



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



**RANCANG BANGUN MODUL LATIH *AUTOMATIC BOTTLE FILLING AND CAPPING SYSTEM* BERBASIS PLC DENGAN
MONITORING SCADA**

TUGAS AKHIR

Radhifka Defandy Putra

2303321069

PROGRAM STUDI ELEKTRONIKA INDUSTRI

JURUSAN TEKNIK ELEKTRO

POLITEKNIK NEGERI JAKARTA

2026



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



**PENGEMBANGAN SISTEM MONITORING DAN SUPERVISI
SCADA TERINTEGRASI IOT PADA *AUTOMATIC BOTTLE
FILLING AND CAPPING SYSTEM* BERBASIS PLC**

TUGAS AKHIR

**Diajukan guna melengkapi sebagian syarat dalam mencapai gelar
Program Pendidikan Diploma III (D3)**

**Radhifka Defandy Putra
2303321069**

**PROGRAM STUDI ELEKTRONIKA INDUSTRI
JURUSAN TEKNIK ELEKTRO
POLITEKNIK NEGERI JAKARTA
2026**



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

HALAMAN PERYATAAN ORISINALITAS

Tugas Akhir ini adalah hasil karya saya sendiri dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama : Radhifka Defandy Putra
NIM : 2303321069
Program Studi : Elektronika Industri
Tanggal : Depok, 24/6/2026
Tanda Tangan : 





© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

LEMBAR PENGESAHAN

TUGAS AKHIR

Tugas Akhir diajukan oleh :
Nama : Radhifka Defandy Putra
NIM : 2303321069
Program Studi : Elektronika Industri
Judul Tugas Akhir : Rancang Bangun Modul Latih *Automatic Bottle Filling and Capping System* Berbasis PLC dengan Monitoring SCADA.
Sub-Judul Tugas Akhir : Pengembangan Sistem Monitoring dan Supervisi SCADA Terintegrasi IoT pada *Automatic Bottle Filling and Capping System* Berbasis PLC

Telah diuji oleh tim penguji dalam sidang Tugas Akhir pada hari senin
Tanggal 29-06-2026 dan dinyatakan **LULUS**.

Pembimbing I : Rizdam Firly Muzakki, S.Pd., M.T. ()
NIP. 199311082024061001

Depok, 10 Juli 2026.....

Disahkan oleh :

Ketua Jurusan Teknik Elektro




Dr. Murni Dwiyanti, S.T., M.T.

NIP. 197803312003122002



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

KATA PENGANTAR

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan Tugas Akhir ini. Penulisan Tugas Akhir ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Diploma Tiga. Tugas Akhir yang penulis buat adalah Pengembangan Sistem Monitoring dan Supervisi SCADA Terintegrasi IoT pada *Automatic Bottle Filling and Capping System* Berbasis PLC. Penulis menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan tugas akhir ini, sangatlah sulit bagi penulis untuk menyelesaikan tugas akhir ini. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Ibu Dr. Murie Dwiyaniti, S.T., M.T. selaku Ketua Jurusan Teknik Elektro;
2. Bapak Ihsan Auditia Akhinov, S.T., M.T., selaku dosen Kepala Program Studi D3-Elektronika Industri
3. Bapak Rizdam Firly Muzakki, S.Pd., M.T., selaku dosen pembimbing yang telah memberikan arahan, dukungan, dan bantuan dalam penyelesaian Tugas Akhir.
4. Orang tua dan keluarga penulis yang telah memberikan dukungan moral, material, serta doa dalam menyusun dan menyelesaikan Tugas Akhir.
5. Rekan satu tim yang telah banyak membantu penulis dalam menyusun dan menyelesaikan Tugas Akhir.

Akhir kata, penulis berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga Tugas Akhir ini membawa manfaat bagi pengembangan ilmu.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

ABSTRAK

Tugas Akhir ini mengimplementasikan dan menguji sistem gateway berbasis mikrokontroler ESP32S untuk menjembatani komunikasi lokal Modbus RTU pada *Programmable Logic Controller* (PLC) Huikong FX3U – 28MR dengan protokol MQTT. Hasil pengujian menunjukkan bahwa *gateway* beroperasi secara andal; proses polling data Modbus pada baud rate 9600bps berjalan stabil dengan waktu rata-rata 84 ms, sementara konversi data ke format JSON hingga proses *publish* ke broker MQTT tereksekusi dalam 4,46ms. Pada pengujian komunikasi jaringan WiFi dengan RSSI sinyal -62dBm, sistem menghasilkan latensi rata-rata 192,54ms dengan rasio keberhasilan pengiriman data mencapai 100% untuk kebutuhan monitoring real-time. Integrasi perangkat keras dengan perangkat lunak Haiwell Cloud SCADA berhasil dilakukan sepenuhnya (100%), memungkinkan instruksi kontrol dari antarmuka SCADA dikirim secara presisi ke plant melalui perintah MQTT yang diteruskan menjadi instruksi tulis Modbus. Selain itu, mekanisme pemulihan perangkat lunak terbukti efektif dengan tingkat keberhasilan sebesar 100% saat terjadi simulasi kegagalan jaringan, sehingga sistem terhindar dari *blocking* permanen.

Kata kunci: SCADA, ESP32, IoT *Gateway*, PLC, Modbus RTU, MQTT.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

ABSTRACT

This Final Assignment implements and tests an ESP32S microcontroller-based gateway system to bridge local Modbus RTU communication on a Huikong FX3U-28MR Programmable Logic Controller (PLC) with the MQTT protocol. The test results indicate that the gateway operates reliably; the Modbus data polling process at a 9600 bps baud rate runs stably with an average time of 84ms, while the data conversion to JSON format up to the publishing process to the MQTT broker is executed in 4.46ms. During Wi-Fi network communication testing with a signal attenuation of -62dBm, the system produced an average latency of 192.54ms with a 100% data transmission success rate for real-time monitoring needs. Hardware integration with the Haiwell Cloud SCADA software was fully accomplished (100%), enabling control instructions from the SCADA interface to be sent precisely to the plant via MQTT commands that are forwarded as Modbus write instructions. Furthermore, the software recovery mechanism proved effective with a 100% success rate during simulated network failures, thereby preventing the system from experiencing permanent blocking.

Keywords: SCADA, ESP32, IoT Gateway, PLC, Modbus RTU, MQTT.

**POLITEKNIK
NEGERI
JAKARTA**



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

DAFTAR ISI

HALAMAN SAMPUL	i
HALAMAN JUDUL.....	ii
HALAMAN PERNYATAAN ORISINALITAS	iii
LEMBAR PENGESAHAN	iv
KATA PENGANTAR.....	v
ABSTRAK.....	vi
<i>ABSTRACT</i>	vii
DAFTAR ISI	viii
DAFTAR TABEL.....	xi
DAFTAR GAMBAR	xii
DAFTAR LAMPIRAN	xiv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah.....	2
1.3 Tujuan.....	3
1.4 Batasan Masalah.....	3
1.5 Luaran.....	3
BAB II TINJAUAN PUSTAKA	5
2.1 State of The Art	5
2.2 Alat Pengisi Air dan Penutup Botol Otomatis	8
2.3 PLC Huikong HK3U	9
2.4 Microcontroller Unit (MCU).....	10
2.4.1 ESP-32S.....	10



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

2.4.2 FreeRTOS	11
2.4.3 Modul esp-wifi.....	11
2.4.4 Modul esp-mqtt.....	12
2.4.5 Modul esp-modbus	12
2.4.6 NVS dan LittleFS.....	12
2.5 SCADA.....	13
2.5.1 Haiwell Cloud SCADA 3.....	13
2.6 Modul Latih PLC.....	14
2.7 Internet of Things (IoT).....	14
2.8 Protokol Komunikasi.....	15
2.8.1 MQTT	15
2.8.2 Modbus	15
2.9 MAX485.....	16
BAB III PERENCANAAN DAN RELISASI	17
3.1 Perancangan Alat.....	17
3.1.1 Deskripsi Alat	17
3.1.2 Cara Kerja Alat	19
3.1.3 Spesifikasi Alat	23
3.1.4 Diagram Blok.....	23
3.2 Realisasi Alat.....	26
3.2.1 Perancangan perangkat keras (<i>Hardware</i>).....	27
3.2.2 Perancangan Perangkat Lunak (<i>Software</i>).....	31
3.2.3 Integrasi Haiwell Cloud SCADA 3 dengan MQTT.....	37
BAB IV PEMBAHASAN.....	46
4.1 Pengujian WiFi dan MQTT	46



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

4.1.1 Deskripsi Pengujian	46
4.1.2 Prosedur Pengujian	47
4.1.3 Data Hasil Pengujian	47
4.1.4 Analisis Data / Evaluasi	49
4.2 Pengujian Modbus	50
4.2.1 Deskripsi Pengujian	50
4.2.2 Prosedur Pengujian	50
4.2.3 Data Hasil Pengujian	51
4.2.4 Analisis Data / Evaluasi	53
4.3 Pengujian SCADA	53
4.3.1 Deskripsi Pengujian	54
4.3.2 Prosedur Pengujian	54
4.3.3 Data Hasil Pengujian	55
4.3.4 Analisis Data / Evaluasi	56
4.4 Pengujian Mekanisme Retry pada WiFi	56
4.4.1 Deskripsi Pengujian	57
4.4.2 Prosedur Pengujian	57
4.4.3 Data Hasil Pengujian	58
4.4.4 Analisis Data / Evaluasi	59
BAB V PENUTUP	60
5.1 Kesimpulan	60
5.2 Saran	61
DAFTAR PUSTAKA	62
LAMPIRAN	xv



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

DAFTAR TABEL

Tabel 2.1 penelitian terdahulu oleh (Achmad Asrori Al Sarfini, Denny Irawan, 2022)	5
Tabel 2.2 penelitian terdahulu oleh (Nuha Nadhiroh, Arum Kusuma Wardhany, Hatib Setiana, Raihan Renaldy, Amanda Alifya Putri, Mutiara Dwi Handayani, 2021)	6
Tabel 3.1 Spesifikasi Alat	23
Tabel 3.2 Variable pada SCADA	37
Tabel 4.1 Alat dan Bahan Pengujian WiFi dan MQTT	47
Tabel 4.2 Data Hasil Pengujian WiFi dan MQTT	47
Tabel 4.3 Alat dan Bahan Pengujian Modbus	50
Tabel 4.4 Data Hasil Pengujian Modbus	51
Tabel 4.5 Alat dan Bahan Pengujian SCADA	54
Tabel 4.6 Data Hasil Pengujian SCADA	55
Tabel 4.7 Alat dan Bahan Pengujian Mekanisme Retry pada WiFi	57
Tabel 4.8 Data Hasil Pengujian Mekanisme Retry pada WiFi	58



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

DAFTAR GAMBAR

Gambar 2.1PLC Huikong FX3U - 28MR.....	10
Gambar 2.2ESP-32S with Antenna port.....	11
Gambar 2.3Haiwell Cloud SCADA 3	13
Gambar 2.4Modul Latih PLC	14
Gambar 2.5MAX485.....	16
Gambar 3.1Flowchart perancangan alat.....	17
Gambar 3.2Flowchart Cara kerja Gateway Modbus MQTT	20
Gambar 3.3Flowchart cara kerja Haiwell SCADA Cloud 3	22
Gambar 3.4Blok Diagram Alat	24
Gambar 3.5 Skematik Gateway Modbus MQTT	30
Gambar 3.6Tampak depan Gateway Modbus MQTT.....	31
Gambar 3.7Tampak belakang Gateway Modbus MQTT.....	31
Gambar 3.8Struk tur Program Gateway Modbus MQTT	32
Gambar 3.9Flowchart Program Gateway Modbus MQTT	34
Gambar 3.10Tampilan Landing Page SCADA.....	39
Gambar 3.11Tampilan Status System SCADA.....	39
Gambar 3.12Tampilan Plant SCADA	40
Gambar 3.13Tampilan Flowcontrol Popup SCADA	40
Gambar 3.14Tampilan Pengaturan MQTT Device pada SCADA	41
Gambar 3.15Tampilan Pengaturan Group Variable pada SCADA	41
Gambar 3.16Tampilan Pengaturan Mapping Tabel Coil pada SCADA.....	42
Gambar 3.17Tampilan Pengaturan Mapping Tabel Register pada SCADA	42



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Gambar 3.18Tampilan Pengaturan MQTT sebagai reporting server pada SCADA	43
Gambar 3.19Tampilan Pengaturan Mapping Tabel Internal Variable pada SCADA	43
Gambar 3.20Tampilan Pengaturan Group Variable pada SCADA	44
Gambar 3.21Tampilan Pengaturan Mapping Reporting server group coil dengan Internal Variable pada SCADA	44
Gambar 3.22Tampilan Pengaturan Mapping Reporting server group registerl dengan Internal Variable pada SCADA	45





Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

DAFTAR LAMPIRAN

Lampiran 1.	Daftar Riwayat Hidup	xv
Lampiran 2.	Datasheet ESP32	xvi
Lampiran 3.	Datasheet MAX485.....	xvi
Lampiran 4.	Poster Alat.....	xvii
Lampiran 5.	SOP Alat.....	xvii
Lampiran 6.	Struktur Program.....	xviii
Lampiran 7.	Foto Alat.....	xviii
Lampiran 8.	Pengujian Alat	xix
Lampiran 9.	Program Modbus MQTT Gateway – config.html.....	xx
Lampiran 10.	Program Modbus MQTT Gateway – wifi_config.html	xxii
Lampiran 11.	Program Modbus MQTT Gateway – main.c.....	xxiii
Lampiran 12.	Program Modbus MQTT Gateway – fs_handle.c dan fs_handle.h xxiii	
Lampiran 13.	Program Modbus MQTT Gateway – gpiocon.c dan gpiocon.h xxiv	
Lampiran 14.	Program Modbus MQTT Gateway – modbus_handle.c dan modbus_handle.h	xxiv
Lampiran 15.	Program Modbus MQTT Gateway – mqtt_handle.c dan mqtt_handle.h	xxvi
Lampiran 16.	Program Modbus MQTT Gateway – wifi_handle.h	xxvi
Lampiran 17.	Program Modbus MQTT Gateway – webhandle.c dan webhandle.h	xxvii



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

BAB I PENDAHULUAN

1.1 Latar Belakang

Perkembangan industri manufaktur sudah dari lama berjalan dengan kontrol dan indikator fisik, lalu berkembang ke HMI, SCADA dan yang sedang pesat sekarang adalah IoT (Aliyu Enemosah & Ogbonna George Ifeanyi, 2024).

