi perangkat keras dan perangkat lunak yang digunakan untuk mendukung kinerja sistem secara keseluruhan. Tahapan pembuatan dilakukan berdasarkan hasil perancangan yang telah ditentukan pada tahap sebelumnya, sehingga setiap komponen yang digunakan dapat direalisasikan sesuai fungsi dan kebutuhan sistem. Realisasi alat mencakup pembuatan bagian mekanik, pemasangan sistem pneumatik, perakitan rangkaian kelistrikan, pemasangan sensor, serta pengembangan program pada mikrokontroler ESP32.

Pembuatan alat dilakukan melalui beberapa tahapan, dimulai dari persiapan alat dan bahan, pembuatan dan penyesuaian konstruksi mekanik, pemasangan silinder pneumatik dan solenoid valve 5/2, pemasangan kompresor dan jalur udara bertekanan, hingga perakitan rangkaian elektronik. Pada bagian kendali digunakan ESP32 sebagai pusat pengolahan data dan pengendalian sistem. ESP32 dihubungkan dengan modul relay untuk mengendalikan solenoid, modul dan untuk mengetahui beban, sensor tekanan untuk memantau tekanan udara, serta LCD 20×4 sebagai media untuk menampilkan informasi hasil pembacaan sistem.

Setelah seluruh komponen terpasang, dilakukan integrasi antara sistem mekanik, pneumatik, dan elektronik agar dapat bekerja sebagai satu kesatuan. Sistem pneumatik menggunakan satu solenoid valve 5/2 dengan satu coil untuk mengatur gerakan maju dan mundur silinder. Ketika coil mendapatkan tegangan, valve berpindah sehingga silinder bergerak menuju posisi kerja. Setelah coil dimatikan, valve kembali ke posisi awal melalui mekanisme *spring return* sehingga silinder bergerak kembali. Proses tersebut dikendalikan oleh ESP32 dan dapat dilakukan secara berulang selama sistem masih dalam kondisi aktif.

Selain proses pembuatan dan integrasi, pada bab ini juga dilakukan pengujian awal terhadap setiap bagian alat untuk mengetahui apakah komponen telah bekerja sesuai dengan fungsi yang direncanakan. Pengujian meliputi pengoperasian sistem pneumatik, respons solenoid terhadap perintah ESP32, pembacaan tekanan melalui sensor G1/4, pembacaan beban menggunakan load cell, serta penampilan data pada LCD 20×4. Hasil dari proses pembuatan dan pengujian tersebut digunakan untuk mengetahui kondisi awal alat sebelum dilakukan pengujian kinerja secara keseluruhan.

4.1 Alat dan Bahan

Persiapan alat dan bahan dilakukan dengan memperhatikan kesesuaian spesifikasi setiap komponen terhadap kebutuhan alat. Pada bagian pneumatik, komponen dipilih berdasarkan kebutuhan tekanan dan mekanisme gerak silinder, sedangkan pada bagian elektronik disesuaikan dengan kebutuhan tegangan kerja dan kemampuan ESP32 dalam menerima maupun mengirimkan sinyal. Pemeriksaan awal juga dilakukan terhadap kondisi fisik komponen, terutama sambungan pneumatik, coil solenoid, relay..

Tabel 4- 1 Alat dan Bahan

Nama	Satuan	Jumlah
ESP32	Unit	1
Power Supply 24V 5A	Unit	1
LCD 20x4	Unit	1
LM2596 Buck Converter	Unit	1
Capacitor Elco 470uf	Unit	1
Resistor 10K	Unit	1
Resistor 22k	Unit	1
Kabel Jumper Female to Female	Unit	20
Push Button	Unit	2
Modul Relay 5V	Unit	2
Pneumatik Silinder 50 x 150 mm	Unit	1
Kabel Serabut ukuran 0.75mm	Unit	5meter
Panel Box Listrik 25cm x 30 cm x 15cm	Unit	1
PCB	Cm ²	
Solenoid Valve 5/2	Unit	1
Solder 70W	Unit	1
Kompresor D25WZ1	Unit	1
Sensor Pressure G1/4	Unit	1
Napel 1/4	Unit	3
Napel 1/2	Unit	1
Push Button	Unit	2
Timah	m	25
Bracket Pneumatik Silinder	Unit	1

4.2 Perancangan Perangkat Keras (Hardware)

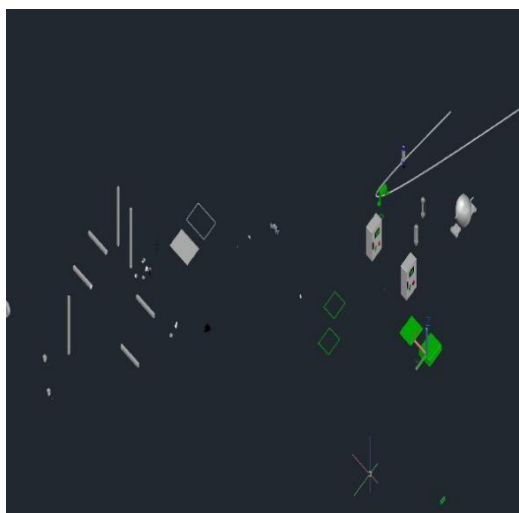
Perancangan perangkat keras dilakukan dengan merealisasikan hubungan antara bagian mekanik, sistem pneumatik, dan sistem elektronik menjadi satu kesatuan alat. Pada sistem ini, ESP32 digunakan sebagai pusat kendali yang mengatur kerja aktuator serta mengolah data dari sensor. Sistem pneumatik berfungsi menghasilkan gerakan stempel melalui silinder pneumatik, sedangkan bagian elektronik digunakan untuk mengatur aktuator, membaca tekanan dan beban, serta menampilkan informasi hasil pembacaan pada LCD 20×4. Perancangan perangkat keras dilakukan secara bertahap agar setiap bagian dapat diperiksa sebelum dihubungkan dengan bagian lainnya.

