Penerapan Digital Quantum
Mata Kuliah Elektronika Digital

Dosen: Ir. Dwi Oktavianto Wahyu Nugroho, S.T., M.T.

Oleh :

Mochamad Rafly Firmansyah

2042241130 – 3A

PRODI D4 TEKNOLOGI REKAYASA INSTRUMENTASI
DEPARTEMEN TEKNIK INSTRUMENTASI
FAKULTAS VOKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER

2025

Sistem Kontrol Lingkungan Otomatis untuk Ruang Penyimpanan Bahan Baku Makanan Berbasis Multi-Sensor Digital dan Aktuator Terintegrasi

Mochamad Rafly Firmansyah/2042241130/Teknik Instrumentasi/Elektronika Digital.
Ir. Dwi Oktavianto Wahyu Nugroho, S.T., M.T.

1. Latar Belakang

Bahan baku kering seperti tepung terigu, biji-bijian, rempah-rempah, kacang-kacangan, gula, dan susu bubuk memerlukan kondisi lingkungan yang stabil selama penyimpanan. Perubahan suhu, kelembapan, aliran udara, dan kualitas udara dapat memicu jamur, penggumpalan, penurunan mutu, hingga kerusakan bahan. Banyak ruang penyimpanan di industri makanan masih mengandalkan pemantauan manual, sehingga respons terhadap perubahan lingkungan sering terlambat dan kurang akurat.

Untuk menjaga kualitas bahan kering secara konsisten, diperlukan sistem kontrol lingkungan otomatis yang mampu memonitor parameter fisik secara real-time menggunakan multi-sensor digital. Sistem tersebut kemudian menyesuaikan kondisi ruangan melalui aktuator seperti exhaust fan, damper udara, humidifier, dehumidifier, dan pendingin. Pendekatan ini memungkinkan ruang penyimpanan tetap berada pada kondisi optimal dan mendukung kelancaran proses manufaktur.

2. Sensor Dalam Pengukuran Besaran Fisis

a) Temperatur Sensor (DS18B20)
- Besaran Fisis : Mengukur besaran fisis derajat (°)
- Referensi : Berdasarkan jurnal IEEE berjudul “Utilization of DS18B20 Temperature Sensor for Predictive Maintenance of Reciprocating Compressor” Pada jurnal ini, DS18B20 digunakan untuk membaca suhu kompresor, di mana tabel hasil pengukuran menampilkan suhu aktual yang dianggap sebagai baseline (kondisi normal) untuk kondisi mesin yang tidak mengalami masalah, serta suhu tinggi yang dianggap sebagai indikasi abnormal [1].
- 0 (Abnormal) : ±41.125°C – ±97.587°C
- 1 (Normal) : ±19.8°C – ±20.2°C

b) Humidity Sensor (SHT31)
- Besaran Fisis : Mengukur kelembapan relatif (%RH)
- Referensi : Berdasarkan jurnal IEEE “A Novel LoRaWAN-based Real-time Traffic Analysis Approach for Vehicle Congestion Estimation”, sensor SHT31 digunakan sebagai humidity sensor untuk mengukur parameter cuaca [2].
- 0 (Abnormal) : ≥ 69% Dianggap abnormal karena termasuk *external factors.*
- 1 (Normal) : ≤ 68% Tidak dikaitkan sebagai parameter yang memperburuk kondisi.

c) VOC Sensor (SGP30)
- Besaran Fisis : Mengukur konsentrasi VOC (Volatile Organic Compounds) dalam satuan mg/m³
- Referensi : Berdasarkan jurnal IEEE “Indoor Air Quality Monitors using IoT Sensors and LPWAN”, VOC diukur menggunakan sensor SGP30 [3].
- 0 (Abnormal) : VOC > 150
- 1 (Normal) : VOC ≤ 50

d) Dust/Particulate Sensor (PMS5003)
- Besaran Fisis : Konsentrasi particulate matter (PM2.5 dan PM10) dalam satuan µg/m³.

- Referensi : Berdasarkan jurnal IEEE berjudul "Air Pollutant Detection System Utilizing an IoT-based Electronic Nose for Air Purifier", sensor PMS5003 digunakan untuk mendeteksi particulate matter PM2.5 dan PM10 [4].
- 0 (Abnormal) : PM10 ≥ 25 μg/m³, PM2.5 ≥ 10 μg/m³
- 1 (Normal) : PM10 < 25 μg/m³, PM2.5 < 10 μg/m³

e) Airflow Sensor (MPXV7002DP)
- Besaran Fisis : Mengukur kecepatan aliran udara dalam satuan m/s.
- Referensi : Berdasarkan jurnal IEEE "Advancements in Microcontroller Technology for Wind Speed Measurement in Wind Tunnels", airflow diukur sebagai wind speed [5].
- 0 (Abnormal) : Tinggi dan tidak stabil ≥ 3.5 m/s, ≥ 4.2 m/s
- 1 (Normal) : Rendah dan stabil 1.8 m/s, 2.6 m/s

f) Light Intensity Sensor (BH1750)
- Besaran Fisis : Mengukur intensitas cahaya (illuminance) dalam satuan lux.
- Referensi : Berdasarkan jurnal "Prototype Design of Automatic Light Intensity Control in Smart Green House", sensor BH1750 digunakan untuk mengukur intensitas cahaya pada sistem greenhouse [6].
- 0 (Abnormal) : Kondisi Lux < 500
- 1 (Normal) : Kondisi Lux ≥ 500

3. Aktuator Dalam Pengukuran Besaran Fisis

a) Exhaust Fan
- IEEE PAPER: Prototype Design of Automatic Switching Speed of Exhaust Fan For Air Quality Control Based On IoT [7].
- AKSI FISIS : Mengatur pembuangan udara kotor secara mekanis dengan mengubah kecepatan putaran exhaust fan berdasarkan konsentrasi CO_2 (ppm).
- 0 (Idle/Off) : Tidak ada pembuangan udara 0 mA
- 1 (Aktif/On) : 500 – 600 ppm – speed 1 (1.3-1.4 mA), 600 – 700 ppm – Speed 2 (1.8 mA).

b) Inline Duct Fan
- IEEE PAPER: VENTI: experimental controller for inline duct fan [8].
- AKSI FISIS : Sebagai aktuator ventilasi untuk mengatur aliran udara dengan menghidupkan/mematikan kipas dan mengatur kecepatan kipas.
- 0 (Idle/Off) : Duct fan tidak menyala karena nilai vapor pressure berada di dalam ambang yang tidak memicu kontrol.
- 1 (Aktif/On) : Duct fan menyala, fan speed aktif karena nilai vapor pressure melewati ambang batas kontrol.

c) Humidifier
- IEEE PAPER: Design and Implementation of a Wireless Control Intelligent Humidifier [9].
- AKSI FISIS : Pada sistem ini, relay digunakan untuk mengatur *working gear* dari peralatan humidifikasi. Ketika sistem berjalan normal, gear humidifier akan diatur (otomatis atau manual) untuk menambah kelembapan udara ruangan berdasarkan data kelembapan yang diukur sensor.
- 0 (Abnormal): Humidifier Off / gear = 0
- 1 (Normal) : Humidifier aktif (gear > 0), relay mengatur gear sesuai mode otomatis/manual untuk menjaga kelembapan ruangan.

d) Dehumidifier
- IEEE PAPER: Automatic Usage of IoT-Based Dehumidifiers in High-Humidity Spaces [10].

