Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X
Volume 3, Issue 12, 2025

INTELLIGENT DEEP LEARNING-BASED SDN FRAMEWORK FOR REAL-TIME FAULT MANAGEMENT IN SMART WIND ENERGY SYSTEMS

Khawaja Tahir Mehmood

Department of Electrical Engineering, Bahauddin Zakariya University, Multan, 60000, Pakistan

ktahir@bzu.edu.pk

DOI: http://doi.org/10.5281/zenodo.19127612

Keywords
Wind turbine energy system; Software Defined Network (SDN); Message Queuing Telemetry Transport (MQTT); Fault detection and mitigation; Graph Neural Network (GNN); K-Nearest Neighbors (KNN); Hybrid SCADA models.

Article History
Received: 02 November, 2025
Accepted:18 December, 2025
Published: 31 December, 2025

Corresponding Author: *
Khawaja Tahir Mehmood

Abstract
The modern smart grid system using renewable energy (wind energy) as the basic energy-producing source requires an Artificial Intelligence (AI) based fault management system for handling the structural and operational limitations of the wind energy system to have high efficiency and standability. This study proposes real-time fault detection and mitigation in wind turbine systems using an SDN-POX controller integrated with the Temporal Convolutional Method (TCM). The POX controller deployed in Mininet extracts the real-time sensor data streams (e.g., vibration, temperature, rotor speed, pitch angle, and torque) of a wind turbine via Message Queuing Telemetry Transport (MQTT). Incoming data from each sensor channel is first structured into normalized multi-channel auto-updating sliding windows to facilitate the automatic extraction of reference operating points and dynamic tolerances via real-time streaming mean and standard deviation estimations. In order to fuse sensor modalities, the TCM features residual stacks (to record both short-term and long-term dependencies without recurrence), parallel sequence modeling, and channel mixing via 1-D causal-dilated convolutions. A SoftMax-based multi-class head is used to classify fault states, and an anomaly score measures deviations from nominal circumstances. The controller compares real-time feature vectors against learned reference-tolerance pairs to trigger Finite State Machine (FSM)-based mitigation in MATLAB/Simulink, including torque limitation, pitch correction, or braking. However, the fault management (involving detection and mitigation) take place within one control loop, with a fault detection delay (TFD) under 300 ms and a fault mitigation delay (TFSM)of less than 50 ms. By conducting experiments, we discovered that our approach detects faults more accurately and faster in mitigating and normalizing faults as compared to traditional methods (Graph Neural Network-GNN, K-Nearest Neighbors-KNN, and Supervisory Control and Data Acquisition-SCADA models).

1 INTRODUCTION

The renewable wind energy emerged as the backbone of modern smart power networks as a bedrock of the global shift to sustainable electricity. About 30% of global electricity generation came from renewable sources in 2023; this percentage has significantly increased in recent years [1]. Following such growth, wind power is expected to surpass hydropower as the world's second-largest

renewable electricity source by 2030 [2-3]. 2023 featured the deployment of a record 117 GW of new wind energy systems (worldwide), the highest ever in a single year [4]. Wind energy system rapid expansion emphasizes the significance of accomplishing climate goals and advancing smart grids, which use cutting-edge ICT (information and communication technologies) to effectively integrate renewable energy sources. But it also presents new technological difficulties that need to be resolved to guarantee the electricity system operates dependably [5]. Notably, the scattered and fluctuating nature of wind power necessitates the use of intelligent control techniques for preserving grid stability. Identification and resolution of faults in network infrastructures associated with wind turbines is a major operational challenge in wind energy systems. There are several possible failure modes for wind turbines because of the severe climatic conditions and ongoing mechanical stress they endure [6-7]. The most common problems that result in unexpected outages or decreased power output include electrical (such as generator faults, converter problems, etc.) and mechanical (such as gearbox wear, blade cracks, and bearing failures, etc.) issues. The research [8] indicates that more than 70% of wind turbine failures are caused by crucial equipment such as gearboxes, generators, and blades, highlighting the necessity of efficient monitoring of these components. In addition to expensive repairs, these issues cause turbine outages, which raise the levelized cost of electricity. Therefore, industry experts emphasize that improving the financial viability of wind farms requires eliminating costly maintenance and unplanned downtime through early defect detection [9]. Additionally, a lot of wind farms are situated in hard-to-reach offshore or remote locations, making manual inspection and maintenance costly and time-consuming [10]. An undetected fault in a sizable wind farm could cause power imbalances
even cascading interruptions in a smart grid environment if it sets off protection devices in other parts of the network [11]. Therefore, there is a great push to create automated fault diagnosis systems that can locate and detect turbine faults quickly so that control systems can isolate or fix the issue by initiating preventative measures (such as unplugging a malfunctioning turbine or rerouting power flow), before it gets worse [12-14]. Today, the modern wind power system requires a real-time intelligent fault-resilient necessity that detection techniques should accomplish both objectives: (a) 5G/IoT connection and (b) cloud-edge collaborative architectures to guarantee accurate and timely issue detection [15].In recent years, the incorporation of Internet of Things (IoT)-based protocols and Software-Defined Networking (SDN) into smart wind energy systems has become increasingly popular as a solution to these issues [16-17]. SDN, by separating the control plane from the data plane, provides a programmable and centralized control architecture that facilitates traffic monitoring, dynamic policy enforcement, and quick network layer reconfiguration [18]. The SDN can be used for real-time fault management entity for any energy system (i.e., wind energy) by using any light (less overhead) communication broker protocol [19]. The MQTT can be integrated with the SDN controller to have access to the energy system and perform the fault and energy management services [20]. In parallel, Finite State Machine (FSM), because of its predictable nature, modular design, and simplicity in integrating with fault management procedures, has become more and more common for modelling control logics in cyber-physical systems [21]. Offline diagnostic models and Supervisory Control and Data Acquisition (SCADA) systems have historically been key components of wind turbine condition monitoring and control [22-23]. However, the

latency, scalability, and centralized communication restrictions that SCADA-based systems encounter make them less than ideal for adaptive control and real-time fault detection in large wind farms [24-25]. Furthermore, while data-driven methods like Graph Neural Networks (GNN) and K-Nearest Neighbours (KNN) have demonstrated promise in fault classification, they frequently lack the speed and responsiveness needed for control choices in dynamic contexts [26-27]. However, the combined use of SDN, MQTT, and FSM-based logic control for real-time problem detection and mitigation in wind turbine systems is still understudied, despite individual advancements in these technologies.

1.1. Work Contribution

To have a wind turbine energy system highly efficient, reliable, and sustainable energy-producing system, the proposed framework deploys a Software Defined Network (SDN)- POX controller integrated with MQTT and FSM using a proposed intelligent sensing real- time fault management algorithm that is Temporal Convolutional Method (TCM) to efficiently manage (fault detection and mitigation) faults in wind energy systems. For this goal, the proposed framework requires the integrated protocol, namely, Message Queuing Telemetry Transport (MQTT) and MATLAB/Simulink-based control. In Simulink, a detailed model of a wind turbine is designed with dynamic subsystems, namely: (a)generator, (b)gearbox, (c) shaft, and (d) pitch controllers. The multiple sensors are employed specifically to maintain the logs containing data sets of wind turbine vibration, temperature, rotor speed, pitch angle, etc. These data sets are published to an MQTT broker. The POX-based SDN controller (designed on Mininet) decodes the messages from sensors via MQTT and uses a proposed (TCM) algorithm to identify any deviation from the usual data patterns. After the fault is detected, control signals are sent back to the Finite State Machine (FSM) controller in Simulink using MQTT that operates under the instructions of the SDN controller to strategize the torque limitation, pitch correction, or braking, etc., until the instruction obtained from the SDN controller indicates that the fault is normalized. The working model is shown in Fig.1.The key characteristics and benefits of the proposed approach are as follows:

a) The real-time and rapid fault detection with low latency delay is possible in the proposed framework that uses SDN-POX controller with (TCM) algorithm logic to monitor the sensor data and compare with reference thresholds to have detection of real-time developed anomalies with minimum processing delays.

b) The proposed algorithm is equipped with an FSM model design that operates rapidly in four states (normal mode, fault mode, recovery mode, and resume normalized mode) with the minimum delay under the influence of the SDN-POX controller.

c) In our proposed method, there are minimal overhead values as the light-weight broker protocol MQTT is used for integration between the SDN-POX controller and FSM.

d) In the proposed framework, the fault management (involving detection and mitigation) take place within one control loop, with a fault detection delay (TFD) of less than 300 ms and a fault mitigation delay (TFSM)of less than 50 ms.

e) The objective cost function (J) is minimum in the proposed framework as compared to traditional fault management systems (SCADA, KNN, GNN). However, the metric (J) involves fault detection delay (T_{FD}), fault mitigation delay (T_{FSM}), Percentage Detection Accuracy (%A), Percentage False Positive Rate (%FPR), factor (A) represents the detection accuracy and Percentage False Negative Rate (%FNR). The objective function is explained in Eq.A. Where (α, β, γ, δ, ϵ) are the weights to optimize the importance of each term.

$$J_{Min} = \alpha \cdot T_{det} + \beta \cdot T_{FSM} + \gamma \cdot FPR + \delta \cdot FNR - \epsilon . A \quad (A)$$

In Eq. A, the higher (α, β) prioritize low latency, the higher (γ, δ) prioritize reliability, and the higher (ϵ) accuracy. In our model, the typical values of these weights are set as (α= 0.4, β = 0.3, γ = 0.1, δ = 0.1, ϵ = 0.1).