Sistem IoT yang umumnya mengandalkan protokol TCP Rest API terasa kurang efisien karena memiliki sifat *stateless* yang mengharuskan perangkat berulang kali membuka dan menutup koneksi TCP (Esposito et al., 2023) sehingga dikembangkanlah WebSocket, namun kedua protokol tersebut memiliki kelemahan, dari yang Rest API yang boros karena membutuhkan *request* untuk mengecek setiap perubahan data, WebSocket yang memiliki header yang bisa dibilang boros untuk penggunaan IoT (Tsaqief & Sutopo, 2025) serta kekurangan mutlak dari kedua protokol tersebut adalah memerlukan *routing*, server dan logika kode untuk setiap proyek (Amirkhanov et al., 2025).

MQTT menjawab kekurangan tersebut dengan sistem *publish* dan *subscribe* dengan penengah broker, server hanya dibutuhkan 1 yaitu server untuk hosting broker (Jesus et al., 2025). MQTT juga memiliki topik, setiap proyek IoT yang dibuat dapat terhubung ke 1 broker dengan topik yang berbeda, setiap topik juga dapat dikunci dengan *username* dan *password* tertentu untuk privasi dan keamanan setiap proyek.

Adapun masalah lain yaitu dari segi kontroler, masih banyak PLC yang tidak memiliki fitur komunikasi MQTT atau bahkan Ethernet secara default, tetapi masih memiliki cara komunikasi untuk area local seperti RS-485 dan RS-232 dengan protokol komunikasi Modbus (Nurahmat & Suryo, 2025).

Menjawab masalah tersebut, suatu hub atau *gateway* diperlukan agar perubahan data pada PLC dapat dikirim ke tempat lain melalui jaringan



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

internet secara realtime. *Gateway* tersebut harus mampu untuk melakukan banyak *request* ke Modbus Slave pada jalur komunikasi RS-485 lalu memuat data tersebut menjadi paket yang dapat dikirim secara efisien melalui protokol MQTT, dan juga melakukan sebaliknya yaitu menunggu paket data tiba dari MQTT dan mengubah nilai pada alamat modbus yang berkaitan.

Oleh karena itu, modul IoT sebagai Modbus MQTT *Gateway* ini dibuat untuk mengatasi masalah *polling task* yang umumnya terjadi pada implementasi MQTT pada ESP32 dengan arsitektur superloop pada sistem bare metal berbasis *Arduino Framework* yang akan menghabiskan kebanyakan siklus instruksi pada CPU hanya untuk melakukan check apakah ada paket MQTT yang datang (Espino et al., 2024). Masalah ini diselesaikan dengan cara menggunakan arsitektur *Event Driven* yang dimana *Task* MQTT tersebut akan dalam keadaan *sleep* hingga ada paket MQTT yang diterima dari topik yang di-*subscribe* (Bertoli et al., 2024).

Selain itu, modul ini juga mengatasi masalah pengembangan sistem IoT yang memakan waktu lama karena memerlukan pemrograman terlebih dahulu, yaitu dirancang Website yang berjalan lokal untuk konfigurasi WiFi, broker MQTT serta kredensialnya, topik MQTT untuk *publish* dan *subscribe*, Modbus, serta *mapping* pada alamat modbus dan variable tag SCADA agar pengguna tidak memerlukan pemrograman sama sekali dan dapat melakukan semua konfigurasi tersebut melalui antarmuka Website yang interaktif.

1.2 Perumusan Masalah

Berdasarkan latar belakang tersebut, dapat dirumuskan beberapa masalah yang akan dibahas dalam penelitian ini, yaitu:

1. Bagaimana mengimplementasikan sistem monitoring *SCADA* pada Modul Latih *Automatic Bottle Filling & Capping System* untuk menampilkan status proses secara *real-time* sebagai bagian dari sistem pengendalian industri?
2. Bagaimana mentransmisikan data dari Modbus yang ada di PLC ke MQTT



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

melalui jaringan internet secara efisien?

3. Optimisasi kode seperti apa yang akan dilakukan agar mikrokontroler dapat menanggulangi masalah ketika terjadi crash atau jaringan yang terputus saat tengah digunakan?

1.3 Tujuan

Berdasarkan rumusan masalah di atas, maka tujuan dari penelitian ini adalah sebagai berikut:

1. Membuat Antarmuka Web yang dapat digunakan untuk konfigurasi kredensial WiFi, MQTT, Konfigurasi dasar Modbus serta Konfigurasi Mapping alamat Modbus.
2. Membuat algoritma untuk melakukan pengecekan perubahan nilai pada alamat modbus tertentu dan mengelompokkannya pada JSON supaya pengiriman data ke MQTT menjadi lebih efisien dan lebih hemat *bandwidth* internet.
3. Mengimplementasikan sistem monitoring SCADA untuk menampilkan status proses secara *real-time* sebagai bagian dari sistem pengendalian dan pengawasan operasi.
4. Melakukan optimisasi pada penggunaan memori yang terbatas pada ESP32, penggunaan heap supaya memori digunakan secara dinamis sesuai kebutuhan, dan melakukan *error handling* pada sistem yang krusial seperti koneksi WiFi dan MQTT.

1.4 Batasan Masalah

Dalam penyusunan penelitian ini, terdapat batasan masalah agar pembahasan lebih fokus dan terarah. Batasan tersebut yaitu :

1. Pengujian dilakukan pada lingkungan di kondisi yang sama yaitu dengan sumber WiFi yang sama dan jarak yang sama dengan halangan 1 tembok, jarak 2 meter dan RSSI -62dBm.

1.5 Luaran

Luaran yang diharapkan dari penelitian tugas akhir ini yaitu:



© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

1. Laporan tugas akhir dan jurnal ilmiah yang memuat perancangan, implementasi, serta pengujian Modul Latih *Automatic Bottle Filling & Capping System* berbasis PLC dengan monitoring SCADA sebagai media pembelajaran otomasi industri.
2. Modul Latih *Automatic Bottle Filling & Capping System* berbasis PLC yang dilengkapi dengan sistem monitoring SCADA dan dapat digunakan sebagai sarana praktikum untuk memahami konsep pengendalian, otomasi, dan monitoring sistem industri.
3. *Jobsheet* (modul praktikum) Modul Latih sebagai panduan penggunaan alat dalam kegiatan pembelajaran.
4. Hasil analisis kinerja sistem yang meliputi lama pengambilan data dari PLC, Pengiriman data ke internet, akurasi sistem *SCADA* dalam menampilkan kondisi operasi secara *real-time*, Serta keberhasilan sistem untuk melakukan pemulihan setelah terjadinya kegagalan koneksi WiFi.

**POLITEKNIK
NEGERI
JAKARTA**

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

BAB V PENUTUP

5.1 Kesimpulan

Adapun kesimpulan yang didapatkan dari hasil penelitian, perancangan, dan pengujian yang telah dilakukan terhadap sistem adalah sebagai berikut:

1. Mikrokontroler ESP32S telah berhasil diimplementasikan sebagai *gateway* yang andal untuk menjembatani komunikasi lokal Modbus RTU pada PLC dengan protokol internet MQTT. Proses *polling* data Modbus pada baud rate 9600 bps berjalan stabil dengan rata-rata waktu 84 ms, dan proses konversi data ke format JSON hingga *publish* ke broker MQTT dieksekusi dengan sangat cepat (rata-rata 4,46 ms) tanpa membebani kinerja PLC.
2. Pengujian komunikasi MQTT via jaringan WiFi menunjukkan performa yang sangat baik untuk kebutuhan *real-time monitoring*. Pada kondisi RSSI sinyal -62dBm (terhalang 1 tembok pada jarak 2 meter), latensi rata-rata yang dihasilkan adalah 192,54 ms dengan rasio keberhasilan pengiriman data mencapai 100%.
3. Integrasi antara perangkat keras dan perangkat lunak Haiwell Cloud SCADA terbukti berhasil 100%. Antarmuka SCADA mampu menjalankan instruksi kontrol (*Start, Stop, Emergency*, pengaturan aktuator, dan pompa) secara presisi ke plant (modul latih) melalui perintah *publish* MQTT yang diteruskan menjadi instruksi tulis (*write*) Modbus ke PLC.
4. Mekanisme *retry* dan perlindungan sistem (pengaman perangkat lunak) beroperasi sesuai dengan perancangan. Saat terjadi simulasi kegagalan jaringan WiFi, sistem mampu melakukan *auto-recovery* pada koneksi WiFi, MQTT, dan *polling* Modbus dengan tingkat keberhasilan pemulihan 100%, sehingga sistem terhindar dari *crash* atau *blocking* permanen.



5.2 Saran

1. Pengujian Modbus saat ini menggunakan baud rate standar 9600bps. Disarankan untuk melakukan eksperimen konfigurasi baud rate yang lebih tinggi (seperti 19200bps) guna memangkas waktu polling jika akan dilakukan penambahan jumlah data yang akan dikirimkan ke SCADA berbasis IoT.
2. Sistem SCADA saat ini berfokus pada monitoring dan kontrol real-time. Ke depannya, data dari broker MQTT dapat dihubungkan ke sistem backend berbasis database relasional atau time-series (seperti PostgreSQL dengan ekstensi TimescaleDB) agar riwayat pencatatan status alat dapat disimpan untuk keperluan analisis data historis yang lebih mendalam.

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

**POLITEKNIK
NEGERI
JAKARTA**

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

DAFTAR PUSTAKA

- Al Sarfini, A. A., & Irawan, D. (2024). Sistem Kontrol Jarak Jauh Plc Menggunakan Esp32 Berbasis Iot. *JURNAL AMPLIFIER : JURNAL ILMIAH BIDANG TEKNIK ELEKTRO DAN KOMPUTER*, 14(1). <https://doi.org/10.33369/jamplifier.v14i1.33484>
- Aliyu Enemosah, & Ogbonna George Ifeanyi. (2024). SCADA in the Era of IoT: Automation, Cloud-driven security, and machine learning applications. *International Journal of Science and Research Archive*, 13(1). <https://doi.org/10.30574/ijrsra.2024.13.1.1975>
- Amirkhanov, B., Amirkhanova, G., Kunelbayev, M., Adilzhanova, S., & Tokhtassyn, M. (2025). Evaluating HTTP, MQTT over TCP and MQTT over WEBSOCKET for digital twin applications: A comparative analysis on latency, stability, and integration. *International Journal of Innovative Research and Scientific Studies*, 8(1), 679–694. <https://doi.org/10.53894/ijirss.v8i1.4414>
- Arrahma, S. A., & Mukhaiyar, R. (2023). Pengujian Esp32-Cam Berbasis Mikrokontroler ESP32. *JTEIN: Jurnal Teknik Elektro Indonesia*, 4(1). <https://doi.org/10.24036/jtein.v4i1.347>
- Bertoli, G. de C., Fernandes, G. V. C., Monici, P. H. B., Guibo, C. H. de A., Dos Santos, A. L., & Pereira Junior, L. A. (2024). Design and implementation of intelligent packet filtering in IoT microcontroller-based devices. *Journal of Internet Services and Applications*, 15(1). <https://doi.org/10.5753/jisa.2024.3835>
- Duguma, A. L., & Bai, X. (2025). How the internet of things technology improves agricultural efficiency. *Artificial Intelligence Review*, 58(2). <https://doi.org/10.1007/s10462-024-11046-0>
- Espino, C. S., Vargas, F. G., Paiva-Peredo, E., & Segura, G. W. Z. (2024). Internet of things and radio frequency identification based embedded system to reduce shopping time in supermarkets. *Bulletin of Electrical Engineering and Informatics*, 13(4). <https://doi.org/10.11591/eei.v13i4.7343>



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Esposito, M., Belli, A., Palma, L., & Pierleoni, P. (2023). Design and Implementation of a Framework for Smart Home Automation Based on Cellular IoT, MQTT, and Serverless Functions. *Sensors*, 23(9). <https://doi.org/10.3390/s23094459>

Gnana, G. S., Karthikeyan, A., Karthikeyan, K., Sanjeevikumar, P., Karapparambil Thomas, S., & Babu, A. (2024). Critical review Of SCADA And PLC in smart buildings and energy sector. In *Energy Reports* (Vol. 12). <https://doi.org/10.1016/j.egyr.2024.07.041>

Jesus, B., Lins, F., & Laranjeiro, N. (2025). An approach to assess robustness of MQTT-based IoT systems. *Internet of Things (The Netherlands)*, 31. <https://doi.org/10.1016/j.iot.2025.101590>

Kachman, O., Malík, P., Baláz, M., Majer, L., & Gyepes, G. (2025). A Lightweight and Configurable Flash Filesystem for Low-Power Devices. *Journal of Low Power Electronics and Applications*, 15(2). <https://doi.org/10.3390/jlpea15020022>

Kusuma, J. F., Rifa'i, M., & Saukani, I. (2024). Implementasi Protokol Komunikasi Modbus Untuk Mini Scada Pada Plant Pengisian Serbuk Temulawak. *Mutiara: Multidiciplinary Scientifict Journal*, 2(5). <https://doi.org/10.57185/mutiara.v2i5.182>

Nadhiroh, N., Wardhany, A. K., Setiana, H., Renaldy, R., Putri, A. A., & Handayani, M. D. (2024). Penyiram Tanaman Hidroponik Otomatis Berbasis IoT Dengan PLC Outseal Dan ESP32. *ELECTRICES*, 6(1). <https://doi.org/10.32722/ees.v6i1.6361>

Nurahmat, H., & Suryo, Y. A. (2025). Implementation of Outseal PLC with RS-485 as a Multi Node in Chicken Farming Using Raspberry Pi and Node RED. *G-Tech: Jurnal Teknologi Terapan*, 9(4). <https://doi.org/10.70609/g-tech.v9i4.8155>

Priyatna, I. A., Darmawan, B., & Paniran. (2024). Rancang Bangun Sistem Monitoring Parkir Mobil Indoor Dengan Wireless Sensor Network Menggunakan Nodemcu Esp8266 Berbasis Internet Of Things.



Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

DIELEKTRIKA, 11(2). <https://doi.org/10.29303/dielektrika.v11i2.366>

Rafique, W. (2025). ML-RASPF: A Machine Learning-Based Rate-Adaptive Framework for Dynamic Resource Allocation in Smart Healthcare IoT. *Algorithms*, 18(6). <https://doi.org/10.3390/a18060325>

Tsaqief, M. F., & Sutopo, J. (2025). Comparative Performance Analysis Between the MQTT and WebSocket Protocols. *Bit-Tech*, 8(2). <https://doi.org/10.32877/bt.v8i2.3223>

Xu, J., Fan, L., Du, J., Wang, X., Qin, L., Ding, X., Wang, Z., & Tao, Y. (2024). Research on the Application of Littlefs and FATFS File Systems in Smart IoT Electricity Meters. *2024 IEEE 4th International Conference on Power, Electronics and Computer Applications, ICPECA 2024*. <https://doi.org/10.1109/ICPECA60615.2024.10471164>

Yanyachi, P. R., Mendoza-Chok, J., Espinoza-Garcia, B., Cutipa Luque, J. C., & Yanyachi Aco Cardenas, D. (2025). OpenNavSense platform: A low-cost, open-source inertial navigation system for the evaluation of estimation algorithms. *HardwareX*, 21. <https://doi.org/10.1016/j.ohx.2024.e00621>

**POLITEKNIK
NEGERI
JAKARTA**

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

LAMPIRAN

Lampiran 1. Daftar Riwayat Hidup



Radhifka Defandy Putra

Lahir di Bekasi pada tanggal 15 Maret 2005. Lulus dari SDIT Assalam Terpadu pada tahun 2017, SMPN 3 Tambun Utara pada tahun 2020, dan SMKS Karya Guna Bhakti 1 pada tahun 2023. Kemudian melanjutkan kuliah dengan Program Studi Teknik Elektronika Industri, Teknik Elektro di Politeknik Negeri Jakarta pada tahun 2023

**POLITEKNIK
NEGERI
JAKARTA**



Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Lampiran 2. Datasheet ESP32

Table 1 provides the specifications of ESP-32S.

Table 1 ESP-32S Specifications

Categories	Items	Values
WiFi	Standards	
	Protocols	802.11 b/g/n/d/e/i/k/r (802.11n up to 150 Mbps)
	Frequency Range	2.4GHz-2.5GHz (2400M-2483.5M)
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and BLE specification
	Radio	NZIF receiver with -98 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH
	Audio	CVSD and SBC
Hardware	Module interface	SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, I2C, IR
		GPIO, capacitive touch sensor, ADC, DAC, LNA pre-amplifier
	On-chip sensor	3.0~3.6V
	On-board clock	Average value: 80mA
	Operating voltage	-40~125°
	Operating current	Normal temperature
	Operating temperature range	14.3mm*24.8mm*3mm
	Ambient temperature range	N/A
Software	Package size	
	Wi-Fi mode	Station/softAP/SoftAP+station/P2P
	Security	WPA/WPA2/WPA2-Enterprise/WPS
	Encryption	AES/RS4/ECC/SHA
	Firmware Upgrade	UART Download / OTA (via network) / download and write firmware via host

Shenzhen Anxinke Technology CO.,LTD

<http://www.ai-thinker.com>

6

Lampiran 3. Datasheet MAX485



MAX481/MAX483/MAX485/ MAX487-MAX491/MAX1487

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

General Description

The MAX481, MAX483, MAX485, MAX487-MAX491, and MAX1487 are low-power transceivers for RS-485 and RS-422 communication. Each part contains one driver and one receiver. The MAX483, MAX487, MAX488, and MAX489 feature reduced slew-rate drivers that minimize EMI and reduce reflections caused by improperly terminated cables, thus allowing error-free data transmission up to 250kbps. The driver slew rates of the MAX481, MAX485, MAX490, MAX491, and MAX1487 are not limited, allowing them to transmit up to 2.5Mbps.

These transceivers draw between 120µA and 500µA of supply current when unloaded or fully loaded with disabled drivers. Additionally, the MAX481, MAX483, and MAX487 have a low-current shutdown mode in which they consume only 0.1µA. All parts operate from a single 5V supply. Drivers are short-circuit current limited and are protected against excessive power dissipation by thermal shutdown circuitry that places the driver outputs into a high-impedance state. The receiver input has a fail-safe feature that guarantees a logic-high output if the input is open circuit.

The MAX487 and MAX1487 feature quarter-unit-load receiver input impedance, allowing up to 128 MAX487/MAX1487 transceivers on the bus. Full-duplex communications are obtained using the MAX488-MAX491, while the MAX481, MAX483, MAX485, MAX487, and MAX1487 are designed for half-duplex applications.

Applications

Low-Power RS-485 Transceivers
Low-Power RS-422 Transceivers
Level Translators
Transceivers for EMI-Sensitive Applications
Industrial-Control Local Area Networks

Next Generation Device Features

- For Fault-Tolerant Applications
MAX3430: ±80V Fault-Protected, Fail-Safe, 1/4 Unit Load, +3.3V, RS-485 Transceiver
MAX3440E-MAX3444E: ±15kV ESD-Protected, ±60V Fault-Protected, 10Mbps, Fail-Safe, RS-485/1708 Transceivers
- For Space-Constrained Applications
MAX3460-MAX3464: ±5V, Fail-Safe, 20Mbps, Profibus RS-485/RS-422 Transceivers
MAX3362: ±3.3V, High-Speed, RS-485/RS-422 Transceiver in a SOT23 Package
MAX3280E-MAX3284E: ±15kV ESD-Protected, 52Mbps, ±3V to +5.5V, SOT23, RS-485/RS-422, True Fail-Safe Receivers
MAX3293/MAX3294/MAX3295: 20Mbps, ±3.3V, SOT23, RS-485/RS-422 Transmitters
- For Multiple Transceiver Applications
MAX3000E-MAX3003E: ±15kV ESD-Protected, ±3.3V, Quad RS-422 Transmitters
- For Fail-Safe Applications
MAX3080-MAX3089: Fail-Safe, High-Speed (10Mbps), Slew-Rate-Limited RS-485/RS-422 Transceivers
- For Low-Voltage Applications
MAX3483E/MAX3485E/MAX3486E/MAX3488E/MAX3490E/MAX3491E: ±3.3V Powered, ±15kV ESD-Protected, 12Mbps, Slew-Rate-Limited, True RS-485/RS-422 Transceivers

Ordering Information appears at end of data sheet.

Selection Table

PART NUMBER	HALF/FULL DUPLEX	DATA RATE (Mbps)	SLEW-RATE LIMITED	LOW-POWER SHUTDOWN	RECEIVER/DRIVER ENABLE	QUIESCENT CURRENT (µA)	NUMBER OF RECEIVERS ON BUS	PIN COUNT
MAX481	Half	2.5	No	Yes	Yes	300	32	8
MAX483	Half	0.25	Yes	Yes	Yes	120	32	8
MAX485	Half	2.5	No	No	Yes	300	32	8
MAX487	Half	0.25	Yes	Yes	Yes	120	128	8
MAX488	Full	0.25	Yes	No	No	120	32	8
MAX489	Full	0.25	Yes	No	Yes	120	32	14
MAX490	Full	2.5	No	No	No	300	32	8
MAX491	Full	2.5	No	No	Yes	300	32	14
MAX1487	Half	2.5	No	No	Yes	230	128	8

For pricing, delivery, and ordering information, please contact Maxim Direct at 1-888-629-4642, or visit Maxim Integrated's website at www.maximintegrated.com.

19-0122; Rev 10; 9/14

Lampiran 4. Poster Alat



**RANCANG BANGUN MODUL LATHI
AUTOMATIC BOTTLE FILLING AND
CAPPING SYSTEM BERBASIS PLC
DENGAN MONITORING SCADA**

Revaldi Kusnadi 2303321011
Rachifika Defandy Putra 2303321069
Raihan Mubaraq Purba 2303321014
Tugas Akhir Elektronika Industri

LATAR BELAKANG

Proses pengisian cairan ke dalam botol secara manual sering kali tidak konsisten dan menyebabkan banyak variasi dalam produksi. Selain itu, proses pengisian air dan penutupan botol secara manual juga dapat membuat waktu yang dibutuhkan menjadi lama. Tanpa adanya sistem monitoring SCADA, pengawasan menjadi sangat terbatas. Oleh karena itu dibutuhkan sistem yang dapat melakukan Pengisian cairan (Filling) serta penutupan botol (Capping) secara otomatis, dan dilengkapi dengan monitoring SCADA IoT untuk keperluan supervisi yang dapat dilakukan dari lokal maupun dari jarak jauh (remote).

CARA KERJA ALAT

Proses kerja dimulai ketika tombol start ditekan, baik melalui panel modul lath maupun antarmuka SCADA. Motor konveyor akan menyala dan bergerak hingga sensor Proximity Start mendeteksi keberadaan botol pada sistem. Ketika botol mencapai bagian filling, sensor Proximity Filling akan mendeteksi botol tersebut, yang kemudian memicu motor konveyor untuk berhenti. Pompa cairan (pompa) selanjutnya akan menyedot, sementara Flowmeter secara otomatis menghitung volume air yang mengalir. Setelah target pengisian tercapai, pompa akan mati dan konveyor kembali berjalan.

Botol kemudian bergerak melalui cap storage, di mana mekanisme secara otomatis akan melakukan tutup ke atas botol. Setelah itu, di stasiun capping, sensor Capping Proximity mendeteksi posisi botol dan menghentikan konveyor. Aktuator linear kemudian bergerak untuk membuka motor capping, dan motor konveyor untuk menggerakkan tutup botol. Setelah proses pengisian selesai, aktuator linear kembali naik, motor capping berhenti, dan konveyor kembali berjalan. Pada tahap akhir, sensor Proximity Counter mendeteksi botol yang telah selesai proses, menghentikan konveyor, serta menambahkan nilai 1 pada memori Total Production.

Secara bersamaan dari sisi komunikasi data, mikrokontroler ESP32 secara kontinu melakukan polling ke Modbus Slave pada PLC untuk mendeteksi perubahan nilai pada register, jika terdapat perubahan data, nilai dari register tersebut akan dikumpulkan ke dalam satu paket format JSON (misalnya, "Parameter": "nilai register"). Data tersebut kemudian dikirimkan ke broker menggunakan metode publish melalui protokol MQTT. Di sisi penerima, sistem SCADA akan membaca paket data tersebut dan memberikan nilai pada variabel yang bersangkutan. Alur komunikasi yang sama juga berlaku secara dua arah (bidirectional) untuk pengaman perintah dari SCADA ke PLC.

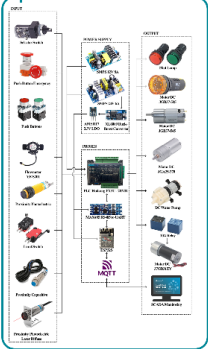
TUJUAN

Membangun Sistem otomatis untuk melakukan pengisian cairan (Filling) serta penutupan botol (Capping) sebagai modul lath, dilengkapi dengan sistem SCADA yang dapat dioperasikan secara IoT sehingga dapat melakukan supervisi dan kontrol secara lokal maupun jarak jauh.