- AKSI FISIS : Dehumidifier diaktifkan secara otomatis ketika tingkat kelembapan ruangan lebih tinggi dari batas yang telah ditentukan, dan dinonaktifkan ketika kelembapan berada di bawah batas tersebut.
- 0 (Idle/Off) : Dehumidifier Off ketika kelembapan dibawah *threshold.*
- 1 (Aktif/On) : Dehumdifier On ketika kelembapan melebihi *threshold.*

e) Cooling System
- IEEE PAPER: Design and Performance Evaluation of LH2 Cooling System for HTS Motor Electric Propulsion Platform [11].
- AKSI FISIS : Cooling system bekerja dengan mensirkulasikan Liquid Hydrogen (LH2) melalui sistem pendingin untuk menurunkan suhu komponen HTS motor. Sistem pendingin menjaga temperatur tetap rendah selama operasi.
- 0 (Idle/Off) : Cooling system tidak mensirkulasikan LH2
- 1 (Aktif/On) : Cooling system aktif mensirkulasikan LH2

f) LED Light System
- IEEE PAPER : A Spectrally Tunable Smart LED Lighting System With Closed-Loop Control [12].
- AKSI FISIS : LED bekerja sebagai aktuator pencahayaan dengan mengeluarkan intensitas cahaya (luminous flux) yang *ditingkatkan atau dikurangi* melalui sistem kontrol *closed-loop* untuk mencapai target. iluminansi.
- 0 (Idle/Off) : LED berada dalam kondisi tidak mengeluarkan cahaya atau di bawah level operasi.
- 1 (Aktif/On) : LED aktif menyala, mengeluarkan luminous flux sesuai kebutuhan sistem untuk mencapai target iluminansi.

4. Truth Tabel

No	S1	S2	S3	S4	S5	S6	A1	A2	A3	A4	A5	A6	Keterangan State
1	1	1	1	1	1	1	0	0	0	0	0	0	Semua normal / IDLE
2	0	1	1	1	1	1	0	0	0	0	1	0	Suhu tinggi → Cooling ON
3	1	0	1	1	1	1	0	0	1	0	0	0	Kelembapan tinggi → Dehumidifier ON
4	1	1	0	1	1	1	1	0	0	0	0	0	VOC tinggi → Exhaust Fan ON
5	1	1	1	0	1	1	1	0	0	0	0	0	Debu tinggi → Exhaust Fan ON
6	1	1	1	1	0	1	0	0	0	1	0	0	Airflow kurang → Inline Duct Fan ON
7	1	1	1	1	1	0	0	0	0	0	0	1	Cahaya kurang → LED System ON

8	0	1	0	1	1	1	1	0	0	0	1	0	Suhu tinggi + VOC tinggi
9	0	1	1	0	1	1	1	0	0	0	1	0	Suhu tinggi + Debu tinggi
10	0	1	1	1	0	1	0	0	0	1	1	0	Suhu tinggi + Airflow kurang
11	0	1	1	1	1	0	0	0	0	0	1	1	Suhu tinggi + Cahaya kurang
12	1	0	0	1	1	1	1	0	1	0	0	0	RH tinggi + VOC tinggi
13	1	0	1	0	1	1	1	0	1	0	0	0	RH tinggi + Debu tinggi
14	1	0	1	1	0	1	0	0	1	1	0	0	RH tinggi + Airflow kurang
15	1	0	1	1	1	0	0	0	1	0	0	1	RH tinggi + Cahaya kurang
16	1	1	0	0	1	1	1	0	0	0	0	0	VOC + Debu tinggi
17	1	1	0	1	0	1	1	0	0	1	0	0	VOC tinggi + Airflow kurang
18	1	1	0	1	1	0	1	0	0	0	0	1	VOC tinggi + Cahaya kurang
19	1	1	1	0	0	1	1	0	0	1	0	0	Debu tinggi + Airflow kurang
20	1	1	1	0	1	0	1	0	0	0	0	1	Debu tinggi + Cahaya kurang
21	1	1	1	1	0	0	0	0	0	1	0	1	Airflow kurang + Cahaya kurang
22	0	0	1	1	1	1	0	0	1	0	1	0	Suhu tinggi + RH tinggi
23	0	0	0	1	1	1	1	0	1	0	1	0	Suhu tinggi + RH tinggi + VOC tinggi
24	0	0	1	0	1	1	1	0	1	0	1	0	Suhu tinggi + RH tinggi + Debu tinggi
25	0	0	1	1	0	1	0	0	1	1	1	0	Suhu tinggi + RH tinggi + Airflow kurang