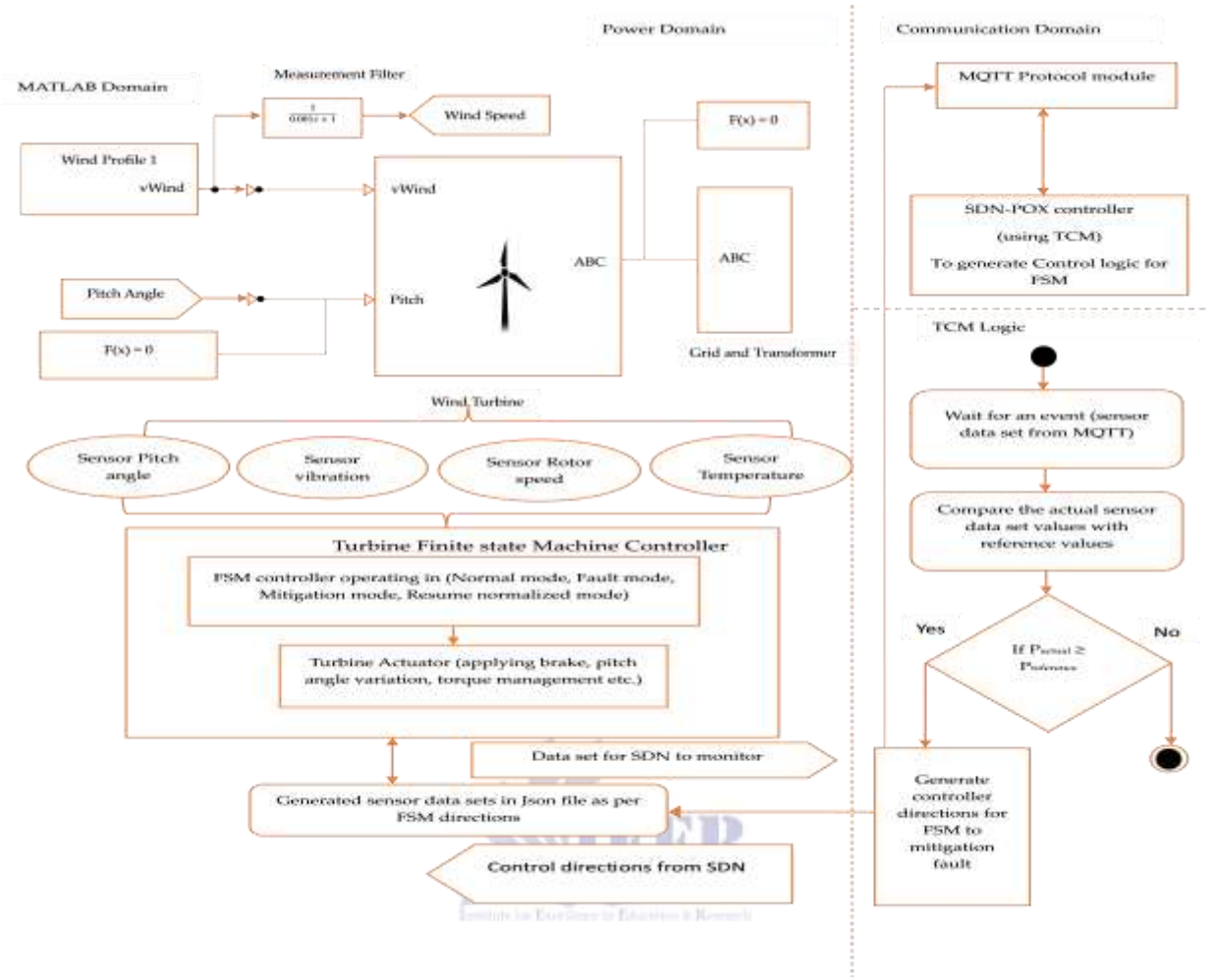

Figure 1: The working scenario of the proposed framework (SDN-POX controller using TCM logic).

1.2. Research objectives

a) To design a real-time intelligent fault management framework for wind energy systems benefiting from the SDN-POX controller functionality in terms of minimum overhead and faster data delivery.

b) To design an FSM model that switches efficiently wind energy system parametric values concerning the transition in four states (normal mode, fault mode, recovery mode, resume normalized mode) under the control instruction of the SDN-POX controller using the proposed algorithm logic.

c) To ensure fast detection and low overhead, a lightweight protocol (MQTT) is implemented between SDN-POX controllers and wind sensors.

d) To minimize the objective cost function (J).

e) To evaluate the performance of the proposed framework with traditional fault management approaches (SCADA, GNN, and KNN).

The literature research and a comparison of the proposed framework with traditional fault management strategies are covered in Section 2. Section 3 discusses the methodology to achieve the intended goal. The simulation findings and the conclusion are discussed in the final section.

2 Literature Review

To make wind turbines' energy system fast, reliable, highly efficient, and have sustainable energy production, with rapid problem identification and mitigation, there is an essential need for an Artificial Intelligence (AI) based fault detection and mitigation system. One such method could be as proposed in this research method, a real-time intelligently controlled fault management framework that involves the Software Defined networking (SDN-POX) controller using the proposed intelligent sensing fault management algorithm that is Temporal Convolutional Method (TCM) for rapid problem identification and mitigation. This is possible with the integration of the SDN-POX controller with Finite State Machine (FSM) using a broker protocol, Message Queuing Telemetry **Transport** (MQTT). The FSM offers quick, targeted mitigation actions with state-aware transitions, and comparison of real-time sensor data with reference criteria, and initiates exact FSM states, including Mitigation Active, Recovery, and Resume Normal Mode. Over the years, authors have put up various strategies to address this problem. The contributions and limits of various research methods to improve wind energy efficiency and make wind energy systems fault resilient are explained in Table 1, and a comparison of these research methods is also drawn with the proposed framework (SDN-POX controller integrated with FSM via MQTT protocol) in light of research objectives defined in Section 1.

Table 1: Contributions and drawbacks of various research methods for improving the wind energy system efficiency.

References	The author's method contribution	Restrictions and contrast with the proposed framework
G. Sasikala.et.al [28]	Proposes wind turbine fault monitoring using sensors + MQTT protocol.	• Detects only specific faults (temperature, bearing). • No coordinated control response. • Proposed SDN–FSM ensures real-time (<300 ms) detection with active, state-aware mitigation
Y. Vidal.et.al [29]	SCADA-based multi-fault detection using SVM (98.2% accuracy).	• Depends on dense historical SCADA data. • Relies on offline learning (PCA, feature selection). • Lacks rapid in-situ mitigation. • Proposed TCM + SDN achieves live streaming, sub-second detection, and FSM-based automatic response.
P. Sudhakar.et.al [30]	AI-based fault detection is more accurate than conventional methods.	• Focuses on prediction, not a fast detect-and-act cycle. May lack unified low-latency mitigation. • The proposed framework integrates intelligent sensing + SDN + FSM for instant detection and response.

C.Y. Lee. et.al [31]	Deep CNN + PSO-optimized XG-Boost on resampled SCADA data	• Heavy preprocessing (SMOTE, feature extraction). • No real-time control loop. • Proposed TCM provides on-the-fly detection (TFD < 300 ms) and FSM-based rapid response (TFSM < 50 ms).
M. Nithya. et.al[32]	Neural networks for wind turbine fault detection.	• Requires large training data. • Risk of overfitting. • Scalability and real-time deployment issues. • Proposed MQTT + SDN + FSM avoids overfitting, ensures fast state transitions, and accurate real-time detection.
D. Zhang. et.al[33]	Ensemble (RF + XG-Boost) for robust fault detection	• Requires handcrafted features and offline training. • Limited adaptability. • Proposed TCM enables real-time sensing, SDN-based routing, and FSM-driven instant fault response.
G. Jiang.et.al[34]	Denoising autoencoder with temporal context for anomaly detection	• Only anomaly detection (no fault type classification). Requires threshold tuning. • No integrated control. • Proposed SDN–POX–MQTT–FSM ensures classification + real-time mitigation with coordinated control.
A.Bouzekri.et.al[35]	AI-based fault-tolerant control for converter failures.	• Niche: limited to converter switch faults. • Involves complex reconfigurations. • Proposed TCM generalizes to all turbine subsystems with SDN global view + FSM for rapid mitigation.
P.H.F.Sousa.et.al[36]	IoT-based generator fault detection (94.4% accuracy, low false alarms).	• Mainly diagnostic (not fast in-loop control). • Proposed SDN + MQTT + FSM enables real-time monitoring, instant fault localization, and proactive fault handling.
A.Santolamazza.et.al[37]	ANN-based SCADA data analysis for predictive maintenance	• Requires preprocessing and thresholding. • Higher latency in fault alarms.

		• Proposed TCM + SDN processes data in real time, and FSM adjusts turbine modes (mitigation < 50 ms).
Z. Zemali.et al[38]	ANFIS + Kalman observer hybrid for precise fault detection	• Computation-heavy. • Works in siloed scenarios. • No integration with communication networks. • Proposed SDN–FSM handles diverse faults simultaneously with faster response.
T. Han. et.al[39]	Semi-supervised adversarial deep learning for fault diagnosis	• Computationally intensive. • Focuses on accuracy, not real-time mitigation. • Proposed TCM ensures <300 ms detection and <50 ms response without adversarial overhead.
Y. Yu et al. [40]	SDN-based fault management survey for networks.	• Focus on network-level resilience (not turbine-specific). • No turbine fault classification or mitigation. • Proposed TCM integrates SDN with FSM for turbine-aware state-driven fault handling.
A.G.Kavaz. et.al[41]	ANN-based sensor fault detection for turbines.	• Limited to anomaly detection at the sensor level. • No system-level corrective action. • Proposed TCM includes FSM transitions (safe mode, rerouting) for real-time sensor fault handling.
M.J. Morshed.et.al [42]	Fault-tolerant control for grid faults using microgrid integration.	• Focus only on grid faults. • Requires complex control tuning. • Proposed TCM unifies electrical + mechanical fault management with SDN sensing + FSM mitigation (≈350 ms).

3. Research Methodology

To have highly efficient, reliable, and sustainable energy production along with intelligent fault-resilient wind energy systems, the proposed framework is implemented with dynamic features of POX-based SDN controller (designed on Mininet) using a proposed intelligent sensing real-time fault management algorithm that is Temporal Convolutional Method (TCM) to identify any

deviation from the usual data patterns integrated with Message Queuing Telemetry Transport (MQTT) protocol. In this section complete methodology of the proposed framework is presented, along with simulation parameters and a mathematical-based objective function.

3.1. Design and Implementation of the Proposed Framework:

In order to fully explain the design of the proposed framework, this section is further subdivided into three sub-sections. Starting from the design of the wind energy system along with sensor integrations on MATLAB, secondly, the design of the Finite State Machine (FSM), and then integrating the wind energy system with SDN-based POX controller to perform fault detection and mitigation process using the broker protocol Message Queuing Telemetry Transport (MQTT).