Hubungan antarbagian dirancang dengan mempertimbangkan fungsi masing-masing komponen. ESP32 mengendalikan solenoid melalui rangkaian relay, sedangkan load cell dihubungkan ke ESP32 melalui modul HX711. Sensor tekanan G1/4 digunakan untuk memperoleh informasi mengenai tekanan pada sistem pneumatik. Data hasil pembacaan sensor kemudian dapat ditampilkan pada LCD bersama dengan nilai beban dan jumlah stempel yang telah dilakukan. Dengan susunan tersebut, perangkat keras tidak hanya berfungsi sebagai penggerak mekanisme, tetapi juga sebagai sistem pemantauan kondisi alat selama proses produksi.

4.2.1 Pembuatan 3D Desain

Pembuatan desain 3D dilakukan sebagai tahap awal untuk menentukan bentuk dan susunan komponen mekanik sebelum dilakukan proses perakitan. Desain digunakan untuk memperkirakan ukuran rangka, posisi silinder pneumatik, posisi mekanisme stempel, serta penempatan komponen pendukung lainnya. Melalui desain tiga dimensi, hubungan antarbagian dapat diperiksa terlebih dahulu sehingga kemungkinan terjadinya kesalahan posisi atau ketidaksesuaian ukuran dapat dikurangi sebelum alat dirakit secara langsung.

Desain juga digunakan untuk menentukan posisi komponen agar gerakan silinder pneumatik dapat diteruskan ke mekanisme stempel secara tepat. Posisi load cell dan bagian yang berhubungan dengan proses pembebanan diperhatikan agar gaya yang diterima sensor dapat diukur dengan baik. Selain itu, penempatan komponen elektronik dan jalur pneumatik diperhitungkan agar tidak mengganggu pergerakan bagian mekanik. Hasil desain tersebut selanjutnya digunakan sebagai acuan dalam proses pembuatan dan perakitan alat.



(a)



(b)

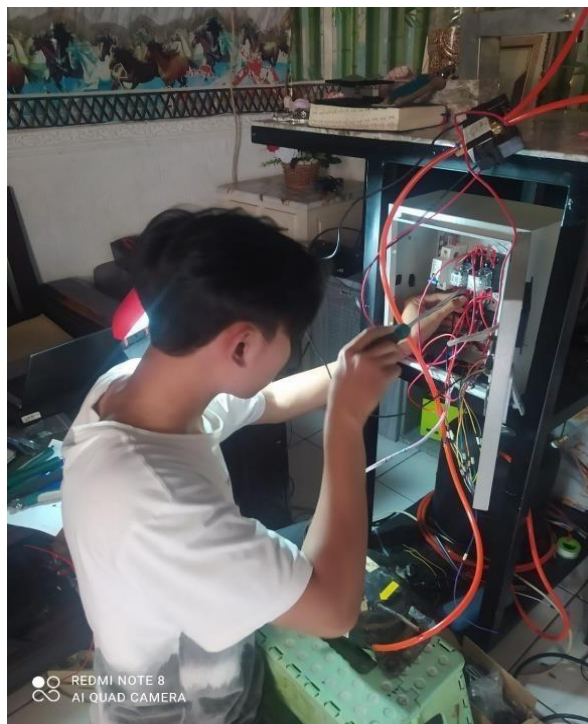
Gambar 4- 1 Desain 3D

4.2.2 Perakitan Mekanik Alat

Perakitan mekanik dilakukan dengan menyusun rangka alat dan memasang komponen yang berhubungan langsung dengan proses pemberian tekanan atau stempel. Silinder pneumatik dipasang pada bagian rangka yang telah ditentukan sehingga batang piston dapat bergerak pada arah yang sesuai dengan mekanisme stempel. Bagian ujung batang piston kemudian dihubungkan dengan mekanisme penekan sehingga gerakan maju silinder dapat digunakan untuk melakukan proses stempel pada benda kerja.

Pada tahap ini juga dilakukan pemeriksaan terhadap posisi dan kekuatan pemasangan setiap komponen. Silinder harus terpasang secara lurus agar batang

piston tidak menerima gaya samping yang berlebihan selama bergerak. Selain itu, bagian mekanisme stempel disesuaikan dengan posisi benda yang akan diproses sehingga perpindahan silinder dapat menghasilkan tekanan pada posisi yang telah ditentukan. Setelah perakitan selesai, dilakukan pemeriksaan gerakan secara manual untuk memastikan tidak terdapat bagian yang menghambat pergerakan silinder maupun mekanisme stempel.



Gambar 4- 2 Tampilan Perakitan Panel(a),(b) Tampilan Hasil Perakitan Panel

Gambar 4-2(a) menunjukkan proses perakitan panel kelistrikan yang dilakukan dengan memasang dan menghubungkan komponen-komponen sistem pada bagian dalam panel. Pada tahap ini dilakukan penyusunan jalur kabel dari sumber tegangan menuju komponen kendali dan komponen keluaran. Penataan kabel dilakukan secara bertahap dengan menyesuaikan posisi masing-masing komponen agar hubungan antarbagian dapat dirakit sesuai dengan rancangan perangkat keras yang telah dibuat. Proses perakitan juga mencakup pemasangan komponen padaudukan panel serta penyambungan terminal menggunakan kabel sesuai dengan kebutuhan sistem.

Gambar 4-2(b) menunjukkan kondisi panel setelah proses perakitan kelistrikan selesai dilakukan. Komponen-komponen yang telah dipasang tersusun pada bagian dalam panel dan dihubungkan menggunakan jalur kabel yang telah ditentukan. Bagian bawah panel dilengkapi dengan terminal sebagai titik penyambungan antara rangkaian kontrol dan komponen yang berada di luar panel. Penempatan komponen dibuat secara teratur untuk mempermudah proses pemeriksaan, pemeliharaan, dan penelusuran jalur kabel apabila terjadi gangguan pada sistem.