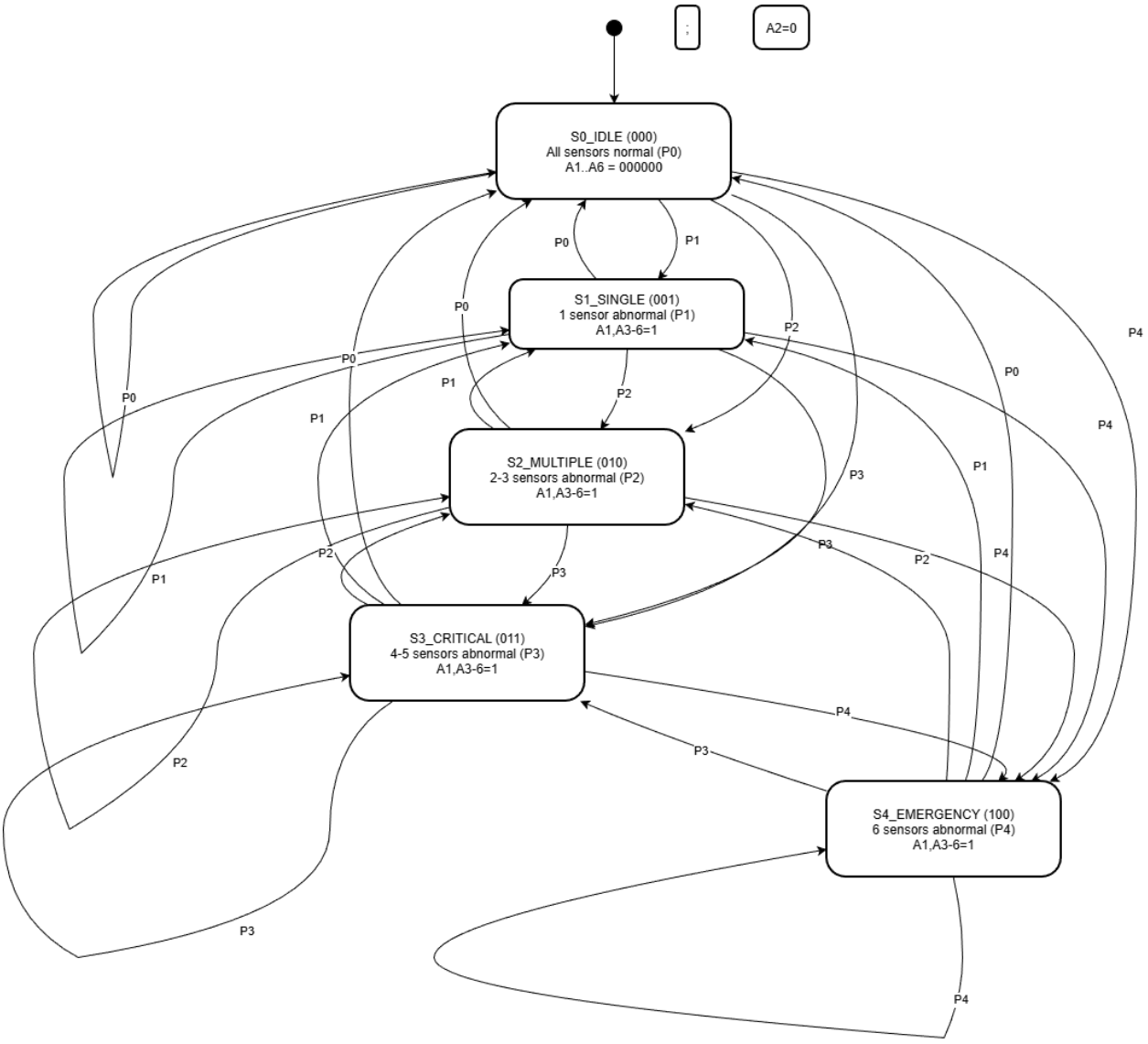
26	0	0	1	1	1	0	0	0	1	0	1	1	Suhu tinggi + RH tinggi + Cahaya kurang
27	0	1	0	0	1	1	1	0	0	0	1	0	Suhu tinggi + VOC + Debu tinggi
28	0	1	0	0	0	1	1	0	0	1	1	0	Suhu tinggi + VOC + Debu + Airflow kurang
29	0	1	1	0	0	0	1	0	0	1	1	1	Suhu tinggi + Debu + Airflow + Cahaya kurang
30	1	0	0	0	1	1	1	0	1	0	0	0	RH tinggi + VOC + Debu tinggi
31	1	0	0	0	0	1	1	0	1	1	0	0	RH tinggi + VOC + Debu + Airflow kurang
32	0	0	0	0	0	0	1	0	1	1	1	1	Semua sensor abnormal → semua aktuator proteksi menyala

1) Idle State
   - Semua sensor = 1 (normal).
   - Semua aktuator OFF (A1-A6 = 0).
   - Sistem standby, tidak ada tindakan koreksi.
   - Contoh: Baris 1 pada *truth table*
2) Single Sensor Abnormal – Minimal Alert
   - Hanya 1 sensor yang = 0 (abnormal)
   - Aktivasi terjadi pada aktuator yang berhubungan langsung dengan sensor itu
   - Sistem melakukan koreksi ringan
   - Contoh kasus:
     - Suhu tinggi – Cooling system ON
     - VOC tinggi – Exhaust Fan ON
     - Debu tinggi – Exhaust Fan ON
     - Airflow kurang – Inline Duct Fan ON
     - Cahaya kurang – LED ON
     - RH tinggi – Dehumidifier ON
       - Contoh: Baris 2-7 *truth table*
3) Multiple Sensors Abnormal – Moderate Correction
   - Terdapat 2-3 sensor yang abnormal
   - Beberapa aktuator aktif secara bersamaan untuk melakukan koreksi gabungan
   - Intensitas koreksi lebih besar dibanding single issue

- Contoh:
    - Suhu tinggi + VOC tinggi – Cooling + Exhaust
    - RH tinggi + Airflow kurang – Dehumidifier + Duct Fan
    - Debu tinggi + Cahaya kurang – Exhaust + LED
        - Contoh: Baris 8-20 *truth table*

4) Critical Condition – Banyak Sensor Bermasalah
    - 4-5 sensor = 0 (abnormal)
    - Hampir semua aktuator ON (sesuai pengaturan masing-masing)
    - Sistem berusaha mempertahankan kondisi ruang pada batas aman
    - Contoh:
        - Suhu tinggi + RH tinggi + VOC tinggi + Debu tinggi
        - Banyak polutan + airflow drop + suhu tinggi
            - Contoh: Baris 21-31 *truth table*
5) Emergency Mode – Semua Sensor Abnormal
    - Semua S1-S6 = 0
    - Semua aktuator proteksi diaktifkan (A1-A6 = 1)
    - Sistem bekerja dalam mode darurat untuk melindungi bahan makanan dari kerusakan berat
    - Exhaust, duct fan, cooling, LED, dan dehumidifier bekerja bersamaan
    - Contoh: Baris 32 *truth table*

5. Skema FSM Sistem Kontrol Lingkungan Ruang Penyimpanan Bahan Makanan



Definisi State

- S0 – IDLE State
  Semua sensor normal (S1–S6 = 1), semua aktuator OFF.
- S1 – Single Sensor Abnormal
  Tepat 1 sensor bernilai 0 (abnormal). Aktuator yang aktif hanya yang terkait dengan sensor tersebut.
- S2 – Multiple Sensors Abnormal (Moderate)
  Terdapat 2–3 sensor abnormal. Beberapa aktuator aktif secara bersamaan.
- S3 – Critical Condition
  Terdapat 4–5 sensor abnormal. Hampir seluruh aktuator proteksi aktif.
- S4 – Emergency Mode
  Semua sensor abnormal (S1–S6 = 0). Semua aktuator proteksi aktif.