3.1.1 Designing a wind energy system with integrated sensors:

This is the first operational step in the design of the proposed framework. The wind turbine energy system is designed on MATLAB, containing the following components:

a) Wind source: In the proposed framework to replicate the real-time behavior, we have opted for realistic gusty wind behavior (i.e., wind speed is simulated as a variable wind with features of base speed: 10 m/s, fluctuation: ± 2 m/s as sinusoidal gusts, and noise: ±0.5 m/s as random variation). The variable wind speed is obtained by using Eq.1.

$$S_W(t) = 10 + 2sin(0.2\pi t) + 0.5randn(size(t)) \quad (1)$$

Where S_w is the wind speed, the factor $(2sin(0.2\pi t))$ shows the periodic fluctuation in wind speed, with amplitude = 2 m/s, providing the wind speed can vary ±2 m/s from the mean, and the factor $0.5randn(size(t))$ adds Gaussian noise of 0.5 scales.

b) Rotor blades: Converts the wind energy to mechanical torque using aerodynamic equations based on speed ratio (λ) and power coefficient (C_p), explained in Eq. (2-5).

$$\lambda = \frac{\omega \cdot R}{S_W} \quad (2)$$

(where ω is the angular velocity of the rotor and R is the rotor radius)

$$C_p = (0.22).\left(\frac{116}{\lambda - 0.4(\beta - 5)}\right).e^{\frac{-12.5}{\lambda}} \quad (3)$$

(β = 0 here for fixed-pitch blades.)

$$P_m = 0.5.rho.\ \pi.\ R^2.\ Cp.\ Sw^3 \quad (4)$$

(P_m is the mechanical power extracted from wind, and rho is the air density, a constant for aerodynamic calculations.)

$$T_m = \frac{P_m}{\omega + 0.1} \quad (5)$$

(T_m is the torque generated from the mechanical power, and 0.1 is intentionally added to the denominator to avoid zero at the start.)

c) Low-speed shaft and Gearbox: It helps to scale up the rotational speed based on the defined gear ratio. This phenomenon is calculated by Eq. 6

$$\omega_{scaled\_up} = \omega_{rotor} + G_{Ratio} \quad (6)$$

d) High-speed shaft and Generator: It helps to calculate the electrical output power. This is shown in Equation 7

$$P_E = \omega_{scaled\_up} \cdot Tm \quad (7)$$

Pitch controller: It helps to optimize the output power and prevent overspending. Based on the above-mentioned components (as explained in headings a-e). The algorithm for designing a wind energy system on MATLAB is explained in Appendix A.

f) Sensor modeling: In the proposed framework, five sensors are used to calculate the fluctuations and record the readings to be forwarded to the SDN-POX controller to detect and mitigate the fault and make the proposed wind energy system fault resilient. (i)Vibrational sensor measures the mechanical vibrations on the shaft. (ii) The temperature sensor senses the thermal behavior of the generator. (iii)Rotor speed sensor-measure the angular velocity using derivatives and gain blocks. (iv)Pitch angle encoder -tracks blade angle using PID controller output, and (v) Torque sensor-measures the dynamic torque. The algorithm for the integration of sensors with a wind energy system designed on MATLAB is explained in Appendix B.

3.1.2 Design of Finite State Machine (FSM):

In the proposed framework, the Finite State Machine (FSM) is designed on MATLAB, based on the state flow principle, and provides a wind energy system with safety and continuous operation by interpreting the control signal from the SDN-POX controller via the MQTT protocol. It performs the mitigation actions (i.e., torque reduction, adjustment of pitch angle, and emergency braking, etc.) under the instructions of the SDN-POX controller. The FSM works in five states: (1) normal mode (NM), (2) fault detection (FD), (3) mitigation mode (MM), (4) recovery mode (RD), and (5) resume normal mode (RNM). The FSM input ports are used for fault detection and recovery signals from the SDN-POX controller via the MQTT protocol. The output ports of the FSM are used for mitigation actions (i.e., torque reduction, adjustment of pitch angle, and emergency braking, etc.). The whole scenario, based on five stages (NM, FD, MM, RD, and RNM) using the state flow principle, is explained in Appendix C.

e)

3.1.3 Designing and implementing SDN-POX logic:

The SDN-POX controller is designed on the Mininet and integrated with a wind energy system for real-time fault management (detection and mitigation) via the MQTT protocol. The SDN controller subscribes to the topic (addressing label type of MQTT protocol) as explained in [43], containing the data sets (data logs of wind energy system sensor data) in the form of a JSON file. The POX controller is loaded with a proposed intelligent sensing real-time fault management algorithm that is the Temporal Convolutional Method (TCM), which features convolutional operations over time-sequenced data to capture dependencies and patterns using 1-D causal-dilated convolutional architectures. The 1-D causal convolution feature makes the filters slide over the time axis, ensuring output at time t only depends on input (t) or before (t). The dilated helps to skip input points to exponentially increase the receptive field without adding layers. These combined features help to process the sequence in parallel, unlike a Recurrent Neural Network (RNN) that processes the sequence step by step. The proposed algorithm (TCM) is also more efficient than the Graphical Neural Network (GNN), adding graph processing overhead without benefits. The same is the case with K-Nearest Neighbors (KNN), which is also very slow as compared to the proposed method because it needs to compare the new samples with all historical samples. The complete working is shown in the form of steps:

Step1-Formation of sliding window (X_t):

Each sensor of the wind turbine produces data continuously and sends it to the SDN-POX

controller using the MQTT protocol. The SDN-POX controller arranges them into the matrix of $R^{C\times T}$. Let the data set produced by the vibration sensor is (x_t^{vib}), data set for rotor speed sensor is (x_t^{w}), data set for pitch angle is (x_t^{θ})' and data set for torque is given as (x_t^{τ}) then the auto shifting sliding window (X_t) generated by POX controller is given by Eq.8. The sliding window belongs to matrix $R^{C\times T}$ (whose rows are the number of channels, that are the number of sensors, while columns are the window length factor T).

$$X_t=\begin{bmatrix} x_{t-T+1}^{vib} & x_{t-T+2}^{vib} & \cdots & x_t^{vib} \\ x_{t-T+1}^{w} & x_{t-T+2}^{w} & \cdots & x_t^{w} \\ x_{t-T+1}^{\theta} & x_{t-T+2}^{\theta} & \cdots & x_t^{\theta} \\ x_{t-T+1}^{\tau} & x_{t-T+2}^{\tau} & \cdots & x_t^{\tau} \end{bmatrix} \text{ belongs to } R^{C\times T} \quad (8)$$

In Eq.8, the (t) represents the current value of time, while (T) is the last time step. For e.g (if t = 100 and T=5 then we will have)

t-4	t-3	t-2	t-1	t
96	97	98	99	100

The general form of the sliding window is given by Eq.9

$X_t= [x_{t-T+1}, x_{t-T+1}\ldots\ldots\ldots\ldots x_t]$ (9)

Step2-Normalization of different sensor scales:
The data set from different wind turbine sensors arranged in the form auto-updating sliding window next undergoes to the step of normalization before feeded as input to TCM algorithm due to following reasons:(1) turbine sensors measure different things (i.e., vibration from 0 to 0.002 g range, torque could be of hundreds Nm, rotor speed tens of rad/s) if these raw values are directed feeded to TCM model then large magnitude will dominate and small magnitude values will be ignored. (2) Sensors do not stay stable, and values resulting from slow drift will be picked as a fault by the TCM model. So, in order to overcome all the above-mentioned issues, the normalization of the dataset is required. The normalization processing involves running mean ($\mu_{t,c}$), finding variance ($6^2_{t,c}$), evaluating standard deviation ($6_{t,c}$), and calculating the Z-test score ($Z_{t,c}$). The complete normalization process is expressed in the form of Eq. (10-14).

$\mu_{t,c} = \frac{1}{n_t}\sum_{i=1}^{n_t} x_{i,c}$ at time (t) for channel (C) (10)

The mean is the reference normal value of all past measurements.

$6^2_{t,c} = \frac{1}{n_{t-1}}\sum_{i=1}^{n_t}(x_{i,c} - \mu_{t,c})^2$ variance of channel C after n_t readings (11)

The variance shows how spread out the values are from the mean.

$6_{t,c} = \sqrt{6^2_{t,c}}$ The standard deviation is the average distance of the readings from the mean. (12)

$Z_{t,c} = \frac{x_{t,c}-\mu_{t,c}}{6_{t,c}}$ (13)

After the normalization process, Equation 9 becomes

$X_t= [z_{t-T+1}, z_{t-T+1}\ldots\ldots\ldots\ldots z_t]$ (14)

Step3-Working of the TCM algorithm:

i.Channel mixing with 1×1 convolution 1D blends sensor: TCM takes all sensors' readings at the same time step and blends them using learned weights.

ii. Causal convolution with dilation factor: The TCM performs convolution of past points spaced by the (d) steps, multiplied by weights, and summed together to produce output. This statement is expressed in Eq.15.

$Y_t=\sum_{i=0}^{k-1} W_i . X_{t-id}$ (15)

(K)is the kernel size, (d) is the dilation factor, (W_i) is the learned weight, and(X_{t-id}) is the input feature vector from the past time step.

iii. First of all, the TCM is trained adopting a supervised learning model by feeding it with N=10,000 data samples (of sensor channel) in the form of Equation (8). The TCM for each sensor channel C calculates the mean (that serves as the reference value) and tolerance using (standard deviation with K=3) as shown in Eq. (16-19).

$\mu_c = \frac{1}{N}\sum_{i=1}^{N} x_{i,c}=P_{normal}$ (Reference value of particular sensor channel C) (16)

$6^2_c = \frac{1}{N-1}\sum_{i=1}^{N}(x_{i,c} - \mu_c)^2$ (17)

$6_c = \sqrt{6^2_{t,c}}$ standard deviation (18)

$(Tol)_c=k.\ 6_c=P_{tolerance}$ (19)

In our model, the P_{normal} (for temperature, vibration, rotor speed, and torque) obtained from TCM are (temperature_ref_ ref =65.0, vibration_ ref =0.03, rotor speed_ ref =1500, torque_ ref =400.0, and pitch_angle_ ref =12.0). The Ptolerance (for temperature, vibration, rotor speed, and torque) obtained from TCM is (Temperature_tol_ tol =5.0, vibration_ tol =0.01, rotor speed_tol =200, torque_ tol =50.0, and pitch_angle_ tol =3.0)

iv. The TCM looks at the sensor normalized data set in the form of a sliding window (X_t) and gives 1 score for each possible condition (normal, gearbox fault, shaft fault, pitch fault, and generator fault). These scores are called logits (O_k) and use the SoftMax formula to calculate the probability of each fault type ($P_k(t)$) and anomaly score $S_a(t)$. These two factors are calculated by Eqs. (20 and 21).