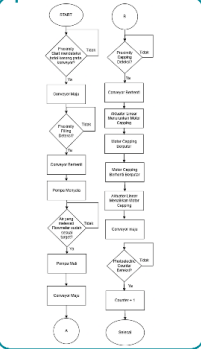
SPEKIFIKASI

PLC: Hologram FX3U - 28MR
SCADA: HMI Cloud SCADA 3
Komunikasi: Modbus RTU, MQTT
Software: GX Works2, HMI Cloud SCADA Develop 3

BLOK DIAGRAM



FLOWCHART ALAT



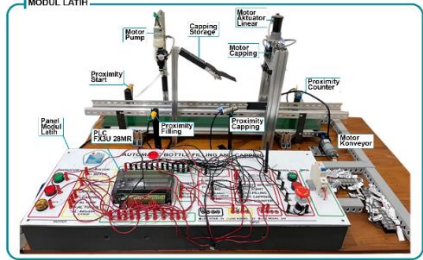
Lampiran 5. SOP Alat



**RANCANG BANGUN MODUL LATHI AUTOMATIC
BOTTLE FILLING AND CAPPING SYSTEM
BERBASIS PLC DENGAN MONITORING SCADA**

SOP ALAT

MODUL LATH



ALAT DAN BAHAN

- Modul Lath Automatic Bottle Filling and Capping
- Kabel Jumper
- Laptop/PC
- Jaringan Internet
- Broker MQTT

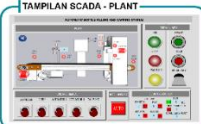
DIRANCANG OLEH

Revaldi Kusnadi 2303321011
Rachifika Defandy Putra 2303321069
Raihan Mubaraq Purba 2303321014
Penyemir: Rikdam Firy Muzakki, S.Pd., M.T. NIP. 199311082624081001

CARA PENGGUNAAN ALAT

1. Hubungkan Modul Lath ke Catu daya 220V
2. Lakukan Wiring sesuai dengan jobsheet.
3. Lakukan Program PLC sesuai dengan Jobsheet dan upload program ke PLC.
4. Hidupkan Modul lath dengan menghidupkan MCB.
5. Pastikan selektor ada pada mode Auto.
6. Tekan tombol Start untuk memulai dan Stop untuk menghentikan.
7. Tekan tombol Emergency ketika ada keadaan darurat.
8. Ubah selektor ke mode Manual untuk mode jogging, lakukan test kontrol setiap aktuator melalui SCADA.

TAMPLAN SCADA - PLANT



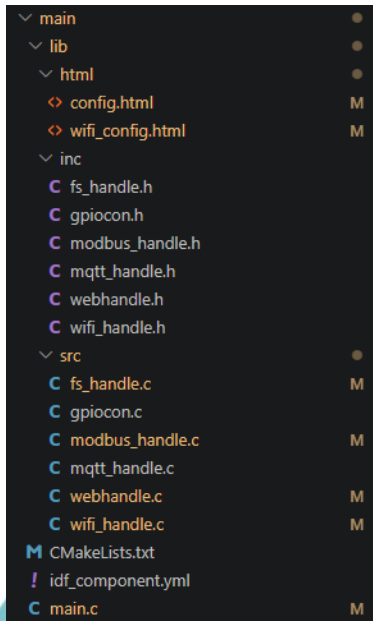
CARA MENYHUBUNGKAN KE SCADA

1. Hubungkan RS 485 plus dan minus dari PLC ke ESP32.
2. Konfigurasi Wifi dengan pergi ke <http://192.168.4.1>
3. Konfigurasi MQTT, Modbus dan Mapping dengan pergi ke <http://imgateway.local>
4. Buat project SCADA sesuai pada jobsheet.
5. Mulai SCADA.

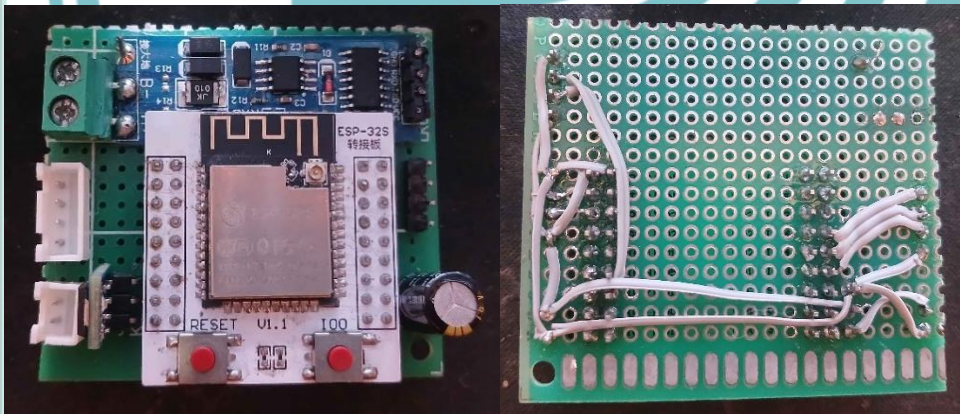
TAMPLAN SCADA - PLANT



Lampiran 6. Struktur Program



Lampiran 7. Foto Alat

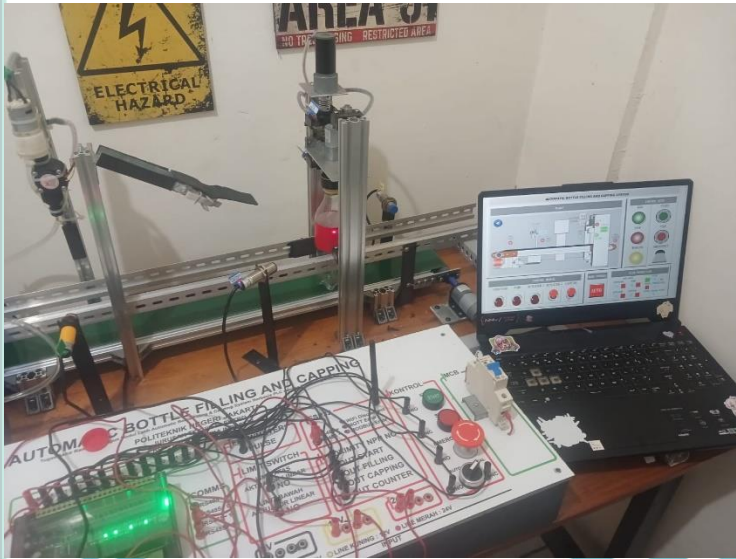


- Hak Cipta :**
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
 2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



© Hak Cipta milik Politeknik Negeri Jakarta

Lampiran 8. Pengujian Alat



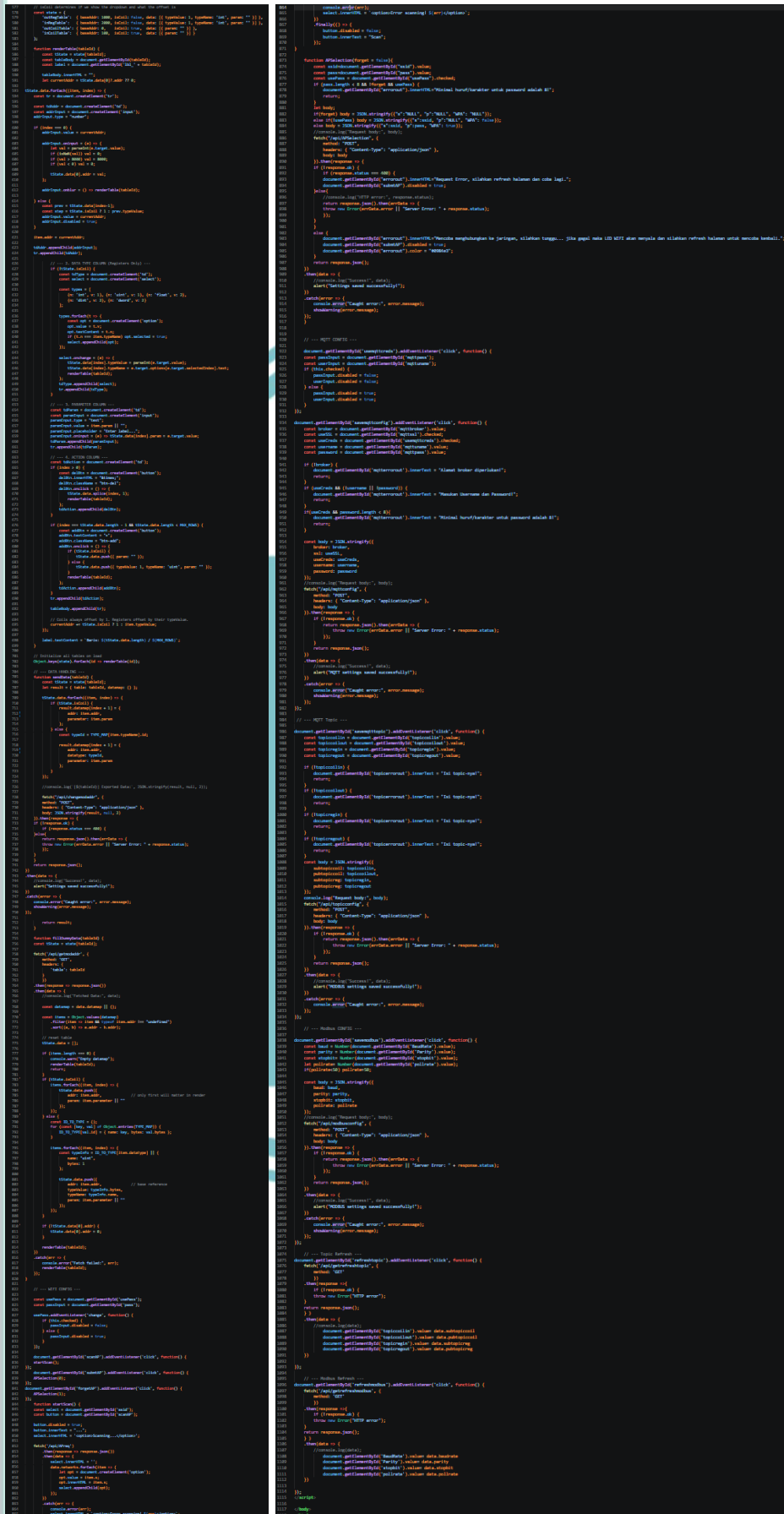
Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

POLITEKNIK
NEGERI
JAKARTA

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Lampiran 10. Program Modbus MQTT Gateway – wifi_config.html

```

<!-- HTML header for the page -->
<!-- Title -->
<!-- Meta tags -->
<!-- Stylesheets -->
<!-- Scripts -->
<!-- Body -->
<!-- Footer -->
<!-- End of page -->

```

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengemukakan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Lampiran 11. Program Modbus MQTT Gateway – main.c

```

1 #include <string.h>
2 #include "freertos/FreeRTOS.h"
3 #include "freertos/task.h"
4 #include "esp_log.h"
5 #include "nvs_flash.h"
6 #include "lwip/err.h"
7 #include "lwip/sys.h"
8 #include "webhandle.h"
9 #include "wifi_handle.h"
10 #include "pinconf.h"
11 #include "mqtt_handle.h"
12 #include "modbus_handle.h"
13 #include "esp_heap_caps.h"
14 #include "fs_handle.h"
15
16 //static const char *TAG = "MEM";
17 //static const char *TAG = "Main";
18
19 void app_main(void){
20     esp_err_t ret = nvs_flash_init();
21     if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
22         ESP_ERROR_CHECK(nvs_flash_erase());
23         ret = nvs_flash_init();
24     }
25     ESP_ERROR_CHECK(ret);
26     ESP_ERROR_CHECK(esp_netif_init());
27     ESP_ERROR_CHECK(esp_event_loop_create_default());
28     esp_netif_create_default_wifi_sta();
29
30     esp_log_level_set("ESP_LOG_WARN");
31     wifi_event_init();
32     wifi_init();
33     init_littlefs();
34     server_init();
35     wificonfig_serveron();
36     gpio_init();
37     init_modbus_config();
38     xTaskCreate(modbus_polling_task, "modbus_task", 4096, NULL, 5, NULL);
39 }