4.2.3 Perakitan Sistem Elektronik

Perakitan sistem elektronik dilakukan setelah seluruh komponen yang digunakan dalam sistem monitoring tekanan dan pengendalian pneumatik dipersiapkan. Pada tahap ini, komponen utama yang dirakit meliputi ESP32 DevKit V1, sensor tekanan G1/4, modul relay 1 channel, LCD 20×4 I2C, push button ON, push button OFF, serta rangkaian catu daya. Sistem dirancang agar ESP32 berfungsi sebagai pusat pengolahan data sensor sekaligus pengendali solenoid valve melalui modul relay. Kompresor D25WZ1 tidak dikendalikan secara langsung oleh ESP32, tetapi dioperasikan secara manual sebagai sumber udara bertekanan pada sistem pneumatik.

Perakitan diawali dengan menghubungkan sensor tekanan G1/4 ke pin analog GPIO34 pada ESP32. Sensor digunakan untuk mendeteksi tekanan udara yang dihasilkan oleh kompresor. Karena tegangan keluaran sensor perlu disesuaikan dengan batas tegangan pembacaan ADC ESP32, digunakan rangkaian pembagi tegangan yang terdiri dari resistor 10 k Ω dan 22 k Ω . Sinyal keluaran sensor terlebih dahulu melewati resistor 10 k Ω , kemudian titik pembagi dihubungkan ke GPIO34 dan resistor 22 k Ω dihubungkan menuju ground. Dengan konfigurasi tersebut, tegangan yang diterima ESP32 dapat diturunkan sebelum diproses menjadi nilai tekanan dalam satuan bar.

Selanjutnya, LCD 20×4 dengan komunikasi I2C dirakit sebagai media untuk menampilkan hasil monitoring. Jalur SDA LCD dihubungkan ke GPIO22,

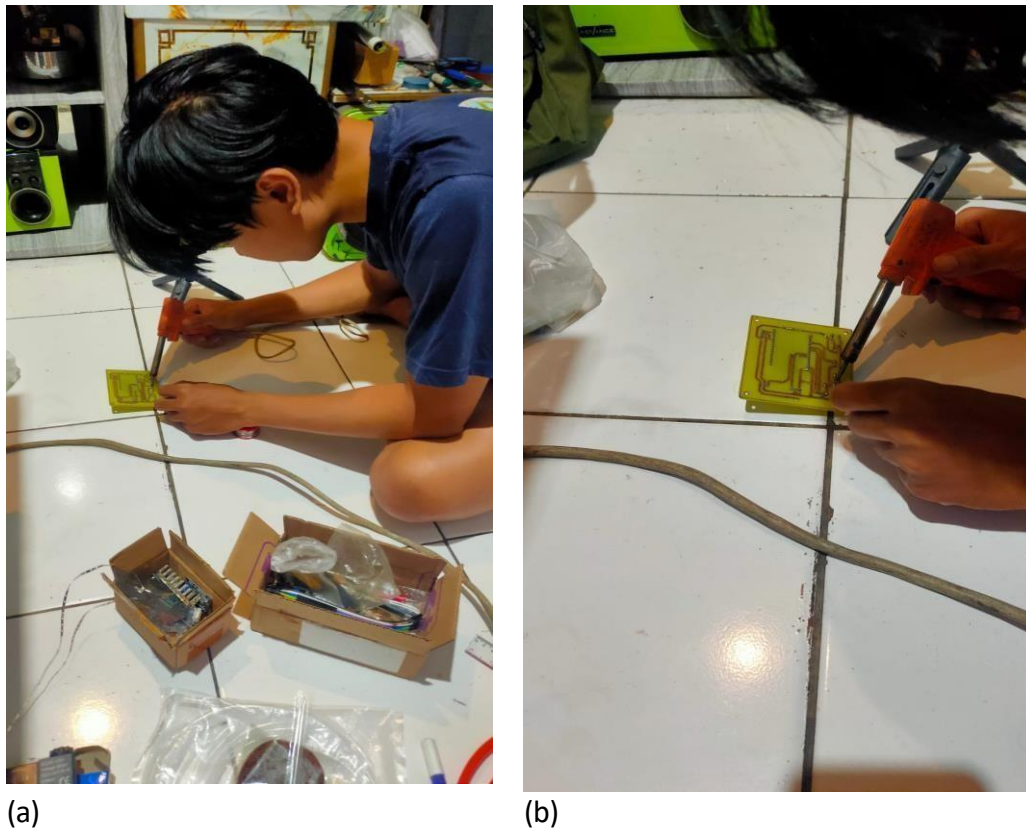
sedangkan jalur SCL dihubungkan ke GPIO21. LCD mendapatkan catu daya dan ground dari rangkaian sistem kontrol. Melalui LCD, pengguna dapat melihat nilai tekanan udara yang terbaca oleh sensor serta status sistem dan solenoid. Dengan adanya tampilan secara langsung, kondisi sistem dapat diamati tanpa harus menggunakan komputer atau Serial Monitor.

Bagian pengendalian sistem menggunakan dua buah push button momentary, yaitu push button ON dan push button OFF. Push button ON dihubungkan ke GPIO25, sedangkan push button OFF dihubungkan ke GPIO35. Push button ON menggunakan konfigurasi INPUT_PULLUP, sehingga tombol aktif ketika GPIO25 berada pada kondisi LOW. Sementara itu, GPIO35 menggunakan resistor pull-down eksternal karena pin tersebut tidak memiliki pull-up atau pull-down internal. Tombol OFF akan memberikan kondisi HIGH ketika ditekan. Kedua tombol digunakan untuk memberikan perintah manual kepada ESP32 dalam mengaktifkan atau menonaktifkan sistem solenoid.