$P_k(t)=\frac{e^{Ok(t)}}{\sum e^{Oj(t)}}$ is the probability of each fault type (20)

$S_a(t)$= fault likelihood (0-1) (21)

v. The TCM performs the real-time parallel processing (comparison) of all the sensor channel data arranged in a sliding window obtained by the SDN-POX controller with known (P_{normal} and $P_{tolerance}$) parameters and generates the error function. The error function result, along with mitigation details, is forwarded to the POX controller to further direct FSM to perform the (normal mode, fault mode, recovery mode, and resume normal mode) as per instructions. The logic of fault management by the proposed algorithm (TCM) is as follows:

```
{
If (P_actual – P_normal) <P_tolerance
  {
  fault_detected =FALSE
        fault_flags= []
   }
Else
If (P_actual – P_normal)> P_tolerance
  {
  fault_detected =True
 CREATE control logic for FSM
command: "mitigate"
```

brake: TRUE
torque_limit: 0.3
pitch_correction: 5
timestamp: CURRENT_TIME
}
Else
If ($P_{actual} - P_{normal}$) = $P_{tolerance}$
{
fault_detected =True
CREATE control logic for FSM
command: "recovery"
brake: false
torque_limit: 0.6
pitch_correction: 2
timestamp: CURRENT_TIME
}
Else
If ($P_{actual} - P_{normal}$) < $P_{tolerance}$
{
fault_detected =False
CREATE control logic for FSM
command: "resume normalized operation"
brake: false
torque_limit: 1.0
pitch_correction: 0
timestamp: CURRENT_TIME
}
End
}

The working of the proposed algorithm, Temporal Convolutional Method (TCM), is explained in Algorithm 1.

Algorithm #1: Working of the proposed intelligent sensing real-time fault management Temporal Convolutional Method (TCM) algorithm

1. Begin
2. # Training of the TCM model based on supervised learning
Calculate the mean (μ_c) and tolerance $(Tol)_c$ of the sensor channel
Pnormal= μ_c and Ptolerance= $(Tol)_c$
3. # Connect to MQTT Broker
CONNECT to broker at "localhost"
SUBSCRIBE to topic "wind/sensor_data."
WAIT for MQTT message from "wind/sensor_data."
PARSE message as sensor_data
SET fault_detected ← FALSE
INITIALIZE fault_flags ← EMPTY_LIST
4. # Comparing sensor value with reference value
5. LOOP START
IF (wind/sensor_data − ref_ sensor_data) > tolerance_ sensor_data

```
6.          THEN
      fault_detected ← TRUE
7.          ADD " sensor_data " TO fault_flags
8.          # direct FSM to work in mitigation state
CREATE control logic for FSM
command: "mitigate"
brake: TRUE
torque_limit: 0.3
pitch_correction: 5
timestamp: CURRENT_TIME
PUBLISH control message TO "wind/fsm_command"
END IF
~~~~~~~~~~~~~~~~Recovery State~~~~~~~~~~~~~~~~~~~~
IF (wind/sensor_data − ref_ sensor_data) = tolerance_ sensor_data
THEN
  # direct FSM to work in recovery state
CREATE control logic for FSM
command: "recovery state"
brake: FALSE
torque_limit: 0.6
pitch_correction: 2
timestamp: CURRENT_TIME
PUBLISH control message TO "wind/fsm_command"
END IF
~~~~~~~~~~~~~~~~Resume normal State~~~~~~~~~~~~~~~~~~~~
IF (wind/sensor_data − ref_ sensor_data) < tolerance_ sensor_data
THEN
# direct FSM to Resume normalized operation _ RNM
CREATE control logic for FSM
command: "resume normal operation "
brake: FALSE
torque_limit: 1
pitch_correction: 0
timestamp: CURRENT_TIME
PUBLISH control message TO "wind/fsm_command"
END IF
7.    END LOOP
8.    END
```

Working of Algorithm 1

The POX-SDN controller, under the influence of the proposed intelligent sensing real-time fault management (TCM) algorithm, compares the actual sensor data obtained by the MQTT JSON file with reference wind energy system parameters (obtained from training of the TCM model using mean and standard deviations). If the final result is greater than tolerance_sensor data, the SDN controller detects this output result as a fault state and directs the FSM to operate in mitigation state (by applying emergency braking, limiting torque to 30%, increasing blade pitch to reduce energy capture). If the final result is equal to the tolerance_sensor data, the SDN controller directs the FSM to operate in recovery state (by releasing emergency braking, limiting torque to 60%, and starting to decrease blade pitch to increase energy

capture). If the final result is less than the tolerance, the SDN controller directs the FSM to operate in resume normalized state (by releasing emergency braking, torque to 100%, and setting blade pitch to normal value to get normalized energy capture). The working of proposed Algorithm 1 in the form of a flow diagram is shown in Fig.2.

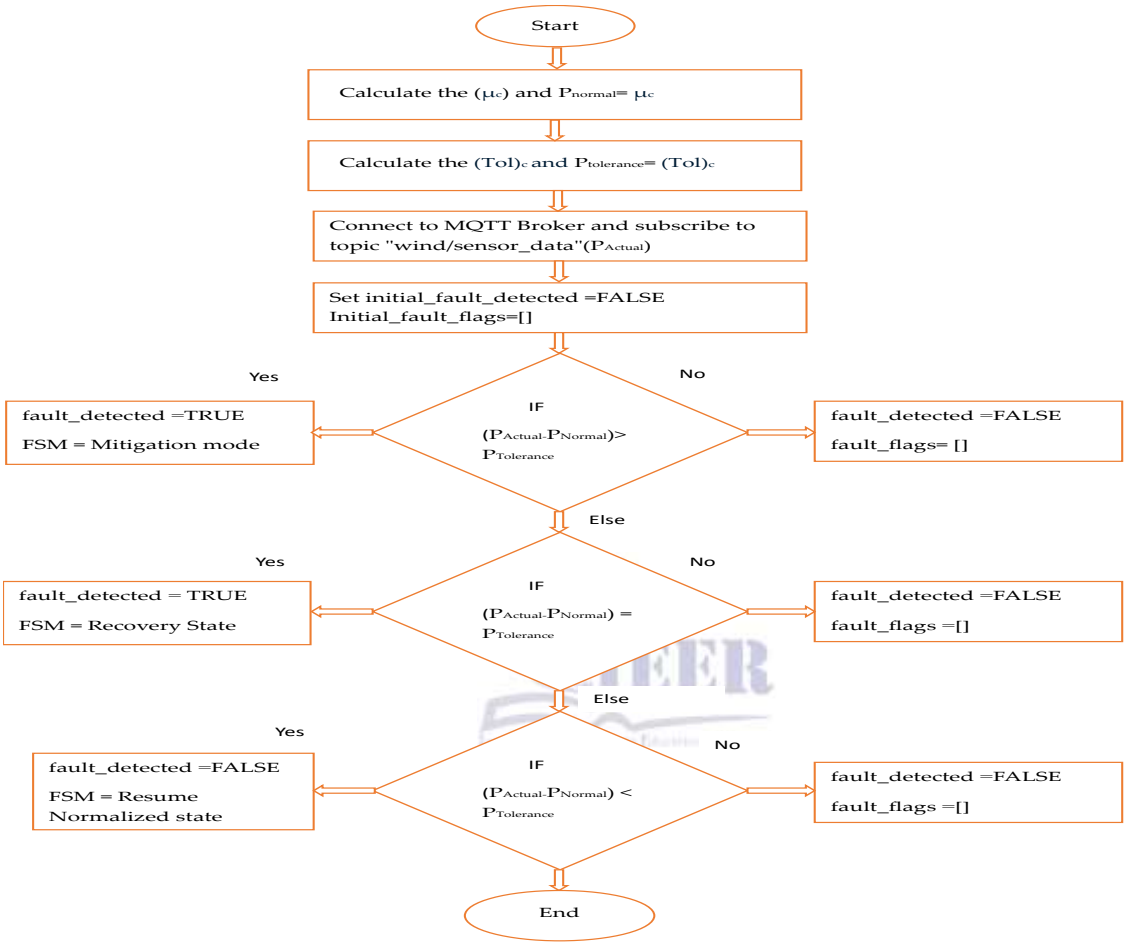

Figure 2: The flow diagram of the proposed framework (SDN-POX controller using TCM logic).

3.2. Simulation Parameters:

The details about the simulation parameters for the proposed framework for real-time fault management using the SDN-POX controller under the proposed algorithm are mentioned in Tab. 2

Table 2. Simulation parameters for the proposed framework for real-time fault management using SDN-POX controller under (TCM)

Parameters	Descriptions	Values
Time_Simulation	total time for which the simulation is performed	20
Time_Sample	time steps (i.e. 0.1,0.2,0.3, ...,19.8,19.9,20)	0.1
R	Rotor radii (m)	40
rho	Air density (kg/m3)	1.225
G_{Ratio}	Gearbox ratios	90
J	Moment of inertia (kg·m²)	10000
Damping_coeff	Shaft damping	50
P_{normal} for Temp	temperature_ ref	65.0
P_{normal} for Vib	vibration_ ref	0.03
P_{normal} for S_{Rotor}	rotor speed_ ref	1500
N_{ormal} for Torque	torque_ ref	400.0
P_{normal} for Pitch	pitch_angle_ ref	12.0
$P_{Tolerance}$ for Temp	tolerance_temperature	5
$P_{Tolerance}$ for Vib	tolerance_vibration	0.01
$P_{Tolerance}$ for S_{Rotor}	tolerance_rotor speed	200
$P_{Tolerance}$ for Torque	tolerance_torque	50.0
$P_{Tolerance}$ for Pitch	tolerance_pitch_angle	3.0

4. Results and Discussions

This section is subdivided into two subsections. In the first sub-section, to check the performance and effectiveness of the proposed framework with the (TCM) algorithm, a scenario is developed to make a real-time intelligent fault-resilient wind energy system, as explained in Algorithm 2. In this scenario, the total simulation time is set to 20 sec, and the time stamp =0.1 sec. The system is allowed to operate in a normalized state till 10 sec (this is done by setting the parameter value and constant used to obtain the sensor's data set to remain under $P_{Tolerance}$ as defined in Table 2). At 11 sec, the fault is induced (this is done by setting the parameter value and constant used to obtain the sensor's data set to increase beyond the level of $P_{Tolerance}$ as defined in Table 2), and the performance is calculated in terms of response delay of the proposed framework. At 15 sec, the recovery mode is initiated (this is done by setting the parameter value and constant used to obtain the sensor's data set to become equal to the level of $P_{Tolerance}$ as defined in Table 2), and the response delay of the proposed framework is calculated. At 17 sec, the fault is cleared and the system is allowed to operate in a resume normalized state (the response delay of the proposed framework is calculated). In the second sub-section, the above-mentioned scenario is again utilized, and a comparative analysis is performed between the proposed framework and conventional fault management methods (SCADA, KNN, GNN).