```

Lampiran 12. Program Modbus MQTT Gateway – fs_handle.c dan fs_handle.h

```

1 #include "esp_littlefs.h"
2 #include "esp_log.h"
3 #include "fs_handle.h"
4 #include "nvs.h"
5
6 static const char *TAG = "FS_Handle";
7
8 void init_littlefs(void){
9     esp_err_t ret = nvs_flash_init();
10     if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
11         ESP_ERROR_CHECK(nvs_flash_erase());
12         ret = nvs_flash_init();
13     }
14     ESP_ERROR_CHECK(ret);
15
16     esp_err_t ret = esp_littlefs_register_backend();
17     ESP_ERROR_CHECK(ret);
18
19     void setupFile(const char * filename, const void * data, size_t size) {
20         char path[64];
21         sprintf(path, "%s/%s", "littlefs/h", filename);
22         FILE * f = fopen(path, "w");
23         if (f) {
24             ESP_LOG1(TAG, "Failed to open file for writing");
25             return;
26         }
27         if (fdata) {
28             ESP_LOG1(TAG, "Null data");
29             return;
30         }
31         fwrite(data, size, 1, f);
32         fclose(f);
33     }
34
35     void readfile(const char * filename, void * data, size_t size) {
36         char path[64];
37         sprintf(path, "%s/%s", "littlefs/h", filename);
38         if (fdata) {
39             ESP_LOG1(TAG, "Null data");
40             return;
41         }
42         FILE * f = fopen(path, "r");
43         if (f) {
44             ESP_LOG1(TAG, "Failed to open file for reading");
45             return;
46         }
47         if (fdata) {
48             ESP_LOG1(TAG, "Null data");
49             return;
50         }
51         fread(data, size, 1, f);
52         fclose(f);
53     }
54
55     void remove(const char * key, void * data, size_t size) {
56         nvs_handle_t nvs;
57         esp_err_t ret = nvs_open("storage", NVS_READWRITE, &nvs);
58         if (ret != ESP_OK) {
59             ESP_LOG1(TAG, "NVS open fail");
60             return;
61         }
62         ret = nvs_get_blob(key, data, size);
63         if (ret != ESP_OK) {
64             ESP_LOG1(TAG, "NVS get fail");
65             return;
66         }
67         nvs_close(nvs);
68     }
69
70     void readnvs(const char * key, void * data, size_t size) {
71         nvs_handle_t nvs;
72         esp_err_t ret = nvs_open("storage", NVS_READWRITE, &nvs);
73         if (ret != ESP_OK) {
74             ESP_LOG1(TAG, "NVS open fail");
75             return;
76         }
77         nvs_get_blob(key, data, size);
78         if (ret != ESP_OK) {
79             ESP_LOG1(TAG, "NVS get fail");
80             return;
81         }
82         nvs_close(nvs);
83     }
84
85     void writenvs(const char * key) {
7
86         nvs_handle_t nvs;
87         esp_err_t ret = nvs_open("storage", NVS_READWRITE, &nvs);
88         if (ret != ESP_OK) {
89             ESP_LOG1(TAG, "NVS open fail");
90             return;
91         }
92         ret = nvs_set_blob(key, data, size);
93         if (ret != ESP_OK) {
94             ESP_LOG1(TAG, "NVS set fail");
95             return;
96         }
97         ret = nvs_commit(nvs);
98         if (ret != ESP_OK) {
99             ESP_LOG1(TAG, "NVS commit fail");
100             return;
101         }
102         nvs_close(nvs);
103     }
104
105     void removeFile(const char * filename) {
106         char path[64];
107         sprintf(path, "%s/%s", "littlefs/h", filename);
108         if (fdata) {
109             ESP_LOG1(TAG, "Null data");
110             return;
111         }
112         FILE * f = fopen(path, "r");
113         if (f) {
114             ESP_LOG1(TAG, "Failed to open file for reading");
115             return;
116         }
117         if (fdata) {
118             ESP_LOG1(TAG, "Null data");
119             return;
120         }
121         fclose(f);
122     }
123
124     void readfile(const char * filename, void * data, size_t size) {
125         char path[64];
126         sprintf(path, "%s/%s", "littlefs/h", filename);
127         FILE * f = fopen(path, "r");
128         if (f) {
129             ESP_LOG1(TAG, "Failed to open file for reading");
130             return;
131         }
132         fread(data, size, 1, f);
133         fclose(f);
134     }
135
136     void setupFile(const char * filename, const void * data, size_t size) {
137         char path[64];
138         sprintf(path, "%s/%s", "littlefs/h", filename);
139         FILE * f = fopen(path, "w");
140         if (f) {
141             ESP_LOG1(TAG, "Failed to open file for writing");
142             return;
143         }
144         fwrite(data, size, 1, f);
145         fclose(f);
146     }
147
148     void readfile(const char * filename, void * data, size_t size) {
149         char path[64];
150         sprintf(path, "%s/%s", "littlefs/h", filename);
151         FILE * f = fopen(path, "r");
152         if (f) {
153             ESP_LOG1(TAG, "Failed to open file for reading");
154             return;
155         }
156         fread(data, size, 1, f);
157         fclose(f);
158     }
159
160     void removeFile(const char * filename) {
161         char path[64];
162         sprintf(path, "%s/%s", "littlefs/h", filename);
163         if (fdata) {
164             ESP_LOG1(TAG, "Null data");
165             return;
166         }
167         FILE * f = fopen(path, "r");
168         if (f) {
169             ESP_LOG1(TAG, "Failed to open file for reading");
170             return;
171         }
172         fclose(f);
173     }
174
175     void readfile(const char * filename, void * data, size_t size) {
176         char path[64];
177         sprintf(path, "%s/%s", "littlefs/h", filename);
178         FILE * f = fopen(path, "r");
179         if (f) {
180             ESP_LOG1(TAG, "Failed to open file for reading");
181             return;
182         }
183         fread(data, size, 1, f);
184         fclose(f);
185     }
186
187     void setupFile(const char * filename, const void * data, size_t size) {
188         char path[64];
189         sprintf(path, "%s/%s", "littlefs/h", filename);
190         FILE * f = fopen(path, "w");
191         if (f) {
192             ESP_LOG1(TAG, "Failed to open file for writing");
193             return;
194         }
195         fwrite(data, size, 1, f);
196         fclose(f);
197     }
198
199     void readfile(const char * filename, void * data, size_t size) {
200         char path[64];
201         sprintf(path, "%s/%s", "littlefs/h", filename);
202         FILE * f = fopen(path, "r");
203         if (f) {
204             ESP_LOG1(TAG, "Failed to open file for reading");
205             return;
206         }
207         fread(data, size, 1, f);
208         fclose(f);
209     }
210
211     void removeFile(const char * filename) {
212         char path[64];
213         sprintf(path, "%s/%s", "littlefs/h", filename);
214         if (fdata) {
215             ESP_LOG1(TAG, "Null data");
216             return;
217         }
218         FILE * f = fopen(path, "r");
219         if (f) {
220             ESP_LOG1(TAG, "Failed to open file for reading");
221             return;
222         }
223         fclose(f);
224     }
225
226     void readfile(const char * filename, void * data, size_t size) {
227         char path[64];
228         sprintf(path, "%s/%s", "littlefs/h", filename);
229         FILE * f = fopen(path, "r");
230         if (f) {
231             ESP_LOG1(TAG, "Failed to open file for reading");
232             return;
233         }
234         fread(data, size, 1, f);
235         fclose(f);
236     }
237
238     void setupFile(const char * filename, const void * data, size_t size) {
239         char path[64];
240         sprintf(path, "%s/%s", "littlefs/h", filename);
241         FILE * f = fopen(path, "w");
242         if (f) {
243             ESP_LOG1(TAG, "Failed to open file for writing");
244             return;
245         }
246         fwrite(data, size, 1, f);
247         fclose(f);
248     }
249
250     void readfile(const char * filename, void * data, size_t size) {
251         char path[64];
252         sprintf(path, "%s/%s", "littlefs/h", filename);
253         FILE * f = fopen(path, "r");
254         if (f) {
255             ESP_LOG1(TAG, "Failed to open file for reading");
256             return;
257         }
258         fread(data, size, 1, f);
259         fclose(f);
260     }
261
262     void removeFile(const char * filename) {
263         char path[64];
264         sprintf(path, "%s/%s", "littlefs/h", filename);
265         if (fdata) {
266             ESP_LOG1(TAG, "Null data");
267             return;
268         }
269         FILE * f = fopen(path, "r");
270         if (f) {
271             ESP_LOG1(TAG, "Failed to open file for reading");
272             return;
273         }
274         fclose(f);
275     }
276
277     void readfile(const char * filename, void * data, size_t size) {
278         char path[64];
279         sprintf(path, "%s/%s", "littlefs/h", filename);
280         FILE * f = fopen(path, "r");
281         if (f) {
282             ESP_LOG1(TAG, "Failed to open file for reading");
283             return;
284         }
285         fread(data, size, 1, f);
286         fclose(f);
287     }
288
289     void setupFile(const char * filename, const void * data, size_t size) {
290         char path[64];
291         sprintf(path, "%s/%s", "littlefs/h", filename);
292         FILE * f = fopen(path, "w");
293         if (f) {
294             ESP_LOG1(TAG, "Failed to open file for writing");
295             return;
296         }
297         fwrite(data, size, 1, f);
298         fclose(f);
299     }
300
301     void readfile(const char * filename, void * data, size_t size) {
302         char path[64];
303         sprintf(path, "%s/%s", "littlefs/h", filename);
304         FILE * f = fopen(path, "r");
305         if (f) {
306             ESP_LOG1(TAG, "Failed to open file for reading");
307             return;
308         }
309         fread(data, size, 1, f);
310         fclose(f);
311     }
312
313     void removeFile(const char * filename) {
314         char path[64];
315         sprintf(path, "%s/%s", "littlefs/h", filename);
316         if (fdata) {
317             ESP_LOG1(TAG, "Null data");
318             return;
319         }
320         FILE * f = fopen(path, "r");
321         if (f) {
322             ESP_LOG1(TAG, "Failed to open file for reading");
323             return;
324         }
325         fclose(f);
326     }
327
328     void readfile(const char * filename, void * data, size_t size) {
329         char path[64];
330         sprintf(path, "%s/%s", "littlefs/h", filename);
331         FILE * f = fopen(path, "r");
332         if (f) {
333             ESP_LOG1(TAG, "Failed to open file for reading");
334             return;
335         }
336         fread(data, size, 1, f);
337         fclose(f);
338     }
339
340     void setupFile(const char * filename, const void * data, size_t size) {
341         char path[64];
342         sprintf(path, "%s/%s", "littlefs/h", filename);
343         FILE * f = fopen(path, "w");
344         if (f) {
345             ESP_LOG1(TAG, "Failed to open file for writing");
346             return;
347         }
348         fwrite(data, size, 1, f);
349         fclose(f);
350     }
351
352     void readfile(const char * filename, void * data, size_t size) {
353         char path[64];
354         sprintf(path, "%s/%s", "littlefs/h", filename);
355         FILE * f = fopen(path, "r");
356         if (f) {
357             ESP_LOG1(TAG, "Failed to open file for reading");
358             return;
359         }
360         fread(data, size, 1, f);
361         fclose(f);
362     }
363
364     void removeFile(const char * filename) {
365         char path[64];
366         sprintf(path, "%s/%s", "littlefs/h", filename);
367         if (fdata) {
368             ESP_LOG1(TAG, "Null data");
369             return;
370         }
371         FILE * f = fopen(path, "r");
372         if (f) {
373             ESP_LOG1(TAG, "Failed to open file for reading");
374             return;
375         }
376         fclose(f);
377     }
378
379     void readfile(const char * filename, void * data, size_t size) {
380         char path[64];
381         sprintf(path, "%s/%s", "littlefs/h", filename);
382         FILE * f = fopen(path, "r");
383         if (f) {
384             ESP_LOG1(TAG, "Failed to open file for reading");
385             return;
386         }
387         fread(data, size, 1, f);
388         fclose(f);
389     }
390
391     void setupFile(const char * filename, const void * data, size_t size) {
392         char path[64];
393         sprintf(path, "%s/%s", "littlefs/h", filename);
394         FILE * f = fopen(path, "w");
395         if (f) {
396             ESP_LOG1(TAG, "Failed to open file for writing");
397             return;
398         }
399         fwrite(data, size, 1, f);
400         fclose(f);
401     }
402
403     void readfile(const char * filename, void * data, size_t size) {
404         char path[64];
405         sprintf(path, "%s/%s", "littlefs/h", filename);
406         FILE * f = fopen(path, "r");
407         if (f) {
408             ESP_LOG1(TAG, "Failed to open file for reading");
409             return;
410         }
411         fread(data, size, 1, f);
412         fclose(f);
413     }
414
415     void removeFile(const char * filename) {
416         char path[64];
417         sprintf(path, "%s/%s", "littlefs/h", filename);
418         if (fdata) {
419             ESP_LOG1(TAG, "Null data");
420             return;
421         }
422         FILE * f = fopen(path, "r");
423         if (f) {
424             ESP_LOG1(TAG, "Failed to open file for reading");
425             return;
426         }
427         fclose(f);
428     }
429
430     void readfile(const char * filename, void * data, size_t size) {
431         char path[64];
432         sprintf(path, "%s/%s", "littlefs/h", filename);
433         FILE * f = fopen(path, "r");
434         if (f) {
435             ESP_LOG1(TAG, "Failed to open file for reading");
436             return;
437         }
438         fread(data, size, 1, f);
439         fclose(f);
440     }
441
442     void setupFile(const char * filename, const void * data, size_t size) {
443         char path[64];
444         sprintf(path, "%s/%s", "littlefs/h", filename);
445         FILE * f = fopen(path, "w");
446         if (f) {
447             ESP_LOG1(TAG, "Failed to open file for writing");
448             return;
449         }
450         fwrite(data, size, 1, f);
451         fclose(f);
452     }
453
454     void readfile(const char * filename, void * data, size_t size) {
455         char path[64];
456         sprintf(path, "%s/%s", "littlefs/h", filename);
457         FILE * f = fopen(path, "r");
458         if (f) {
459             ESP_LOG1(TAG, "Failed to open file for reading");
460             return;
461         }
462         fread(data, size, 1, f);
463         fclose(f);
464     }
465
466     void removeFile(const char * filename) {
467         char path[64];
468         sprintf(path, "%s/%s", "littlefs/h", filename);
469         if (fdata) {
470             ESP_LOG1(TAG, "Null data");
471             return;
472         }
473         FILE * f = fopen(path, "r");
474         if (f) {
475             ESP_LOG1(TAG, "Failed to open file for reading");
476             return;
477         }
478         fclose(f);
479     }
480
481     void readfile(const char * filename, void * data, size_t size) {
482         char path[64];
483         sprintf(path, "%s/%s", "littlefs/h", filename);
484         FILE * f = fopen(path, "r");
485         if (f) {
486             ESP_LOG1(TAG, "Failed to open file for reading");
487             return;
488         }
489         fread(data, size, 1, f);
490         fclose(f);
491     }
492
493     void setupFile(const char * filename, const void * data, size_t size) {
494         char path[64];
495         sprintf(path, "%s/%s", "littlefs/h", filename);
496         FILE * f = fopen(path, "w");
497         if (f) {
498             ESP_LOG1(TAG, "Failed to open file for writing");
499             return;
500         }
501         fwrite(data, size, 1, f);
502         fclose(f);
503     }
504
505     void readfile(const char * filename, void * data, size_t size) {
506         char path[64];
507         sprintf(path, "%s/%s", "littlefs/h", filename);
508         FILE * f = fopen(path, "r");
509         if (f) {
510             ESP_LOG1(TAG, "Failed to open file for reading");
511             return;
512         }
513         fread(data, size, 1, f);
514         fclose(f);
515     }
516
517     void removeFile(const char * filename) {