Sinyal kendali solenoid diberikan melalui modul relay 1 channel **optocoupler** yang terhubung pada GPIO26. Pada konfigurasi sistem yang digunakan, modul relay diatur pada mode *active* HIGH, sehingga ketika GPIO26 berada pada kondisi HIGH, relay aktif dan menghubungkan sumber tegangan menuju solenoid valve. Sebaliknya, ketika GPIO26 berada pada kondisi LOW, relay tidak aktif sehingga solenoid kehilangan suplai dan valve kembali ke posisi semula melalui mekanisme spring return. Penggunaan relay sebagai perantara memungkinkan ESP32 mengendalikan solenoid tanpa memberikan tegangan 24 V secara langsung ke pin mikrokontroler.

Seluruh rangkaian kontrol memperoleh suplai sesuai kebutuhan masing-masing komponen, sedangkan solenoid valve menggunakan sumber 24 VDC dan kompresor D25WZ1 menggunakan sumber 220 VAC. Pada tahap perakitan, hubungan antar komponen diperiksa kembali berdasarkan diagram pengawatan untuk memastikan polaritas catu daya, koneksi ground, jalur input sensor, jalur komunikasi I2C, serta jalur kendali relay telah terpasang dengan benar. Setelah

perakitan selesai, dilakukan pengujian awal dengan mengaktifkan ESP32, memeriksa tampilan LCD, menguji respons kedua push button, serta memastikan relay dan solenoid dapat bekerja sesuai perintah sebelum sistem digunakan untuk pengujian tekanan dan proses stamping.



Gambar 4- 3 Tampilan Perakitan PCB(a), (b) Tampilan Proses Penyolderan PCB

Gambar 4-3(a) menunjukkan proses perakitan PCB yang digunakan sebagai bagian dari rangkaian elektronik alat. Pada tahap ini dilakukan pemasangan komponen elektronik pada PCB berdasarkan susunan rangkaian yang telah dirancang. Komponen ditempatkan pada posisi yang telah ditentukan kemudian dilakukan penyambungan antarbagian menggunakan kabel dan jalur rangkaian pada PCB. Proses perakitan dilakukan secara bertahap dengan memperhatikan posisi dan hubungan setiap komponen agar rangkaian dapat berfungsi sesuai dengan kebutuhan sistem.

Gambar 4-3(b) menunjukkan proses penyolderan komponen pada PCB. Penyolderan dilakukan untuk menghubungkan kaki komponen dengan jalur

konduktor pada PCB sehingga terbentuk hubungan listrik yang kuat dan tidak mudah terlepas ketika rangkaian digunakan. Pada proses ini, setiap titik sambungan diperiksa setelah penyolderan untuk memastikan tidak terdapat sambungan yang terputus maupun hubungan antarjalur yang tidak seharusnya.

4.3 Perancangan Perangkat Lunak

Perancangan perangkat lunak dilakukan untuk mengatur proses kerja sistem monitoring tekanan dan pengendalian solenoid valve menggunakan mikrokontroler ESP32. Perangkat lunak dirancang agar ESP32 dapat membaca data dari sensor tekanan G1/4, mengolah hasil pembacaan menjadi nilai tekanan dalam satuan bar, menampilkan informasi pada LCD 20×4, serta mengendalikan solenoid valve melalui modul relay berdasarkan perintah dari push button ON dan OFF. Selain itu, ESP32 juga dilengkapi dengan koneksi Wi-Fi untuk mengirimkan data hasil pembacaan tekanan ke Google Spreadsheet sebagai media pencatatan data pengujian.

Pemrograman sistem dilakukan menggunakan **Arduino** IDE dengan bahasa pemrograman C/C++. Program diawali dengan inisialisasi pin input dan output yang digunakan pada sistem. Sensor tekanan G1/4 terhubung pada GPIO34 sebagai input analog, push button ON terhubung pada GPIO25, push button OFF pada GPIO35, dan modul relay terhubung pada GPIO26. LCD 20×4 menggunakan komunikasi I2C dengan GPIO22 sebagai SDA dan GPIO21 sebagai SCL. Pada sistem ini modul relay menggunakan konfigurasi **Active HIGH**, sehingga GPIO26 berada pada kondisi HIGH ketika solenoid diperintahkan aktif dan LOW ketika solenoid diperintahkan tidak aktif.

Proses pembacaan tekanan dilakukan dengan membaca tegangan keluaran sensor melalui ADC ESP32. Tegangan yang diterima pada GPIO34 terlebih dahulu dikonversi menjadi tegangan keluaran sensor dengan memperhitungkan rangkaian pembagi tegangan menggunakan resistor 10 kΩ dan 22 kΩ. Nilai tegangan tersebut kemudian dikonversikan menjadi tekanan berdasarkan karakteristik sensor yang digunakan dan ditampilkan dalam satuan bar pada LCD. Pada saat sistem pertama

kali diaktifkan, dilakukan kalibrasi titik nol (zero calibration) ketika sensor tidak mendapatkan tekanan udara. Nilai rata-rata pembacaan sensor pada kondisi tersebut digunakan sebagai *zero offset*, sehingga kondisi tanpa tekanan dapat ditampilkan mendekati 0,00 bar pada LCD.

4.3.1 Pembuatan Program Arduino

solenoid mendapatkan suplai, sedangkan ketika GPIO26 bernilai LOW relay tidak aktif dan solenoid dalam kondisi OFF. Pembuatan program Arduino dilakukan sebagai tahap implementasi dari perancangan perangkat lunak pada sistem monitoring tekanan pengisian udara otomatis. Program dibuat menggunakan Arduino IDE dengan bahasa pemrograman C/C++ dan ditanamkan pada mikrokontroler ESP32 DevKit V1. Program yang dibuat berfungsi untuk mengatur pembacaan sensor tekanan G1/4, membaca kondisi push button ON dan OFF, mengendalikan modul relay untuk mengaktifkan solenoid valve, menampilkan informasi pada LCD 20×4, serta mengirimkan data tekanan melalui koneksi Wi-Fi ke Google Spreadsheet.