Algorithm #2: Scenario formation to check the effectiveness of the proposed framework

```
1.  Begin
2.  # Defining simulation time and time stamp
SET total_sim_time ← 20
    SET time_step ← 0.1        % CREATE time vector t from 0 to 20 with step size 0.1
3.  # Design of wind turbine model
FOR each time step t[k]:
GENERATE wind_speed[k] ← base_speed + gust + noise
IF t[k] >= 11 AND t[k] <= 15 THEN
INJECT FAULT (temperature, vibration, rotor_speed,torque)
ELSE
OPERATE NORMALLY
PUBLISH sensor_data[k] via MQTT
4.  #  SDN Controller (POX)
EXTRACT data
COMPARE actual_values with reference_values
IF deviation > threshold
THEN
fault_flags ← TRUE
SEND command to FSM: mitigate
IF values return to normal
THEN
Fault_flag ← FALSE
SEND command to FSM: resume
5.  # FSM (Simulink State flow)
IF "mitigate" THEN
ENTER Mitigation Active
activate brake ()
reduce torque ()
```

```
  change pitch ()
  IF "resume" THEN
ENTER Resume Operation
restore_full_torque ()
reset pitch ()

6.  Repeat until t = 20 sec

7.  END
```

4.1 (Case-A) Measuring the effectiveness of the proposed framework with the scenario developed in Algorithm 2:

In this case, the effectiveness of the proposed algorithm is judged by making the SDN-POX controller and FSM connected via the MQTT protocol to operate in four states (normal mode, mitigation mode, recovery mode, and resume normalized mode). As per the logic defined in Algorithm 2, the wind energy system parametric values are set to the normal range for 10 seconds of the total simulation time (Sim_Time). At 11 seconds, the fault is induced by disturbing the wind energy system's parametric values. At 15 seconds of the simulation time, these values are again forced to come close to normalized values and finally become normalized at 16 seconds. The performance of the SDN-POX controller under the (TCM) algorithm in directing FSM to make the wind energy system fault resilient with minimum response time and processing delay is shown in Tab.3

Table 3. The performance of the proposed framework with the scenario developed in Algorithm 2

Sim_Time (sec)	Temp (C°)	Vibration (g)	S_{Rotor} (RPM)	Torque (Nm)	Pitch Angle (°)	SDN_Action	FSM_Response
0-10	65	0.03	1500	400	12	No fault	Normal mode
11	78	0.043	1750	500	16	sensing	No action directed
11.280	78	0.043	1750	500	16	Fault detected and mitigation triggered	Fault mode
11.310-15	74	0.041	525	150	21	Mitigation triggered	Mitigation activated
15	70	0.04	525	150	21	Sensing	No action directed
15.310-16	70	0.04	1050	240	15	Recovery Triggered	Recovery mode activated
16	65	0.03	1050	240	15	Sensing	No action directed
16.310	65	0.03	1500	400	12	Resume normalized mode triggered	Resume normalized mode activated

It is evident from the table that the SDN-POX controller under the proposed algorithm requires 280ms to detect the fault and 30ms for directing FSM to act upon the control instructions. From (0 to 10 sec), the temperature (65 C°) and shaft vibrations (0.03 g) were under normal range and satisfying condition ($P_{actual} - P_{normal}$) $<P_{tolerance}$, so SDN-POX controller detects no fault and no control instructions are provided to FSM. At 11 sec, the temperature (78 C°) and shaft vibrations (0.043 g) exceed normal range, and the fault is detected by the SDN-POX controller after 11.280 sec with fulfillment of the logical condition ($P_{actual} - P_{normal}$) $> P_{tolerance}$. The mitigation mode is triggered by SDN-POX and is activated after 30ms at 11.310 sec by FSM by following the control instructions of the POX controller (to apply breaking, reducing torque to have only 30% and reducing energy gain by increasing pitch angle by 5 degrees) as per the logic explained in Algorithm (1). At 15 sec, the temperature (70 C°) and shaft vibrations (0.04g) falls to allowed $P_{Tolerance}$ range and the SDN-POX controller after 15.280 sec with fulfillment of logical condition ($P_{actual} - P_{normal}$) = $P_{tolerance}$, triggers the recovery mode and is activated after 30ms at 15.310 sec by FSM by following control instructions of POX controller (to somewhat release breaking, having torque to 60% and reducing pitch angle to 15 degree) as per logic explained in Algorithm (1). At 16 sec, the temperature (65C°) and shaft vibrations (0.03g) falls to allowed P_{normal} range and the SDN-POX controller after 16.280 sec with fulfillment of logical condition ($P_{actual} - P_{normal}$) < $P_{tolerance}$, triggers the resume normalized mode and is activated after 30ms at 16.310 sec by FSM by following control instructions of POX controller (no breaking, having torque to 100% and reducing pitch angle to 12 degree) as per logic explained in Algorithm (1). Figure 3(a-c) is obtained with the help of Gnuplot and represents the variations in the values of torque, rotor speed, and pitch angle for the designed scenario as per Algorithm 2.

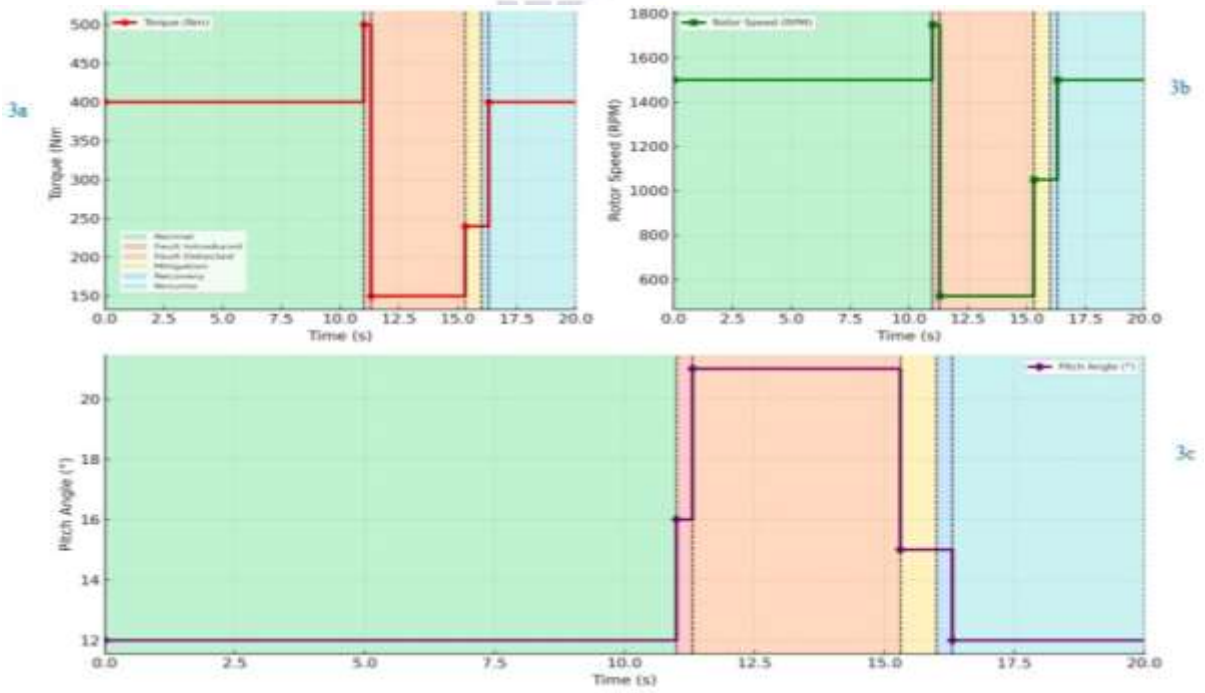

Figure 3: Variation in wind turbine parameters as per the designed algorithm 2. (a) Variation of Torque in Nm. (b) Variation of Rotor speed in rpm. (c) Variation of Pitch angle in degrees.

The proposed framework is simulated for 20 seconds with a time stamp of 0.1 seconds. The values of all metrices as per the defined scenario in Algorithm 2 remain the same for 0-10 sec, and the main transitions take place between 11 and 16 sec time frames. These Figures show the variation in the values of (Torque in Nm, Rotor speed in rpm, and Pitch angle in degree) when FSM is directed to operate in normal mode from (0-10 sec), mitigation mode (11.310sec-15.310sec), recovery mode from (15.310 sec to 16.310 sec), and resumed normalized mode from (16.310 sec to 20 sec). The delay in the FSM response (i.e., how fast FSM responds to SDN instructions) is shown in Figure 4.

