Autonomous Drone MedEvac Coordination Using Multi-Sensor Cyber-Physical System With FSM-Based Safety Control

Mahendra Dwi Fahreza
Instrumentation Engineering Department
Vocational Faculty
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
mahendrafhrz1202@gmail.com
ORCID: 0009-0001-8809-0957

Dwi Oktavianto Wahyu Nugroho, S.T., M.T.
Instrumentation Engineering Department
Vocational Faculty
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
Oktavianto_2wn@its.ac.id
ORCID: 0000-0002-6781-0732

***Abstract*—This paper discusses the design and testing of an autonomous MedEvac drone control system. It is based on a multi-sensor Cyber-Physical System (CPS) architecture and a safety controller using a deterministic Finite State Machine (FSM). The system monitors patient health and cabin conditions with six digital sensors. It produces Warning (W), Critical (C), and Acknowledgment (ACK) flags as its main decision indicators. Digital logic modeling uses Boolean functions, truth tables, and synchronous next-state equations. These are formally extended with a quantum logic transformation that employs bra–ket notation and reversible operators. This approach allows for a mathematically verifiable representation of the controller using CNOT and Toffoli gates. The hardware implementation is tested through Verilog HDL simulations and VCD-based waveform analysis in GTKWave, which demonstrate correct transitions among NORMAL, WARNING, CRITICAL, and ACKED states. A C# behavioral model also confirms that software-level logic aligns with hardware-level execution. Although the system has not yet been deployed on a real FPGA platform, the simulation results prove the correctness, reliability, and feasibility of the proposed MedEvac controller. This system lays the groundwork for future real-time drone-based medical stabilization and integration with physical biomedical sensors.**

***Index Terms*—Cyber-Physical Systems (CPS), Finite State Machine (FSM), MedEvac Drone, Multi-Sensor Monitoring, Digital Logic Modeling, Quantum Logic Transformation, Verilog HDL, GTKWave Simulation, Autonomous Medical Systems.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) have advanced considerably in emergency medical services because they increase response times and allow access to inaccessible areas [1]. During health emergencies when the movement of healthcare workers is restricted to help slow the spread of disease, drones have demonstrated their value in delivering medical supplies and drugs [2]. In major disasters, UAVs are a vital way of accelerating the distribution of medical supplies while guaranteeing the safety of people in hazardous areas [3]. Beyond logistical assistance, drones are also becoming increasingly used for delivering medical samples to laboratories, therefore helping to maintain the integrity of the specimens at a high rate [4].

Corresponding author: Oktovianto_2wn@its.ac.id

Soni et al. proved that by autonomously delivering and collecting medical data during the COVID-19 epidemic, drones could lighten the operational burden on medical staff hence lowering direct exposure to infection [5]. Other work reveals that drones can transport medications, tourniquets, and oximeters with good flight stability across short to medium distances [6]. Embedded sensors have helped UAVs grow not only as a delivery vehicle but also as a platform for enabling remote physiological measurements.

Despite these achievements, existing research focuses more on logistics rather than direct patient stabilization (MedEvac). Cabin stability—temperature, humidity, oxygen levels, CO_2, patient motion—is not addressed in traditional UAV designs [7]. For real-time in-flight stabilization, UAV-based MedEvac systems must integrate multi-sensors, adaptive actuators, and an autonomous safety controller.

This research develops an Autonomous Drone MedEvac Coordination system that automatically monitors patient vital signs and controls cabin environment using a Finite State Machine (FSM). The FSM acts as the deterministic decision center, mapping sensor abnormalities to actuator responses. Formal design using quantum logic transformation ensures reversible, verifiable computation. Verilog HDL and C# simulation confirm that the system behaves consistently in NORMAL, WARNING, CRITICAL, and ACKED conditions.

II. Related Work

A thorough examination of the most recent publications pertinent to the development of unmanned aerial medical evacuation (MedEvac) technologies is provided in this chapter. The debate is broken down three key points: the development of drone usage in emergency medical logistics services; the application of Cyber-Physical Systems (CPS) design in health transport; and the detection of technological shortcomings (gap analysis) in current systems. This review aspires to show where

research fits in defeating the restrictions of dynamic patient monitoring and automated safety control during flight.

A. Drone Delivery and Emergency Systems

The usage of drones for emergency medical services has exploded and shows great promise for speeding the delivery of vital medical equipment in disaster- or pandemic-struck regions [10]. Many research reveal that UAVs can deliver medications and also medical equipment including defibrillators much more quickly than traditional ambulances, therefore improving patient survival rates early in therapy [1]. Drones have also been very helpful for humanitarian missions in inaccessible regions, especially when infrastructure is damaged and ground access is restricted [2].

During the COVID-19 pandemic, drones have shown they can effectively deliver medical kits, medicines, and collect laboratory samples. This approach reduces direct contact between medical personnel and patients [5]. The use of drones for blood transport has also been confirmed, with results showing that sample quality stays stable during flight [4]. However, all of these studies still focus on logistics delivery, not on the real-time patient monitoring needed in Medical Evacuation (MedEvac) operations. livery, not on real-time patient monitoring needed in Medical Evacuation (MedEvac) operations.

B. Cyber-Physical Systems in Medical Transport

Cyber-physical systems (CPS) are common in today's healthcare systems. They are used for things like sensor-based patient monitoring and automatic control of medical devices in hospitals [11]. In medical transportation, several studies show that combining real-time sensors with adjustable actuators is crucial for keeping patients stable while being transported [12]. However, current drone-based medical transportation systems do not fully use Cyber-physical systems (CPS) design, especially in terms of processing physiological sensors, and managing cabin environments, also ensuring automatic actuator response during flight.

C. Gaps Identified

Medical Evacuation (MedEvac) scenarios requiring direct patient stabilization are not yet supported by UAVs because the majority of UAV research in the medical field focuses on logistics delivery rather than monitoring patients physiological conditions during flight [5]. According to the literature on medical drones, current UAV systems are limited to transporting equipment, medications, or samples; they lack multi-sensor integration to monitor patients in real-time, including body temperature, oxygen, CO2, humidity, and movement response.

Existing systems remain passive and do not adapt because there is no research using cabin actuators, such as heaters, humidity controllers, oxygen mixers, ventilation controls, or patient body stabilization mechanisms, to maintain physiological conditions during flight. Additionally, even though there is a strong need for reliable methods such as FSM to ensure stable and predictable control decisions in critical systems, studies on medical drones have not adopted FSM-based safety control mechanisms.

Lastly, there is no foundation for advanced computational verification for autonomous medical UAV systems since no research has transformed the safety logic of MedEvac drones into formal representations like quantum logic mapping.

III. System Architecture

The Autonomous Drone MedEvac Coordination system is set up as a cyber-physical system (CPS). It integrates multiple sensors that monitor health state, controls environmental actions, and uses automatic computing units. The system operates on a closed-loop control, which includes four main parts: the sensor subsystem, the controller and FSM subsystem, the actuator subsystem, and the flight control platform.

A. Overall Cyber-Physical System Architecture

The system's general setup is a step-by-step process that measures patient and environmental conditions before taking corrective actions with the actuator. All sensors send digital signals to the embedded controller. The controller evaluates the conditions using Warning Flag (W) and Critical Flag (C) logic. The W and C values are the main inputs for the Finite State Machine (FSM), which acts as the decision-making center.

The FSM determines the drone's operating mode: NORMAL, WARNING, CRITICAL, or ACKED. Each state triggers a different response from the actuator. The actuator then adjusts conditions in the cabin, including temperature, humidity, ventilation, and patient body position.

The CPS component works with the flight controller to ensure that the medical system does not interfere with drone navigation stability. This overall setup allows for automatic patient handling without manual intervention during the evacuation process.