Pada tahap awal pembuatan program dilakukan deklarasi library dan konfigurasi pin yang digunakan. Sensor tekanan G1/4 dihubungkan pada GPIO34, push button ON pada GPIO25, push button OFF pada GPIO35, dan modul relay pada GPIO26. LCD 20×4 menggunakan komunikasi I2C dengan jalur SDA pada GPIO22 dan SCL pada GPIO21. Modul relay dikonfigurasi sebagai Active HIGH, sehingga ketika GPIO26 bernilai HIGH relay aktif dan

Program kemudian dilengkapi dengan fungsi pembacaan sensor tekanan. Nilai analog yang diterima oleh GPIO34 dikonversikan menjadi nilai tegangan dengan memanfaatkan ADC pada ESP32. Karena sensor terhubung melalui rangkaian pembagi tegangan 10 k Ω dan 22 k Ω , tegangan pada pin ADC dikonversikan kembali menjadi tegangan keluaran sensor. Nilai tersebut selanjutnya dikonversikan menjadi tekanan dalam satuan bar berdasarkan karakteristik sensor. Untuk mengurangi pembacaan tekanan ketika sensor berada pada kondisi tanpa

tekanan, program dilengkapi dengan proses zero calibration pada saat sistem mulai dijalankan. Nilai rata-rata pembacaan sensor pada kondisi tanpa tekanan digunakan sebagai nilai *zero offset* yang kemudian dikurangkan dari hasil pembacaan sensor.

Pengendalian solenoid dilakukan melalui dua push button. Push button ON menggunakan konfigurasi INPUT_PULLUP, sehingga ketika tombol ditekan ESP32 membaca kondisi LOW pada GPIO25. Kondisi tersebut menyebabkan GPIO26 menjadi HIGH sehingga relay aktif dan solenoid mendapatkan tegangan. Sementara itu, push button OFF menggunakan GPIO35 dengan resistor pull-down eksternal. Ketika tombol OFF ditekan, GPIO35 membaca kondisi HIGH dan program memberikan kondisi LOW pada GPIO26 sehingga relay menjadi OFF. Dengan metode tersebut, kondisi solenoid tetap berada pada keadaan ON atau OFF setelah tombol dilepaskan sampai terdapat perintah berikutnya.

4.3.1.1 Cara Kerja Program

Cara kerja program dimulai ketika ESP32 mendapatkan catu daya dan menjalankan proses inisialisasi seluruh perangkat yang terhubung. Pada tahap awal, ESP32 mengatur fungsi setiap pin sesuai dengan perancangannya, yaitu GPIO34 sebagai input sensor tekanan G1/4, GPIO25 sebagai input push button ON, GPIO35 sebagai input push button OFF, GPIO26 sebagai output untuk modul relay, serta GPIO22 dan GPIO21 sebagai jalur komunikasi I2C untuk LCD 20×4. Setelah proses inisialisasi selesai, sistem melakukan proses zero calibration pada sensor tekanan. Pada tahap ini sensor harus berada dalam kondisi tidak mendapatkan tekanan udara. ESP32 mengambil beberapa sampel pembacaan sensor dan menghitung nilai rata-ratanya sebagai zero offset. Nilai tersebut digunakan sebagai acuan agar pembacaan tekanan pada kondisi tanpa tekanan dapat ditampilkan mendekati 0,00 bar.

Ketika push button ON ditekan, ESP32 mendeteksi kondisi LOW pada GPIO25. Program kemudian mengubah status sistem menjadi ON dan memberikan sinyal HIGH pada GPIO26. Karena modul relay dikonfigurasi sebagai Active HIGH, kondisi HIGH pada GPIO26 menyebabkan relay aktif. Relay selanjutnya

menghubungkan sumber tegangan 24 VDC menuju coil solenoid valve sehingga solenoid menjadi aktif dan mengubah posisi valve 5/2. Ketika push button ON dilepaskan, kondisi sistem tetap ON sehingga solenoid tetap aktif sampai terdapat perintah OFF. Sebaliknya, ketika push button OFF ditekan, ESP32 mendeteksi kondisi HIGH pada GPIO35. Program kemudian mengubah status sistem menjadi OFF dan memberikan sinyal LOW pada GPIO26 sehingga relay menjadi tidak aktif. Ketika suplai coil terputus, solenoid menjadi OFF dan valve kembali ke posisi awal melalui mekanisme spring return.

Pada saat sistem bekerja, ESP32 secara terus-menerus membaca tegangan keluaran sensor tekanan melalui GPIO34. Tegangan tersebut diproses dengan memperhitungkan rangkaian pembagi tegangan yang terdiri dari resistor 10 k Ω dan 22 k Ω . Hasil pengolahan kemudian dikonversikan menjadi nilai tekanan dalam satuan bar dan dikurangi dengan nilai *zero offset* yang diperoleh pada proses kalibrasi awal. Nilai tekanan yang telah dikoreksi kemudian ditampilkan pada LCD 20 $\times$ 4 bersama dengan status sistem dan status solenoid. Dengan demikian, pengguna dapat mengetahui kondisi tekanan udara dan keadaan aktuator secara langsung selama sistem beroperasi.

Tabel 4- 2 Inisialisasi Program WiFi

```
#include <WiFi.h>

#include <HTTPClient.h>

#include <WiFiClientSecure.h>

#include <Wire.h>

#include <LiquidCrystal_I2C.h>
```

Pada tabel 4-2 ditampilkan bagian awal program yang berisi proses pemanggilan beberapa *library* yang digunakan untuk mendukung pengoperasian sistem monitoring. *Library* tersebut diperlukan agar mikrokontroler ESP32 dapat menjalankan fungsi komunikasi jaringan WiFi, pengiriman data melalui protokol HTTP, komunikasi I2C, serta pengoperasian LCD 20×4. Adapun *library* yang digunakan meliputi WiFi.h, HTTPClient.h, Wire.h, dan LiquidCrystal_I2C.h.

