BAB IV
PENGUJIAN DAN ANALISA

4.1. Data Sensor dan *Input*

Pada sistem ini digunakan sembilan buah *smoke detector* analog yang dibagi ke dalam tiga segmen, dimana masing-masing segmen terdiri dari tiga buah sensor. Konfigurasi sembilan *smoke detector* yang dibagi menjadi tiga segmen ini dibuat supaya sistem bisa mendeteksi asap secara lebih merata di tiap area. Dengan begitu, sistem tidak hanya tahu ada asap, tetapi juga bisa menentukan tingkat kedaruratan berdasarkan berapa banyak sensor yang aktif di setiap segmen.

Berdasarkan hasil pengujian awal dengan *smoke detector* yang dirangkai secara parallel pada tabel 4.1, *smoke detector* menghasilkan data berupa kondisi *Standby* dengan arus sangat kecil, berkisar antara 0,27 mA – 0,54 mA dan pada kondisi *Alarm* arus meningkat signifikan, mencapai 38 mA – 116 mA tergantung jumlah sensor aktif.

Tabel 4.1 Pengukuran Arus Sensor

Kondisi Smoke	Pengukuran (Unit Sensor)				Perhitungan (Unit Sensor)			
	Led Mati	1 (mA)	2 (mA)	3 (mA)	Led Mati	1 (mA)	2 (mA)	3 (mA)
Standby	0.29	0.37	0.47	0.50	0.27	0.36	0.45	0.54
Alarm	-	38.57	77	114	-	38.75	77.5	116.25

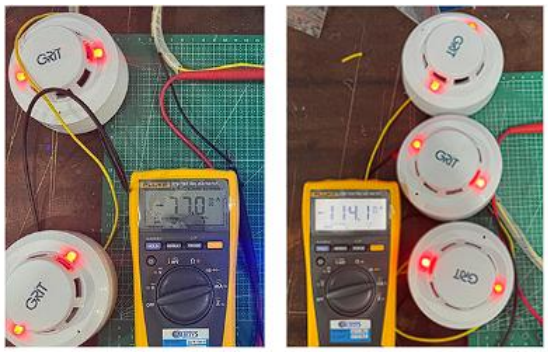
Data ini kemudian akan dikonversikan menjadi kondisi biner, yaitu 0 untuk tidak aktif atau *Standby* dan 1 untuk aktif atau *Alarm*. Sebelum digunakan dalam proses inferensi *fuzzy*, data hasil pembacaan sensor melalui STM32F4 *Discovery* terlebih dahulu dilakukan *preprocessing*. Proses ini meliputi pembacaan nilai arus dan pengelompokan data berdasarkan segmen, yang kemudian akan diolah untuk mendapatkan informasi jumlah sensor aktif dalam setiap segmen di ESP32.

Berdasarkan hasil pengukuran dan perhitungan seperti pada Tabel 4.2, terlihat bahwa pada kondisi *standby* nilai *error* relatif lebih besar dibandingkan dengan kondisi *alarm*. Pada saat *standby*, *error* berada pada rentang 2,70%–7,40%, sedangkan pada kondisi *alarm error* jauh lebih kecil yaitu hanya sekitar 0,46%–0,64%, bahkan mencapai 0,019% ketika tiga sensor aktif. Hal ini karena pada kondisi *standby* arus yang terbaca sangat kecil, sehingga sedikit perbedaan hasil ukur akan menghasilkan persentase *error* yang lebih besar. Sementara itu, pada kondisi *alarm* arus yang terbaca jauh lebih besar sehingga perbedaan kecil tidak berpengaruh signifikan terhadap persentase *error*.

Tabel 4.2 Rekapitulasi *Error Smoke Detector*

Kondisi Smoke	1 Sensor	2 Sensor	3 Sensor
Standby	2.70%	4.44%	7.40%
Alarm	0.46%	0.64%	0.019%

Selain itu, *smoke detector* memiliki LED indikator pada setiap sensor yang berkedip dalam selang waktu tertentu. Kedipan LED ini tidak serempak antara satu sensor dengan yang lain, sehingga saat dilakukan pembacaan bisa terjadi perbedaan nilai sesaat akibat posisi kedipan yang tidak sama. Hal tersebut terutama berpengaruh pada kondisi *standby* karena arus yang dihasilkan kecil, sehingga pengaruh kedipan LED relatif lebih besar terhadap hasil pengukuran. Sedangkan pada kondisi *alarm*, LED indikator pada sensor akan menyala terus menerus, sehingga tidak ada pengaruh kedipan terhadap hasil pengukuran dan arus yang terbaca menjadi lebih stabil.



Gambar 4.1 Pengujian Smoke Detector

Gambar 4.1 menunjukkan hasil pengujian *smoke detector* pada kondisi *alarm* dengan konfigurasi dua sensor dan tiga sensor yang terpasang.

4.2. Pengujian RFID dan *Doorlock*

Pengujian parsial RFID dan *doorlock* dilakukan untuk memastikan bahwa modul RFID dapat membaca kartu/tag dengan benar pada Gambar 4.2 serta sistem dapat mengendalikan aktuator *doorlock* sesuai hasil autentikasi seperti pada Gambar 4.3.

Gambar 4.2 RFID dan Doorlock

```
--- RFID & DOOR LOCK STATUS ---
RFID Reader: ACTIVE
Door Lock Status: OPEN (0s remaining)
Access Granted: 13
Access Denied: 0
Last Authorized Card: 30:5C:B2:9F
```

Gambar 4.3 Pengujian Akses RFID dan Doorlock

Pengujian dilakukan dengan cara menyiapkan kartu RFID yang telah terdaftar. Kartu tersebut digunakan untuk menguji keakuratan sistem dalam membedakan hak akses pengguna. Selanjutnya, aktuator doorlock berupa *solenoid doorlock* dihubungkan ke mikrokontroler sehingga dapat dikendalikan melalui sinyal keluaran. Setelah itu, kartu RFID didekatkan ke pembaca, dan respon sistem diamati dari dua sisi, yaitu hasil pembacaan UID kartu serta kondisi doorlock yang berubah sesuai dengan status autentikasi.

Berdasarkan Tabel 4.3 hasil pengujian RFID dan *doorlock* dengan menggunakan satu kartu yang telah terdaftar menunjukkan bahwa dari 15 kali percobaan, seluruhnya berhasil dibaca dengan benar dan menghasilkan respon *Access Granted* sehingga *doorlock* terbuka sesuai perintah. Berdasarkan perhitungan, akurasi sistem mencapai 100% dengan tingkat error 0%. Hal ini membuktikan bahwa modul RFID mampu bekerja secara konsisten dan andal dalam mengenali kartu terdaftar serta mengendalikan aktuator *doorlock* tanpa terjadi kesalahan selama pengujian.

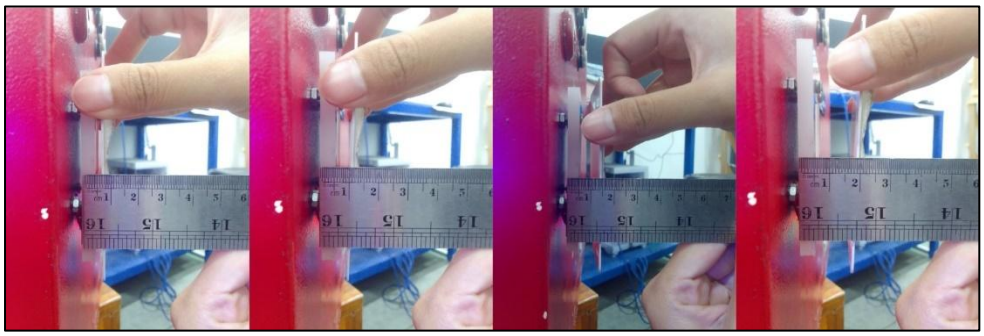
Tabel 4.3 Pengujian RFID dan *Doorlock*

Pengujian Ke-	UID	Respon Sistem	Kondisi *Doorlock*
1	30:5C:B2:9F	*Access Granted*	OPEN
2	30:5C:B2:9F	*Access Granted*	OPEN
3	30:5C:B2:9F	*Access Granted*	OPEN
4	30:5C:B2:9F	*Access Granted*	OPEN
5	30:5C:B2:9F	*Access Granted*	OPEN
6	30:5C:B2:9F	*Access Granted*	OPEN
7	30:5C:B2:9F	*Access Granted*	OPEN
8	30:5C:B2:9F	*Access Granted*	OPEN
9	30:5C:B2:9F	*Access Granted*	OPEN
10	30:5C:B2:9F	*Access Granted*	OPEN
11	30:5C:B2:9F	*Access Granted*	OPEN

Tabel Lanjutan 4.3 Pengujian RFID dan *Doorlock*

Pengujian Ke-	UID	Respon Sistem	Kondisi *Doorlock*
12	30:5C:B2:9F	*Access Granted*	OPEN
13	30:5C:B2:9F	*Access Granted*	OPEN
14	30:5C:B2:9F	*Access Granted*	OPEN
15	30:5C:B2:9F	*Access Granted*	OPEN

Selanjutnya, dilakukan pengujian tambahan terhadap jarak pembacaan kartu RFID dengan variasi jarak 0.8 cm, 1 cm, 1.5 cm, dan 2 cm seperti pada Gambar 4.4.



