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

1st Mochamad Rafly Firmansyah
Department of Instrumentation
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
2042241130@student.its.ac.id
https://orcid.org/0009-0002-2456-0419

2nd Fitri Adi I., S.T., M.T.
Department of Instrumentation
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
fiskandarianto@gmail.com
https://orcid.org/0009-0001-9106-2201

3rd Ir. Dwi Oktavianto W. N., S.T., M.T.
Department of Instrumentation
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
goldcells@gmail.com
https://orcid.org/0000-0002-6781-0732

***Abstract*—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.**

***Index Terms*—Kontrol Lingkungan, Multi-Sensor, FPGA, FSM, Otomatisasi Industri.**

I. 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.

II. 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): $\pm 41.125°\text{C}$ – $\pm 97.587°\text{C}$
- 1 (Normal): $\pm 19.8°\text{C}$ – $\pm 20.2°\text{C}$

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): $\geq 69\%$ Dianggap abnormal karena termasuk external factors.
- 1 (Normal): $\leq 68\%$ Tidak dikaitkan sebagai parameter yang memperburuk kondisi.

C. VOC Sensor (SGP30)

Besaran Fisis: Mengukur konsentrasi VOC (Volatile Organic Compounds) dalam satuan mg/m^3. 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/m^3. 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/m^3, PM2.5 ≥ 10 μg/m^3
- 1 (Normal): PM10 < 25 μg/m^3, PM2.5 < 10 μg/m^3

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

III. 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.

IV. Logika State

A. Idle State

Semua sensor = 1 (normal). Semua aktuator OFF (A1-A6 = 0). Sistem standby, tidak ada tindakan koreksi. Contoh: Baris 1 pada truth table.

Tabel I. 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

B. 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; dll. Contoh: Baris 2-7 truth table.

C. Multiple Sensors Abnormal – Moderate Correction

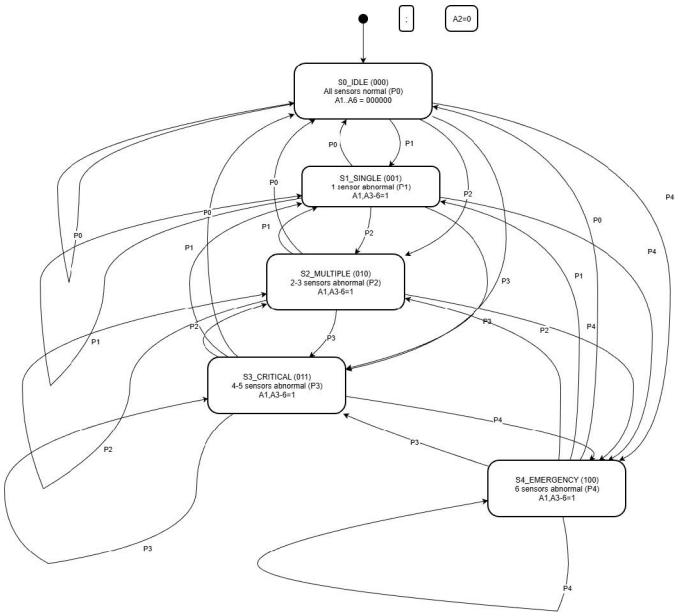
Terdapat 2-3 sensor yang abnormal. Beberapa aktuator aktif secara bersamaan untuk melakukan koreksi gabungan. Intensitas koreksi lebih besar dibanding single issue. Contoh: Baris 8-20 truth table.

D. Critical Condition – Banyak Sensor Bermasalah

4-5 sensor = 0 (abnormal). Hampir semua aktuator ON. Sistem berusaha mempertahankan kondisi ruang pada batas aman. Contoh: Baris 21-31 truth table.

E. Emergency Mode – Semua Sensor Abnormal

Semua S1-S6 = 0. Semua aktuator proteksi diaktifkan (A1-A6 = 1). Sistem bekerja dalam mode darurat. Contoh: Baris 32 truth table.



V. Definisi dan Transisi State

A. 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.

B. Aturan Transisi Antar State

Dari S0 (IDLE): Jika 1 sensor menjadi 0 $\rightarrow$ S1; Jika 2–3 sensor $\rightarrow$ S2; Jika 4–5 sensor $\rightarrow$ S3; Jika semua sensor 0 $\rightarrow$ S4.
Dari S1, S2, S3, S4 mengikuti logika jumlah sensor abnormal yang bertambah atau berkurang, dan transisi balik (Recovery) terjadi jika kondisi membaik.

C. Output Aktuator per State (Model Moore)

- S0: A1..A6 = 0
- S1: Aktuator ON hanya yang sesuai sensor.
- S2: Kombinasi beberapa aktuator ON.
- S3: Mayoritas aktuator proteksi ON.
- S4: Semua aktuator proteksi ON (A1, A3, A4, A5, A6 = 1).