Library WiFi.h digunakan untuk mengaktifkan dan mengatur koneksi WiFi pada ESP32. Dengan adanya *library* ini, ESP32 dapat terhubung ke jaringan internet sehingga data hasil pembacaan sistem dapat dikirimkan secara daring. Selanjutnya, *library* HTTPClient.h digunakan untuk melakukan komunikasi HTTP antara ESP32 dengan Google Apps Script sebagai media pengiriman data ke Google Spreadsheet.

Selain itu, *library* Wire.h digunakan untuk mendukung komunikasi I2C (*Inter-Integrated Circuit*) pada ESP32. Komunikasi ini digunakan dalam proses pertukaran data antara mikrokontroler dengan perangkat yang menggunakan antarmuka I2C, khususnya LCD 20×4 yang digunakan pada sistem. Sementara itu, *library* LiquidCrystal_I2C.h digunakan untuk mengatur fungsi LCD, seperti proses inisialisasi, pengaturan posisi kursor, dan penampilan nilai tekanan serta beban pada layar LCD.

Dengan pemanggilan keempat *library* tersebut, program memiliki dasar fungsi yang diperlukan untuk mendukung proses monitoring, yaitu menampilkan data hasil pembacaan sistem pada LCD dan mengirimkan data melalui jaringan WiFi ke Google Spreadsheet.

Tabel 4- 3 Konfigurasi Pin

```
define BUTTON_ON    25
#define BUTTON_OFF   35
#define RELAY_PIN    26
#define PRESSURE_PIN 34

#define LCD_SDA      22
#define LCD_SCL      21
```

Pada Gambar 4-3 ditampilkan bagian program yang digunakan untuk melakukan konfigurasi pin pada mikrokontroler ESP32. Konfigurasi ini bertujuan untuk menentukan pin yang digunakan oleh masing-masing komponen pada sistem agar proses pembacaan input, pengendalian output, dan komunikasi dengan LCD dapat dilakukan sesuai dengan rancangan perangkat keras.

Pada program tersebut, BUTTON_ON ditetapkan pada GPIO25 dan BUTTON_OFF pada GPIO35. Kedua pin tersebut digunakan sebagai masukan dari push button untuk memberikan perintah mengaktifkan dan menonaktifkan sistem. Selanjutnya, RELAY_PIN ditetapkan pada GPIO26 yang berfungsi sebagai keluaran untuk memberikan sinyal kendali kepada modul relay dalam mengaktifkan atau menonaktifkan solenoid valve. Sensor tekanan G1/4 menggunakan GPIO34 sebagai pin input analog untuk membaca tegangan keluaran sensor dan mengolahnya menjadi nilai tekanan.

Selain itu, komunikasi antara ESP32 dan LCD 20×4 menggunakan antarmuka I2C. Pada konfigurasi tersebut, jalur SDA menggunakan GPIO22, sedangkan jalur SCL menggunakan GPIO21. Penggunaan deklarasi #define pada bagian program tersebut memudahkan pengaturan dan pemanggilan pin pada bagian program selanjutnya, sehingga setiap komponen memiliki konfigurasi pin yang sesuai dengan wiring sistem yang telah dirancang.

Tabel 4- 4 Konfigurasi WiFi

```
// =====
// LCD
// =====

LiquidCrystal_I2C lcd(0x27, 20, 4);

// =====
// WIFI
// =====

const char* ssid = "realme 5 Pro";
```

```

const char* password = "Program_Tugas_Akhir";

// =====
// GOOGLE APPS SCRIPT
// =====

const char* scriptURL =
"https://script.google.com/macros/s/AKfycbzXD8JWJlVo44DGCJczpzq0DuHY-
QXuW0OP0oBg9IGZT82KWHZKB3IPguXNPWoa8jwZ/exec";

```

Pada Tabel 4-4 ditampilkan bagian program yang digunakan untuk melakukan konfigurasi LCD, koneksi Wi-Fi, dan alamat Google Apps Script pada sistem. Pada bagian LCD, objek LiquidCrystal_I2C dikonfigurasi dengan alamat I2C **0x27** dan ukuran 20×4, sehingga ESP32 dapat melakukan komunikasi dengan LCD untuk menampilkan informasi hasil monitoring sistem.

Pada bagian konfigurasi Wi-Fi, program mendefinisikan nama jaringan melalui variabel ssid dan kata sandi jaringan melalui variabel password. Informasi tersebut digunakan oleh ESP32 untuk menghubungkan sistem dengan jaringan Wi-Fi sehingga mikrokontroler dapat melakukan komunikasi melalui internet. Setelah koneksi berhasil, ESP32 dapat mengirimkan data hasil pembacaan sensor ke layanan pencatatan secara daring.

Selanjutnya, variabel scriptURL digunakan untuk menyimpan alamat Google Apps Script Web App yang berfungsi sebagai tujuan pengiriman data dari ESP32. Alamat tersebut digunakan ketika program melakukan permintaan HTTP menuju Google Apps Script. Data yang dikirim kemudian dapat diterima dan dicatat ke dalam Google Spreadsheet sehingga hasil monitoring tekanan dapat tersimpan secara otomatis selama proses pengujian. Dengan konfigurasi tersebut, ESP32 tidak

hanya menampilkan data secara langsung melalui LCD, tetapi juga mendukung pencatatan data hasil pengukuran secara digital.