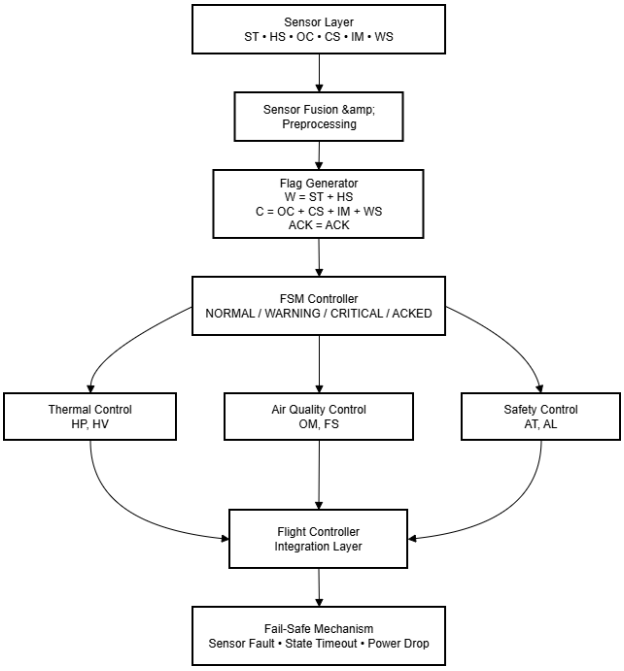

Fig. 1. Overall Cyber-Physical System (CPS) architecture of the Autonomous Drone MedEvac Controller

B. Sensor Subsystem

The sensor subsystem monitors the patient's physiological condition and the cabin environment in real time. The system uses six types of sensors:

1) Skin Temperature Sensor (ST): This sensor device is used to detect hypothermia by measuring skin temperature. When the temperature is $< 35°\mathrm{C}$, the sensor gives a HIGH signal that triggers thermal correction (13).

2) Humidity Sensor (HS): This sensor measures the relative humidity in the cabin. Values below 40% or above 60% are considered abnormal and trigger humidity stabilization measures (14).

3) O_2 Concentration Sensor (OC): This sensor monitors the patient's inhaled oxygen level. $\mathrm{FiO_2} < 21\%$ (hypoxia) generates a HIGH signal as a critical condition indicator (15).

4) CO_2 Sensor (CS): In the system, this sensor measures the CO_2 concentration in the cabin. $CO_2 > 1000$ ppm triggers additional ventilation via the fan actuator (16).

5) Infant Movement Sensor (IM): IMU sensor that detects the frequency of the baby's movements. Movements with frequency > 2 Hz indicate distress or abnormal activity (17).

6) Weight Sensor (WS): The weight sensor detects changes in mass distribution in the patient's position. $\Delta\mathrm{Weight} > 12.5$ g indicates an imbalance in load and triggers a position correction (18).

This block cleans the signal, digitizes it, and compresses the information into two main flags: W (Warning) and C (Critical), which become inputs for the FSM. However, the MedEvac system also receives one important input from the operator, namely ACK (Acknowledgement Flag), which does not originate from physiological sensors but from a human-in-the-loop safety structure that is activated when the operator or ground staff confirms that the flight supervisor recognizes a critical condition.

The combined sensor–FSM system raises the correct alert on time, triggers the emergency protocol, and maintains stable cabin environmental conditions for the patient inside the MedEvac system.

C. Actuator Subsystem

The actuator subsystem provides a physical response to cabin conditions and patient status based on FSM decisions. This system has six actuators that work in three areas: thermal, air quality, and patient safety.

1) Heater Power (HP): It increases the cabin temperature when the Skin Temperature (ST) sensor detects hypothermia. This heating system works in stages to prevent excessive temperature rises.

2) Humidifier Valve (HV): It controls cabin humidity. When the Humidity Sensor (HS) shows a significant drop in humidity, the HV adds water vapor to keep the patient's respiratory tract stable. When humidity is too high, the HV automatically lowers its output.

3) O_2 Mixer (OM): The O_2 Mixer adjusts the oxygen level based on signals from the O_2 Concentration (OC) sensor. A digital valve regulates the mix of oxygen and ambient air to ensure the patient maintains a safe $\mathrm{FiO_2}$ level.

4) FanSpeed (FS): It increases cabin ventilation when the CO_2 Sensor (CS) detects CO_2 buildup, keeping the atmosphere safe and stable.

5) Auto Tilt (AT): It helps maintain the patient's body position and the drone's load distribution. When the Weight Sensor (WS) detects a shift in mass, the AT corrects the platform tilt to avoid aerodynamic instability.

6) Alarm (AL): It gives visual and audio warnings when the system reaches a critical state or if a sensor has a fault and also receives acknowledgment input from the operator.

All of these actuators work in a closed-loop control pattern managed by FSM and respond quickly to changes in the environment. With six actuators working together, the drone can achieve automatic physiological stabilization without any manual input.

D. Embedded Control Platform

The embedded control platform is the computing core of the MedEvac drone system. This platform can be a high-performance microcontroller or FPGA that manages all sensor signal flows, executes FSM, and controls actuators. This unit also integrates data between the medical subsystem and flight controller.

1) Sensor Processing Unit: This component converts and standardizes signals. All analog inputs are transformed using an ADC. They are then assigned HIGH or LOW conditions based on medical thresholds. This unit maintains signal stability before it goes into the Flag Generator.

2) Flag Generator: This section calculates the W and C statuses through simple logical operations. W (Warning) indicates non-critical abnormalities that require moderate action, while C (Critical) indicates conditions that threaten safety. There is also an acknowledgment (ACK) to ensure that critical conditions do not recur without intervention, and it is part of the fail-safe control mechanism.

3) FSM Controller: This component is responsible for determining the drone's operational mode: NORMAL, WARNING, CRITICAL, or ACKED. The FSM uses two D-Flip Flops (D1, D0) as state registers, resulting in deterministic and reversible logical decisions. The advantages of this approach are fast response times, predictable system behavior, and resistance to noise.

4) Actuator Control Module: This component sends PWM or digital commands to the actuators based on the FSM state. This module ensures that all actuators get the right command signals according to priority conditions.

5) Safety and Fail-Safe Unit: This section checks for problems like sensor failure, state timeout, input issues, or low battery power. If it finds an unusual condition, the system automatically switches to CRITICAL mode or activates a safety alarm.

IV. Digital Logic Modeling

Digital logic modeling converts the medical and environmental behavior of the MedEvac system into a clear logic representation. This representation can be directly implemented on digital hardware like FPGAs or microcontrollers. In this system, each sensor sends a binary signal (HIGH/LOW). This signal is processed into main flags (W, C, ACK), which are then handled by a Finite State Machine (FSM) to control the actuator. This section explains the design of truth tables, flag functions, FSM modeling, and D Flip-Flop implementation.

A. Truth Table Design

Truth tables are used to describe the logical relationship between sensors, flags, states, and actuators. The six sensors (ST, HS, OC, CS, IM, WS) are first grouped into two categories, namely Environment Sensors: ST, HS and Physiological/Critical Sensors: OC, CS, IM, WS Each sensor produces a HIGH (1) output when it is in an abnormal condition according to a predefined medical threshold.

Although the initial system analysis involved a complete truth table enumeration consisting of 26 (64) states, the analysis reveals that the control logic is decoupled. Most actuators are directly driven by their respective sensor states, while the alert systems (AT and AL) operate on a logical disjunction of the safety sensors.