Gambar 4.4 Pengujian Jarak Deteksi RFID

Hasil pengujian menunjukkan bahwa pada jarak 0.8 cm, 1 cm, dan 1.5 cm sistem mampu membaca kartu secara konsisten dengan akurasi 100% dan error 0% seperti pada Tabel 4.4.

Tabel 4.4 Pengujian Pembacaan Jarak RFID

Jarak (mm)	Jumlah Percobaan	Berhasil	Gagal	Error (%)
8	5	5	0	0%
10	5	5	0	0%
15	5	5	0	0%
20	5	0	5	100%

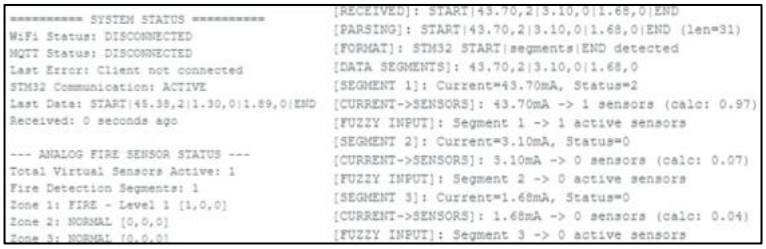
Namun, pada jarak 2 cm pembacaan kartu tidak berhasil sama sekali sehingga akurasi menjadi 0% dengan error 100%. Hal ini menunjukkan bahwa jarak efektif pembacaan RFID pada sistem adalah kurang dari 2 cm,

dengan jarak optimal berada pada kisaran 0.8–1.5 cm agar sistem tetap bekerja dengan baik dan stabil.

Jika dibandingkan dengan spesifikasi pada *datasheet*, modul RFID seharusnya mampu membaca kartu hingga jarak maksimum sekitar 5 cm. Akan tetapi, pada implementasi sistem ini modul RFID ditempatkan di dalam *cover* akrilik sebagai pelindung, sehingga daya pancar dan sensitivitas pembacaan kartu mengalami penurunan. Akibatnya, jarak pembacaan menjadi lebih pendek daripada spesifikasi ideal. Meskipun demikian, dengan jarak baca 0.8–1.5 cm sistem masih dapat bekerja dengan andal dalam mendukung proses autentikasi akses pada *doorlock*.

4.3. Hasil Pengujian Komunikasi STM32 ke ESP32

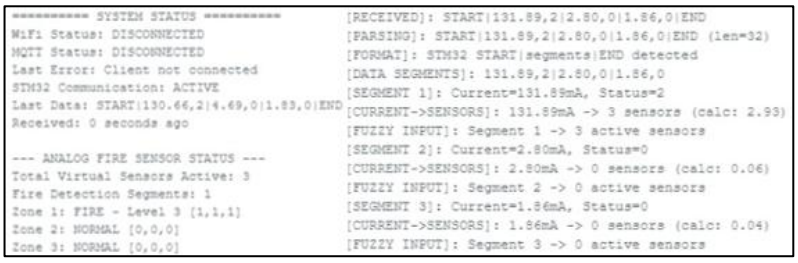
Komunikasi data sensor dilakukan melalui UART (USART6 STM32 ke RX ESP32) menggunakan *library HardwareSerial*. Data yang diterima dari STM32 memiliki format *string* khusus, yaitu START|current,status|...|END. Data ini diproses dalam fungsi *handleSTMData()* yang bertugas membaca aliran data, mem-parsing pesan, serta mengonversi nilai arus yang terbaca menjadi jumlah sensor aktif pada tiap segmen.



Gambar 4.5 Pengujian Komunikasi 1 Sensor

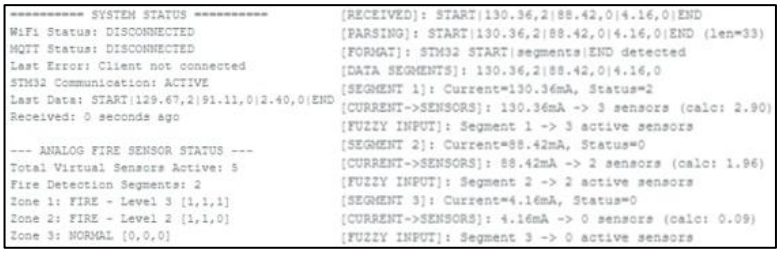
Gambar 4.5 menunjukkan status sistem saat koneksi WiFi dan MQTT belum aktif, tetapi komunikasi dengan STM32 sudah berjalan. Data yang diterima berupa string START|43.70|3.10|1.68|END berhasil terbaca dan ditampilkan kembali oleh ESP32. Sistem kemudian menampilkan jumlah sensor aktif beserta status masing-masing segmen. Pada contoh tersebut, terdapat 1 sensor aktif pada segmen 1, sedangkan

segmen 2 dan segmen 3 berada pada kondisi normal atau tidak ada sensor aktif.



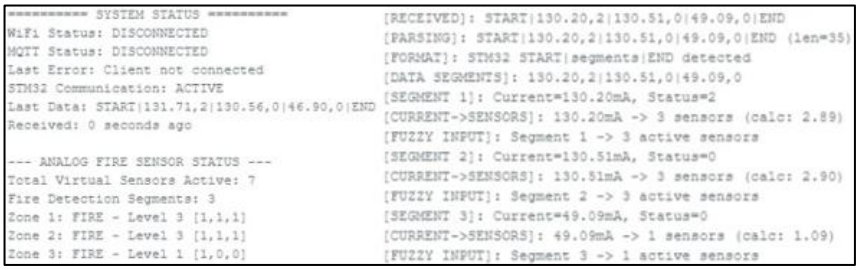
Gambar 4.6 Pengujian Komunikasi 3 Sensor

Gambar 4.6 merupakan pengujian komunikasi menggunakan 3 sensor. Data yang diterima berupa string START|131.89|2.80|1.86|END berhasil terbaca. Sistem kemudian menampilkan jumlah sensor aktif beserta status masing-masing segmen. Pada contoh tersebut, terdapat 3 sensor aktif pada segmen 1, sedangkan segmen 2 dan segmen 3 berada pada kondisi normal atau tidak ada sensor aktif.



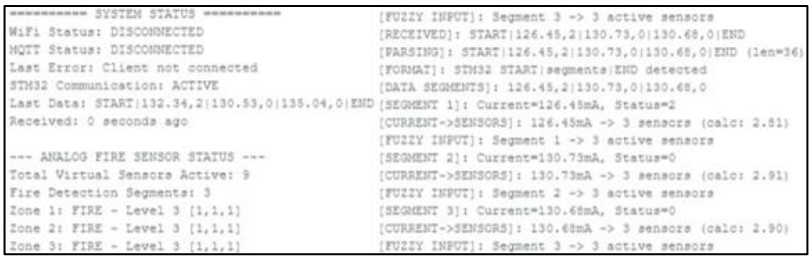
Gambar 4.7 Pengujian Komunikasi 5 Sensor

Pengujian juga dilakukan menggunakan 5 sensor seperti pada Gambar 4.7. Data yang diterima pada ESP32 dari STM32F4 adalah START|130.36|88.42|4.16|END. Data tersebut berhasil diparsing dengan hasil terdapat 3 sensor aktif pada segmen 1, 2 sensor aktif di segmen 2, dan tidak ada sensor aktif di segmen 3.