Tabel 4- 5 Konfigurasi Sensor Tekanan

```
//
=====
// PRESSURE SENSOR
//
=====

// 0.5 V = 0 MPa
// 4.5 V = 1.2 MPa
//
// 1 MPa = 10 bar

const float SENSOR_MIN_VOLTAGE = 0.50;
const float SENSOR_MAX_VOLTAGE = 4.50;
const float SENSOR_MAX_MPA = 1.20;
// Kenaikan tekanan
const float KENAIKAN_TEKANAN = 0.05;
```

Pada Tabel 4-5 ditampilkan bagian program yang digunakan untuk menentukan parameter karakteristik **sensor** tekanan G1/4 yang digunakan pada sistem. Program menetapkan tegangan minimum sensor sebesar 0,5 V sebagai kondisi tekanan **0** MPa, sedangkan tegangan maksimum sensor sebesar 4,5 V digunakan sebagai batas pembacaan 1,2 MPa. Nilai tersebut menjadi dasar bagi ESP32 dalam melakukan konversi tegangan keluaran sensor menjadi nilai tekanan.

Selain itu, program mencantumkan hubungan satuan tekanan, yaitu 1 MPa = 10 bar, sehingga hasil pengolahan data tekanan dapat ditampilkan dalam satuan bar

pada LCD. Pada bagian berikutnya juga terdapat parameter `SENSOR_MAX_MPA` sebesar 1,20 MPa yang digunakan sebagai batas maksimum tekanan sensor. Dengan parameter tersebut, program dapat mengolah tegangan analog yang diterima ESP32 menjadi nilai tekanan sesuai dengan rentang kerja sensor yang digunakan pada penelitian.

Tabel 4- 6 Konfigurasi Serial Print

```
void calibrateZeroPressure() {

    const int samples = 30;

    float totalPressure = 0.0;

    Serial.println();
    Serial.println("=====");
    Serial.println(" KALIBRASI ZERO PRESSURE");
    Serial.println("=====");
    Serial.println("Pastikan sensor tidak mendapat");
    Serial.println("tekanan udara.");
    Serial.println();
}
```

Pada Tabel 4-6 ditampilkan bagian program yang digunakan untuk melakukan kalibrasi titik nol (zero calibration) pada sensor tekanan. Fungsi `calibrateZeroPressure()` digunakan untuk menentukan nilai awal pembacaan sensor ketika sensor berada dalam kondisi tidak mendapatkan tekanan udara. Pada fungsi tersebut ditentukan sebanyak 30 sampel melalui perintah `const int samples = 30`, yang digunakan sebagai dasar dalam proses pengambilan data untuk menentukan nilai zero offset.

Bagian `float totalPressure = 0.0;` digunakan untuk menyediakan variabel penampung hasil penjumlahan pembacaan tekanan selama proses kalibrasi. Sementara itu, perintah `Serial.println()` digunakan untuk menampilkan informasi proses kalibrasi pada Serial Monitor. Pesan yang ditampilkan memberikan informasi bahwa sistem sedang melakukan “KALIBRASI ZERO PRESSURE”

serta mengingatkan pengguna untuk memastikan sensor tidak mendapatkan tekanan udara selama proses kalibrasi.

Dengan adanya proses tersebut, program dapat memperoleh nilai pembacaan awal sensor pada kondisi tanpa tekanan yang kemudian digunakan sebagai acuan dalam pengolahan data tekanan. Proses kalibrasi ini bertujuan agar kondisi sensor ketika tidak mendapatkan tekanan dapat digunakan sebagai titik nol, sehingga hasil pembacaan tekanan yang ditampilkan pada sistem dapat dikoreksi berdasarkan kondisi awal sensor.

Tabel 4- 7 Void Setup

```
void setup() {

  Serial.begin(115200);

  //
  =====
  // BUTTON
  //
  =====

  pinMode(
    BUTTON_ON,
    INPUT_PULLUP
  );

  pinMode(
    BUTTON_OFF,
    INPUT
  );

  //
  =====
  // RELAY
  //
  =====

  pinMode(
    RELAY_PIN,
    OUTPUT
```

```

);

// Kondisi awal OFF
solenoidState = false;

solenoidOFF();

//
=====
// PRESSURE SENSOR
//
=====

analogReadResolution(12);

analogSetPinAttenuation(
  PRESSURE_PIN,
  ADC_11db
);

```

Pada Tabel 4-7 ditampilkan bagian program pada fungsi void setup() yang digunakan untuk melakukan inisialisasi perangkat input, modul relay, dan sensor tekanan sebelum sistem mulai bekerja. Program diawali dengan Serial.begin(115200) yang digunakan untuk mengatur komunikasi antara ESP32 dengan Serial Monitor pada baud rate 115200.

Pada bagian konfigurasi push button, BUTTON_ON diatur menggunakan INPUT_PULLUP, sehingga GPIO25 dapat membaca kondisi tombol ON dengan memanfaatkan resistor pull-up internal ESP32. Sementara itu, BUTTON_OFF menggunakan konfigurasi INPUT pada GPIO35 karena pin tersebut menggunakan resistor pull-down eksternal. Konfigurasi tersebut digunakan untuk membedakan kondisi tombol ketika ditekan dan tidak ditekan.

Selanjutnya, bagian konfigurasi relay menggunakan pinMode(RELAY_PIN, OUTPUT) untuk menetapkan GPIO26 sebagai pin keluaran yang memberikan sinyal kendali kepada modul relay. Pada kondisi awal, variabel solenoidState diatur menjadi false dan fungsi solenoidOFF() dijalankan sehingga sistem memulai


```
if (  
    buttonOn == LOW &&  
    buttonOff == LOW &&  
    millis() - lastButtonTime >  
    debounceDelay  
) {  
  
    lastButtonTime =  
        millis();  
  
    solenoidState = true;  
  
    // ACTIVE HIGH  
    solenoidON();  
  
    Serial.println();  
    Serial.println(  
        "PUSH BUTTON ON"  
    );  
  
    Serial.println(  
        "GPIO26 = HIGH"  
    );  
  
    Serial.println(  
        "RELAY = ON"  
    );  
  
    Serial.println(  
        "SOLENOID = ON"  
    );  
}
```

```

while (
  digitalRead(BUTTON_ON) ==
  LOW
) {

  delay(10);
}
}

//
=====
// BUTTON OFF
//
=====

if (
  buttonOff == HIGH &&
  buttonOn == HIGH &&
  millis() - lastButtonTime >
  debounceDelay
) {

  lastButtonTime =
    millis();

  solenoidState = false;

  // ACTIVE HIGH
  solenoidOFF();