Therefore, the control strategy is synthesized into the following Boolean functions, specifically described as:

$$HP = ST, \quad HV = HS, \quad OM = OC, \quad FS = CS \quad (1)$$

Meanwhile, the alarm and trip actuators follow the logic:

$$AT = AL = IM + WS \quad (2)$$

As shown in TABLE I, the system logic is fully deterministic. Each of the 64 binary states maps to a specific set of actuator outputs, confirming that there are no floating or undefined conditions in the control loop. The table illustrates the direct mapping for the primary actuators (HP, HV, OM, FS) and the logical disjunction used for the safety alerts (AT, AL).

In the context of autonomous MedEvac operations, the logical mapping between sensor abnormalities and actuator responses is crucial for maintaining system stability. Each subsystem must react deterministically to ensure that no unsafe or unpredictable behavior occurs during flight. This deterministic behavior is particularly important in emergency scenarios, where rapid stabilization of cabin conditions directly affects patient safety. By structuring the control rules through Boolean logic, the entire system can be validated formally and tested exhaustively using digital simulation tools, ensuring that every possible sensor combination yields a predictable actuator outcome.

TABLE I
TRUTH TABLE OF LOGIC RELATIONSHIPS BETWEEN SENSORS AND ACTUATORS

Sensor						**Actuator**					
ST	HS	OC	CS	IM	WS	HP	HV	OM	FS	AT	AL
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	1	0	1	0	0	0	1	1	1
0	0	0	1	1	0	0	0	0	1	1	1
0	0	0	1	1	1	0	0	0	1	1	1
0	0	1	0	0	0	0	0	1	0	0	0
0	0	1	0	0	1	0	0	1	0	1	1
0	0	1	0	1	0	0	0	1	0	1	1
0	0	1	0	1	1	0	0	1	0	1	1
0	0	1	1	0	0	0	0	1	1	0	0
0	0	1	1	0	1	0	0	1	1	1	1
0	0	1	1	1	0	0	0	1	1	1	1
0	0	1	1	1	1	0	0	1	1	1	1
0	1	0	0	0	0	0	1	0	0	0	0
0	1	0	0	0	1	0	1	0	0	1	1
0	1	0	0	1	0	0	1	0	0	1	1
0	1	0	0	1	1	0	1	0	0	1	1
0	1	0	1	0	0	0	1	0	1	0	0
0	1	0	1	0	1	0	1	0	1	1	1
0	1	0	1	1	0	0	1	0	1	1	1
0	1	0	1	1	1	0	1	0	1	1	1
0	1	1	0	0	0	0	1	1	0	0	0
0	1	1	0	0	1	0	1	1	0	1	1
0	1	1	0	1	0	0	1	1	0	1	1
0	1	1	0	1	1	0	1	1	0	1	1
0	1	1	1	0	0	0	1	1	1	0	0
0	1	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	0	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1	1
1	0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	1	1	0	0	0	1	1
1	0	0	0	1	0	1	0	0	0	1	1
1	0	0	0	1	1	1	0	0	0	1	1
1	0	0	1	0	0	1	0	0	1	0	0
1	0	0	1	0	1	1	0	0	1	1	1
1	0	0	1	1	0	1	0	0	1	1	1
1	0	0	1	1	1	1	0	0	1	1	1
1	0	1	0	0	0	1	0	1	0	0	0
1	0	1	0	0	1	1	0	1	0	1	1
1	0	1	0	1	0	1	0	1	0	1	1
1	0	1	0	1	1	1	0	1	0	1	1
1	0	1	1	0	0	1	0	1	1	0	0
1	0	1	1	0	1	1	0	1	1	1	1
1	0	1	1	1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0	1	1	1	1
1	1	0	0	0	0	1	1	0	0	0	0
1	1	0	0	0	1	1	1	0	0	1	1
1	1	0	0	1	0	1	1	0	0	1	1
1	1	0	0	1	1	1	1	0	0	1	1
1	1	0	1	0	0	1	1	0	1	0	0
1	1	0	1	0	1	1	1	0	1	1	1
1	1	0	1	1	0	1	1	0	1	1	1
1	1	0	1	1	1	1	1	0	1	1	1
1	1	1	0	0	0	1	1	1	0	0	0
1	1	1	0	0	1	1	1	1	0	1	1
1	1	1	0	1	0	1	1	1	0	1	1
1	1	1	0	1	1	1	1	1	0	1	1
1	1	1	1	0	0	1	1	1	1	0	0
1	1	1	1	0	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1

B. Flag Logic

Flag logic simplifies six sensors into two logic channels, W and C, and one control channel, ACK. Flag calculations use Boolean logic, which makes the system easy to implement in hardware.

Warning Flag (W) means the environmental sensors indicate abnormal conditions when humidity or temperature is outside safe limits, so:

$$W = ST + HS \quad (3)$$

W = 1 means that the environmental conditions have the potential to deteriorate, requiring moderate correction but not directly threatening patient safety.

Critical Flag (C) is a critical condition indicating a dangerous medical or environmental condition, so that:

$$C = OC + CS + IM + WS \tag{4}$$

Acknowledgement Flag (ACK) is a manual input from the operator. ACK = 1 indicates that the operator is aware of the patient's critical condition and allows the system to continue operation without activating repeated critical alarms, so that the actuator commands remain stable.

TABLE II
TRUTH TABLE OF FLAG LOGIC

Sensor						Flag		
ST	HS	OC	CS	IM	WS	W	C	ACK
0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0
0	0	0	0	1	0	0	1	0
0	1	0	0	0	0	1	0	0
1	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0
0	0	0	1	0	0	0	1	0
1	1	0	0	0	1	1	1	0

The flag logic in Table II shows how the six sensor inputs are reduced to three clear control flags: W, C, and ACK. These flags represent the system's key situational states. By turning raw sensor issues into simple Boolean indicators, the controller can assess the MedEvac system's performance without heavy computational demands. This approach clearly separates low-level sensing from higher-level decision-making. It ensures that the next FSM outputs work with checked and simplified digital signals. As a result, the flag logic acts as an important interface layer. It allows reliable and hardware-friendly state changes in the overall autonomous medical evacuation system.

C. Finite State Machine Modeling

Finite State Machine (FSM) modeling forms the main decision-making part of the Autonomous Drone MedEvac system. The FSM takes the three flag inputs W, C, and ACK to figure out the system's operational state and actuator commands.

The proposed FSM has four different states: NORMAL, WARNING, CRITICAL, and ACKED. Each state represents a specific operational condition based on combinations of the W, C, and ACK flags. NORMAL shows stable conditions with no abnormalities. WARNING points to mild environmental changes. CRITICAL indicates dangerous medical or environmental conditions that need immediate system response. ACKED represents a critical monitored state in which the operator has confirmed the emergency. These states help the system respond appropriately to the urgency level detected by the sensors and verified by the operator.

State transitions depend only on the logical relationships among the three flags. When C = 1 and ACK = 0, the system goes into CRITICAL. If C = 1 and ACK = 1, the system shifts to ACKED. On the other hand, W = 1 with C = 0 leads to a WARNING state. The system goes back to NORMAL only when both W and C = 0.

To represent the four operating states in hardware, the FSM uses a two-bit state encoding, where the current state is stored in the registers S1 and S0. The NORMAL, WARNING, CRITICAL and ACKED states are encoded as 00, 01, 10, and 11, respectively. At every clock cycle, the FSM computes the next-state values S1' and S0' based on the input flags W, C, and ACK. The next-state logic is defined such that

$$S1' = C \tag{5}$$

Indicating that any critical condition forces the system into a state with S1 = 1; and:

$$S0' = (W \cdot \overline{C}) + (C \cdot ACK) \tag{6}$$

which selects WARNING or ACKED when environmental deviations or acknowledged critical events occur. These relationships ensure that state transitions follow medically consistent rules, as summarized in Table III.