VI. Implementasi Memori State

A. Kuantitas Bit State

Untuk menyimpan 5 state FSM secara fisik: $[\log_2(5)] = 3$ bits. Maka dibutuhkan 3 buah Flip-Flop (Q2,Q1,Q0). Total kombinasi 8 state, 5 digunakan, 3 unused.

B. Jenis Flip-Flop

Menggunakan D-Type Flip-Flop (D-FF) karena memetakan langsung ke arsitektur FPGA dan persamaan eksitasi paling sederhana ($D = Q_{next}$).

C. Representasi Matriks

FSM menggunakan 3 Flip-Flop, state direpresentasikan vektor kolom $|\mathbf{Q}\rangle = Q_2Q_1Q_0$. Kategori jumlah sensor abnormal:

- P0: N = 0 (IDLE)
- P1: N = 1 (Single)
- P2: N = 2–3 (Multiple)
- P3: N = 4–5 (Critical)
- P4: N = 6 (Emergency)

Matriks transisi M:

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Hubungan: $|\mathbf{Q}^+\rangle = M|\text{Input}\rangle$. Karena D-FF, maka $D = |\mathbf{Q}^+\rangle$.

D. Representasi Bra–Ket (Dirac Notation)

Basis State FSM: $|S_0\rangle = |000\rangle$, $|S_1\rangle = |001\rangle$, $|S_2\rangle = |010\rangle$, $|S_3\rangle = |011\rangle$, $|S_4\rangle = |100\rangle$. Operator Output $\hat{O}$:

$$\hat{O} = |000000\rangle\langle S_0| + |101111\rangle(\langle S_1| + \langle S_2| + \langle S_3| + \langle S_4|)$$

Operator Transisi $\hat{T}$:

$$\hat{T} = |S_0\rangle\langle P_0| + |S_1\rangle\langle P_1| + |S_2\rangle\langle P_2| + |S_3\rangle\langle P_3| + |S_4\rangle\langle P_4|$$

E. Gate yang Digunakan

NOT, AND, OR, BUFFER, dan Ground/Constant Gate digunakan untuk membentuk kategori sensor (P0-P4), Next State (Q2, Q1, Q0), dan Output Aktuator.

VII. Implementasi Kode

Listing 1. 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
);
// State encoding
localparam S0_IDLE = 3'b000;
localparam S1_SINGLE = 3'b001;
localparam S2_MULTIPLE = 3'b010;
localparam S3_CRITICAL = 3'b011;
localparam S4_EMERGENCY = 3'b100;

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
always @(posedge clk or posedge reset) begin
if (reset) current_state <= S0_IDLE;
else current_state <= next_state;
end

// Output Logic
always @(*) begin
A1 = 0; A2 = 0; A3 = 0; A4 = 0; A5 = 0; A6 = 0;
case (current_state)
S0_IDLE: begin end
S1_SINGLE: begin
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
S2_MULTIPLE: begin
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
if (!S1) A5 = 1;
if (!S2) A3 = 1;
if (!S3) A1 = 1;
```

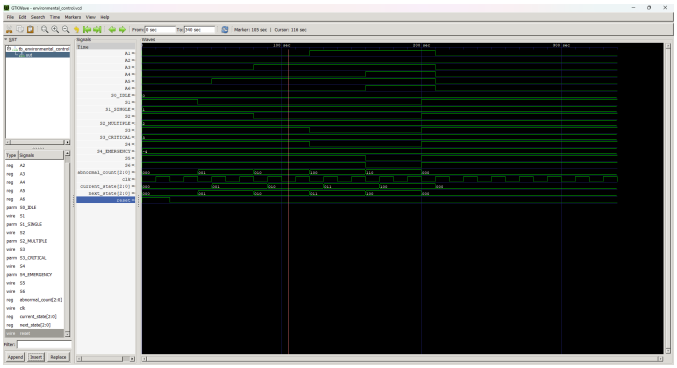
```
                if (!S4) A1 = 1;
                if (!S5) A4 = 1;
                if (!S6) A6 = 1;
                end
                S4_EMERGENCY: begin
                A1 = 1; A3 = 1; A4 = 1; A5 = 1; A6 = 1; A2 = 0;
                end
                default: begin end
                endcase
                end
                endmodule