Gambar 4.8 Pengujian Komunikasi 7 Sensor

Pengujian dengan 7 sensor aktif menunjukkan hasil bahwa daya yang berhasil di terima di ESP32 pada Gambar 4.8 adalah START|130.20|130.51|49.09|END. Data yang sudah diterima tersebut berhasil diparsing sehingga didapatkan informasi bahwa terdapat 3 sensor aktif di segmen 1, 3 sensor aktif di segmen 2, dan 1 sensor aktif di segmen 3.



Gambar 4.9 Pengujian Komunikasi 9 Sensor

Pengujian dengan sensor aktif terbanyak yaitu 9 sensor juga berhasil dilakukan. Data yang diterima ESP32 berisi string START|126.45|130.73|130.68|END pada Gambar 4.9. Data berhasil diparsing dengan hasil informasi terdapat 3 sensor aktif di semua segmen.

Hasil pengujian pada Tabel 4.5 menunjukkan bahwa data yang dikirim STM32 dan diterima ESP32 memiliki kesesuaian penuh tanpa kehilangan informasi. Sebagai contoh, pada data 43.70 ; 3.10 ; 1.68, ESP32 berhasil menerima dengan format START|43.70|3.10|1.68|END dan memproses menjadi Seg1=43.70, Seg2=3.10, Seg3=1.68. Setelah dilakukan parsing sensor, sistem dapat mengklasifikasikan bahwa hanya Segmen 1 yang aktif dengan nilai 1 unit sensor.

Tabel 4.5 Pengujian Komunikasi STM32 ke ESP32

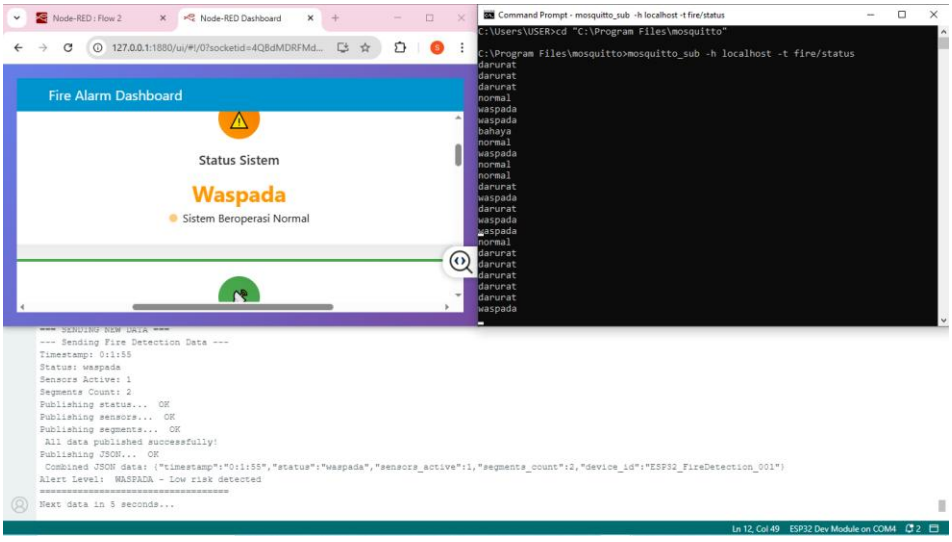
Data STM32	Data ESP32	Parsing Segmen (mA)	Parsing Sensor (unit)	*Error*
43.70 ; 3.10 ; 1.68	START|43.70|3.10|1.68|END	Seg1=43.70, Seg2=3.10, Seg3=1.68	Seg1=1, Seg2 = 0, Seg3 = 0	0%
89.61 ; 2.53 ; 1.32	START|89.61|2.53|1.32|END	Seg1=89.61, Seg2=2.53, Seg3=1.32	Seg1=2, Seg2 = 0, Seg3 = 0	0%
131.89 ; 2.80 ; 1.86	START|131.89|2.80|1.86|END	Seg1=131.89, Seg2=2.80, Seg3=1.86	Seg1=3, Seg2 = 0, Seg3 = 0	0%
131.77 ; 44,81 ; 3.29	START|131.77|44.81|3.29|END	Seg1=131.77, Seg2=44.81, Seg3=3.29	Seg1=3, Seg2 = 1, Seg3 = 0	0%
130.36 ; 88.42 ; 4.16	START|130.36|88.42|4.16|END	Seg1=130.36, Seg2=88.43, Seg3=4.16	Seg1=3, Seg2 = 2, Seg3 = 0	0%
131.45 ; 130.53 ; 4.19	START|131.45|130.53|4.19|END	Seg1=131.45, Seg2=130.53, Seg3=4.19	Seg1=3, Seg2 = 3, Seg3 = 0	0%
130.20 ; 130.51 ; 49.09	START|130.20|130.51|49.09|END	Seg1=130.20, Seg2=130.51, Seg3=49.09	Seg1=3, Seg2 = 3, Seg3 = 1	0%
130.61 ; 132.22 ; 88.06	START|130.61|132.22|88.06|END	Seg1=130.61, Seg2=132.22, Seg3=88.06	Seg1=3, Seg2 = 3, Seg3 = 2	0%
126.45 ; 130.73 ; 130.68	START|126.45|130.73|130.68|END	Seg1=126.45, Seg2=130.73, Seg3=130.68	Seg1=3, Seg2 = 3, Seg3 = 3	0%

Seiring peningkatan nilai arus pada tiap segmen, ESP32 juga mampu memetakan jumlah sensor yang aktif secara bertahap. Misalnya, pada data 131.77 ; 44.81 ; 3.29, sistem membaca Seg1=3 sensor aktif dan Seg2=1 sensor aktif. Begitu juga saat ketiga segmen diberi *input*, seperti pada data 126.45 ; 130.73 ; 130.68, parsing menghasilkan Seg1=3, Seg2=3, dan Seg3=3 sensor aktif, yang sesuai dengan kondisi sebenarnya.

Secara keseluruhan, pengujian ini membuktikan bahwa komunikasi STM32 ke ESP32 berjalan stabil, akurat dengan 0% *error*, dan *real-time*. Proses parsing mampu mengonversi data analog menjadi jumlah sensor aktif per segmen tanpa *error*, sehingga sistem siap digunakan untuk tahap integrasi ke *fuzzy* logic dan pengiriman data ke *server* MQTT.

4.4. Pengujian Komunikasi dan Pengiriman Data dari ESP ke MQTT

Langkah awal pengujian ini adalah menghubungkan ESP32 dan *device* yang akan digunakan untuk monitoring sistem atau PC *server* dengan koneksi jaringan yang sama, pada penelitian ini menggunakan IP yang disesuaikan ketika terhubung dengan jaringan.