518         char path[64];
519         sprintf(path, "%s/%s", "littlefs/h", filename);
520         if (fdata) {
521             ESP_LOG1(TAG, "Null data");
522             return;
523         }
524         FILE * f = fopen(path, "r");
525         if (f) {
526             ESP_LOG1(TAG, "Failed to open file for reading");
527             return;
528         }
529         fclose(f);
530     }
531
532     void readfile(const char * filename, void * data, size_t size) {
533         char path[64];
534         sprintf(path, "%s/%s", "littlefs/h", filename);
535         FILE * f = fopen(path, "r");
536         if (f) {
537             ESP_LOG1(TAG, "Failed to open file for reading");
538             return;
539         }
540         fread(data, size, 1, f);
541         fclose(f);
542     }
543
544     void setupFile(const char * filename, const void * data, size_t size) {
545         char path[64];
546         sprintf(path, "%s/%s", "littlefs/h", filename);
547         FILE * f = fopen(path, "w");
548         if (f) {
549             ESP_LOG1(TAG, "Failed to open file for writing");
550             return;
551         }
552         fwrite(data, size, 1, f);
553         fclose(f);
554     }
555
556     void readfile(const char * filename, void * data, size_t size) {
557         char path[64];
558         sprintf(path, "%s/%s", "littlefs/h", filename);
559         FILE * f = fopen(path, "r");
560         if (f) {
561             ESP_LOG1(TAG, "Failed to open file for reading");
562             return;
563         }
564         fread(data, size, 1, f);
565         fclose(f);
566     }
567
568     void removeFile(const char * filename) {
569         char path[64];
570         sprintf(path, "%s/%s", "littlefs/h", filename);
571         if (fdata) {
572             ESP_LOG1(TAG, "Null data");
573             return;
574         }
575         FILE * f = fopen(path, "r");
576         if (f) {
577             ESP_LOG1(TAG, "Failed to open file for reading");
578             return;
579         }
580         fclose(f);
581     }
582
583     void readfile(const char * filename, void * data, size_t size) {
584         char path[64];
585         sprintf(path, "%s/%s", "littlefs/h", filename);
586         FILE * f = fopen(path, "r");
587         if (f) {
588             ESP_LOG1(TAG, "Failed to open file for reading");
589             return;
590         }
591         fread(data, size, 1, f);
592         fclose(f);
593     }
594
595     void setupFile(const char * filename, const void * data, size_t size) {
596         char path[64];
597         sprintf(path, "%s/%s", "littlefs/h", filename);
598         FILE * f = fopen(path, "w");
599         if (f) {
600             ESP_LOG1(TAG, "Failed to open file for writing");
601             return;
602         }
603         fwrite(data, size, 1, f);
604         fclose(f);
605     }
606
607     void readfile(const char * filename, void * data, size_t size) {
608         char path[64];
609         sprintf(path, "%s/%s", "littlefs/h", filename);
610         FILE * f = fopen(path, "r");
611         if (f) {
612             ESP_LOG1(TAG, "Failed to open file for reading");
613             return;
614         }
615         fread(data, size, 1, f);
616         fclose(f);
617     }
618
619     void removeFile(const char * filename) {
620         char path[64];
621         sprintf(path, "%s/%s", "littlefs/h", filename);
622         if (fdata) {
623             ESP_LOG1(TAG, "Null data");
624             return;
625         }
626         FILE * f = fopen(path, "r");
627         if (f) {
628             ESP_LOG1(TAG, "Failed to open file for reading");
629             return;
630         }
631         fclose(f);
632     }
633
634     void readfile(const char * filename, void * data, size_t size) {
635         char path[64];
636         sprintf(path, "%s/%s", "littlefs/h", filename);
637         FILE * f = fopen(path, "r");
638         if (f) {
639             ESP_LOG1(TAG, "Failed to open file for reading");
640             return;
641         }
642         fread(data, size, 1, f);
643         fclose(f);
644     }
645
646     void setupFile(const char * filename, const void * data, size_t size) {
647         char path[64];
648         sprintf(path, "%s/%s", "littlefs/h", filename);
649         FILE * f = fopen(path, "w");
650         if (f) {
651             ESP_LOG1(TAG, "Failed to open file for writing");
652             return;
653         }
654         fwrite(data, size, 1, f);
655         fclose(f);
656     }
657
658     void readfile(const char * filename, void * data, size_t size) {
659         char path[64];
660         sprintf(path, "%s/%s", "littlefs/h", filename);
661         FILE * f = fopen(path, "r");
662         if (f) {
663             ESP_LOG1(TAG, "Failed to open file for reading");
664             return;
665         }
666         fread(data, size, 1, f);
667         fclose(f);
668     }
669
670     void removeFile(const char * filename) {
671         char path[64];
672         sprintf(path, "%s/%s", "littlefs/h", filename);
673         if (fdata) {
674             ESP_LOG1(TAG, "Null data");
675             return;
676         }
677         FILE * f = fopen(path, "r");
678         if (f) {
679             ESP_LOG1(TAG, "Failed to open file for reading");
680             return;
681         }
682         fclose(f);
683     }
684
685     void readfile(const char * filename, void * data, size_t size) {
686         char path[64];
687         sprintf(path, "%s/%s", "littlefs/h", filename);
688         FILE * f = fopen(path, "r");
689         if (f) {
690             ESP_LOG1(TAG, "Failed to open file for reading");
691             return;
692         }
693         fread(data, size, 1, f);
694         fclose(f);
695     }
696
697     void setupFile(const char * filename, const void * data, size_t size) {
698         char path[64];
699         sprintf(path, "%s/%s", "littlefs/h", filename);
700         FILE * f = fopen(path, "w");
701         if (f) {
702             ESP_LOG1(TAG, "Failed to open file for writing");
703             return;
704         }
705         fwrite(data, size, 1, f);
706         fclose(f);
707     }
708
709     void readfile(const char * filename, void * data, size_t size) {
710         char path[64];
711         sprintf(path, "%s/%s", "littlefs/h", filename);
712         FILE * f = fopen(path, "r");
713         if (f) {
714             ESP_LOG1(TAG, "Failed to open file for reading");
715             return;
716         }
717         fread(data, size, 1, f);
718         fclose(f);
719     }
720
721     void removeFile(const char * filename) {
722         char path[64];
723         sprintf(path, "%s/%s", "littlefs/h", filename);
724         if (fdata) {
725             ESP_LOG1(TAG, "Null data");
726             return;
727         }
728         FILE * f = fopen(path, "r");
729         if (f) {
730             ESP_LOG1(TAG, "Failed to open file for reading");
731             return;
732         }
733         fclose(f);
734     }
735
736     void readfile(const char * filename, void * data, size_t size) {
737         char path[64];
738         sprintf(path, "%s/%s", "littlefs/h", filename);
739         FILE * f = fopen(path, "r");
740         if (f) {
741             ESP_LOG1(TAG, "Failed to open file for reading");
742             return;
743         }
744         fread(data, size, 1, f);
745         fclose(f);
746     }
747
748     void setupFile(const char * filename, const void * data, size_t size) {
749         char path[64];
750         sprintf(path, "%s/%s", "littlefs/h", filename);
751         FILE * f = fopen(path, "w");
752         if (f) {
753             ESP_LOG1(TAG, "Failed to open file for writing");
754             return;
755         }
756         fwrite(data, size, 1, f);
757         fclose(f);
758     }
759
760     void readfile(const char * filename, void * data, size_t size) {
761         char path[64];
762         sprintf(path, "%s/%s", "littlefs/h", filename);
763         FILE * f = fopen(path, "r");
764         if (f) {
765             ESP_LOG1(TAG, "Failed to open file for reading");
766             return;
767         }
768         fread(data, size, 1, f);
769         fclose(f);
770     }
771
772     void removeFile(const char * filename) {
773         char path[64];
774         sprintf(path, "%s/%s", "littlefs/h", filename);
775         if (fdata) {
776             ESP_LOG1(TAG, "Null data");
777             return;
778         }
779         FILE * f = fopen(path, "r");
780         if (f) {
781             ESP_LOG1(TAG, "Failed to open file for reading");
782             return;
783         }
784         fclose(f);
785     }
786
787     void readfile(const char * filename, void * data, size_t size) {
788         char path[64];
789         sprintf(path, "%s/%s", "littlefs/h", filename);
790         FILE * f = fopen(path, "r");
791         if (f) {
792             ESP_LOG1(TAG, "Failed to open file for reading");
793             return;
794         }
795         fread(data, size, 1, f);
796         fclose(f);
797     }
798
799     void setupFile(const char * filename, const void * data, size_t size) {
800         char path[64];
801         sprintf(path, "%s/%s", "littlefs/h", filename);
802         FILE * f = fopen(path, "w");
803         if (f) {
804             ESP_LOG1(TAG, "Failed to open file for writing");
805             return;
806         }
807         fwrite(data, size, 1, f);
808         fclose(f);
809     }
810
811     void readfile(const char * filename, void * data, size_t size) {
812         char path[64];
813         sprintf(path, "%s/%s", "littlefs/h", filename);
814         FILE * f = fopen(path, "r");
815         if (f) {
816             ESP_LOG1(TAG, "Failed to open file for reading");
817             return;
818         }
819         fread(data, size, 1, f);
820         fclose(f);
821     }
822
823     void removeFile(const char * filename) {
824         char path[64];
825         sprintf(path, "%s/%s", "littlefs/h", filename);
826         if (fdata) {
827             ESP_LOG1(TAG, "Null data");
828             return;
829         }
830         FILE * f = fopen(path, "r");
831         if (f) {
832             ESP_LOG1(TAG, "Failed to open file for reading");
833             return;
834         }
835         fclose(f);
836     }
837
838     void readfile(const char * filename, void * data, size_t size) {
839         char path[64];
840         sprintf(path, "%s/%s", "littlefs/h", filename);
841         FILE * f = fopen(path, "r");
842         if (f) {
843             ESP_LOG1(TAG, "Failed to open file for reading");
844             return;
845         }
846         fread(data, size, 1, f);
847         fclose(f);
848     }
849
850     void setupFile(const char * filename, const void * data, size_t size) {
851         char path[64];
852         sprintf(path, "%s/%s", "littlefs/h", filename);
853         FILE * f = fopen(path, "w");
854         if (f) {
855             ESP_LOG1(TAG, "Failed to open file for writing");
856             return;
857         }
858         fwrite(data, size, 1, f);
859         fclose(f);
860     }
861
862     void readfile(const char * filename, void * data, size_t size) {
863         char path[64];
864         sprintf(path, "%s/%s", "littlefs/h", filename);
865         FILE * f = fopen(path, "r");
866         if (f) {
867             ESP_LOG1(TAG, "Failed to open file for reading");
868             return;
869         }
870         fread(data, size, 1, f);
871         fclose(f);
872     }
873
874     void removeFile(const char * filename) {
875         char path[64];
876         sprintf(path, "%s/%s", "littlefs/h", filename);
877         if (fdata) {
878             ESP_LOG1(TAG, "Null data");
879             return;
880         }
881         FILE * f = fopen(path, "r");
882         if (f) {
883             ESP_LOG1(TAG, "Failed to open file for reading");
884             return;
885         }
886         fclose(f);
887     }
888
889     void readfile(const char * filename, void * data, size_t size) {
890         char path[64];
891         sprintf(path, "%s/%s", "littlefs/h", filename);
892         FILE * f = fopen(path, "r");
893         if (f) {
894             ESP_LOG1(TAG, "Failed to open file for reading");
895             return;
896         }
897         fread(data, size, 1, f);
898         fclose(f);
899     }
900
901     void setupFile(const char * filename, const void * data, size_t size) {
902         char path[64];
903         sprintf(path, "%s/%s", "littlefs/h", filename);
904         FILE * f = fopen(path, "w");
905         if (f) {
906             ESP_LOG1(TAG, "Failed to open file for writing");
907             return;
908         }
909         fwrite(data, size, 1, f);
910         fclose(f);
911     }
912
913     void readfile(const char * filename, void * data, size_t size) {
914         char path[64];
915         sprintf(path, "%s/%s", "littlefs/h", filename);
916         FILE * f = fopen(path, "r");
917         if (f) {
918             ESP_LOG1(TAG, "Failed to open file for reading");
919             return;
920         }
921         fread(data, size, 1, f);
922         fclose(f);
923     }
924
925     void removeFile(const char * filename) {
926         char path[64];
927         sprintf(path, "%s/%s", "littlefs/h", filename);
928         if (fdata) {
929             ESP_LOG1(TAG, "Null data");
930             return;
931         }
932         FILE * f = fopen(path, "r");
933         if (f) {
934             ESP_LOG1(TAG, "Failed to open file for reading");
935             return;
936         }
937         fclose(f);
938     }
939
940     void readfile(const char * filename, void * data, size_t size) {
941         char path[64];
942         sprintf(path, "%s/%s", "littlefs/h", filename);
943         FILE * f = fopen(path, "r");
944         if (f) {
945             ESP_LOG1(TAG, "Failed to open file for reading");
946             return;
947         }
948         fread(data, size, 1, f);
949         fclose(f);
950     }
951
952     void setupFile(const char * filename, const void * data, size_t size) {
953         char path[64];
954         sprintf(path, "%s/%s", "littlefs/h", filename);
955         FILE * f = fopen(path, "w");
956         if (f) {
957             ESP_LOG1(TAG, "Failed to open file for writing");
958             return;
959         }
960         fwrite(data, size, 1, f);
961         fclose(f);
962     }
963
964     void readfile(const char * filename, void * data, size_t size) {
965         char path[64];
966         sprintf(path, "%s/%s", "littlefs/h", filename);
967         FILE * f = fopen(path, "r");
968         if (f) {
969             ESP_LOG1(TAG, "Failed to open file for reading");
970             return;
971         }
972         fread(data, size, 1, f);
973         fclose(f);
974     }
975
976     void removeFile(const char * filename) {
977         char path[64];
978         sprintf(path, "%s/%s", "littlefs/h", filename);
979         if (fdata) {
980             ESP_LOG1(TAG, "Null data");
981             return;
982         }
983         FILE * f = fopen(path, "r");
984         if (f) {
985             ESP_LOG1(TAG, "Failed to open file for reading");
986             return;
987         }
988         fclose(f);
989     }
990
991     void readfile(const char * filename, void * data, size_t size) {
992         char path[64];
993         sprintf(path, "%s/%s", "littlefs/h", filename);
994         FILE * f = fopen(path, "r");
995         if (f) {
996             ESP_LOG1(TAG, "Failed to open file for reading");
997             return;
998         }
999         fread(data, size, 1, f);
1000         fclose(f);
1001     }
1002
1003     void setupFile(const char * filename, const void * data, size_t size) {
1004         char path[64];
1005         sprintf(path, "%s/%s", "littlefs/h", filename);
1006         FILE * f = fopen(path, "w");
1007         if (f) {
1008             ESP_LOG1(TAG, "Failed to open file for writing");
1009             return;
1010         }
1011         fwrite(data, size, 1, f);
1012         fclose(f);
1013     }
1014
1015     void readfile(const char * filename, void * data, size_t size) {
1016         char path[64];
1017         sprintf(path, "%s/%s", "littlefs/h", filename);
1018         FILE * f = fopen(path, "r");
1019         if (f) {
1020             ESP_LOG1(TAG, "Failed to open file for reading");
1021             return;
1022         }
1023         fread(data, size, 1, f);
1024         fclose(f);
1025     }
1026
1027     void removeFile(const char * filename) {
102
```