TABLE III
FSM TRUTH TABLE FOR STATE TRANSITIONS AND ACTUATOR OUTPUTS

W	C	ACK	S1	S0	S1'	S0'	HP	HV	OM	FS	AT	AL
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	1	1	1
0	1	1	0	0	1	1	0	0	1	1	1	0
1	0	0	0	0	0	1	1	1	0	0	0	0
1	0	1	0	0	0	1	1	1	0	0	0	0
1	1	0	0	0	1	0	1	1	1	1	1	1
1	1	1	0	0	1	1	1	1	1	1	1	0
0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	0	0	1	1	1	1
0	1	1	0	1	1	1	0	0	1	1	1	0
1	0	0	0	1	0	1	1	1	0	0	0	0
1	0	1	0	1	0	1	1	1	0	0	0	0
1	1	0	0	1	1	0	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	0
0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0	0	1	1	1	1
0	1	1	1	0	1	1	0	0	1	1	1	0
1	0	0	1	0	0	1	1	1	0	0	0	0
1	0	1	1	0	0	1	1	1	0	0	0	0
1	1	0	1	0	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	0
0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0
0	1	0	1	1	1	0	0	0	1	1	1	1
0	1	1	1	1	1	1	0	0	1	1	1	0
1	0	0	1	1	0	1	1	1	0	0	0	0
1	0	1	1	1	0	1	1	1	0	0	0	0
1	1	0	1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	0

The complete truth table provides a combination of flags and current states leading to the next-state transitions and actuator responses. Because the table contains many numbers, a graphical representation is beneficial for illustrating the overall system behavior in a more intuitive form. The FSM state transition diagram in Fig. 3 summarizes the logical flow between the NORMAL, WARNING, CRITICAL, and ACKED states.

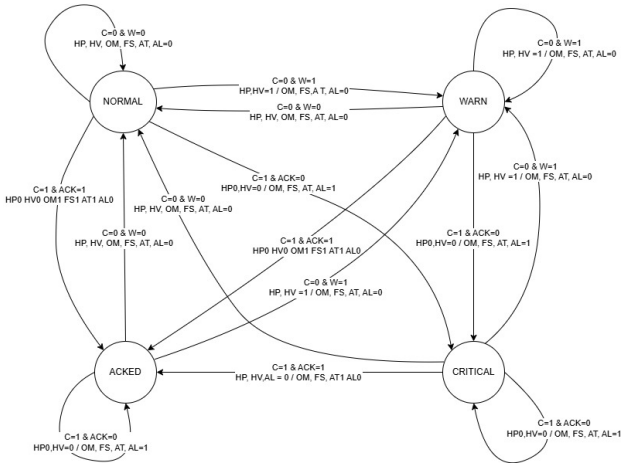

Fig. 2. Finite state machine diagram of the proposed MedEvac controller.

D. Flip-Flop Based Implementation

The proposed FSM uses two D Flip-Flops (DFFs) to store the state bits S1 and S0. A DFF is a synchronous element that captures the input value D at the rising edge of the system clock and outputs it in the next cycle. This feature makes the DFF suitable for state machines, providing stable and predictable behavior in safety-critical medical applications like the MedEvac drone controller. A DFF operates according to the following relation:

$$Q_{\text{next}} = D \tag{7}$$

i.e., the output at the next clock cycle is equal to the input at the current cycle. Because the FSM next-state equations produce S1' and S0', these values directly serve as the D inputs for the two DFFs. They form the register that stores the current state. Based on the next-state logic developed earlier, the two DFF inputs are:

$$D1 = S1' = C \tag{8}$$

$$D0 = S0' = (W \cdot \overline{C}) + (C \cdot ACK) \tag{9}$$

Here, D1 shows whether the system must switch to a critical state (CRITICAL or ACKED) since any critical condition requires S1 = 1. Meanwhile, D0 decides if the system enters WARNING or ACKED based on environmental issues or operator acknowledgment. These equations ensure that the FSM transitions match the medical priority of each sensor condition.

During operation, the DFF that stores S1 differentiates between non-critical and critical states. The DFF that stores S0 narrows the state down to one of four modes: NORMAL (00), WARNING (01), CRITICAL (10), or ACKED (11). This register-based design lets the hardware update the system state in a single clock cycle, reducing latency and ensuring a quick system response to abnormal physiological or environmental signals.

The full mapping of input flags, current-state bits, next-state bits, and D-input signals is shown in Table IV. It highlights the predictable behavior of the FSM when implemented with D flip-flops. This table builds on the previous FSM truth table by clearly displaying the D1 and D0 inputs for each combination of system conditions

TABLE IV
FSM STATE TRANSITION TABLE WITH D FLIP-FLOP INPUTS (D1 AND D0)

W	C	ACK	S1	S0	S1'	S0'	HP	HV	OM	FS	AT	AL	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	1	1	1	1	0
0	1	1	0	0	1	1	0	0	1	1	1	0	1	1
1	0	0	0	0	0	1	1	1	0	0	0	0	0	1
1	0	1	0	0	0	1	1	1	0	0	0	0	0	1
1	1	0	0	0	1	0	1	1	1	1	1	1	1	0
1	1	1	0	0	1	1	1	1	1	1	1	0	1	1
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	0	0	1	1	1	1	1	0
0	1	1	0	1	1	1	0	0	1	1	1	0	1	1
1	0	0	0	1	0	1	1	1	0	0	0	0	0	1
1	0	1	0	1	0	1	1	1	0	0	0	0	0	1
1	1	0	0	1	1	0	1	1	1	1	1	1	1	0
1	1	1	0	1	1	1	1	1	1	1	1	0	1	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0	0	1	1	1	1	1	0
0	1	1	1	0	1	1	0	0	1	1	1	0	1	1
1	0	0	1	0	0	1	1	1	0	0	0	0	0	1
1	0	1	1	0	0	1	1	1	0	0	0	0	0	1
1	1	0	1	0	1	0	1	1	1	1	1	1	1	0
1	1	1	1	0	1	1	1	1	1	1	1	0	1	1
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	0	0	0	1	1	1	1	1	0
0	1	1	1	1	1	1	0	0	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	0	0	0	0	0	1
1	0	1	1	1	0	1	1	1	0	0	0	0	0	1
1	1	0	1	1	1	0	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1	0	1	1

The D Flip-Flop implementation of the FSM logic is illustrated in Fig. 4, showing how the W, C, and ACK flags propagate through the next-state equations into the D1 and D0 inputs.

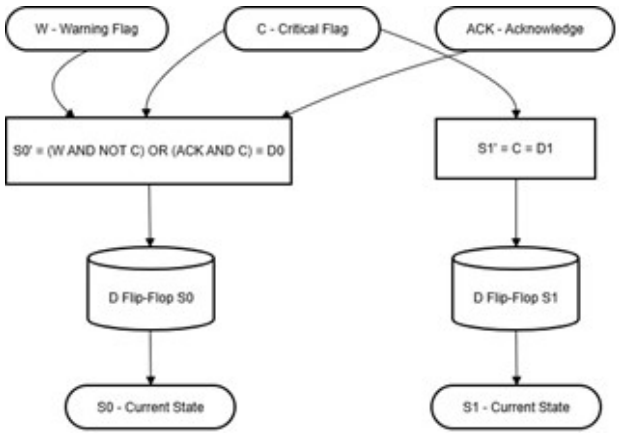

Fig. 3. Flip-Flop implementation of the FSM next-state logic.