Gambar 4.10 Pengujian Pengiriman Data Status ke MQTT

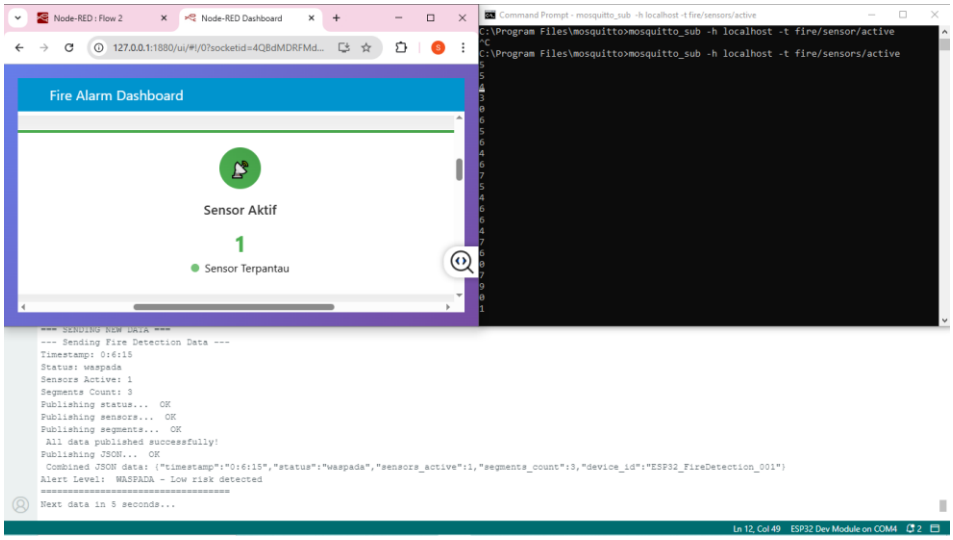
Gambar 4.10 menunjukkan tampilan *Node-RED Dashboard* pada bagian status sistem. Informasi yang ditampilkan adalah status Waspada dengan indikator warna kuning serta keterangan bahwa sistem beroperasi

normal. Di sisi kanan tampak jendela *Command Prompt* yang menjalankan perintah *mosquitto_sub* pada topik *fire/status*.

Berdasarkan Tabel 4.6, hasilnya terlihat bahwa setiap status yang dikirimkan oleh ESP32 berhasil diterima secara bergantian, mulai dari darurat, bahaya, normal, hingga waspada. Hal ini mengindikasikan bahwa komunikasi antara perangkat ESP32 dengan *broker* MQTT berjalan dengan baik dan dapat menyalurkan data status secara *real-time* tanpa adanya *error* atau *error* 0%.

Tabel 4.6 Pengujian Pengiriman Data Status ESP32 ke MQTT

Pengujian Ke-	Data ESP32	Data MQTT	*Error*
1	Normal	Normal	0%
2	Waspada	Waspada	0%
3	Waspada	Waspada	0%
4	Bahaya	Bahaya	0%
5	Normal	Normal	0%
6	Waspada	Waspada	0%
7	Normal	Normal	0%
8	Normal	Normal	0%
9	Darurat	Darurat	0%
10	Waspada	Waspada	0%
11	Darurat	Darurat	0%
12	Waspada	Waspada	0%
13	Waspada	Waspada	0%
14	Normal	Normal	0%
15	Darurat	Darurat	0%
16	Darurat	Darurat	0%
17	Darurat	Darurat	0%
18	Darurat	Darurat	0%
19	Darurat	Darurat	0%
20	Waspada	Waspada	0%



Gambar 4.11 Pengujian Pengiriman Data Sensor Aktif ke MQTT

Gambar 4.11 memperlihatkan *card* Sensor Aktif pada *dashboard* yang menampilkan angka 1 sebagai jumlah sensor yang sedang terdeteksi aktif. Data ini sejalan dengan hasil *subscriber* di *Command Prompt* pada topik *fire/sensors/active*, yang juga menunjukkan angka 1.

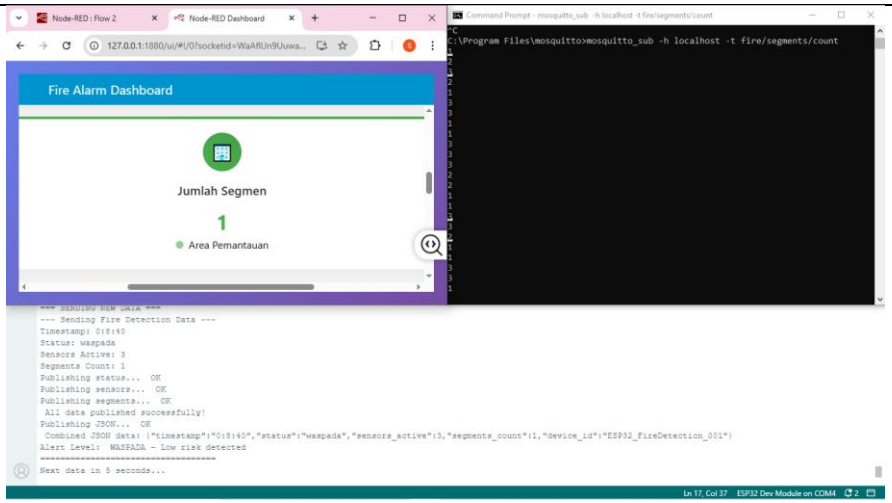
Berdasarkan Tabel 4.7 dengan 20 kali pengujian, sistem berhasil mengirimkan data deteksi kebakaran ke broker MQTT dan menampilkannya di *dashboard* Node-RED secara konsisten tanpa adanya kegagalan pengiriman. Hal ini menunjukkan bahwa komunikasi antara ESP32 dan broker berjalan stabil serta mampu merepresentasikan kondisi sensor di lapangan. Pada versi idealnya, sistem tidak hanya menampilkan data secara *real-time*, tetapi juga dilengkapi akses pemantauan jarak jauh, serta fitur notifikasi otomatis.

Tabel 4.7 Pengujian Pengiriman Data Sensor ESP32 ke MQTT

Pengujian Ke-	Data ESP32	Data MQTT	*Error*
1	3	3	0%
2	0	0	0%
3	6	6	0%
4	5	5	0%
5	6	6	0%

Tabel Lanjutan 4.7 Pengujian Pengiriman Data Sensor ESP32 ke MQTT

Pengujian Ke-	Data ESP32	Data MQTT	*Error*
6	4	4	0%
7	6	6	0%
8	7	7	0%
9	5	5	0%
10	4	4	0%
11	6	6	0%
12	6	6	0%
13	4	4	0%
14	7	7	0%
15	6	6	0%
16	0	0	0%
17	7	7	0%
18	9	9	0%
19	0	0	0%
20	1	1	0%



Gambar 4.12 Pengujian Pengiriman Data Segmen Aktif ke MQTT

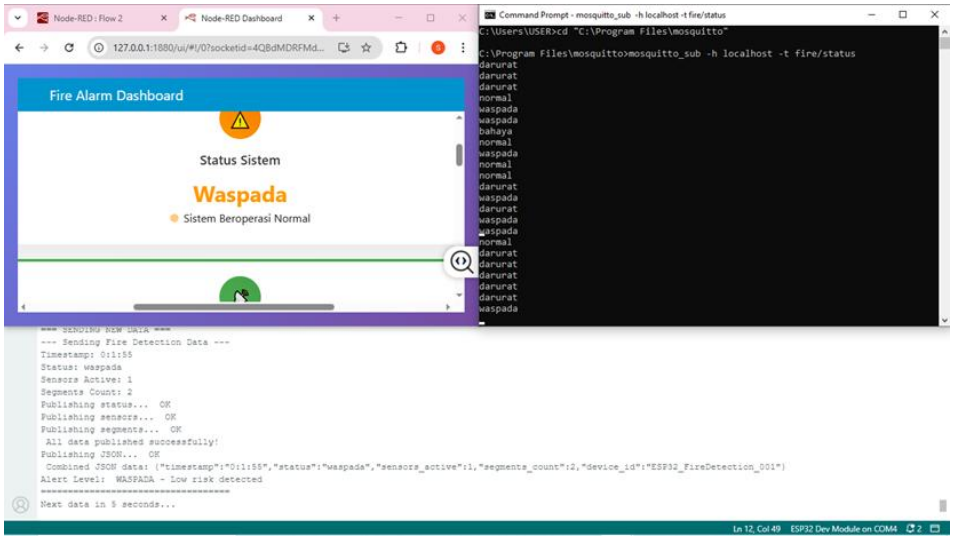
Gambar 4.12 menampilkan *card* Jumlah Segmen yang menunjukkan angka 1 pada *dashboard*, sesuai dengan data yang dikirimkan melalui topik *fire/segments_count*. Hasil *subscriber* di *Command Prompt* juga

konsisten dengan nilai yang sama. Dengan demikian, dapat dipastikan bahwa data jumlah area pemantauan berhasil dikirim dan ditampilkan dengan benar.

Secara keseluruhan, ketiga gambar tersebut memperlihatkan bahwa pengiriman data dari ESP32 menuju *broker* MQTT dan penyajiannya pada *Node-RED Dashboard* berlangsung sesuai rancangan. Data dapat dipublikasikan dan disubskripsi pada topik yang berbeda tanpa gangguan.