Aturan Transisi Antar State

- Dari S0 (IDLE):

    - Jika 1 sensor menjadi 0 → pindah ke S1
    - Jika 2–3 sensor menjadi 0 → pindah ke S2
    - Jika 4–5 sensor menjadi 0 → pindah ke S3
    - Jika semua sensor 0 → pindah ke S4
- Dari S1 (Single Abnormal):
    - Jika semua sensor kembali 1 → S0
    - Jika jumlah sensor abnormal bertambah jadi 2–3 → S2
    - Jika bertambah jadi 4–5 → S3
    - Jika semua sensor 0 → S4
- Dari S2 (Multiple Abnormal):
    - Jika hanya tersisa 1 sensor abnormal → S1
    - Jika semua normal → S0
    - Jika bertambah jadi 4–5 abnormal → S3
- Dari S3 (Critical):
    - Jika semua sensor 0 → S4
    - Jika jumlah sensor abnormal berkurang (misalnya hanya 2–3) → S2
    - Jika hanya 1 sensor abnormal → S1
    - Jika semua normal → S0
- Dari S4 (Emergency):
    - Jika sebagian sensor mulai kembali normal (misalnya tinggal 4–5 abnormal) → S3
    - Jika berkurang jadi 2–3 abnormal → S2
    - Jika hanya satu abnormal → S1
    - Jika semua normal → S0

Transisi Balik (Recovery)

- Dari S1, S2, S3, S4 kembali ke S0 jika semua sensor kembali normal (S1–S6 = 1).
- Dari S4 → S3 → S2 → S1 → S0 mengikuti penurunan jumlah sensor abnormal.

Output Aktuator per State (Model Moore)

- S0 (IDLE):
  A1=0, A2=0, A3=0, A4=0, A5=0, A6=0
- S1 (Single Abnormal):
  Aktuator ON hanya yang sesuai dengan sensor yang 0 (Contoh: S1=0 → A5=1).
- S2 (Multiple Abnormal):
  Kombinasi beberapa aktuator ON sesuai kombinasi sensor abnormal.
- S3 (Critical):
  Mayoritas aktuator proteksi ON (A1, A3, A4, A5, A6 = 1, tergantung kombinasi spesifik).
- S4 (Emergency):
  Semua aktuator proteksi ON: A1=1, A3=1, A4=1, A5=1, A6=1.

6. Implementasi Memori State (Analisis Flip-Flop)

Untuk menyimpan 5 state FSM secara fisik, diperlukan elemen memori sekuensial.

1) Kuantitas Bit State
   Untuk merepresentasikan 5, jumlah bit minimum adalah:
   - $[Log_2(5)] = 3$ bits

- Maka dibutuhkan 3 buah Flip-Flop untuk menyimpan Current State *(Q2,Q1,Q0)*

Total kombinasi yang tersedia: $2^3 = 8$ state

Tetapi yang digunakan hanya 5, 3 sisanya dapat menjadi *unused state*.

2) Jenis Flip-Flop yang digunakan

Jenis flip-flop paling sesuai Adalah D-Type Flip-Flop (D-FF).

Alasannya:

- D-FF memetakan langsung ke arsitektur FPGA modern (slice LUT + DFF)
- Persamaan eksitasi paling sederhana:

$$D = Q_{next}$$

- Menyederhanakan sintesis dibanding JK-FF atau T-FF
- State machine digital modern, termasuk kontrol HVAC, hampir selalu menggunakan D-FF

State Name	Q2	Q1	Q0
S0 – IDLE (Semua sensor normal)	0	0	0
S1 – Single Abnormal	0	0	1
S2 – Multiple Abnormal	0	1	0
S3 – Critical Condition	0	1	1
S4 – Emergency Mode	1	0	0
(unused)	1	0	1
(unused)	1	1	0
(unused)	1	1	1

Kesimpulan Teknis, sinyal RESET akan memaksa state kembali ke:

$$Q_2Q_1Q_0 = 000\ (State\ S0 - IDLE)$$

7. Representasi Matriks dari Truth Table dan Flip-Flop

- Representasi Vektro State

FSM menggunakan 3 Flip-Flop (Q2, Q1, Q0) sehingga state direpresentasikan sebagai vektor kolom:

$$|Q\rangle = \begin{bmatrix} Q_2 \\ Q_1 \\ Q_0 \end{bmatrix}$$

Dengan encoding state berdasarkan Step 4:

State	Makna	Q2	Q1	Q0
S0	IDLE	0	0	0
S1	Single Abnormal	0	0	1
S2	Multiple Abnormal	0	1	0
S3	Critical	0	1	1
S4	Emergency	1	0	0

- Dari *Truth Table* step 2, jumlah sensor abnormal ditandai sebagai kategori berikut:

Kategori	Jumlah Sensor Abnormal	Nama
P0	N = 0	IDLE Condition
P1	N = 1	Single Abnormal
P2	N = 2–3	Multiple Abnormal
P3	N = 4–5	Critical
P4	N = 6	Emergency

Kategori dituliskan sebagai vector one-hot:

$$|Input\rangle = \begin{bmatrix} P0 \\ P1 \\ P2 \\ P3 \\ P4 \end{bmatrix}$$

Dimana nilai setiap Pi adalah 1 jika kondisi tersebut terpenuhi, dan 0 jika tidak.

Truth Table mendefinisikan hubungan:

$$|Q^{+}\rangle = M \cdot |Input\rangle$$

Dimana $|Q^+\rangle$ Adalah *Next State* dan M Adalah matrix 3×5.
Berdasarkan kategori – *next state:*

- P0 → 000 (S0)
- P1 → 001 (S1)
- P2 → 010 (S2)
- P3 → 011 (S3)
- P4 → 100 (S4)

Maka matriks transisi adalah:

$$M = \begin{bmatrix} 0 & 0 & 00 & 1 \\ 0 & 0 & 11 & 0 \\ 0 & 1 & 01 & 0 \end{bmatrix}$$

- Interpretasi kolom:

Kolom	Kategori	Next State
1	P0	0
2	P1	1
3	P2	10
4	P3	11
5	P4	100

Sehingga:

$$|Q^+\rangle = M|Input\rangle$$

- Hubungan Dengan D Flip-Flop

Karena menggunakan D Flip-Flop:

$$D = Q^+$$

Maka vektor input Flip-Flop adalah:

$$|D\rangle = |Q^+\rangle = M|Input\rangle$$

- Hubungan ini direpresentasikan oleh matrix identitas:

$$|D\rangle = I_3|Q^+\rangle = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} |Q^+\rangle$$

8. Representasi Bra–Ket (Dirac Notation)

1) Representasi Basis State FSM

FSM menggunakan 3-bit sehingga state dapat ditulis sebagai ket:

State	Ket
S0 (IDLE)	000⟩
S1 (Single Abnormal)	001⟩
S2 (Multiple Abnormal)	010⟩
S3 (Critical)	011⟩
S4 (Emergency)	100⟩

State |101⟩, |110⟩, |111⟩ tidak digunakan dan dipetakan kembali ke |S0⟩.