E. FSM Matrix Representation

1) Transition State Matrix (T): is a mathematical representation of all FSM state transitions based on the system's logic input. If the system is in state j, then on the next clock cycle, it will move to state i. Below are the mathematical equations and definitions.

$$t_{ij} = \begin{cases} 1, & \text{if state } j \text{ transitions to state } i \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

TABLE V
INDEX MAPPING TABLE TRANSITION STATE MATRIX

j	State (W, C, ACK, S1, S0)	State (W, C, ACK, S1, S0)	i
0	00000	00000	0
1	00001	00000	0
2	00010	00000	0
3	00011	00000	0
4	00100	01000	4
5	00101	01000	4
6	00110	01000	4
7	00111	01000	4
8	01000	01010	10
9	01001	01010	10
10	01010	01010	10
11	01011	01010	10
12	01100	01111	15
13	01101	01111	15
14	01110	01111	15
15	01111	01111	15
16	10000	10001	17
17	10001	10001	17
18	10010	10001	17
19	10011	10001	17
20	10100	10101	21
21	10101	10101	21
22	10110	10101	21
23	10111	10101	21
24	11000	11010	26
25	11001	11010	26
26	11010	11010	26
27	11011	11010	26
28	11100	11111	31
29	11101	11111	31
30	11110	11111	31
31	11111	11111	31

2) FSM Truth Table & D Flip-Flop Matrix: This matrix combines logic inputs, current state, next state, and DFF inputs into one table. It clearly explains the relationship between input, next state, next state, and DFF input. The column order is W, C, ACK, S1, S0, S1', S0', D1, D0.

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

3) Output Matrix: This matrix explains the actions taken by the system for each combination of state and input. Column order (HP, HV, OM, FS, AT, AL).

$$O = \begin{bmatrix} 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&1&1&1&1 \\ 0&0&1&1&1&0 \\ 1&1&0&0&0&0 \\ 1&1&0&0&0&0 \\ 0&0&1&1&1&1 \\ 0&0&1&1&1&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&1&1&1&1 \\ 0&0&1&1&1&0 \\ 1&1&0&0&0&0 \\ 1&1&0&0&0&0 \\ 0&0&1&1&1&1 \\ 0&0&1&1&1&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&1&1&1&1 \\ 0&0&1&1&1&0 \\ 1&1&0&0&0&0 \\ 1&1&0&0&0&0 \\ 0&0&1&1&1&1 \\ 0&0&1&1&1&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&1&1&1&1 \\ 0&0&1&1&1&0 \\ 1&1&0&0&0&0 \\ 1&1&0&0&0&0 \\ 0&0&1&1&1&1 \\ 0&0&1&1&1&0 \end{bmatrix} \tag{12}$$

V. Quantum Logic Transformation

The goal of developing a quantum-level structure for the MedEvac FSM safety controller is to create a solid mathematical framework. This framework will help verify state transitions that are deterministic, using reversible computation models. Recent studies in quantum automata show that classical finite-state controllers can be transformed into unitary

operators that operate on Hilbert spaces. This transformation does not change their computational behavior.

Puram et al. showed that "the action of the input alphabet of the automaton is modeled as a unitary matrix acting on states modeled as vectors in a Hilbert space", establishing a general approach for expressing classical transition systems via quantum operators [19]. Another study also confirmed the importance of FSM decomposition for creating sequential circuits that work with current reversible and flux-based quantum logic families [20]. These studies show that the MedEvac controller, while using traditional hardware, can gain from a reversible, quantum-compatible approach. This change would improve its verifiability and support future quantum enhanced cyber-physical systems.

A. Qubit Representation of System Variables

To embed the MedEvac safety controller into a quantum framework, all classical binary variables are represented using the computational basis of qubit states. The five core logical variables such as Warning Flag (W), Critical Flag (C), Acknowledgment (ACK), and the state bits S1 and S0 are encoded as:

$$|0\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{13}$$

The full system state is expressed as a tensor-product qubit register:

$$|W,\, C,\, ACK,\, S1,\, S0\rangle = |W\rangle \otimes |C\rangle \otimes |ACK\rangle \otimes |S1\rangle \otimes |S0\rangle \tag{14}$$

This representation ensures that all FSM inputs are embedded in a 5-qubit Hilbert space, providing a reversible and fully deterministic basis for quantum operator construction. The output of the FSM corresponds to two next-state qubits:

$$|S1',\, S0'\rangle \tag{15}$$

which ultimately drive the D Flip-Flops (D1 and D0) that store the controller's operational mode.

B. Bra–Ket Formalism for FSM Transition Representation

Bra–ket notation is used to describe the transformation from every possible input basis state to its corresponding next-state output. The quantum operator implementing the classical next-state logic is defined as:

$$U_{\text{logic}} = \sum_i |\text{Output}_i\rangle\langle\text{Input}_i| \tag{16}$$

where each Input $|\text{Input}_i\rangle$ is one of the $2^5 = 32$ possible combinations of $(W, C, ACK, S1, S0)$, and each $|\text{Output}_i\rangle$ is the next-state vector $(S1', S0')$ derived from the FSM truth table.

1) Transition to NORMAL ($S1' = 0$, $S0' = 0$): All input combinations that satisfy $W = 0$ and $C = 0$ produce the NORMAL state. These inputs correspond to indices $j = 0, 1, 2, 3, 4, 5, 6, 7$. The projection operator is:

$$\widehat{P}_{00} = \sum_{j=0}^{7} |00\rangle\langle j| \tag{17}$$

2) Transition to WARNING ($S1' = 0$, $S0' = 1$): WARNING occurs when $W = 1$ and $C = 0$, producing indices $j = 16, 17, 18, 19, 20, 21, 22, 23$:

$$\widehat{P}_{01} = \sum_{j=16}^{23} |01\rangle\langle j| \tag{18}$$

3) Transition to CRITICAL ($S1' = 1$, $S0' = 0$): CRITICAL is triggered by $C = 1$ and $ACK = 0$, mapping $j \in \{8, 9, 10, 11, 24, 25, 26, 27\}$:

$$\widehat{P}_{10} = \sum_{j \in \{8,9,10,11,24,25,26,27\}} |10\rangle\langle j| \tag{19}$$

4) Transition to ACKED ($S1' = 1$, $S0' = 1$): ACKED occurs when $C = 1$ and $ACK = 1$, mapping $j \in \{12, 13, 14, 15, 28, 29, 30, 31\}$:

$$\widehat{P}_{11} = \sum_{j \in \{12,13,14,15,28,29,30,31\}} |11\rangle\langle j| \tag{20}$$

5) Total FSM Operator: The complete quantum operator realizing the FSM transition is:

$$\widehat{U}_{\text{state}} = \widehat{P}_{00} + \widehat{P}_{01} + \widehat{P}_{10} + \widehat{P}_{11} \tag{21}$$

This unitary-form operator ensures that all MedEvac state transitions remain reversible and mathematically verifiable.

C. Bra–Ket Operator for DFF-Based Next-State Logic

After deriving the next-state functions from the FSM, the corresponding quantum representation must handle the two D Flip-Flop inputs $D1 = S1'$ and $D0 = S0'$. These outputs are encoded as qubits to preserve reversibility and enable implementation using quantum gates.

1) Qubit Representation for $D1$: The most significant next-state bit is determined by the critical flag:

$$S1' = C \tag{22}$$

Thus, the quantum representation directly transfers the value of the qubit $|C\rangle$ into $|D1\rangle$:

$$|D1\rangle = |C\rangle \tag{23}$$

This operation is reversible and implemented using a single CNOT gate, where C is the control qubit and $D1$ is the target.

2) Qubit Representation for $D0$: The least significant next-state bit is given by:

$$S0' = (W \cdot \overline{C}) + (ACK \cdot C) \tag{24}$$

This expression consists of two reversible Boolean terms: - Term 1: $W \cdot \overline{C}$ (corresponds to WARNING), - Term 2: $ACK \cdot C$ (corresponds to ACKED).