```

```

Serial.println();
Serial.println(
  "PUSH BUTTON OFF"
);

Serial.println(
  "GPIO26 = LOW"
);

Serial.println(
  "RELAY = OFF"
);

Serial.println(
  "SOLENOID = OFF"
);

while (
  digitalRead(BUTTON_OFF) ==
  HIGH
) {

  delay(10);
}
}

//
=====
// SENSOR
//
=====

```

```
static float pressureBar = 0.0;
```

```
if (  
    millis() - lastSensorRead >=  
    sensorInterval  
) {
```

```
    lastSensorRead =  
        millis();
```

```
    pressureBar =  
        readPressureBar();
```

```
    updateLCD(  
        pressureBar  
    );
```

```
    Serial.print(  
        "Pressure: "  
    );
```

```
    Serial.print(  
        pressureBar,  
        2  
    );
```

```
    Serial.print(  
        " bar | System: "  
    );
```

```
    if (solenoidState) {
```

```
    Serial.println("ON");  
  }  
  else {  
    Serial.println("OFF");  
  }  
}
```

Pada bagian program tersebut merupakan bagian dari fungsi void loop() yang digunakan untuk menjalankan proses utama sistem secara berulang selama ESP32 aktif. Program terlebih dahulu membaca kondisi push button ON dan OFF menggunakan fungsi digitalRead(). Nilai yang diperoleh dari kedua tombol tersebut kemudian digunakan sebagai dasar untuk menentukan perintah pengaktifan atau penonaktifan solenoid.

Ketika kondisi buttonOn bernilai LOW, buttonOff bernilai LOW, dan waktu sejak pembacaan tombol terakhir telah melebihi debounceDelay, program menjalankan perintah untuk mengaktifkan solenoid. Variabel solenoidState diubah menjadi true, kemudian fungsi solenoidON() dijalankan. Karena modul relay menggunakan konfigurasi Active HIGH, kondisi tersebut menyebabkan GPIO26 berada pada kondisi HIGH sehingga relay aktif dan solenoid mendapatkan suplai. Informasi mengenai kondisi tombol, GPIO26, relay, dan solenoid juga ditampilkan melalui Serial Monitor. Perintah while digunakan untuk menunggu hingga push button ON dilepaskan sebelum program melanjutkan proses berikutnya.

Selanjutnya, ketika buttonOff bernilai HIGH, buttonOn bernilai HIGH, dan waktu sejak pembacaan tombol terakhir telah melebihi nilai debounce, program menjalankan perintah untuk menonaktifkan solenoid. Variabel solenoidState diubah menjadi false, kemudian fungsi solenoidOFF() dijalankan sehingga GPIO26 berada pada kondisi LOW. Kondisi tersebut menyebabkan relay tidak aktif dan solenoid menjadi OFF. Informasi perubahan kondisi tersebut juga ditampilkan pada Serial

Monitor. Program kemudian menunggu hingga push button OFF dilepaskan melalui perintah `while`.

Setelah proses pembacaan push button, program melakukan proses pembacaan sensor tekanan secara berkala. Variabel `pressureBar` digunakan untuk menyimpan hasil pembacaan tekanan dalam satuan bar. Pembacaan dilakukan ketika waktu yang telah berlalu sejak pembacaan sebelumnya memenuhi `sensorInterval`. Program kemudian menjalankan fungsi `readPressureBar()` untuk memperoleh nilai tekanan yang telah dikoreksi, kemudian hasilnya dikirimkan ke fungsi `updateLCD()` untuk ditampilkan pada LCD 20×4.

4.4 Hasil Akhir Alat

Setelah melalui tahapan perancangan, pembuatan, dan perakitan, diperoleh hasil akhir berupa sistem monitoring tekanan udara pada kompresor D25WZ1 berbasis mikrokontroler ESP32 yang diterapkan pada mesin stempel pneumatik skala kecil. Sistem yang telah dibuat terdiri dari komponen utama berupa ESP32 DevKit V1 sebagai pusat pengolahan data, sensor *pressure transducer* G1/4 sebagai perangkat pengukuran tekanan udara, *load cell* sebagai parameter tambahan pengukuran beban, modul HX711 sebagai pengolah sinyal *load cell*, LCD 20×4 sebagai media tampilan, serta koneksi WiFi untuk mengirimkan data hasil monitoring ke Google Spreadsheet.

Pada kondisi akhir, sistem dirancang untuk menampilkan nilai tekanan dan beban secara langsung pada LCD 20×4. Data hasil pembacaan tersebut juga dapat dikirimkan secara berkala melalui jaringan WiFi menuju Google Spreadsheet sehingga data pengujian dapat tersimpan secara otomatis dan digunakan dalam proses analisis. Pengiriman data dilakukan berdasarkan interval waktu yang telah ditentukan dalam program, yaitu setiap 5 detik.



Gambar 4- 4 Hasil Akhir Alat

Hasil akhir alat menunjukkan bahwa integrasi antara perangkat keras dan perangkat lunak telah dirancang untuk mendukung proses monitoring parameter pengujian. ESP32 berfungsi sebagai pusat sistem yang menerima dan mengolah data, kemudian menampilkan informasi pada LCD serta mengirimkannya ke Google Spreadsheet. Dengan demikian, alat yang telah dibuat dapat digunakan sebagai sistem pendukung dalam proses pengambilan data tekanan udara dan beban pada pengujian mesin stempel pneumatik skala kecil.