2) Representasi Operator Output Ô

Output sistem tergantung state FSM. Dengan ket output 6-bit:

$$A1\ A2\ A3\ A4\ A5\ A6$$

Didefinisikan:

- |A_IDLE⟩ = |000000⟩
- |A_ALERT⟩ = |101111⟩

Sehingga:

Ô|S0⟩ = |000000⟩ Ô|S1⟩ = |101111⟩ Ô|S2⟩ = |101111⟩ Ô|S3⟩ = |101111⟩ Ô|S4⟩ = |101111⟩

Operator output dalam bentuk bra–ket:

Ô = |000000⟩⟨S0| + |101111⟩(⟨S1| + ⟨S2| + ⟨S3| + ⟨S4|)

3) Representasi Operator Transisi T̂

Kategori jumlah sensor abnormal (Step 2):

- P0 → IDLE (S0)
- P1 → S1
- P2 → S2
- P3 → S3
- P4 → S4

Operator transisi:
$\hat{T} = |S0\rangle\langle P0| + |S1\rangle\langle P1| + |S2\rangle\langle P2| + |S3\rangle\langle P3| + |S4\rangle\langle P4|$
Operator ini memetakan input sensor ke next state FSM.

4) Hubungan dengan D-Flip-Flop
Karena menggunakan D-FF:
- $|Q(t+1)\rangle = \hat{T}\,|P\rangle$
- $|A(t)\rangle = \hat{O}\,|Q(t)\rangle$

9. Gate yang Digunakan untuk Penyelesaian

Pada langkah ini ditentukan **jenis-jenis gate** yang digunakan untuk merealisasikan seluruh proses FSM: penentuan kategori sensor, pembentukan next state ($Q_2\ Q_1\ Q_0$), dan pembentukan output aktuator.

GATE YANG DIGUNAKAN:

1) NOT Gate (X Gate)
Digunakan untuk membalik nilai sensor saat menghitung kondisi ”sensor abnormal”
- Bentuk operasi:
$$X\,|\,0\rangle = |\,1\rangle,\ X\,|\,1\rangle = |\,0\rangle$$
- Dipakai untuk memperoleh: $\bar{S}_i$

2) AND Gate
Digunakan untuk mendeteksi kombinasi beberapa sensor abnormal.
- And 2-input, 3-input, 4-input, hingga 6-input.
- Contoh:
  - Semua sensor normal – AND NOT (S1..S6)
  - Semua sensor abnormal – AND (S1..S6)

3) OR Gate
Digunakan untuk menggabungkan beberapa kondisi yang menuju kategori atau state yang sama.
- Contoh:
  - OR seluruh sensor abnormal untuk kategori P1 (1 sensor error)
  - OR output dari beberapa AND untuk kategori P2 dan P3

4) BUFFER / Identity Gate (I Gate)
Gate yang mempertahankan bit apa adanya.
- Dipakai ketika bit Q_2 atau Q_1 atau Q_0 tidak mengalami perubahan khusus.
- Contoh:
  $Q_2 = P_4$

5) Ground / Constant Gate (0 Gate)
Memberikan nilai tetap 0.

- Digunakan untuk aktuator yang tidak dipakai (A2 = 0).

GATE UNTUK PEMBENTUKAN KATEGORI SENSOR (P0-P4)

Penentuan kategori sensor berdasarkan jumlah sensor abnormal menggunakan kombinasi gate:

- P0 (tidak ada sensor abnormal):
  AND dari seluruh NOT(Si)
- P1 (1 sensor abnormal):
  OR dari seluruh Si
- P2 (2–3 sensor abnormal):
  OR dari seluruh AND pasangan dan triplet

- P3 (4–5 sensor abnormal):
  OR dari AND 4-input dan 5-input
- P4 (6 sensor abnormal):
  AND (S1, S2, S3, S4, S5, S6)

GATE UNTUK PEMBENTUKAN NEXT STATE (Q2 Q1 Q0)
Dari hasil kategori sensor:

- Q_2 = P4 → BUFFER Gate
- Q_1 = OR (P2, P3) → OR Gate
- Q_0 = OR (P1, P3) → OR Gate

Semua nilai Q dikirim ke D-Flip-Flop.

GATE UNTUK PEMBENTUKAN OUTPUT AKTUATOR
Output berdasarkan state FSM:

- A1, A3, A4, A5, A6:
  OR (|S1⟩, |S2⟩, |S3⟩, |S4⟩)
- A2:
  Ground/Constant 0 Gate

10. Gate yang Digunakan untuk Penyelesaian

Langkah terakhir adalah mentranslasikan FSM yang telah disintesis (Step 7) ke dalam kode yang dapat di-deploy pada dua platform target: FPGA dan Microcontroller.