In bra–ket notation, the qubit $|D0\rangle$ encodes these terms as:

$$|D0\rangle = |W \cdot \overline{C} \vee (C \cdot ACK)\rangle \tag{25}$$

Or equivalently expressed using XOR:

$$|D0\rangle = |W \cdot \overline{C} \oplus (C \cdot ACK)\rangle \tag{26}$$

3) Final Bra–Ket Operator for FSM and DFF: When combined into a unified operator that writes both next-state bits into their dedicated qubit registers, the system evolves as:

$$|W,\, C,\, ACK,\, S1,\, S0,\, D1,\, D0\rangle \longrightarrow |W,\, C,\, ACK,\, S1,\, S0,\, D1',\, D0'\rangle \tag{27}$$

Thus, the complete quantum operator for implementing the classical FSM next-state logic is:

$$\widehat{U}_{\text{logic}} = |C\rangle\langle D1| \; + \; |W \cdot \overline{C} \vee (C \cdot ACK)\rangle\langle D0| \tag{28}$$

This operator seamlessly integrates the classical FSM next-state logic into a reversible, quantum-compatible model consistent with unitary evolution constraints.

D. Quantum Gate Synthesis and Circuit Realization

The MedEvac controller is synthesized into a complete reversible quantum circuit using X, CNOT, and Toffoli gates after deriving the reversible Boolean expressions for the next-state bits and actuator logic. Each classical logic expression is mapped to its quantum counterpart based on standard reversible gate constructions.

1) Warning Flag: The WARNING flag is computed as:

$$W = ST \vee HS \tag{29}$$

Quantum cost: 2 CNOT + 1 Toffoli (2-input OR).

2) Critical Flag: The CRITICAL flag combines four abnormal physiological sensors:

$$C = OC \vee CS \vee IM \vee WS \tag{30}$$

Quantum cost: 6 CNOT + 3 Toffoli gates with 2 ancilla qubits.

3) Next-State MSB ($D1$): The most significant bit is simply the copy of the critical flag:

$$D1 = C \tag{31}$$

Quantum cost: 1 CNOT.

4) Next-State LSB ($D0$): The reversible form of the next-state least significant bit is:

$$D0 = (ACK \cdot C) \vee (W \cdot \overline{C}) \tag{32}$$

Quantum cost: 1 X + 1 CNOT + 2 Toffoli.

5) Heater Power (HP):

$$HP = \overline{C} \cdot ST \tag{33}$$

Quantum cost: 1 Toffoli.

6) Humidifier Valve (HV):

$$HV = \overline{C} \cdot HS \tag{34}$$

Quantum cost: 1 Toffoli.

7) O_2 Mixer (OM):

$$OM = C \tag{35}$$

Quantum cost: 1 CNOT.

8) FanSpeed (FS):

$$FS = C \tag{36}$$

Quantum cost: 1 CNOT.

9) AutoTilt (AT):

$$AT = C \tag{37}$$

Quantum cost: 1 CNOT.

10) Alarm (AL):

$$AL = C \cdot ACK \tag{38}$$

Quantum cost: 1 Toffoli.

The resulting quantum circuit integrates both the next-state logic ($D1$ and $D0$) and all actuator outputs into a single reversible structure, producing a fully quantum-compatible representation of the MedEvac safety controller.

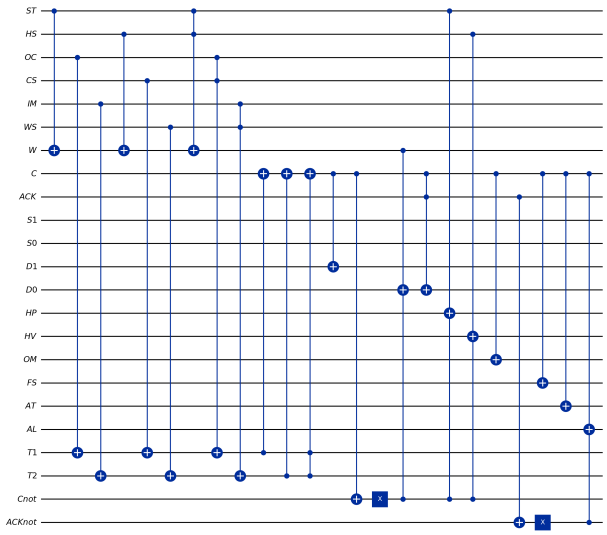

Fig. 4. Quantum circuit representation of the MedEvac FSM, showing CNOT, Toffoli, and X-gate operations used to implement the reversible next-state logic and actuator control pathways.

Shown in Fig. 4, the quantum circuit mixes the entire reversible MedEvac safety controller's formulation. While the middle section encodes the three control flags (W, C, ACK), the upper part of the circuit represents environmental and physiological sensor qubits. The lower part houses the six actuator outputs as well as next-state registers D1 and D0. While CNOT gates help signal duplication and state propagation, Toffoli gates realize all multi-input logical conditions. Consistent with unitary quantum computer demands, the final circuit preserves reversibility and guarantees one-to-one correspondence between input and output states

VI. Result

The digital logic model and quantum-logic mapping presented in earlier chapters were translated onto real hardware and simulation tools to verify the design of the autonomous MedEvac coordination system. The implementing platform is a hardware synthesis of the FSM controller based on an FPGA (described in Section IV–V) combined with software simulation of the system environment using C#-based VCD (Value Change Dump) generation and GTKWave waveform analysis.

A. Verilog HDL Implementation

The MedEvac controller including sensor preprocessing, W/C/ACK flag generation, FSM next-state logic, and actuator mapping was implemented using Verilog HDL. The FSM was encoded using two state bits (S1, S0) and evaluated synchronously on the rising clock edge. The logic directly follows the truth tables, Boolean equations, and quantum-derived next-state functions discussed in Sections IV and V. The HDL design was tested using an event-driven testbench that injects sensor patterns representing NORMAL, WARNING, CRITICAL, and ACKED conditions. This approach enables hardware-accurate validation without requiring a real FPGA board.

B. VCD-Based Simulation Using GTKWave

To visualize internal signal transitions, a Value Change Dump (VCD) file was generated from the Verilog simulation. This file was analyzed using GTKWave, allowing direct observation of sensor signals (ST, HS, OC, CS, IM, WS), control flags (W, C, ACK), FSM state bits (S1, S0), DFF next-state values (D1, D0), and actuator outputs (HP, HV, OM, FS, AT, AL).

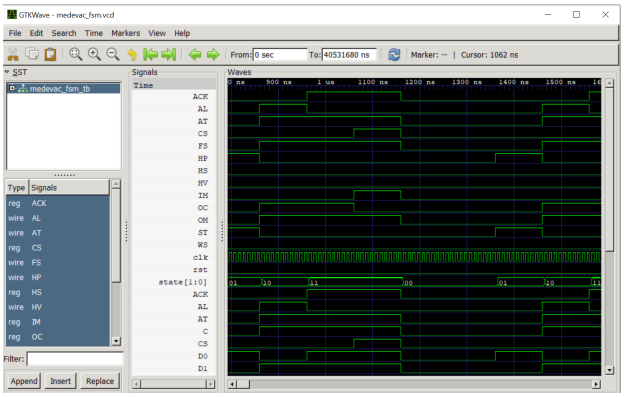

Fig. 5. GTKWave waveform output showing MedEvac FSM transitions, next-state DFF logic, sensor flags, and actuator responses.

The waveform shown in Fig. 8 depicts the MedEvac FSM's full execution across all four operational states. The reset signal initializes the system to the NORMAL state (state = 00) at the beginning of the simulation. The Warning Flag (W) asserts and the FSM properly changes to the WARNING state (01) as soon as the environmental deviation sensors (ST and HS) become active. This is seen in the operation of the Humidifier Valve (HV) and Heater (HP), which react to variations in skin temperature and humidity.