                // TESTBENCH (Singkat)
                module tb_environmental_control;
                reg clk, reset;
                reg S1, S2, S3, S4, S5, S6;
                wire A1, A2, A3, A4, A5, A6;
                wire [2:0] current_state;
                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)
                );
                initial begin
                clk = 0; forever #10 clk = ~clk;
                end
                initial begin
                $dumpfile("environmental_control.vcd");
                $dumpvars(0, tb_environmental_control);
                // Test Case 1: Reset
                reset = 1; S1=1; S2=1; S3=1; S4=1; S5=1; S6=1; #20 reset = 0;
                // Test Case 2: Single sensor abnormal
                S1=0; #40;
                // Test Case 3: Multiple
                S2=0; #40;
                // Test Case 4: Critical
                S3=0; S4=0; #40;
                // Test Case 5: Emergency
                S5=0; S6=0; #40;
                // Recovery
                S1=1; S2=1; S3=1; S4=1; S5=1; S6=1; #40;
                $finish;
                end
                endmodule
```

Listing 2. C# Simulasi Microcontroller

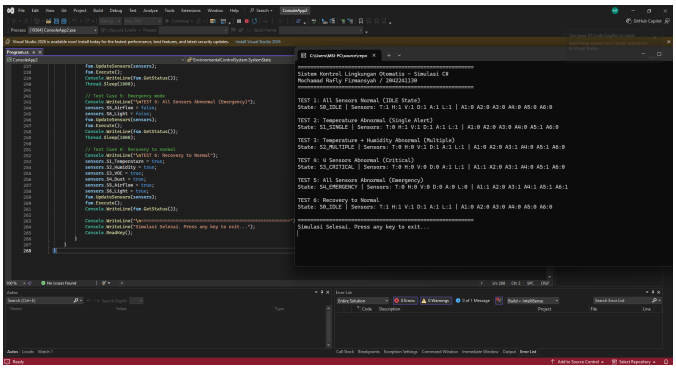
```
                // ====================================================================
                // Sistem Kontrol Lingkungan Otomatis - C# untuk Mikrocontroller
                // Mochamad Rafly Firmansyah / 2042241130
                // ====================================================================
                using System;
                using System.Threading;

                namespace EnvironmentalControlSystem {
                        public enum SystemState {
                                S0_IDLE = 0, S1_SINGLE = 1, S2_MULTIPLE = 2, S3_CRITICAL = 3, S4_EMERGENCY = 4
                        }

                        public struct SensorInputs {
                                public bool S1_Temperature;
                                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;
                                }
                        }

                        public struct ActuatorOutputs {
                                public bool A1_ExhaustFan;
                                public bool A2_InlineDuctFan;
                                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 class EnvironmentalControlFSM {
                        private SystemState currentState;
                        private SensorInputs sensors;
                        private ActuatorOutputs actuators;

                        public EnvironmentalControlFSM() { Reset(); }
                        public void Reset() {
                                currentState = SystemState.S0_IDLE; actuators.Reset();
                        }
                        public void UpdateSensors(SensorInputs newSensors) { sensors = newSensors; }

                        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;
                                }
                        }

                        private void UpdateActuators() {
                                actuators.Reset();
                                switch (currentState) {
                                        case SystemState.S0_IDLE: break;
                                        case SystemState.S1_SINGLE:
                                        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:
                                        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:
                                        actuators.A1_ExhaustFan = true; actuators.A3_Dehumidifier = true;
                                        actuators.A4_InlineDuctFan2 = true; actuators.A5_CoolingSystem = true;
                                        actuators.A6_LEDSystem = true;
                                        break;
                                }
                        }
                        public void Execute() {
                                currentState = DetermineNextState();
                                UpdateActuators();
                        }
                }
        }

```

VIII. Output Visual



Gambar 1. Output HDL Verilog Icarus



Gambar 2. Output C# Visual Studio

Referensi

[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*, IEEE 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*, IEEE 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*, IEEE Inc., 2022, pp. 136–140. doi: 10.1109/ELTICOM57747.2022.10037913.

[5] B. Junaidin et al., "Advancements in Microcontroller Technology for Wind Speed Measurement in Wind Tunnels," in *Proceedings - IEIT 2023*, IEEE 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 *ICAMIMIA 2021 - Proceeding*, IEEE Inc., 2021, pp. 41–46. doi: 10.1109/ICAMIMIA54022.2021.9807698.

[7] A. F. Adziima et al., "Prototype Design of Automatic Switching Speed of Exhaust Fan For Air Quality Control Based On IoT," in *ICAMIMIA 2021 - Proceeding*, IEEE Inc., 2021, pp. 114–119. doi: 10.1109/ICAMIMIA54022.2021.9809416.

[8] Paulo Abreu et al., "VENTI: experimental controller for inline duct fan," IEEE, 2017.

[9] S. Qiaoyun et al., "Design and Implementation of a Wireless Control Intelligent Humidifier," IEEE, Sep. 2025, pp. 1396–1400. doi: 10.1109/itaic64559.2025.11163186.

[10] A. K. Putri et al., "Automatic Usage of IoT-Based Dehumidifiers in High-Humidity Spaces," in *ICISS 2024 - Proceeding*, IEEE Inc., 2024. doi: 10.1109/ICISS62896.2024.10751296.

[11] K. Kim et al., "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 et al., "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.