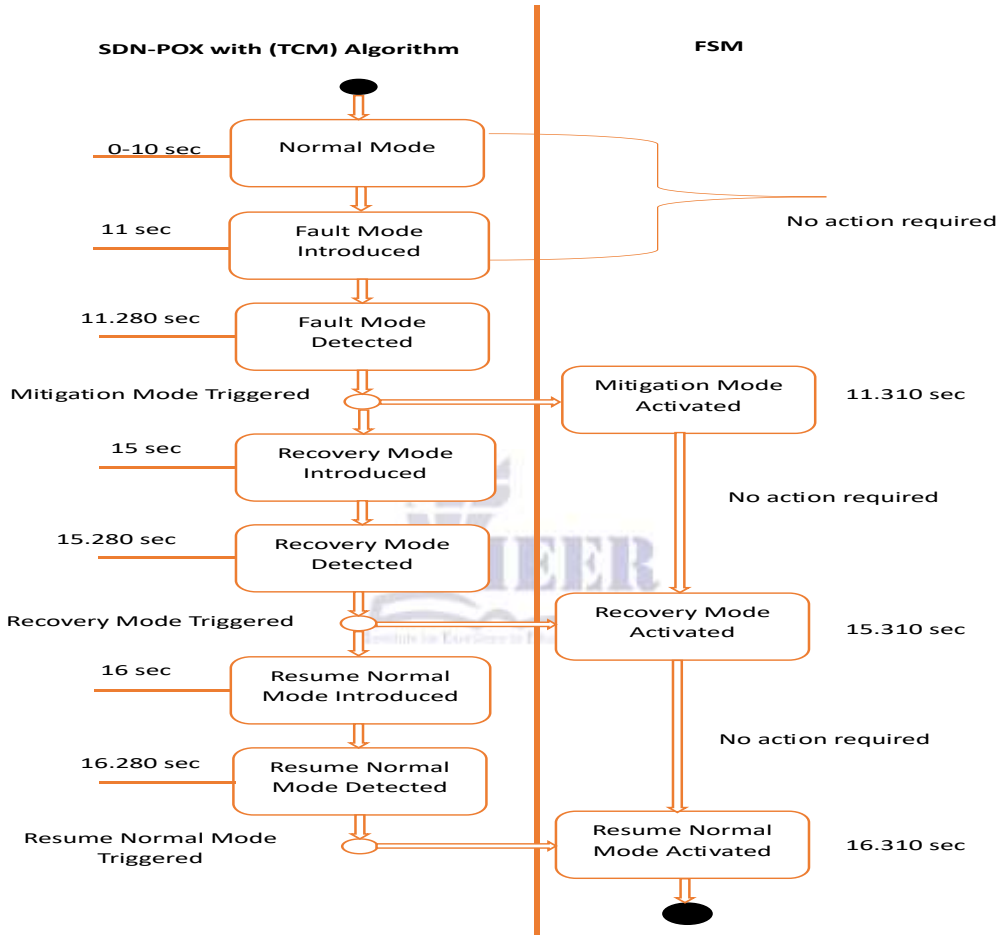

Figure 4. UML activity diagram representing T_{det} and T_{FSM} of both SDN-POX controller and FSM.

4.1.1 Summary of Case-A

From (0 to 10 sec), the SDN-POX controller detects no fault, and no control instructions are provided to the FSM. At 11 sec, the temperature (78 C°) and shaft vibrations (0.043 g) exceed normal range, and the fault is detected by the SDN-POX controller after 11.280 sec. The mitigation mode is triggered by SDN-POX and is activated after 30ms at 11.310 sec by FSM by following the control instructions of the POX controller (to apply braking and reduce the rotor speed from 1750 rpm under faulty conditions to 525rpm by increasing pitch angle from 12 to 21 degrees with a reduction in torque from 500Nm to 150 Nm). During a fault (like overspeed or overvoltage), we don't want the turbine to keep rotating at full speed. To protect components, FSM tells the pitch controller to raise the blade pitch angle in order to safeguard components. As a result, the rotor slows down and lift is decreased as the blades rotate more

against the wind. As a result, working conditions are safer, and the rotor speed and aerodynamic torque are reduced. At 15 sec, the temperature (70 C°) and shaft vibrations (0.04g) falls to allowed $P_{Tolerance}$ range and the SDN-POX controller after 15.280 sec with fulfillment of logical condition ($P_{actual} - P_{normal}$) = $P_{Tolerance}$, triggers the recovery mode and is activated after 30ms at 15.310 sec by FSM by following control instructions of POX controller (to somewhat release breaking, having torque values again increased to 240Nm, incremental rotor speed from 525 rpm to 1050rpm and reducing pitch angle to 116 degree) as per logic explained in Algorithm (1). At 16 sec, the temperature (65C°) and shaft vibrations (0.03g) falls to allowed Pnormal range and the SDN-POX controller after 16.280 sec with fulfillment of logical condition ($P_{actual} - P_{normal}$) < $P_{tolerance}$, triggers the resume normalized mode and is activated after 30ms at 16.310 sec by FSM by following control instructions of POX controller (no breaking, having torque back to its normalized value of 400Nm, rotor speed to 1500 rpm and pitch angle is maintained up to 12 degrees)

4.2 (Case-B)

Comparative analysis of the proposed framework with existing fault management frameworks (SCADA, KNN, GNN) with the designed scenario of Algorithm 5:

In this case, the effectiveness of the proposed (TCM) algorithm is compared with existing fault management frameworks (Supervisory Control and Data Acquisition, SCADA, K-Nearest Neighbors, KNN, Graph Neural Network, GNN) by implementing the logic defined in Algorithm 5. The wind energy system parametric values are set to the normal range for 10 seconds of the total simulation time (Sim_Time). At 11 seconds, the fault is induced by disturbing the wind energy system's parametric values. At 15 seconds of the simulation time, these values are again forced to come close to normalized values and finally become normalized at 16 seconds. The comparative analysis of the proposed framework with existing fault management frameworks (SCADA, KNN, GNN) is shown in Tab.4

Table 4. The comparative analysis of the proposed framework with existing fault management frameworks (SCADA, KNN, GNN) using a scenario developed in Algorithm 2

Parametric Metrices	Proposed Framework SDN-POX Via TCM	SCADA System	KNN System	GNN System
T_{det} (Fault Detection Time in ms)	280ms	1200ms	800ms	650ms
T_{FSM} (delay in the FSM response in ms)	30ms	1000ms	600ms	400ms
Protocol Latency (avg)	280 ms	~1000 ms (SCADA bus)	600 ms	500 ms
Percentage Detection Accuracy (%A)	97.5%	91%	93%	95%
Percentage False Positive Rate (%FPR)	0.9%	3%	2.2%	1.5%
Percentage False Negative Rate (%FNR)	1.2%	4%	5.5%	3.5%

It is evident from Table 4 that the performance of the proposed algorithm, SDN-POX controller using ISFM algorithm, is far superior to traditional fault management methods (Supervisory Control and Data Acquisition, SCADA, K-Nearest Neighbors, KNN, Graph Neural Network, GNN). The comparative analysis of timing metrics (T_{det}, T_{FSM}, Latency) is shown in Fig.5, while the percentage accuracy comparative analysis is shown in Fig.6. Fig.7 shows the trade-off between (%FPR vs %FNR), and comparison of all the parameters is shown in Fig.8.

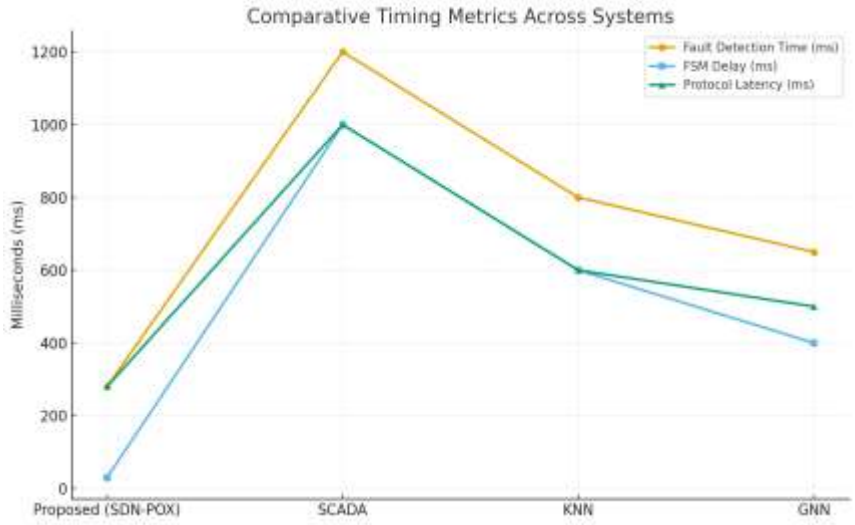

Figure 5: The comparative analysis of timing metrics (T_{det}, $T_{FSM,}$ and Latency)

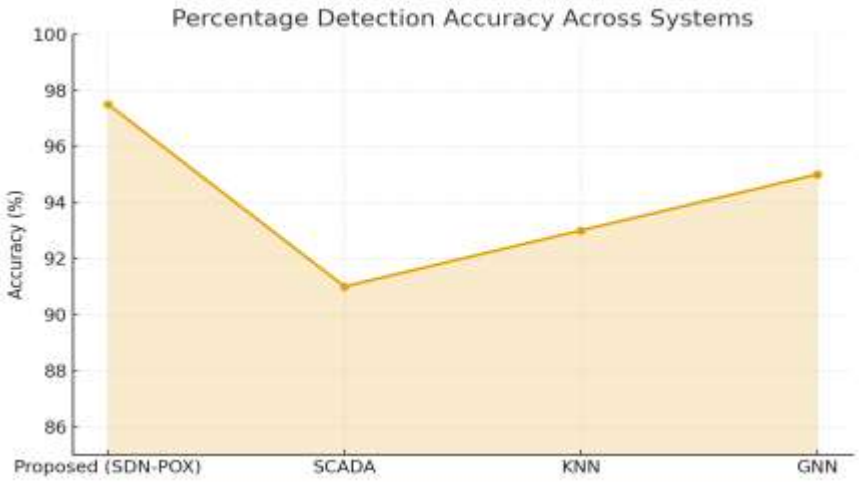

Figure 6: The comparative analysis of percentage accuracy

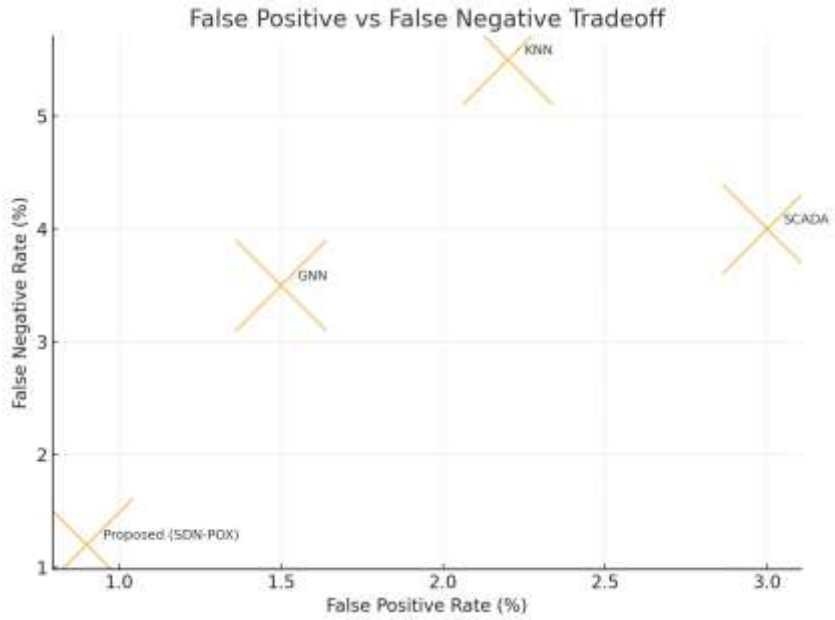

Figure 7: The trade-off between (%FPR vs %FNR)