Lampiran 14. Program Modbus MQTT Gateway – modbus_handle.c dan modbus_handle.h

```

1 // "blinker/blink.h"
2 #include "esp_timer.h"
3 #include "pinmux.h"
4
5
6 typedef struct {
7     int gpio;
8     int delay;
9     blink_t * blinker;
10    esp_timer_handle_t timer;
11 } blink_t;
12
13 #define MAX_BLINK_LED 3
14
15 static blink_t blink[MAX_BLINK_LED];
16 static blink_t * get_blink_gpio()
17 {
18     for (int i = 0; i < MAX_BLINK_LED; i++) {
19         if (blink[i].gpio == gpio)
20             return &blink[i];
21     }
22
23     for (int i = 0; i < MAX_BLINK_LED; i++) {
24         if (blink[i].gpio == 0) {
25             blink[i].gpio = gpio;
26             return &blink[i];
27         }
28     }
29
30     return NULL;
31 }
32
33 static void blink_callback(void *arg)
34 {
35     blink_t * blink = (blink_t *)arg;
36     led_setstate = !led_setstate;
37     gpio_set_level(blink_gpio, led_setstate);
38 }
39
40 void gpioinit(void)
41 {
42     gpio_config_t LED_MQTT_conf = {
43         .pin_bit_mask = (1ULL << LED_MQTT),
44         .mode = GPIO_MODE_OUTPUT,
45         .pull_up_en = GPIO_PULLUP_DISABLE,
46         .pull_down_en = GPIO_PULLDOWN_DISABLE,
47         .io_type = GPIO_MODE_DISABLE
48     };
49
50     gpio_config_t LED_Mosduin_conf = {
51         .pin_bit_mask = (1ULL << LED_Mosduin),
52         .mode = GPIO_MODE_OUTPUT,
53         .pull_up_en = GPIO_PULLUP_DISABLE,
54         .pull_down_en = GPIO_PULLDOWN_DISABLE,
55         .io_type = GPIO_MODE_DISABLE
56     };
57
58     gpio_config_t LED_MQTT_conf = {
59         .pin_bit_mask = (1ULL << LED_MQTT),
60         .mode = GPIO_MODE_OUTPUT,
61         .pull_up_en = GPIO_PULLUP_DISABLE,
62         .pull_down_en = GPIO_PULLDOWN_DISABLE,
63         .io_type = GPIO_MODE_DISABLE
64     };
65
66     gpio_config_t LED_MQTT_conf = {
67         .pin_bit_mask = (1ULL << LED_MQTT),
68         .mode = GPIO_MODE_OUTPUT,
69         .pull_up_en = GPIO_PULLUP_DISABLE,
70         .pull_down_en = GPIO_PULLDOWN_DISABLE,
71         .io_type = GPIO_MODE_DISABLE
72     };
73
74     gpio_config_t LED_MQTT_conf = {
75         .pin_bit_mask = (1ULL << LED_MQTT),
76         .mode = GPIO_MODE_OUTPUT,
77         .pull_up_en = GPIO_PULLUP_DISABLE,
78         .pull_down_en = GPIO_PULLDOWN_DISABLE,
79         .io_type = GPIO_MODE_DISABLE
80     };
81
82     gpio_config_t LED_MQTT_conf = {
83         .pin_bit_mask = (1ULL << LED_MQTT),
84         .mode = GPIO_MODE_OUTPUT,
85         .pull_up_en = GPIO_PULLUP_DISABLE,
86         .pull_down_en = GPIO_PULLDOWN_DISABLE,
87         .io_type = GPIO_MODE_DISABLE
88     };
89
90     gpio_config_t LED_MQTT_conf = {
91         .pin_bit_mask = (1ULL << LED_MQTT),
92         .mode = GPIO_MODE_OUTPUT,
93         .pull_up_en = GPIO_PULLUP_DISABLE,
94         .pull_down_en = GPIO_PULLDOWN_DISABLE,
95         .io_type = GPIO_MODE_DISABLE
96     };
97
98     gpio_config_t LED_MQTT_conf = {
99         .pin_bit_mask = (1ULL << LED_MQTT),
100        .mode = GPIO_MODE_OUTPUT,
101        .pull_up_en = GPIO_PULLUP_DISABLE,
102        .pull_down_en = GPIO_PULLDOWN_DISABLE,
103        .io_type = GPIO_MODE_DISABLE
104    };
105
106    void ledLink(int gpio, int interval_ms)
107    {
108        blink_t * blink = get_blink_gpio();
109        if (blink) return;
110
111        if (led-blinking) return;
112
113        const esp_timer_create_args_t args = {
114            .callback = blink_callback,
115            .name = "blink"
116        };
117
118        esp_timer_create(&args, &timer);
119        esp_timer_start_periodic(timer, interval_ms * 1000);
120    }
121
122    void ledLink(int gpio, int level)
123    {
124        blink_t * blink = get_blink_gpio();
125        if (blink) return;
126
127        if (led-blinking) {
128            esp_timer_stop(timer);
129            esp_timer_delete(timer);
130            led-blinking = 0;
131        }
132
133        led_setstate = level;
134        gpio_set_level(blink_gpio, level);
135    }
136
137    #endif

```

```

1 #ifndef GPIODCON_H
2 #define GPIODCON_H
3
4 //arah wifi
5 //kuning myit
6 //putih modbus
7
8 typedef enum{
9     LED_WIFI= 13,
10    LED_Modbus= 14,
11    LED_MYIT= 26
12 }led_pin_t;
13
14 void ledLink(int gpio, int interval_ms);
15 void ledSet(int gpio, int level);
16 void gpioInit(void);
17
18 #endif

```



1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :

- [illegible]

[illegible][illegible]

tanpa izin Politeknik Negeri Jakarta

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6 #include <sys/stat.h>
7 #include <fcntl.h>
8 #include <sys/mman.h>
9 #include <sys/time.h>
10 #include <sys/resource.h>
11 #include <sys/wait.h>
12 #include <sys/queue.h>
13 #include <sys/uio.h>
14 #include <sys/sysmacros.h>
15 #include <sys/sysinfo.h>
16 #include <sys/param.h>
17 #include <sys/procfs.h>
18 #include <sys/procfs.h>
19 #include <sys/procfs.h>
20 #include <sys/procfs.h>
21 #include <sys/procfs.h>
22 #include <sys/procfs.h>
23 #include <sys/procfs.h>
24 #include <sys/procfs.h>
25 #include <sys/procfs.h>
26 #include <sys/procfs.h>
27 #include <sys/procfs.h>
28 #include <sys/procfs.h>
29 #include <sys/procfs.h>
30 #include <sys/procfs.h>
31 #include <sys/procfs.h>
32 #include <sys/procfs.h>
33 #include <sys/procfs.h>
34 #include <sys/procfs.h>
35 #include <sys/procfs.h>
36 #include <sys/procfs.h>
37 #include <sys/procfs.h>
38 #include <sys/procfs.h>
39 #include <sys/procfs.h>
40 #include <sys/procfs.h>
41 #include <sys/procfs.h>
42 #include <sys/procfs.h>
43 #include <sys/procfs.h>
44 #include <sys/procfs.h>
45 #include <sys/procfs.h>
46 #include <sys/procfs.h>
47 #include <sys/procfs.h>
48 #include <sys/procfs.h>
49 #include <sys/procfs.h>
50 #include <sys/procfs.h>
51 #include <sys/procfs.h>
52 #include <sys/procfs.h>
53 #include <sys/procfs.h>
54 #include <sys/procfs.h>
55 #include <sys/procfs.h>
56 #include <sys/procfs.h>
57 #include <sys/procfs.h>
58 #include <sys/procfs.h>
59 #include <sys/procfs.h>
60 #include <sys/procfs.h>
61 #include <sys/procfs.h>
62 #include <sys/procfs.h>
63 #include <sys/procfs.h>
64 #include <sys/procfs.h>
65 #include <sys/procfs.h>
66 #include <sys/procfs.h>
67 #include <sys/procfs.h>
68 #include <sys/procfs.h>
69 #include <sys/procfs.h>
70 #include <sys/procfs.h>
71 #include <sys/procfs.h>
72 #include <sys/procfs.h>
73 #include <sys/procfs.h>
74 #include <sys/procfs.h>
75 #include <sys/procfs.h>
76 #include <sys/procfs.h>
77 #include <sys/procfs.h>
78 #include <sys/procfs.h>
79 #include <sys/procfs.h>
80 #include <sys/procfs.h>
81 #include <sys/procfs.h>
82 #include <sys/procfs.h>
83 #include <sys/procfs.h>
84 #include <sys/procfs.h>
85 #include <sys/procfs.h>
86 #include <sys/procfs.h>
87 #include <sys/procfs.h>
88 #include <sys/procfs.h>
89 #include <sys/procfs.h>
90 #include <sys/procfs.h>
91 #include <sys/procfs.h>
92 #include <sys/procfs.h>
93 #include <sys/procfs.h>
94 #include <sys/procfs.h>
95 #include <sys/procfs.h>
96 #include <sys/procfs.h>
97 #include <sys/procfs.h>
98 #include <sys/procfs.h>
99 #include <sys/procfs.h>
100 #include <sys/procfs.h>

```

```

1 // #include <stdio.h>
2
3 #define WFFI_LEN 1024
4 #define WFFI_MAXOFF 1
5
6 #include "exp_wffi.h"
7
8 typedef struct {
9     char name[32];
10     char password[64];
11     bool wffi;
12 } wffi_localconfig_t;
13
14 void wffi_init(void);
15 void set_wffi_config();
16 void wffi_set_ip_addr(wffi_localconfig_t *wffi);
17 void wffi_set_name(wffi_localconfig_t *wffi);
18 void wffi_set_passwd(wffi_localconfig_t *wffi);
19 void wffi_set_ip_gateway(wffi_localconfig_t *wffi, const char *ip, const char *gateway);
20
21 #endif

```