- Implementasi HDL untuk FPGA

```
// ======================================================================
// Sistem Kontrol Lingkungan Otomatis - Verilog HDL untuk FPGA
// Mochamad Rafly Firmansyah / 2042241130
// ======================================================================

module environmental_control_fsm (
    input wire clk,          // Clock signal
    input wire reset,        // Reset signal (active high)
    input wire S1,           // Temperature sensor (1=normal, 0=abnormal)
    input wire S2,           // Humidity sensor
    input wire S3,           // VOC sensor
    input wire S4,           // Dust sensor
    input wire S5,           // Airflow sensor
    input wire S6,           // Light sensor
    output reg A1,           // Exhaust Fan
    output reg A2,           // Inline Duct Fan (always 0)
    output reg A3,           // Humidifier
    output reg A4,           // Dehumidifier
```

```
    output reg A5,            // Cooling System
    output reg A6,            // LED Light System
    output reg [2:0] current_state  // Current FSM state (for monitoring)
);

    // State encoding
    localparam S0_IDLE = 3'b000;      // All sensors normal
    localparam S1_SINGLE = 3'b001;    // 1 sensor abnormal
    localparam S2_MULTIPLE = 3'b010;  // 2-3 sensors abnormal
    localparam S3_CRITICAL = 3'b011;  // 4-5 sensors abnormal
    localparam S4_EMERGENCY = 3'b100; // All sensors abnormal

    // Internal registers
    reg [2:0] next_state;
    reg [2:0] abnormal_count;

    // Count abnormal sensors
    always @(*) begin
        abnormal_count = 3'b000;
        if (!S1) abnormal_count = abnormal_count + 1;
        if (!S2) abnormal_count = abnormal_count + 1;
        if (!S3) abnormal_count = abnormal_count + 1;
        if (!S4) abnormal_count = abnormal_count + 1;
        if (!S5) abnormal_count = abnormal_count + 1;
        if (!S6) abnormal_count = abnormal_count + 1;
    end

    // Next State Logic
    always @(*) begin
        case (abnormal_count)
            3'd0: next_state = S0_IDLE;
            3'd1: next_state = S1_SINGLE;
            3'd2, 3'd3: next_state = S2_MULTIPLE;
            3'd4, 3'd5: next_state = S3_CRITICAL;
            3'd6: next_state = S4_EMERGENCY;
            default: next_state = S0_IDLE;
        endcase
    end

    // State Register (D Flip-Flops)
    always @(posedge clk or posedge reset) begin
        if (reset)
            current_state <= S0_IDLE;
        else
            current_state <= next_state;
    end

    // Output Logic (Moore Machine)
```

```
always @(*) begin
   // Default: all actuators OFF
   A1 = 0; A2 = 0; A3 = 0; A4 = 0; A5 = 0; A6 = 0;

   case (current_state)
      S0_IDLE: begin
         // All OFF (already set)
      end

      S1_SINGLE: begin
         // Activate specific actuator based on which sensor is abnormal
         if (!S1) A5 = 1;      // Temperature -> Cooling
         if (!S2) A3 = 1;      // Humidity -> Dehumidifier
         if (!S3) A1 = 1;      // VOC -> Exhaust Fan
         if (!S4) A1 = 1;      // Dust -> Exhaust Fan
         if (!S5) A4 = 1;      // Airflow -> Inline Duct Fan
         if (!S6) A6 = 1;      // Light -> LED System
      end

      S2_MULTIPLE: begin
         // Multiple actuators active
         if (!S1) A5 = 1;
         if (!S2) A3 = 1;
         if (!S3) A1 = 1;
         if (!S4) A1 = 1;
         if (!S5) A4 = 1;
         if (!S6) A6 = 1;
      end

      S3_CRITICAL: begin
         // Most protection actuators active
         if (!S1) A5 = 1;
         if (!S2) A3 = 1;
         if (!S3) A1 = 1;
         if (!S4) A1 = 1;
         if (!S5) A4 = 1;
         if (!S6) A6 = 1;
      end

      S4_EMERGENCY: begin
         // All protection actuators ON
         A1 = 1;  // Exhaust Fan
         A3 = 1;  // Dehumidifier
         A4 = 1;  // Inline Duct Fan
         A5 = 1;  // Cooling System
         A6 = 1;  // LED System
         A2 = 0;  // Always OFF
      end
```

```
            default: begin
                A1 = 0; A2 = 0; A3 = 0; A4 = 0; A5 = 0; A6 = 0;
            end
        endcase
    end

endmodule

// ============================================================================
// TESTBENCH untuk Simulasi
// ============================================================================

module tb_environmental_control;
    reg clk;
    reg reset;
    reg S1, S2, S3, S4, S5, S6;
    wire A1, A2, A3, A4, A5, A6;
    wire [2:0] current_state;

    // Instantiate the module
    environmental_control_fsm uut (
        .clk(clk),
        .reset(reset),
        .S1(S1), .S2(S2), .S3(S3), .S4(S4), .S5(S5), .S6(S6),
        .A1(A1), .A2(A2), .A3(A3), .A4(A4), .A5(A5), .A6(A6),
        .current_state(current_state)
    );

    // Clock generation (50 MHz = 20ns period)
    initial begin
        clk = 0;
        forever #10 clk = ~clk;
    end

    // Test scenarios
    initial begin
        // Initialize waveform dump
        $dumpfile("environmental_control.vcd");
        $dumpvars(0, tb_environmental_control);

        // Display header
        $display("Time\tState\tS1 S2 S3 S4 S5 S6\tA1 A2 A3 A4 A5 A6");
        $display("-------------------------------------------------------------");

        // Test Case 1: Reset
        reset = 1; S1=1; S2=1; S3=1; S4=1; S5=1; S6=1;
        #20 reset = 0;
```

```
        #20 $display("%0t\tS0\t%b %b %b %b %b %b\t%b %b %b %b %b %b",
                $time, S1,S2,S3,S4,S5,S6, A1,A2,A3,A4,A5,A6);

        // Test Case 2: Single sensor abnormal (Temperature)
        S1=0; S2=1; S3=1; S4=1; S5=1; S6=1;
        #40 $display("%0t\tS1\t%b %b %b %b %b %b\t%b %b %b %b %b %b",
                $time, S1,S2,S3,S4,S5,S6, A1,A2,A3,A4,A5,A6);

        // Test Case 3: Multiple sensors abnormal (Temp + Humidity)
        S1=0; S2=0; S3=1; S4=1; S5=1; S6=1;
        #40 $display("%0t\tS2\t%b %b %b %b %b %b\t%b %b %b %b %b %b",
                $time, S1,S2,S3,S4,S5,S6, A1,A2,A3,A4,A5,A6);

        // Test Case 4: Critical condition (4 sensors abnormal)
        S1=0; S2=0; S3=0; S4=0; S5=1; S6=1;
        #40 $display("%0t\tS3\t%b %b %b %b %b %b\t%b %b %b %b %b %b",
                $time, S1,S2,S3,S4,S5,S6, A1,A2,A3,A4,A5,A6);

        // Test Case 5: Emergency (All sensors abnormal)
        S1=0; S2=0; S3=0; S4=0; S5=0; S6=0;
        #40 $display("%0t\tS4\t%b %b %b %b %b %b\t%b %b %b %b %b %b",
                $time, S1,S2,S3,S4,S5,S6, A1,A2,A3,A4,A5,A6);

        // Test Case 6: Recovery to normal
        S1=1; S2=1; S3=1; S4=1; S5=1; S6=1;
        #40 $display("%0t\tS0\t%b %b %b %b %b %b\t%b %b %b %b %b %b",
                $time, S1,S2,S3,S4,S5,S6, A1,A2,A3,A4,A5,A6);

        #100 $finish;
    end

endmodule
```