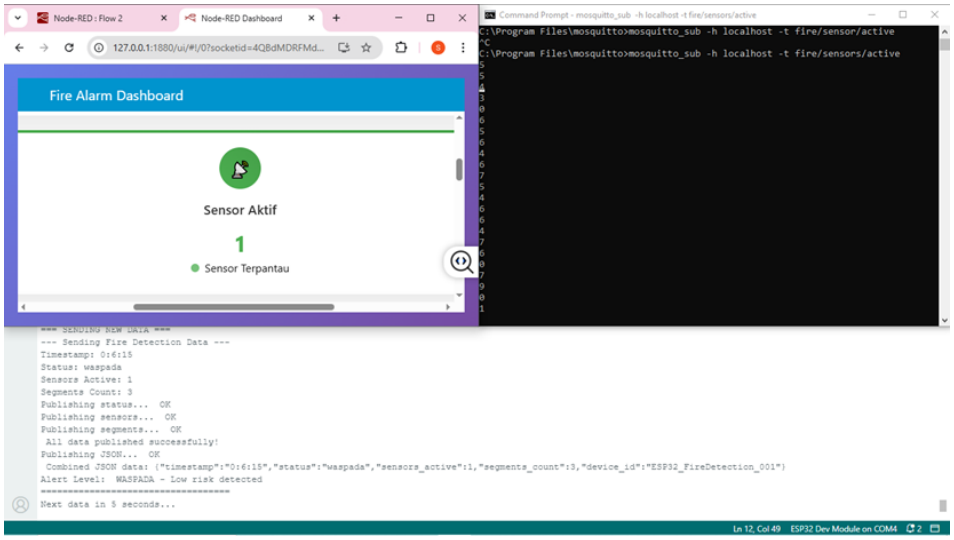
4.5. Pengujian Komunikasi ESP32 ke MQTT *Broker* dan *Node-RED*

Pengujian pengiriman data dari ESP32 ke *broker* MQTT menunjukkan bahwa sistem telah berfungsi dengan baik. Data dikirimkan melalui tiga topik utama, yaitu *fire/status* untuk informasi status sistem, *fire/sensors/active* untuk jumlah sensor yang aktif, serta *fire/segments_count* untuk jumlah segmen pemantauan. Hasil pengujian menggunakan *subscriber* pada *Command Prompt* dengan perintah mosquitto_sub memperlihatkan bahwa setiap data yang dikirim ESP32 dapat diterima secara *real-time* tanpa keterlambatan, baik berupa teks status (normal, waspada, bahaya, darurat) maupun angka jumlah sensor dan segmen yang terpantau. Selain itu, integrasi dengan *Node-RED Dashboard* berhasil menampilkan data dalam bentuk visual, seperti indikator status sistem, jumlah sensor aktif, dan jumlah segmen pemantauan yang sesuai dengan data yang diterima dari *broker*.



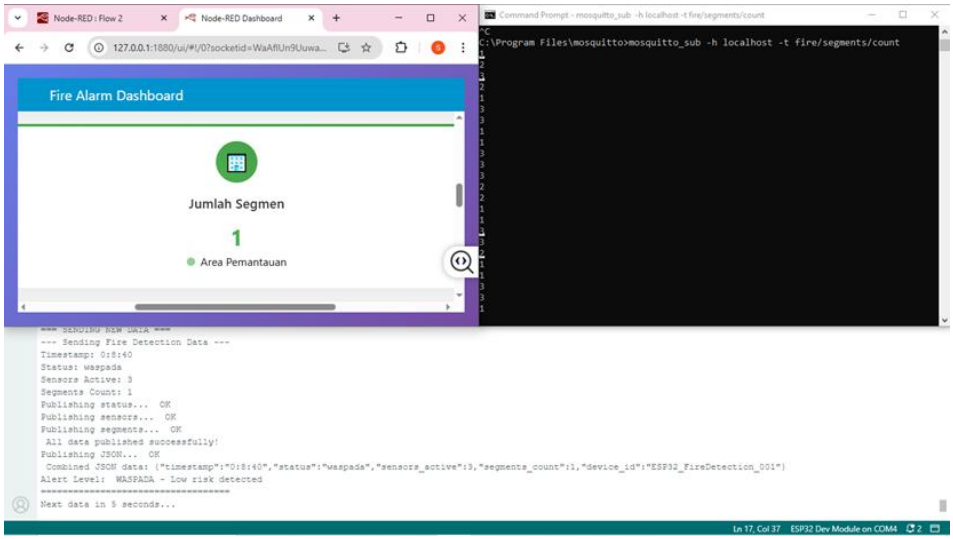
Gambar 4.13 Pengujian Komunikasi Data Status ESP32 ke MQTT dan Node-RED

Gambar 4.13 menunjukkan tampilan *Node-RED Dashboard* pada bagian status sistem. Informasi yang ditampilkan adalah status Waspada dengan indikator warna kuning serta keterangan bahwa sistem beroperasi normal. Di sisi kanan tampak jendela *Command Prompt* yang menjalankan perintah mosquitto_sub pada topik *fire/status*. Hasilnya terlihat bahwa setiap status yang dikirimkan oleh ESP32 berhasil diterima secara bergantian, mulai dari darurat, bahaya, normal, hingga waspada.



Gambar 4.14 Pengujian Komunikasi Data Sensor ESP32 ke MQTT dan Node-RED

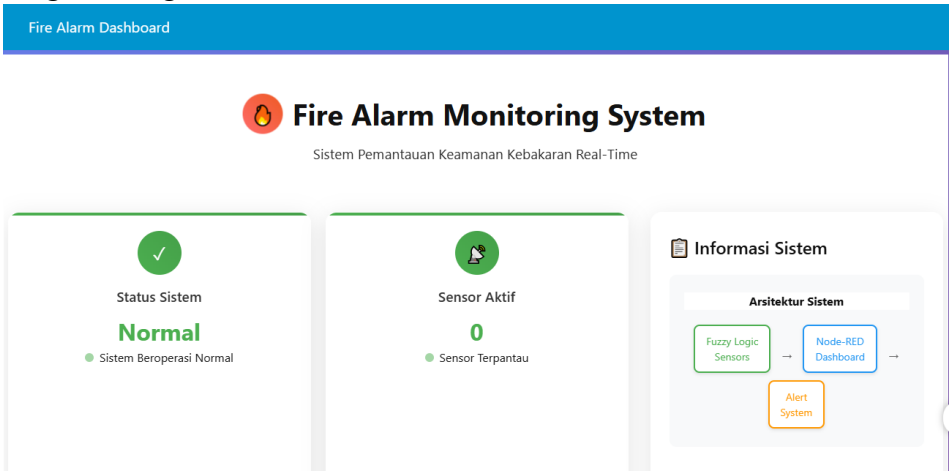
Gambar 4.14 memperlihatkan *card* Sensor Aktif pada *dashboard* yang menampilkan angka 1 sebagai jumlah sensor yang sedang terdeteksi aktif. Data ini sejalan dengan hasil *subscriber* di *Command Prompt* pada topik *fire/sensors/active*, yang juga menunjukkan angka 1. Sinkronisasi antara data yang ditampilkan di *dashboard* dengan hasil *subscriber* membuktikan bahwa sistem mampu memonitor jumlah sensor aktif secara akurat.