The Critical Flag (C) asserts itself when physiological-critical sensors (OC, CS, IM, and WS) exceed their thresholds signifying hypoxia, CO_2 accumulation, excessive infant movement, or mass imbalance. The waveform plainly illustrates the state moving from WARNING (01) to CRITICAL (10). The oxygen mixer (OM), fan speed (FS), and auto-tilt (AT) actuators are activated concurrently in this mode; the alarm (AL) alerts the operator.

Once the operator confirms the ACK input, the FSM enters ACKED (11). By displaying state = 11 while maintaining active critical actuators and rejecting repeated warning signals, the waveform validates this transformation. This conduct is consistent with the planned safety system for operator-supervised stabilisation during emergency evacuation.

Additionally, the next-state registers D1 and D0 toggle exactly according to the derived equation:

$$D1 = C, \qquad D0 = W \cdot C + ACK \cdot C \tag{35}$$

and their updates take place exactly at ascending edges of the clock signal. The absence of metastability effects or glitches in the actuator outputs validates that the combinational logic and synchronous D Flip-Flop design adhere to the FSM truth table, guaranteeing proper hardware-level behavior and timing.

C. C# Console-Based Behavioral Verification

Besides the HDL-level verification, a high-level behavioral validation was carried out using a C# console simulation. This software environment creates sensor patterns, calculates the related W/C/ACK flags, assesses state transitions, and creates actuator outputs in order to represent the logical behavior of the MedEvac controller. This simulation serves as a quick confirmation layer reproducing the HDL behavior without necessitating full waveform inspection.

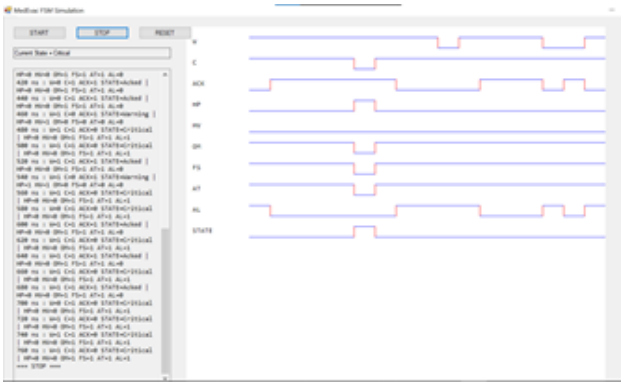

Fig. 6. Console-based C# simulation output showing time-domain state transitions, flag evaluations, and actuator responses.

Evaluated for every simulated time step, the console output shown in Fig. 9 has five columns: Inputs, Flags, State, and Outputs. The Inputs column lists sensor conditions (ST, HS, OC, CS, IM, WS), and the Flags column shows their conversion into W, C, and ACK. The State column displays the encoded FSM state (00= NORMAL, 01 = WARNING, 10 =

CRITICAL, 11 = ACKED), which the Outputs column shows the resulting actuator responses (HP, HV, OM, FS, AT, AL). For example, at early timestamps (0–40 ns), ACK = 1 and C = 1 force the FSM into the ACKED state (11), matching the HDL's next-state rules. Between 60–90 ns, the system momentarily returns to NORMAL (00) when W = 0 and C = 0, confirming the correct implementation of the NORMAL recovery condition.

These results show that the C# high-level behavioral model is logically consistent with the Verilog implementation. The console output also confirms that the Boolean expressions for the actuators, including HP = C'·ST, HV = C'·HS, OM = C, FS = C, AT = C, and AL = C·ACK', are evaluated correctly at each time step. The agreement between C# and HDL simulations offers strong cross-validation, proving that the control logic is accurate at both the algorithmic and hardware levels. This dual-layer verification method is often used in embedded systems development to cut down on debugging time and ensure consistent behavior across different levels of abstraction.

VII. DISCUSSION

The results from the hardware-level simulation, quantum-logic formulation, and digital modeling show that the proposed Autonomous Drone MedEvac Coordination system behaves consistently and predictably in different emergency situations. The multi-sensor CPS architecture keeps track of patient environmental and physiological conditions continuously. The FSM offers a clear decision-making framework that avoids unclear or undefined actions. This fits with the current trend in modern medical and cyber-physical designs, where clear logic and formal correctness are necessary to ensure safety in autonomous systems.

In all experiments, the FSM consistently moves between NORMAL, WARNING, CRITICAL, and ACKED states based on the truth tables, Boolean formulations, and matrix/quantum representations created earlier. The GTKWave waveform results show that actuator signals (HP, HV, OM, FS, AT, AL) respond quickly and accurately to changes in sensor flags. This confirms the correctness of the combinational logic functions and the synchronous D Flip-Flop implementation. Additionally, the C# simulation, while conducted at the behavioral level, produced the same state progressions and actuator patterns. This demonstrates strong consistency among the theoretical model, hardware logic description, and software-level abstraction.

The integration of quantum logic transformation in Section V adds a degree of formal verification that is often missing in undergraduate MedEvac or UAV-control studies. By using bra-ket notation for FSM transitions and breaking down the next-state logic into reversible quantum operators (CNOT, Toffoli), the design achieves provable reversibility. It also remains compatible with future quantum-enhanced embedded platforms. Although the physical system operates on classical hardware, the quantum approach provides a mathematical guarantee that each input state corresponds to a valid and unique output state. This prevents metastable or undefined transitions, which is crucial for life-saving applications such as neonatal transport and hypoxia-CO_2 stabilization.

However, there are still some limitations. The current setup only uses HDL simulation without deploying it on actual FPGA hardware. This means that factors like environmental noise, timing jitter, and real sensor connections have not been tested. Moreover, while the actuator responses follow predictable logic, real cabin conditions such as temperature changes, humidity settling time, and oxygen diffusion rates are not included in the simulation. The CPS design also assumes that sensors work perfectly; in reality, noisy biomedical sensors may need filtering, hysteresis, or sensor fusion methods. Despite these issues, the results strongly suggest that the proposed architecture works, can be implemented, and is a solid starting point for future real-world MedEvac systems.

Overall, the combination of classical FSM modeling, quantum logic transformation, HDL verification, and external C# simulation forms a multi-layer validation pipeline that builds confidence in the system's correctness. These mathematical, logical, and hardware levels together show that the MedEvac controller is strong, can be implemented, and is able to respond in real time to critical physiological conditions during autonomous drone evacuation operations. The system offers a solid foundation for future research involving FPGA deployment, hardware-in-the-loop testing, and integration with real UAV flight controllers.

VIII. CONCLUSION

This work presented an autonomous MedEvac drone control system based on a multi-sensor cyber-physical architecture and a deterministic FSM-based safety controller. The digital design used Boolean logic, truth tables, and synchronous next-state mapping. It was formally expanded through a quantum logic transformation for reversible verification. HDL simulation with Verilog and GTKWave confirmed that the state transitions and actuator outputs matched the expected medical emergency behavior. Similarly, C# behavioral validation produced the same logic responses. Although the system has not been deployed on physical FPGA hardware yet, the simulation results show that the controller is functionally correct, implementable, and suitable for real-time MedEvac stabilization. Future work will focus on hardware deployment, integration with real biomedical sensors, and UAV flight-level testing.

ACKNOWLEDGEMENT

The author would like to thank Dwi Oktavianto Wahyu Nugroho, S.T., M.T., for his guidance, supervision, and valuable insights during this research. They also appreciate the Instrumentation Engineering Department, Faculty of Vocational Studies, Sepuluh Nopember Institute of Technology, for providing academic support and laboratory facilities. The authors acknowledge the use of Verilog HDL, GTKWave, and C# development tools, which allowed for the hardware-level and behavioral verification of the MedEvac control system. Lastly, author express sincere gratitude to colleagues and peers

who provided feedback during the simulation and validation stages.