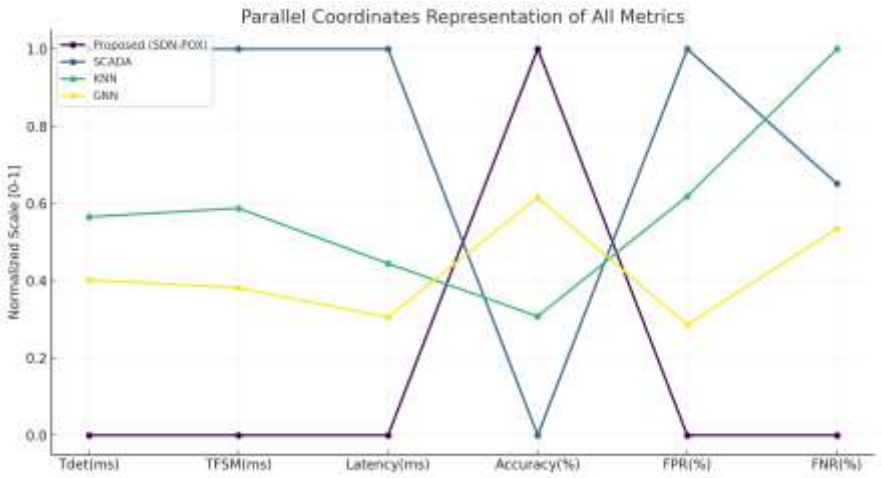

Figure 8: The comparison of all parameters in a normalized scale

Fig.5 shows that the curves connecting the four frameworks (SDN-POX, SCADA, KNN, GNN), and SDN-POX (Proposed) are located at the lowest values (280 ms detection, 30 ms FSM delay, 280 ms latency), indicating a much faster response. The SCADA is at the worst corner (1200 ms detection, 1000 ms FSM delay, ~1000 ms latency), showing that traditional systems are orders of magnitude slower. While the KNN and GNN sit between the two extremes, they still lag behind the proposed framework. Fig. 8 and 9 represent the Accuracy (%) and False Positive Rate (FPR, %) vs False Negative Rate (FNR, %) trade-off, respectively. The SDN-POX (Proposed) sits near the ideal corner: very high accuracy (97.5%) with the lowest FPR (0.9%) and FNR (1.2%). The SCADA performs worst again (91% accuracy, FPR 3%, FNR 4%). The KNN has moderate accuracy but a higher FNR (5.5%). While the GNN is better than SCADA/KNN, it is still not as efficient as the proposed system (as shown in Fig.8). Fig.9 shows the comparison of all six parameters (T_{det}, T_{FSM}, Latency, %A, %FNR, %FPR, %FPR) for four systems (SDN-POX, GNN, KNN, SCADA). The

radar chart has four curves and six axes with all values normalized to the scale (0-1), where 0 is worst and 1 is best optimized. The Radar Plot has optimal zones rings (Outer ring ≈ optimal, Middle rings = moderate, and Middle region = average performance). The proposed SDN-POX polygon is largely in the outer green area, which demonstrates its terms measures.

GNN is close, as it remains close to the outer zones, but it also drops by a small margin in terms of error rates. KNN is biased in terms of good latency but fails on FNR. SCADA is left largely in the inner, poor-performing area.

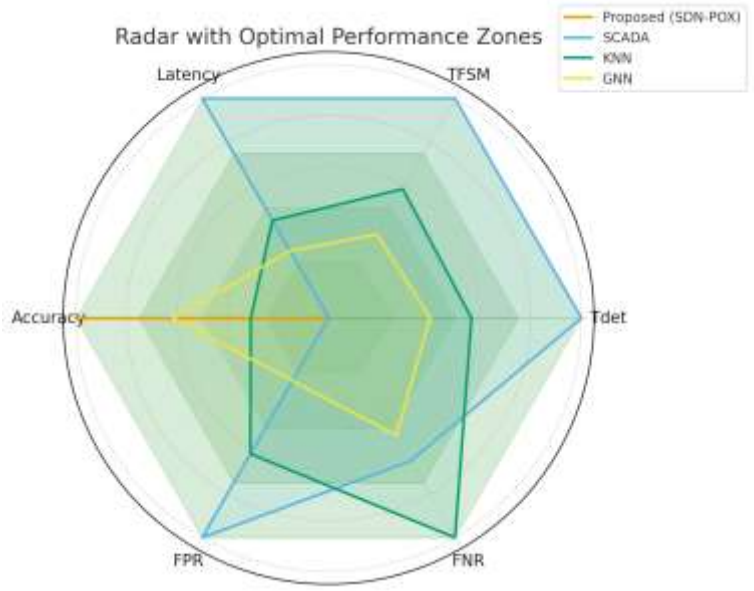

Figure 9. The comparison of all parameters in a normalized scale in a radar chart

The objective function is explained in Eq.A. In this equation, the (J) represents the cost or penalty associated with our system that is to be minimized. The value of metric (J) is calculated by using the data of metrics presented in Tab.4. The comparative analysis of metric (J) of the proposed framework with traditional fault management frameworks is presented in Tab.5.

Table 5. The comparative analysis of the proposed framework with existing fault management frameworks (SCADA, KNN, GNN) for metric (J)

Method	T_{det}	T_{FSM}	%FPR	%FNR	(%A)	J (α= 0.4, β = 0.3, γ = 0.1, δ = 0.1, ϵ = 0.1)
Proposed Framework SDN-POX via TCM	280ms	30ms	0.9	1.2	97.5	98.82
Traditional SCADA System	1200ms	1000ms	3.0	4.0	91.0	252.1
KNN System	800	600	2.2	5.5	93.0	204.7
GNN System	650	400	1.5	3.5	95.0	170.9

In Tab.5, the value of metric (J) representing the cost or penalty associated with our system is minimum in our proposed framework as compared to traditional fault management systems. The comparative graph of metric (J) of the proposed framework with traditional fault management frameworks is presented in Fig.10.

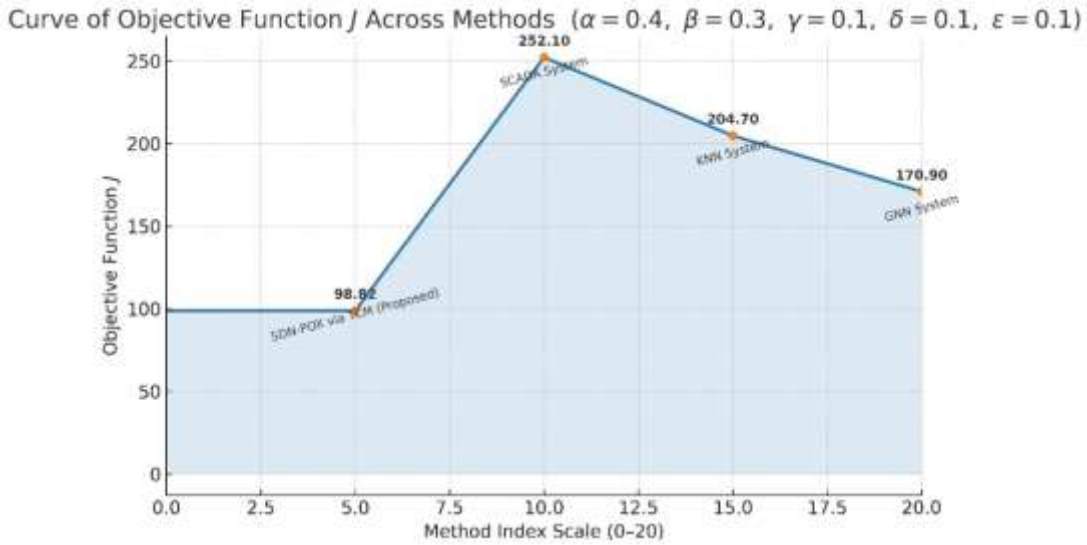

Figure 10: The comparative analysis of metric J

5 Conclusion

In this research, a real-time intelligently controlled fault management framework is designed that involves the Software Defined networking (SDN-POX) controller using a proposed intelligent sensing fault management (TCM) algorithm that enables wind turbines' energy system to produce fast, reliable, highly efficient, and sustainable energy production, with rapid problem identification and mitigation. This is possible with the integration of the SDN-POX controller with Finite State Machine (FSM) using a broker protocol, Message Queuing Telemetry Transport (MQTT). The robust protection and wind energy system stability under variable wind conditions is made possible by the SDN-POX controller using TCM algorithm logic to compare real-time sensor data with reference criteria and initiate exact FSM states, including Mitigation Active, Recovery, and Resume Normal Mode. By using the TCM algorithm, the SDN-POX controller provides rapid and real-time fault management in a designed wind energy system with low detection and mitigation latency (T_{FD} =280 ms and T_{FSM} =30 ms). As these values show, the performance of the proposed framework is far superior to traditional fault management algorithms (i.e., GNN, KNN, and SCADA models). The proposed framework also provides much lower cost value for the metric J (representing the cost or penalty associated with our system that is to be minimized) as defined in the objective function, including detection delay, control delay, false rates, and accuracy as compared to traditional fault management methods (i.e., SCADA, KNN, and GNN).