Gambar 4.15 Pengujian Komunikasi Data Segmen ESP32 ke MQTT dan Node-RED

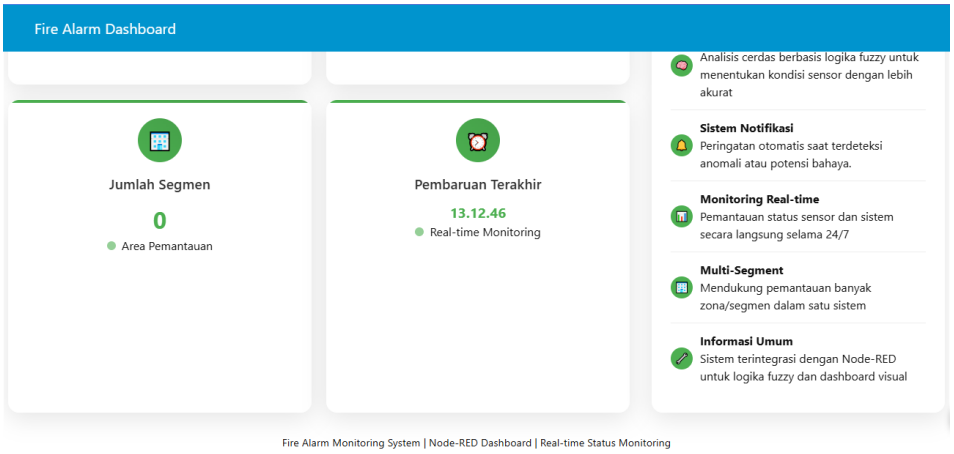
Gambar 4.15 menampilkan *card* Jumlah Segmen yang menunjukkan angka 1 pada *dashboard*, sesuai dengan data yang dikirimkan melalui topik *fire/segments_count*. Hasil *subscriber* di *Command Prompt* juga konsisten dengan nilai yang sama. Dengan demikian, dapat dipastikan bahwa data jumlah area pemantauan berhasil dikirim dan ditampilkan dengan benar.

4.6. Integrasi dengan *Dashboard Node-RED*



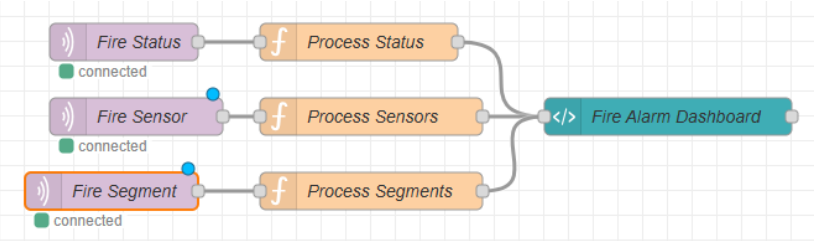
Gambar 4.16 Tampilan Utama *Dashboard Node-RED*

Pada Gambar 4.16 ditampilkan informasi status sistem, jumlah sensor yang aktif, jumlah segmen pemantauan, serta waktu pembaruan terakhir. Status sistem divisualisasikan dalam bentuk indikator hijau dengan keterangan "Normal" ketika sistem beroperasi dalam kondisi tanpa deteksi kebakaran. Bagian Sensor Aktif menampilkan jumlah sensor yang terpicu pada setiap kondisi kebakaran. Selain itu, panel Jumlah Segmen menunjukkan area pemantauan yang sedang dalam pengawasan, yang dalam sistem ini terbagi menjadi beberapa segmen sesuai konfigurasi *smoke detector*. Informasi pembaruan terakhir ditampilkan dalam format waktu *real-time* sehingga pengguna dapat memastikan bahwa data yang tersaji adalah hasil *monitoring* terkini.



Gambar 4.17 Informasi Dashboard Node-RED

Pada sisi kanan *dashboard* di Gambar 4.17 juga tersedia bagian Informasi Sistem yang memuat arsitektur alur data mulai dari sensor yang terhubung dengan logika *fuzzy* pada ESP32, kemudian diteruskan ke *Node-RED* sebagai pengolah data dan penyaji antarmuka. Penjelasan tambahan mengenai fitur sistem, seperti notifikasi, *monitoring real-time*, dukungan multi-segmen, serta penyajian informasi umum, turut ditampilkan untuk memberikan gambaran menyeluruh mengenai kapabilitas sistem.



Gambar 4.18 *Flow Node*

Berdasarkan hasil implementasi pada *Node-RED* seperti Gambar 4.18, alur data sistem *alarm* kebakaran terdiri dari tiga topik utama yang diterima melalui MQTT, yaitu status kebakaran, jumlah sensor aktif, dan jumlah segmen. Masing-masing data diproses terlebih dahulu oleh *function node* sebelum ditampilkan pada antarmuka *dashboard*. *Node Process Status* berfungsi memetakan berbagai kemungkinan input status seperti Normal, Waspada, Bahaya, atau Darurat ke dalam kategori yang sama untuk memudahkan visualisasi. *Node Process Sensors* bertugas memastikan jumlah sensor aktif terbaca dalam format numerik yang valid dengan batas maksimum tertentu, sedangkan *Node Process Segments* melakukan validasi jumlah segmen pemantauan yang dikirimkan oleh sistem.

Seluruh data hasil pemrosesan kemudian dikirim ke *Node Fire Alarm Dashboard* yang dirancang menggunakan elemen HTML dan CSS pada *ui_template*. *Dashboard* ini menampilkan status sistem dengan indikator visual berwarna, jumlah sensor yang sedang aktif, jumlah segmen yang dipantau, serta waktu pembaruan terakhir secara *real-time*.

Tampilan antarmuka juga dilengkapi dengan elemen peringatan visual berupa *alert banner* yang muncul secara otomatis ketika status berubah menjadi Bahaya atau Darurat.

4.7. Uji Coba dan Evaluasi Hasil Klasifikasi

Konversi dilakukan berdasarkan aturan sekitar 38.75 mA mewakili satu sensor aktif, sehingga jika arus 77.5 mA maka setara dua sensor, dan seterusnya. Data hasil parsing ini disimpan ke dalam struktur *SegmentData* yang memuat kondisi tiga segmen deteksi kebakaran.

Setelah data sensor diperoleh, *fuzzy* logic Sugeno orde 0 dijalankan melalui fungsi evaluate*Fuzzy*(). *Input fuzzy* terdiri dari tiga variabel yaitu jumlah total sensor aktif, jumlah segmen yang aktif, dan konsentrasi maksimum sensor dalam satu segmen. Fungsi keanggotaan yang digunakan meliputi kategori sedikit, sedang, banyak, sangat banyak untuk total sensor, serta rendah, sedang, tinggi untuk Terdistribusi segmen, dan rendah, tinggi untuk konsentrasi per segmen. Berdasarkan kombinasi fungsi keanggotaan, diterapkan aturan *fuzzy* IF-THEN yang menghasilkan *output* numerik konstan (orde 0). Hasil akhir kemudian diklasifikasikan menjadi status NORMAL, WASPADA, BAHAYA, atau DARURAT sesuai nilai keluaran.

Hasil klasifikasi *fuzzy* beserta data sensor kemudian dipublikasikan ke *broker* MQTT pada beberapa topik, yaitu *fire/status*, *fire*/*segments_count*, dan *fire/sensors/active*. Dengan demikian, setiap perubahan kondisi sensor dapat segera terlihat pada *dashboard Node-RED*. Agar tidak membanjiri *server*, sistem hanya mengirim data setiap interval 2 detik.

Evaluasi sistem dilakukan melalui serangkaian uji coba dengan berbagai kondisi simulasi kebakaran untuk menilai kinerja klasifikasi *fuzzy* logic Sugeno orde 0 yang diterapkan pada FACP. Uji coba dilakukan dengan memvariasikan jumlah *smoke detector* yang aktif, pola Terdistribusi antarsegmen, serta konsentrasi sensor yang mendeteksi asap.

Pada pengujian kondisi nyata dengan satu sensor aktif, hasil pembacaan pada *serial monitor* menunjukkan bahwa arus yang diterima dari STM32 dan dikirim ke ESP32 berada pada kisaran 44,2 mA hingga 45,28 mA. Berdasarkan data tersebut, sistem kemudian mengolah informasi dan mengidentifikasi bahwa terdapat satu sensor aktif pada segmen 1 dengan *output* WASPADA.

Pengujian dengan kondisi dua sensor aktif pada segmen 1 memperlihatkan hasil pembacaan arus melalui *serial monitor* berada pada rentang 88,91 mA hingga 89,61 mA. Data arus tersebut kemudian diproses oleh sistem sehingga dapat diklasifikasikan bahwa terdapat dua sensor yang aktif pada segmen 1 dengan *output* WASPADA.