- C# Simulasi Microcontroller

Kode ini ditujukan untuk kompilasi pada platform Microcontroller (misal, STM32 dengan .NET nanoFramework). Kode ini mengimplementasikan FSM sebagai class (objek) perangkat lunak, di mana state FSM di-update oleh method call (misal, Update()).

```
// ===================================================================
// Sistem Kontrol Lingkungan Otomatis - C# untuk Mikrocontroller
// Mochamad Rafly Firmansyah / 2042241130
// ===================================================================

using System;
using System.Threading;

namespace EnvironmentalControlSystem
{
```

```
// Enum untuk FSM States
public enum SystemState
{
    S0_IDLE = 0,        // 000 - All sensors normal
    S1_SINGLE = 1,      // 001 - 1 sensor abnormal
    S2_MULTIPLE = 2,    // 010 - 2-3 sensors abnormal
    S3_CRITICAL = 3,    // 011 - 4-5 sensors abnormal
    S4_EMERGENCY = 4    // 100 - All sensors abnormal
}

// Struktur untuk sensor inputs
public struct SensorInputs
{
    public bool S1_Temperature;  // 1 = normal, 0 = abnormal
    public bool S2_Humidity;
    public bool S3_VOC;
    public bool S4_Dust;
    public bool S5_Airflow;
    public bool S6_Light;

    public int CountAbnormal()
    {
        int count = 0;
        if (!S1_Temperature) count++;
        if (!S2_Humidity) count++;
        if (!S3_VOC) count++;
        if (!S4_Dust) count++;
        if (!S5_Airflow) count++;
        if (!S6_Light) count++;
        return count;
    }
}

// Struktur untuk actuator outputs
public struct ActuatorOutputs
{
    public bool A1_ExhaustFan;
    public bool A2_InlineDuctFan;  // Always 0
    public bool A3_Dehumidifier;
    public bool A4_InlineDuctFan2;
    public bool A5_CoolingSystem;
    public bool A6_LEDSystem;

    public void Reset()
    {
        A1_ExhaustFan = false;
        A2_InlineDuctFan = false;
        A3_Dehumidifier = false;
```

```
        A4_InlineDuctFan2 = false;
        A5_CoolingSystem = false;
        A6_LEDSystem = false;
    }

    public override string ToString()
    {
        return $"A1:{(A1_ExhaustFan ? 1 : 0)} A2:{(A2_InlineDuctFan ? 1 : 0)} " +
               $"A3:{(A3_Dehumidifier ? 1 : 0)} A4:{(A4_InlineDuctFan2 ? 1 : 0)} " +
               $"A5:{(A5_CoolingSystem ? 1 : 0)} A6:{(A6_LEDSystem ? 1 : 0)}";
    }
}

// Main FSM Controller Class
public class EnvironmentalControlFSM
{
    private SystemState currentState;
    private SensorInputs sensors;
    private ActuatorOutputs actuators;

    public SystemState CurrentState => currentState;
    public ActuatorOutputs Actuators => actuators;

    public EnvironmentalControlFSM()
    {
        Reset();
    }

    // Reset FSM to initial state
    public void Reset()
    {
        currentState = SystemState.S0_IDLE;
        actuators.Reset();
    }

    // Update sensor readings
    public void UpdateSensors(SensorInputs newSensors)
    {
        sensors = newSensors;
    }

    // State transition logic
    private SystemState DetermineNextState()
    {
        int abnormalCount = sensors.CountAbnormal();

        switch (abnormalCount)
        {
```

```
        case 0:
            return SystemState.S0_IDLE;
        case 1:
            return SystemState.S1_SINGLE;
        case 2:
        case 3:
            return SystemState.S2_MULTIPLE;
        case 4:
        case 5:
            return SystemState.S3_CRITICAL;
        case 6:
            return SystemState.S4_EMERGENCY;
        default:
            return SystemState.S0_IDLE;
    }
}

// Output logic based on current state (Moore Machine)
private void UpdateActuators()
{
    actuators.Reset();  // Start with all OFF

    switch (currentState)
    {
        case SystemState.S0_IDLE:
            // All actuators OFF
            break;

        case SystemState.S1_SINGLE:
            // Activate specific actuator based on abnormal sensor
            if (!sensors.S1_Temperature) actuators.A5_CoolingSystem = true;
            if (!sensors.S2_Humidity) actuators.A3_Dehumidifier = true;
            if (!sensors.S3_VOC) actuators.A1_ExhaustFan = true;
            if (!sensors.S4_Dust) actuators.A1_ExhaustFan = true;
            if (!sensors.S5_Airflow) actuators.A4_InlineDuctFan2 = true;
            if (!sensors.S6_Light) actuators.A6_LEDSystem = true;
            break;

        case SystemState.S2_MULTIPLE:
        case SystemState.S3_CRITICAL:
            // Multiple actuators active
            if (!sensors.S1_Temperature) actuators.A5_CoolingSystem = true;
            if (!sensors.S2_Humidity) actuators.A3_Dehumidifier = true;
            if (!sensors.S3_VOC) actuators.A1_ExhaustFan = true;
            if (!sensors.S4_Dust) actuators.A1_ExhaustFan = true;
            if (!sensors.S5_Airflow) actuators.A4_InlineDuctFan2 = true;
            if (!sensors.S6_Light) actuators.A6_LEDSystem = true;
            break;
```

```
            case SystemState.S4_EMERGENCY:
                // All protection actuators ON
                actuators.A1_ExhaustFan = true;
                actuators.A3_Dehumidifier = true;
                actuators.A4_InlineDuctFan2 = true;
                actuators.A5_CoolingSystem = true;
                actuators.A6_LEDSystem = true;
                actuators.A2_InlineDuctFan = false;  // Always OFF
                break;
        }
    }

    // Execute one FSM cycle
    public void Execute()
    {
        // State transition
        currentState = DetermineNextState();

        // Update outputs based on new state
        UpdateActuators();
    }

    // Get status string
    public string GetStatus()
    {
        return $"State: {currentState} | Sensors: T:{(sensors.S1_Temperature ? 1 : 0)} " +
               $"H:{(sensors.S2_Humidity ? 1 : 0)} V:{(sensors.S3_VOC ? 1 : 0)} " +
               $"D:{(sensors.S4_Dust ? 1 : 0)} A:{(sensors.S5_Airflow ? 1 : 0)} " +
               $"L:{(sensors.S6_Light ? 1 : 0)} | {actuators}";
    }
}

// ============================================================================
// PROGRAM SIMULASI
// ============================================================================
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("============================================================");
        Console.WriteLine("Sistem Kontrol Lingkungan Otomatis - Simulasi C#");
        Console.WriteLine("Mochamad Rafly Firmansyah / 2042241130");
        Console.WriteLine("============================================================\n");

        EnvironmentalControlFSM fsm = new EnvironmentalControlFSM();
        SensorInputs sensors = new SensorInputs();
```

```
// Test Case 1: All sensors normal (IDLE)
Console.WriteLine("TEST 1: All Sensors Normal (IDLE State)");
sensors.S1_Temperature = true;
sensors.S2_Humidity = true;
sensors.S3_VOC = true;
sensors.S4_Dust = true;
sensors.S5_Airflow = true;
sensors.S6_Light = true;
fsm.UpdateSensors(sensors);
fsm.Execute();
Console.WriteLine(fsm.GetStatus());
Thread.Sleep(1000);

// Test Case 2: Single sensor abnormal (Temperature)
Console.WriteLine("\nTEST 2: Temperature Abnormal (Single Alert)");
sensors.S1_Temperature = false;
fsm.UpdateSensors(sensors);
fsm.Execute();
Console.WriteLine(fsm.GetStatus());
Thread.Sleep(1000);

// Test Case 3: Multiple sensors abnormal
Console.WriteLine("\nTEST 3: Temperature + Humidity Abnormal (Multiple)");
sensors.S1_Temperature = false;
sensors.S2_Humidity = false;
fsm.UpdateSensors(sensors);
fsm.Execute();
Console.WriteLine(fsm.GetStatus());
Thread.Sleep(1000);

// Test Case 4: Critical condition
Console.WriteLine("\nTEST 4: 4 Sensors Abnormal (Critical)");
sensors.S3_VOC = false;
sensors.S4_Dust = false;
fsm.UpdateSensors(sensors);
fsm.Execute();
Console.WriteLine(fsm.GetStatus());
Thread.Sleep(1000);

// Test Case 5: Emergency mode
Console.WriteLine("\nTEST 5: All Sensors Abnormal (Emergency)");
sensors.S5_Airflow = false;
sensors.S6_Light = false;
fsm.UpdateSensors(sensors);
fsm.Execute();
Console.WriteLine(fsm.GetStatus());
Thread.Sleep(1000);
```

```
            // Test Case 6: Recovery to normal
            Console.WriteLine("\nTEST 6: Recovery to Normal");
            sensors.S1_Temperature = true;
            sensors.S2_Humidity = true;
            sensors.S3_VOC = true;
            sensors.S4_Dust = true;
            sensors.S5_Airflow = true;
            sensors.S6_Light = true;
            fsm.UpdateSensors(sensors);
            fsm.Execute();
            Console.WriteLine(fsm.GetStatus());

            Console.WriteLine("\n============================================================");
            Console.WriteLine("Simulasi Selesai. Press any key to exit...");
            Console.ReadKey();
        }
    }
}
```