REFERENCES

[1] N. B. Roberts et al., "Current summary of the evidence in drone-based emergency medical services care," Resusc Plus, vol. 13, p. 100347, Mar. 2023, doi: 10.1016/j.resplu.2022.100347.

[2] R. Emad Alfaris, Z. Vafakhah, and M. Jalayer, "Application of Drones in Humanitarian Relief: A Review of State of Art and Recent Advances and Recommendations," Transportation Research Record: Journal of the Transportation Research Board, vol. 2678, no. 7, pp. 689–705, Jul. 2024, doi: 10.1177/03611981231209033.

[3] M. Kunovjanek and C. Wankmüller, "Containing the COVID-19 pandemic with drones - Feasibility of a drone enabled back-up transport system," Transp Policy (Oxf), vol. 106, pp. 141–152, Jun. 2021, doi: 10.1016/j.tranpol.2021.03.015.

[4] M. A. H. Zailani et al., "Influence of drone carriage material on maintenance of storage temperature and quality of blood samples during transportation in an equatorial climate," PLoS One, vol. 17, no. 9, p. e0269866, Sep. 2022, doi: 10.1371/journal.pone.0269866.

[5] S. Soni, P. Chandra, P. Chandra Sharma, J. Gangrade, and D. K. Singh, "Medical kit delivery using Drone: Critical medical infrastructure solution for emergency medical situation," International Journal of Disaster Risk Reduction, vol. 108, p. 104502, Jun. 2024, doi: 10.1016/j.ijdrr.2024.104502.

[6] T. Mesar, A. Lessig, and D. R. King, "Use of Drone Technology for Delivery of Medical Supplies During Prolonged Field Care," Journal of Special Operations Medicine, vol. 18, no. 4, p. 34, 2018, doi: 10.55460/M63P-H7DM.

[7] M. Eichleay, E. Evens, K. Stankevitz, and C. Parker, "Using the Unmanned Aerial Vehicle Delivery Decision Tool to Consider Transporting Medical Supplies via Drone," Glob Health Sci Pract, vol. 7, no. 4, pp. 500–506, Dec. 2019, doi: 10.9745/GHSP-D-19-00119.

[8] Á. Restás, "Drone Applications Fighting COVID-19 Pandemic—Towards Good Practices," Drones, vol. 6, no. 1, p. 15, Jan. 2022, doi: 10.3390/drones6010015.

[9] L. A. Cortés, P. Eles, and Z. Peng, "Modeling and formal verification of embedded systems based on a Petri net representation," Journal of Systems Architecture, vol. 49, no. 12–15, pp. 571–598, Dec. 2003, doi: 10.1016/S1383-7621(03)00096-1.

[10] X. Li, J. Tupayachi, A. Sharmin, and M. Martinez Ferguson, "Drone-Aided Delivery Methods, Challenge, and the Future: A Methodological Review," Drones, vol. 7, no. 3, p. 191, Mar. 2023, doi: 10.3390/drones7030191.

[11] X. Ma, X. Liang, M. Ning, and A. Radu, "METRIC: Toward a Drone-based Cyber-Physical Traffic Management System," in 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, Oct. 2022, pp. 3324–3329. doi: 10.1109/SMC53654.2022.9945433.

[12] S.-W. Huang, S.-Y. Chiou, R.-C. Chen, and C. Sub-r-pa, "Enhancing Hospital Efficiency and Patient Care: Real-Time Tracking and Data-Driven Dispatch in Patient Transport," Sensors, vol. 24, no. 12, p. 4020, Jun. 2024, doi: 10.3390/s24124020.

[13] M. U. H. Al Rasyid, S. Sukaridhoto, A. Sudarsono, and A. N. Kaffah, "Design and Implementation of Hypothermia Symptoms Early Detection With Smart Jacket Based on Wireless Body Area Network," IEEE Access, vol. 8, pp. 155260–155274, 2020, doi: 10.1109/ACCESS.2020.3018793.

[14] M. M. Gouda, "Fuzzy ventilation control for zone temperature and relative humidity," in Proceedings of the 2005, American Control Conference, 2005., IEEE, pp. 507–512. doi: 10.1109/ACC.2005.1469986.

[15] P. Chanyagorn and P. Kiratiwudhikul, "Fuzzy control of oxygen gas content for premature labor infants oxygen therapy," in 2016 International Conference on Electronics, Information, and Communications (ICEIC), IEEE, Jan. 2016, pp. 1–5. doi: 10.1109/ELINFOCOM.2016.7563026.

[16] V. Kumar S, P. Samanvitha S, V. Kumar K, S. S, Y. N, and S. C. A, "Indoor Air Quality Monitoring System for Healthcare," in 2024 Ninth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), IEEE, Apr. 2024, pp. 1–5. doi: 10.1109/ICONSTEM60960.2024.10568669.

[17] K. Hiratsuka, Y. Nishida, and H. Mizoguchi, "Infant drowning prevention system with wireless accelerometer - Evaluation of optimum floating body shape for home-use -," in 2008 IEEE Sensors, IEEE, Oct. 2008, pp. 1218–1221. doi: 10.1109/ICSENS.2008.4716662.

[18] A. Widianto, I. Nurfitri, P. Mahatidana, T. Abuzairi, N. R. Poespawati, and R. W. Purnamaningsih., "Weight monitoring system for newborn incubator application," 2018, p. 040013. doi: 10.1063/1.5023983.

[19] V. Puram, K. Karuppasamy, K. M. George, and J. P. Thomas, "Quantum Implementation of Finite State Automata," 2025, pp. 326–341. doi: 10.1007/978-3-031-92608-2_23.

[20] S. Yang, X. Gao, and J. Ren, "Sequential Circuits Synthesis for Rapid Single Flux Quantum Logic Based on Finite State Machine Decomposition," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 42, no. 10, pp. 3315–3326, Oct. 2023, doi: 10.1109/TCAD.2023.3245542.

BIOGRAPHIES

Mahendra Dwi Fahreza studied Industrial Automation Engineering at SMK Negeri 1 Batam, graduating in 2020. He completed industrial internships as an Assistant Method Engineer at PT TESE Manufacturing Indonesia and as a Facilities Technician at PT SIIX Electronics Indonesia. He is currently pursuing a Bachelor's degree in Instrumentation Engineering at the Department of Instrumentation Engineering, Faculty of Vocational Studies, Institut Teknologi Sepuluh Nopember (ITS), starting in 2024.

His academic interests include digital systems, embedded systems, automation engineering, and cyber-physical systems. He can be contacted at the following email address: mahendrafhrz1202@gmail.com.

Dwi Oktavianto Wahyu Nugroho received his Bachelor's degree in Electro Engineering from the Engineering Faculty, Brawijaya University, in 2009. He completed his Master's degree in Electrical Engineering (Industrial Electronics) at the Institut Teknologi Sepuluh Nopember (ITS) in 2016. His academic interests include medical instrumentation, sensor system development, energy harvesting, and mechatronics.

He began his professional career in 2009 at Cipta Karya, Dinas Pekerjaan Umum, East Java Province, then continued his work at Perusahaan Daerah Air Minum (PDAM) in Sidoarjo from 2010 to 2017. He currently serves as a lecturer in the Department of Instrumentation Engineering, Faculty of Vocational Studies, Institut Teknologi Sepuluh Nopember (ITS), Indonesia. He can be contacted at: Oktavianto@instrum-eng.its.ac.id.