```

100 // Get the number of valid words in the word list
101 int valid_words = 0;
102 for (int i = 0; i < word_list.size(); i++)
103 {
104     // Check if the word is in the word list
105     if (word_list[i] == word)
106     {
107         valid_words++;
108     }
109 }
110
111 // Check if the word is in the word list
112 if (valid_words == 0)
113 {
114     // The word is not in the word list
115     return false;
116 }
117
118 // Check if the word is in the word list
119 if (valid_words == 1)
120 {
121     // The word is in the word list
122     return true;
123 }
124
125 // Check if the word is in the word list
126 if (valid_words == 2)
127 {
128     // The word is in the word list
129     return true;
130 }
131
132 // Check if the word is in the word list
133 if (valid_words == 3)
134 {
135     // The word is in the word list
136     return true;
137 }
138
139 // Check if the word is in the word list
140 if (valid_words == 4)
141 {
142     // The word is in the word list
143     return true;
144 }
145
146 // Check if the word is in the word list
147 if (valid_words == 5)
148 {
149     // The word is in the word list
150     return true;
151 }
152
153 // Check if the word is in the word list
154 if (valid_words == 6)
155 {
156     // The word is in the word list
157     return true;
158 }
159
160 // Check if the word is in the word list
161 if (valid_words == 7)
162 {
163     // The word is in the word list
164     return true;
165 }
166
167 // Check if the word is in the word list
168 if (valid_words == 8)
169 {
170     // The word is in the word list
171     return true;
172 }
173
174 // Check if the word is in the word list
175 if (valid_words == 9)
176 {
177     // The word is in the word list
178     return true;
179 }
180
181 // Check if the word is in the word list
182 if (valid_words == 10)
183 {
184     // The word is in the word list
185     return true;
186 }
187
188 // Check if the word is in the word list
189 if (valid_words == 11)
190 {
191     // The word is in the word list
192     return true;
193 }
194
195 // Check if the word is in the word list
196 if (valid_words == 12)
197 {
198     // The word is in the word list
199     return true;
200 }
201
202 // Check if the word is in the word list
203 if (valid_words == 13)
204 {
205     // The word is in the word list
206     return true;
207 }
208
209 // Check if the word is in the word list
210 if (valid_words == 14)
211 {
212     // The word is in the word list
213     return true;
214 }
215
216 // Check if the word is in the word list
217 if (valid_words == 15)
218 {
219     // The word is in the word list
220     return true;
221 }
222
223 // Check if the word is in the word list
224 if (valid_words == 16)
225 {
226     // The word is in the word list
227     return true;
228 }
229
230 // Check if the word is in the word list
231 if (valid_words == 17)
232 {
233     // The word is in the word list
234     return true;
235 }
236
237 // Check if the word is in the word list
238 if (valid_words == 18)
239 {
240     // The word is in the word list
241     return true;
242 }
243
244 // Check if the word is in the word list
245 if (valid_words == 19)
246 {
247     // The word is in the word list
248     return true;
249 }
250
251 // Check if the word is in the word list
252 if (valid_words == 20)
253 {
254     // The word is in the word list
255     return true;
256 }
257
258 // Check if the word is in the word list
259 if (valid_words == 21)
260 {
261     // The word is in the word list
262     return true;
263 }
264
265 // Check if the word is in the word list
266 if (valid_words == 22)
267 {
268     // The word is in the word list
269     return true;
270 }
271
272 // Check if the word is in the word list
273 if (valid_words == 23)
274 {
275     // The word is in the word list
276     return true;
277 }
278
279 // Check if the word is in the word list
280 if (valid_words == 24)
281 {
282     // The word is in the word list
283     return true;
284 }
285
286 // Check if the word is in the word list
287 if (valid_words == 25)
288 {
289     // The word is in the word list
290     return true;
291 }
292
293 // Check if the word is in the word list
294 if (valid_words == 26)
295 {
296     // The word is in the word list
297     return true;
298 }
299
300 // Check if the word is in the word list
301 if (valid_words == 27)
302 {
303     // The word is in the word list
304     return true;
305 }
306
307 // Check if the word is in the word list
308 if (valid_words == 28)
309 {
310     // The word is in the word list
311     return true;
312 }
313
314 // Check if the word is in the word list
315 if (valid_words == 29)
316 {
317     // The word is in the word list
318     return true;
319 }
320
321 // Check if the word is in the word list
322 if (valid_words == 30)
323 {
324     // The word is in the word list
325     return true;
326 }
327
328 // Check if the word is in the word list
329 if (valid_words == 31)
330 {
331     // The word is in the word list
332     return true;
333 }
334
335 // Check if the word is in the word list
336 if (valid_words == 32)
337 {
338     // The word is in the word list
339     return true;
340 }
341
342 // Check if the word is in the word list
343 if (valid_words == 33)
344 {
345     // The word is in the word list
346     return true;
347 }
348
349 // Check if the word is in the word list
350 if (valid_words == 34)
351 {
352     // The word is in the word list
353     return true;
354 }
355
356 // Check if the word is in the word list
357 if (valid_words == 35)
358 {
359     // The word is in the word list
360     return true;
361 }
362
363 // Check if the word is in the word list
364 if (valid_words == 36)
365 {
366     // The word is in the word list
367     return true;
368 }
369
370 // Check if the word is in the word list
371 if (valid_words == 37)
372 {
373     // The word is in the word list
374     return true;
375 }
376
377 // Check if the word is in the word list
378 if (valid_words == 38)
379 {
380     // The word is in the word list
381     return true;
382 }
383
384 // Check if the word is in the word list
385 if (valid_words == 39)
386 {
387     // The word is in the word list
388     return true;
389 }
390
391 // Check if the word is in the word list
392 if (valid_words == 40)
393 {
394     // The word is in the word list
395     return true;
396 }
397
398 // Check if the word is in the word list
399 if (valid_words == 41)
400 {
401     // The word is in the word list
402     return true;
403 }
404
405 // Check if the word is in the word list
406 if (valid_words == 42)
407 {
408     // The word is in the word list
409     return true;
410 }
411
412 // Check if the word is in the word list
413 if (valid_words == 43)
414 {
415     // The word is in the word list
416     return true;
417 }
418
419 // Check if the word is in the word list
420 if (valid_words == 44)
421 {
422     // The word is in the word list
423     return true;
424 }
425
426 // Check if the word is in the word list
427 if (valid_words == 45)
428 {
429     // The word is in the word list
430     return true;
431 }
432
433 // Check if the word is in the word list
434 if (valid_words == 46)
435 {
436     // The word is in the word list
437     return true;
438 }
439
440 // Check if the word is in the word list
441 if (valid_words == 47)
442 {
443     // The word is in the word list
444     return true;
445 }
446
447 // Check if the word is in the word list
448 if (valid_words == 48)
449 {
450     // The word is in the word list
451     return true;
452 }
453
454 // Check if the word is in the word list
455 if (valid_words == 49)
456 {
457     // The word is in the word list
458     return true;
459 }
460
461 // Check if the word is in the word list
462 if (valid_words == 50)
463 {
464     // The word is in the word list
465     return true;
466 }
467
468 // Check if the word is in the word list
469 if (valid_words == 51)
470 {
471     // The word is in the word list
472     return true;
473 }
474
475 // Check if the word is in the word list
476 if (valid_words == 52)
477 {
478     // The word is in the word list
479     return true;
480 }
481
482 // Check if the word is in the word list
483 if (valid_words == 53)
484 {
485     // The word is in the word list
486     return true;
487 }
488
489 // Check if the word is in the word list
490 if (valid_words == 54)
491 {
492     // The word is in the word list
493     return true;
494 }
495
496 // Check if the word is in the word list
497 if (valid_words == 55)
498 {
499     // The word is in the word list
500     return true;
501 }
502
503 // Check if the word is in the word list
504 if (valid_words == 56)
505 {
506     // The word is in the word list
507     return true;
508 }
509
510 // Check if the word is in the word list
511 if (valid_words == 57)
512 {
513     // The word is in the word list
514     return true;
515 }
516
517 // Check if the word is in the word list
518 if (valid_words == 58)
519 {
520     // The word is in the word list
521     return true;
522 }
523
524 // Check if the word is in the word list
525 if (valid_words == 59)
526 {
527     // The word is in the word list
528     return true;
529 }
530
531 // Check if the word is in the word list
532 if (valid_words == 60)
533 {
534     // The word is in the word list
535     return true;
536 }
537
538 // Check if the word is in the word list
539 if (valid_words == 61)
540 {
541     // The word is in the word list
542     return true;
543 }
544
545 // Check if the word is in the word list
546 if (valid_words == 62)
547 {
548     // The word is in the word list
549     return true;
550 }
551
552 // Check if the word is in the word list
553 if (valid_words == 63)
554 {
555     // The word is in the word list
556     return true;
557 }
558
559 // Check if the word is in the word list
560 if (valid_words == 
```

[illegible][illegible]

Hak Cipta :

1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
 - a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
 - b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumunkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```

153 //
154 httpd_uri_t get_APreq = {
155     .uri = "/api/APreq",
156     .method = HTTP_GET,
157     .handler = handle_APreq,
158     .user_ctx = NULL
159 };
160
161 httpd_uri_t post_APsel = {
162     .uri = "/api/APselection",
163     .method = HTTP_POST,
164     .handler = handle_APsel,
165     .user_ctx = NULL
166 };
167
168 httpd_uri_t post_nqticonf = {
169     .uri = "/api/nqticonf",
170     .method = HTTP_POST,
171     .handler = handle_nqticonf,
172     .user_ctx = NULL
173 };
174
175 httpd_uri_t post_topicconf = {
176     .uri = "/api/topicconf",
177     .method = HTTP_POST,
178     .handler = handle_nqt_topic_change,
179     .user_ctx = NULL
180 };
181
182 httpd_uri_t post_modbusconf = {
183     .uri = "/api/modbusconf",
184     .method = HTTP_POST,
185     .handler = handle_modbus_master_config,
186     .user_ctx = NULL
187 };
188
189 httpd_uri_t post_modaddr = {
190     .uri = "/api/changetmodaddr",
191     .method = HTTP_POST,
192     .handler = handle_change_modaddr,
193     .user_ctx = NULL
194 };
195
196 httpd_uri_t get_modaddr = {
197     .uri = "/api/getmodaddr",
198     .method = HTTP_GET,
199     .handler = handle_get_modaddr,
200     .user_ctx = NULL
201 };
202
203 httpd_uri_t get_refreshtopic = {
204     .uri = "/api/getrefreshtopic",
205     .method = HTTP_GET,
206     .handler = handle_get_refreshtopic,
207     .user_ctx = NULL
208 };
209
210 httpd_uri_t get_refreshmodbus = {
211     .uri = "/api/getrefreshmodbus",
212     .method = HTTP_GET,
213     .handler = handle_get_refreshmodbus,
214     .user_ctx = NULL
215 };
216
217 httpd_handle_t server = NULL;
218
219 void server_init(void){
220     httpd_config_t config = HTTPD_DEFAULT_CONFIG();
221     config_max_uri_handlers = 20;
222     ESP_ERROR_CHECK(httpd_start(&server, &config));
223     if(server != NULL){
224         httpd_register_uri_handler(server, &get_APreq);
225         httpd_register_uri_handler(server, &post_APsel);
226     }
227 }
228
229 httpd_handle_t wificonfig_serveron(void) {
230     if (server != NULL) {
231         httpd_unregister_uri_handler(server, get_configweb.uri, get_configweb.method);
232         httpd_unregister_uri_handler(server, get_configweb.uri, get_configweb.method);
233         httpd_unregister_uri_handler(server, get_modaddr.uri, get_modaddr.method);
234         httpd_unregister_uri_handler(server, post_modaddr.uri, post_modaddr.method);
235         httpd_unregister_uri_handler(server, post_nqticonf.uri, post_nqticonf.method);
236         httpd_unregister_uri_handler(server, post_topicconf.uri, post_topicconf.method);
237         httpd_unregister_uri_handler(server, post_modbusconf.uri, post_modbusconf.method);
238         httpd_unregister_uri_handler(server, get_refreshtopic.uri, get_refreshtopic.method);
239         httpd_unregister_uri_handler(server, get_refreshmodbus.uri, get_refreshmodbus.method);
240         httpd_register_uri_handler(server, &get_wifi_connect);
241     }
242     return server;
243 }
244
245 httpd_handle_t wificonfig_serveroff(void) {
246     if (server != NULL) {
247         httpd_unregister_uri_handler(server, get_wifi_connect.uri, get_wifi_connect.method);
248         httpd_unregister_uri_handler(server, &get_configweb);
249         httpd_unregister_uri_handler(server, &get_modaddr);
250         httpd_unregister_uri_handler(server, &post_modaddr);
251         httpd_unregister_uri_handler(server, &post_nqticonf);
252         httpd_unregister_uri_handler(server, &post_topicconf);
253         httpd_unregister_uri_handler(server, &post_modbusconf);
254         httpd_unregister_uri_handler(server, &get_refreshtopic);
255         httpd_unregister_uri_handler(server, &get_refreshmodbus);
256         ESP_LOGI(TAG, "Webserver stopped");
257     }
258     return server;
259 }

```

```

1
2 #ifndef WEBWDLI_H
3 #define WEBWDLI_H
4 #include "esp_http_server.h"
5
6 httpd_handle_t wificonfig_serveron(void);
7 httpd_handle_t wificonfig_serveroff(void);
8 void server_init(void);
9 void wifi_event_handler(void* arg, esp_event_base_t event_base, int32_t event_id, void* event_data);
10 #endif

```