11. Daftar Pustaka

[1] D. Bora, D. Singh, and B. Negi, "Utilization of DS18B20 Temperature Sensor for Predictive Maintenance of Reciprocating Compressor," in *2023 International Conference on Power Energy, Environment and Intelligent Control, PEEIC 2023*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 147–150. doi: 10.1109/PEEIC59336.2023.10450639.

[2] C. S. Priya and F. S. Francis, "A Novel LoRaWAn-based Real-time Traffic Analysis Approach for Vehicle Congestion Estimation," in *Winter Summit on Smart Computing and Networks, WiSSCoN 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/WiSSCoN56857.2023.10133856.

[3] Jithina Jose and T.Sasipraba, *Indoor air quality monitors using IOT sensors and LPWAN*. [IEEE], 2019.

[4] J. J. R. Balbin, A. J. G. De Guzman, and C. J. C. Rambuyon, "Air Pollutant Detection System Utilizing an IoT-based Electronic Nose for Air Purifier," in *Proceeding - ELTICOM 2022: 6th International Conference on Electrical, Telecommunication and Computer Engineering 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 136–140. doi: 10.1109/ELTICOM57747.2022.10037913.

[5] B. Junaidin, B. D. Adiputra, K. Hariyanto, L. R. Pinandhita, and D. Wahju Santoso, "Advancements in Microcontroller Technology for Wind Speed Measurement in Wind Tunnels," in *Proceedings - IEIT 2023: 2023 International Conference on Electrical and Information Technology*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 71–75. doi: 10.1109/IEIT59852.2023.10335512.

[6] S. N. Patrialova, T. Agasta, and I. N. Sari, "Prototype Design of Automatic Light Intensity Control in Smart Green House," in *2021 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation, ICAMIMIA 2021 - Proceeding*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 41–46. doi: 10.1109/ICAMIMIA54022.2021.9807698.

[7] A. F. Adziima, D. Febrianto, E. Ardiatmajaya, A. B. Putri Susanto, M. D. Susanti, and E. R. Fauzi, "Prototype Design of Automatic Switching Speed of Exhaust Fan For Air Quality Control Based On IoT," in *2021 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation, ICAMIMIA 2021 - Proceeding*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 114–119. doi: 10.1109/ICAMIMIA54022.2021.9809416.

[8] Paulo Abreu, Tiago Andrade, Rafael Tavares, Fernando Carneiro, Maria Teresa Restivo, and Vasco Peixoto de Freitas, *VENTI: experimental controller for inline duct fan*. IEEE, 2017.

[9] S. Qiaoyun, Y. Gong, Z. Bo, and Z. Yuefeng, "Design and Implementation of a Wireless Control Intelligent Humidifier," Institute of Electrical and Electronics Engineers (IEEE), Sep. 2025, pp. 1396–1400. doi: 10.1109/itaic64559.2025.11163186.

[10] A. K. Putri, A. Pramono, N. Anthony, M. J. Firnandi, F. F. Zebua, and I. B. A. Wijaya, "Automatic Usage of IoT-Based Dehumidifiers in High-Humidity Spaces," in *11th International Conference on ICT for Smart Society: Integrating Data and Artificial Intelligence for a Resilient and Sustainable Future Living, ICISS 2024 - Proceeding*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ICISS62896.2024.10751296.

[11] K. Kim, S. Hahn, B. Min, K. D. Sim, and S. Kim, "Design and Performance Evaluation of LH2 Cooling System for HTS Motor Electric Propulsion Platform," *IEEE Transactions on Applied Superconductivity*, vol. 35, no. 5, 2025, doi: 10.1109/TASC.2025.3529421.

[12] I. Chew, V. Kalavally, C. P. Tan, and J. Parkkinen, “A Spectrally Tunable Smart LED Lighting System with Closed-Loop Control,” *IEEE Sens J*, vol. 16, no. 11, pp. 4452–4459, Jun. 2016, doi: 10.1109/JSEN.2016.2542265.