Pada saat dilakukan pengujian dengan tiga sensor aktif di segmen 1, *serial monitor* menampilkan hasil pembacaan arus pada kisaran 129,7 mA hingga 131,89 mA. Nilai arus ini kemudian dikirim dan diolah oleh sistem, sehingga diinterpretasikan bahwa segmen 1 memiliki tiga sensor aktif dengan keluaran sistem berada pada status WASPADA.

Pengujian dengan kondisi empat sensor aktif di segmen 1 serta satu sensor aktif pada segmen 2 menghasilkan pembacaan arus pada *serial monitor* sebesar sekitar 131,89 mA untuk segmen 1 dan 44,81 mA untuk segmen 2. Data arus tersebut selanjutnya diproses oleh sistem dan diidentifikasi bahwa segmen 1 memiliki empat sensor aktif, sedangkan keluaran klasifikasi sistem menunjukkan status BAHAYA.

Berdasarkan Tabel 4.8, dapat dilihat bahwa hasil klasifikasi fuzzy menunjukkan pola yang konsisten dengan desain yang telah ditetapkan. Peningkatan jumlah sensor aktif (X), jumlah segmen terpantau (Y), maupun konsentrasi maksimum sensor dalam satu segmen (Z) berpengaruh langsung terhadap naiknya nilai *output fuzzy* dan status kedaruratan. Misalnya, pada kondisi tiga sensor yang terpusat dalam satu segmen menghasilkan *output* 2,0 dengan status WASPADA, sedangkan

Terdistribusi tiga sensor pada dua segmen menghasilkan *output* lebih tinggi yaitu 2,752 dengan status BAHAYA. Hal ini menunjukkan bahwa sistem menginterpretasikan Terdistribusi sensor pada lebih dari satu segmen sebagai kondisi yang lebih berisiko. Selain itu, hasil *output* yang berada pada rentang ambang, seperti 3,001 dan 3,251, juga tetap menghasilkan klasifikasi yang sesuai, yaitu BAHAYA. Pada kondisi dengan kombinasi sensor lebih banyak, seperti X=7, X=8, maupun X=9, sistem secara konsisten menghasilkan status DARURAT dengan *output* 3,625 hingga 4,0. Seluruh hasil pengujian ini sesuai dengan aturan klasifikasi pada desain awal, sehingga tingkat kesesuaian mencapai 100%. Dengan demikian, dapat disimpulkan bahwa implementasi logika *fuzzy* Sugeno pada sistem ini telah berjalan dengan baik, di mana klasifikasi tingkat kedaruratan yang dihasilkan selalu konsisten dengan rancangan, baik pada kondisi sensor Terkonsentrasi maupun Terdistribusi.

Tabel 4.8 Hasil Klasifikasi Sistem

X	*Y*	*Z*	*Output Fuzzy*	Status	Keterangan	Perbandingan dengan Desain
1	1	1	2	WASPADA	Terkonsentrasi	Sesuai
2	2	1	2,375	WASPADA	Terdistribusi (2 Segmen)	Sesuai
3	1	3	2	WASPADA	Terkonsentrasi	Sesuai
3	2	2	2,752	BAHAYA	Terdistribusi (2 Segmen)	Sesuai
4	2	2	3,001	BAHAYA	Terdistribusi (2 Segmen)	Sesuai
4	3	2	3,251	BAHAYA	Terdistribusi (3 Segmen)	Sesuai
5	2	3	3,273	BAHAYA	Terdistribusi (2 Segmen)	Sesuai

Tabel Lanjutan 4.8 Hasil Klasifikasi Sistem

X	*Y*	*Z*	*Output Fuzzy*	Status	Keterangan	Perbandingan dengan Desain
5	3	2	3,43	BAHAYA	Terdistribusi (3 Segmen)	Sesuai
6	2	3	3,333	BAHAYA	Terkonsentrasi	Sesuai
7	2	3	3,625	DARURAT	Terdistribusi (2 Segmen)	Sesuai
8	3	3	4	DARURAT	Terkonsentrasi	Sesuai
8	3	3	4	DARURAT	Terdistribusi (3 Segmen Max 3)	Sesuai
9	3	3	4	DARURAT	Terkonsentrasi	Sesuai
9	2	3	4	DARURAT	Terdistribusi (2 Segmen)	Sesuai

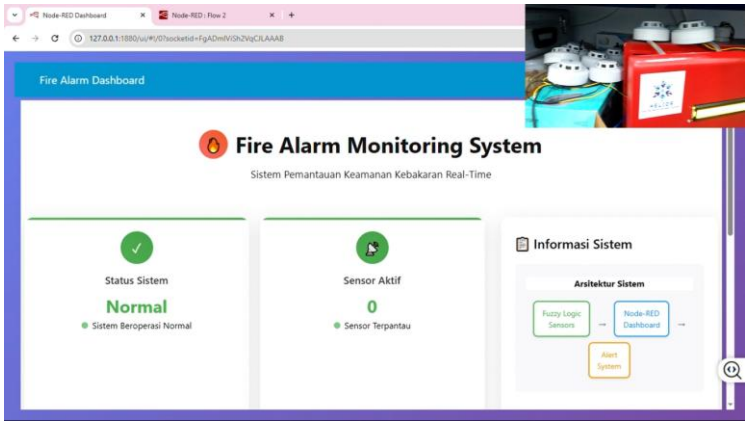
4.8. Pengujian Integrasi Keseluruhan Sistem

Pengujian integrasi keseluruhan sistem dilakukan untuk memastikan bahwa setiap komponen yang telah dirancang dan diimplementasikan dapat berfungsi secara terpadu sesuai dengan tujuan sistem FACP berbasis *fuzzy logic*. Komponen utama yang diuji meliputi *smoke detector* sebagai sensor *input*, modul STM32 sebagai pengolah awal sinyal arus sensor, modul ESP32 sebagai pemroses *fuzzy logic* Sugeno, aktuator berupa *buzzer*, relay, dan *doorlock*, serta antarmuka pemantauan berbasis *dashboard Node-RED*.

Proses pengujian dimulai dengan mengaktifkan *smoke detector* pada kondisi simulasi kebakaran. Sinyal keluaran sensor berupa perubahan arus diterima dan diproses oleh STM32, kemudian dikirimkan ke ESP32 melalui komunikasi serial UART. Program pada ESP32 dibuat sebagai penghubung antara mikrokontroler STM32F4 yang membaca data sensor *smoke detector* dengan *server* MQTT untuk kebutuhan *monitoring* melalui *dashboard*. Alur utama kode dimulai dari inisialisasi *WiFi* dan MQTT, di mana ESP32 dikonfigurasi untuk terhubung ke jaringan *WiFi* dengan SSID dan *password*

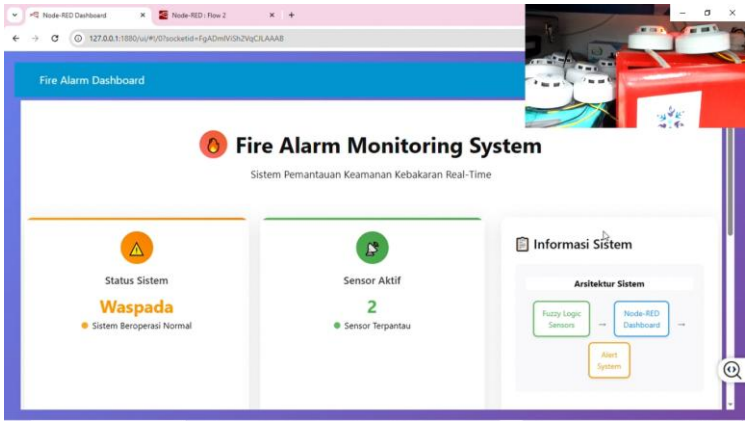
tertentu. Setelah koneksi *WiFi* berhasil, modul juga melakukan koneksi ke IP *broker* MQTT pada port 1883. Status koneksi *WiFi* dan MQTT dipantau secara berkala, dan jika terjadi putus, sistem akan melakukan proses *reconnect* secara otomatis. ESP32 mengubah nilai arus menjadi jumlah sensor aktif pada setiap segmen, kemudian melakukan klasifikasi tingkat kedaruratan menggunakan metode *fuzzy logic* Sugeno orde 0. *Output* berupa kategori status Normal, Waspada, Bahaya, atau Darurat, sekaligus diteruskan ke broker MQTT untuk ditampilkan secara *real-time* pada *dashboard Node-RED*.

Selain pengiriman data, sistem juga memiliki fungsi *monitoring* internal yang ditampilkan melalui *serial monitor*. Informasi ini meliputi status koneksi *WiFi*, status koneksi MQTT, data terakhir yang diterima dari STM32, kondisi sensor pada tiap segmen, serta hasil perhitungan *fuzzy logic*. *Monitoring* ini penting untuk proses *debugging* dan validasi bahwa seluruh alur komunikasi bekerja sesuai rancangan.



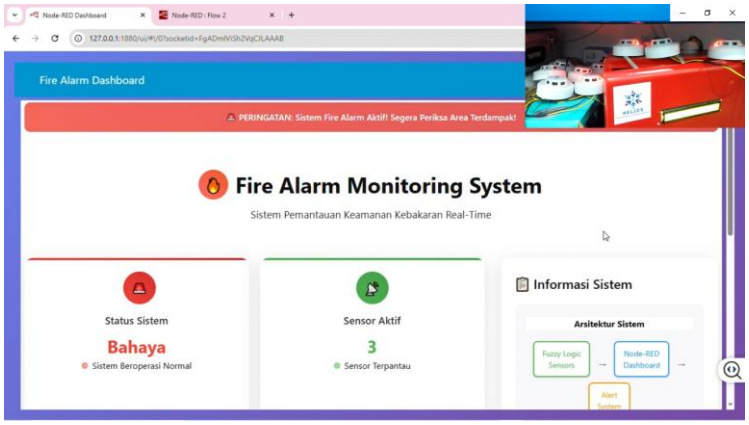
Gambar 4.19 Pengujian Sistem dengan *Output* NORMAL

Dari hasil pengujian pada Gambar 4.23, sistem mampu menghasilkan status Normal ketika tidak ada sensor aktif dan sesuai dengan perancangan.



Gambar 4.20 Pengujian Sistem dengan *Output* WASPADA

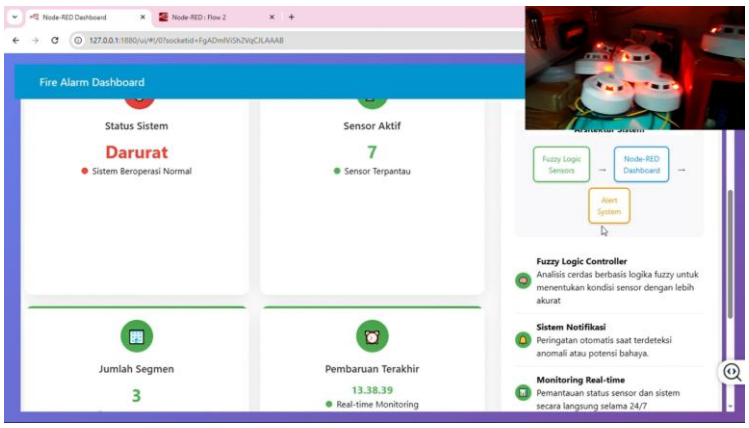
Terlihat pada Gambar 4.24 bahwa dua sensor di kanan atas dalam keadaan aktif atau terpicu oleh simulasi asap/nyala api, yang kemudian dikenali oleh sistem. Pada dashboard, sistem berhasil mendeteksi 2 sensor aktif, ditampilkan dalam panel "Sensor Aktif" sebagai 2 sensor terpantau, dan secara otomatis menghasilkan status sistem Waspada.



Gambar 4.21 Pengujian Sistem dengan *Output* BAHAYA

Pengujian yang ditampilkan pada Gambar 4.25 menunjukkan bahwa input dari sensor kebakaran pada bagian kanan atas berhasil diproses dengan baik oleh sistem. Pada saat pengujian, sebanyak 3 sensor terdeteksi aktif sehingga sistem *fuzzy* melakukan klasifikasi kondisi. Berdasarkan hasil inferensi *fuzzy*, status sistem ditampilkan dalam kondisi Bahaya. *Output* ini ditunjukkan pada *dashboard Node-RED* dengan indikator visual

berupa kotak merah dan peringatan teks “Sistem Fire Alarm Aktif! Segera Periksa Area Terdampak!”.



Gambar 4.22 Pengujian Sistem dengan *Output* DARURAT

Pada kondisi ini seperti Gambar 4.26, jumlah sensor yang aktif meningkat menjadi 7 sensor, yang menunjukkan deteksi kebakaran lebih luas dibandingkan pengujian sebelumnya. Hasil klasifikasi *fuzzy logic* menampilkan status sistem dalam kondisi Darurat, ditandai dengan indikator kotak merah serta tulisan peringatan pada *dashboard*. Hal ini menunjukkan bahwa sistem dapat menyesuaikan *output* klasifikasi sesuai dengan tingkat kebakaran yang disimulasikan.

Pengujian *latency* dilakukan untuk mengetahui waktu tunda (*end-to-end delay*) dari proses pengiriman data mulai dari sensor aktif hingga data diterima dan ditampilkan pada *dashboard Node-RED*.

Berdasarkan Tabel 4.9, dapat dilihat bahwa rata-rata waktu tunda dari saat sensor aktif hingga data berhasil ditampilkan pada *dashboard Node-RED* adalah 2,14, detik. *Latency* terendah diperoleh sebesar 1,38 detik, sedangkan latency tertinggi mencapai 2,58 detik, yang terjadi pada saat pertama kali sistem diinisialisasi. Setelah sistem berada di kondisi *standby*, *latency* berada di kisaran 1,38–2,58 detik, yang masih tergolong baik untuk kebutuhan *monitoring* kebakaran secara *real-time*.

Tabel 4.9 Hasil Pengujian Latency Sistem

Percobaan Ke-	Waktu *Latency* (Detik)
1	2,48
2	2,39
3	2,45
4	1,38
5	2,03
6	1,87
7	2,21
8	1,96
9	2,58
10	2,12
Rata-rata *Latency*	2,14
Latency Minimum	1,38
Latency Maksimum	2,58

Hasil ini menunjukkan bahwa sistem mampu menjaga kestabilan komunikasi data meskipun terdapat sedikit variasi waktu tunda. Perbedaan nilai *latency* terutama dipengaruhi oleh kondisi jaringan *WiFi*, proses antrian data pada *broker* MQTT, serta *refresh rate* pada *dashboard Node-RED*. Dengan rata-rata di bawah 3 detik, sistem dapat dikatakan cukup responsif untuk memberikan informasi kepada pengguna pada saat kondisi kebakaran terdeteksi.

Analisis akurasi dilakukan dengan membandingkan hasil klasifikasi sistem terhadap kondisi nyata simulasi. Data pengujian memperlihatkan bahwa sistem mampu memberikan hasil klasifikasi yang sesuai dengan skenario kebakaran yang dirancang, dengan tingkat akurasi sebesar 100%. Hasil evaluasi memperlihatkan bahwa sistem berbasis *fuzzy logic* tetap memberikan klasifikasi yang lebih fleksibel dibandingkan logika *if-then* sederhana, karena mampu mengakomodasi kondisi ketidakpastian dari *input* sensor.

Selain itu, ditemukan pula kendala pada saat sensor kembali ke kondisi normal. Status sistem pada *dashboard* memang berubah menjadi Normal, tetapi jumlah sensor aktif dan jumlah segmen tidak langsung kembali ke nilai nol, melainkan hanya turun hingga angka 1. Agar tampilan kembali normal sepenuhnya, pengguna perlu melakukan *refresh* pada halaman *dashboard*. Kondisi ini menunjukkan adanya masalah pada mekanisme pembaruan data di *Node-RED*, di mana variabel status berhasil diperbarui, tetapi variabel jumlah sensor dan segmen tidak sepenuhnya tersinkronisasi. Secara umum, hal tersebut tidak memengaruhi fungsi utama sistem dalam mendeteksi dan mengklasifikasikan kondisi kebakaran, namun berdampak pada akurasi tampilan informasi di *dashboard*.