Acknowledgment: The Researcher would like to acknowledge Electrical Engineering department of Bahauddin Zakariya University, Multan,Pakistan for providing the facility to conduct this research experiment.

Funding Statement: No funding received

Author Contributions: The author's contributions to this paper are as follows: study conception and design: K.T. Mehmood; data collection: K.T. Mehmood; analysis and interpretation of results:

K.T. Mehmood; draft manuscript preparation: K.T. Mehmood.

Availability of Data and Materials: The supporting data related to the algorithm and simulation material technical details are with the corresponding author and can be provided upon appropriate request.

APPENDIX A

A1: Designing a wind energy system on MATLAB

1. BEGIN
Defining Simulation Parameters
Time_Simulation 20; % total time for which the simulation is performed
Time_Sample= 0.1; % time steps (i.e. 0.1,0.2,0.3, ...,19.8,19.9,20)
R=40; % Rotor radii (m)
rho=1.225; % Air density (kg/m^3)
G_{Ratio}=90; % Gearbox ratios
J=10000; % Moment of inertia (kg·m^2)
Damping_coeff =50; % Shaft damping
2. # Define Variable Wind Speed in m/s
t = 0: Time_Sample: Time_Simulation;
$S_W(t) = 10 + 2\sin(0.2\pi t) + 0.5\text{randn}(\text{size}(t))$;
3. # Define Vector array
ω_rotor = zeros(size(t)); % Rotor speed (rad/s)
T_m = zeros(size(t)); % Mechanical torque
3. P_E = zeros(size(t)); % Generated electrical power
Loop for dynamic simulation
for k = 2: length(t)

4. # Mechanical Power & Torque calculation
λ = (ω(k-1)·R)/S_W(k) % where ω is the angular velocity of the rotor and R is the rotor radius
4. $C_p = (0.22).\left(\frac{116}{\lambda - 0.4(\beta - 5)}\right).e^{\frac{-12.5}{\lambda}}$ %(β = 0 here for fixed-pitch blades)
P_m=0.5.rho. π. R^2. Cp. Sw^3(k)
$T_m(k) = \frac{P_m}{\omega(k-1)+0.1}$

5. #Shaft Dynamics based on Damping Coefficient
5. (dω/dt = (Tm - Damping_coeff . ω)/J);
ω_rotor(k) = ω_rotor(k-1) + Time_Sample * dω/dt;

6. # Electric Power Generation
6. ω_(scaled_up) = ω_rotor (k) +G_Ratio;
P_E= ω_(scaled_up) . Tm(k)

7. 7. END

Working of Algorithm:

The first command of the algorithm involves defining basic simulation parameters, while the second command explains the definition of the sine function to have variable wind flow (Where Sw is the wind speed, the factor (2sin(0.2πt)) shows

the periodic fluctuation in wind speed, with amplitude = 2 m/s, providing the wind speed can vary ±2 m/s from the mean, and the factor 0.5randn(size(t)) adds Gaussian noise of 0.5 scales. The third statement of the algorithm involves defining vector arrays to store rotor speed (ω), mechanical torque (Tm), and generator power (PE) at each time step, as it improves speed and memory efficiency, and also runs the simulation step-by-step from time t=0 to t=20s.The fourth statement of the algorithm explains
the calculation of mechanical torque using aerodynamic equations based on speed ratio (λ) and power coefficient (Cp). The fifth statement of the algorithm involves rotor speed calculations that change due to torque and damping, and integrate angular acceleration over time to update rotor speed. The last statement explains how electrical energy is calculated with the updated rotor speed and with a defined gear ratio.

APPENDIX B

B1: Integration of sensors with a wind energy system designed on MATLAB.

```
1. Begin
2. #Defining sensor arrays
   sensor _vibration= zeros(size(t));
   sensor _temperature= zeros(size(t));
   sensor _rotor_speed= zeros(size(t));
   sensor _pitch_angle= zeros(size(t));
   sensor _torque= zeros(size(t));
3. # Defining simulation constants (i.e., thermal and pitch angle)
Temperature= 65;          % °C
Gain_temp= 0.005;       % °C per watt
pitch_angle = 12;       % Degrees (fixed in this model)
4. for k = 1: length(t)
% sensor _vibration (simulated as RMS of noise + shaft frequency)
sensor _vibration (k) = rms (0.02 * randn (1,100)) + 0.001 * ω _rotor(k);
% Sensor_Temperature (proportional to power output)
sensor _temperature(k) = Temperature + Gain_temp * P_E(k);
4. % Sensor_Rotor Speed
Sensor_rotor_speed(k) = ω _rotor(k);
% sensor_Pitch Angle Sensor (fixed in proposed framework)
sensor_pitch_angle(k) = pitch_angle;
% sensor_Torque
Sensor_torque(k) = Tm(k)
5. # Generation of data sets from sensors for MQTT Logs
MQTT_data_set = strings(length(t), 1)
for k = 1: length(t)
% Construct data structure
data. Timestamp= datestr (now + seconds(t(k)), 'yyyy-mm-ddTHH:MM:SS.FFF: SS.FFF');
5.      data.vibration= sensor _vibration (k);
        data.temperature = sensor _temperature(k);
        data.rotor_speed= Sensor_rotor_speed(k);
        data.pitch_angle= sensor_pitch_angle(k);
        data.torque = Sensor_torque(k);
6. # Convert to JSON-like string (for MQTT payload)
```

```
MQTT_data_log(k) = jsonencode(data);
7.  end
```

Working of Algorithm :

The first three commands of the algorithm involve defining basic simulation parameters, while the fourth statement of the algorithm explains the calculation of vibration, temperature, rotor speed, pitch angle, and torque data using vibration, temperature, rotor speed, and torque sensors. The fifth statement of the algorithm involves arranging these data in the form of logs, which is continuously updated with a time stamp function and then converted into the MQTT protocol format (JSON file).

APPENDIX C

C1: Working and Designing of Finite State Machine (FSM)

```
1.  Begin
2.  # Defining initial state
    state = "Normal Mode_NM";
3.  # Control Parameters as an input from SDN
fault detection_FD = false;
recovery mode _RM = false;
normalized stage (RNM) = false;
4.  # FSM logic for outputs
Active_ brake = false;
limit_torque = 0;
pitch_correction = 0;
5.  # Defining Simulation Loop
for k = 1: length(t)
6.  # Simulated fault trigger at t = 11s
 if t(k) > 11 && t(k) < 20
 fault_detected = true;
      else
  fault_detected = false;
      end

7.  #Simulated SDN recovery command at t = 15s
if t(k) > 15
recovery_signal = true;
else
recovery_signal = false;
 end
8.  # FSM logic in response to 5 states
9.  # Normal operation state_NM
case "Normal_Operation"
brake_active = false;
torque_limit = 1.0;     % Full torque
pitch_correction = 0;
if fault_detected
state = "Fault_detected";
```

```
end
10.    # Fault Detection_FD
case "Fault_detected"
state = "Mitigation_Active";
11.  # Mitigation state _ MM
case "Mitigation Active"
brake_active = true;
torque_limit = 0.3;      % Reduce torque
pitch_correction = 5; % Increase pitch to reduce load
if recovery_signal
state = "Recovery";
end
12.    # Recovery State _ RM
case "Recovery"
brake_active = false;
torque_limit = 0.6;
pitch_correction = 2;
  if ~fault_detected
state = "Resume_Operation";
end
13.  # Resume normalized operation _ RNM
case "Resume Operation"
brake_active = false;
torque_limit = 1.0;
pitch_correction = 0;
state = "Normal_Operation";
end

14.  End
```

Working of Algorithm:

The FSM starts in the Normal Operation state. In normal operation, the state system runs normally with no braking implemented, there is full torque (no limit), and no pitch correction is required. When a fault is detected by the SDN-POX controller, the FSM immediately enters mitigation mode. In the mitigation mode, the wind energy system is upheld by applying emergency braking, limiting torque to 30%, increasing blade pitch to reduce energy capture, and FSM waits for the SDN recovery signal. If recovery orders are received, then the wind energy system goes to recovery mode. In recovery mode, the system is relaxed somewhat with braking released, and torque is partially restored (60%) with pitch slowly reduced. Now in recovery mode, the FSM waits for the fault to disappear and orders from the SDN-POX controller, and finally enters again into normalized mode (RNM) with full torque (no limit), and no pitch correction is required anymore.

References:

Alagha N, Khairuddin ASM, Haitaamar ZN, Al-Khatib O, Kanesan J. Artificial Intelligence in Wind Turbine Fault Detection and Diagnosis: Advances and Perspectives. Energies. 2025; 18(7):1680. https://doi.org/10.3390/en18071680.

International Energy Agency (IEA). Renewables 2024 – Global Status and Outlook; IEA: Paris, France, 2024. (Accessed online: IEA Renewables 2024 report). https://www.iea.org/reports/renewables-2024.

Sasinthiran A, Gnanasekaran S, Ragala R. A review of artificial intelligence applications in wind turbine health monitoring. International Journal of Sustainable Energy. 2024; 43(1). https://doi.org/10.1080/14786451.2024.2326296

Global Wind Energy Council (GWEC). Global Wind Report (2024). GWEC: Brussels, Belgium, 2024. (Highlights of global wind installations and targets). https://www.gwec.net/reports.

Krivohlava Z, Chren S, Rossi B. Failure and fault classification for smart grids. Energy Inform 5, 33 (2022). https://doi.org/10.1186/s42162-022-00218-3.

Moshtaghi P, Hajialigol N, Rafiee B. A comprehensive review of artificial intelligence applications in wind energy power generation. Sustainable Futures 2025; 9(100638). https://doi.org/10.1016/j.sftr.2025.100638.

Liu H, Wang YZ, Zeng T, Wang HF, Chan SC, Ran L. Wind turbine generator failure analysis and fault diagnosis: A review. IET Renew. Power Gener. (2022); 16(15), 3051–3068. DOI: 10.1049/rpg2.13104

Yang T, Teng J, Li C, Feng Y..Wind turbine fault detection and diagnosis using an LSTM neural network. 2020 39th Chinese Control Conference (CCC), Shenyang, China, (2020). pp. 4042-(4047). Doi: 10.23919/CCC50068.2020.9188709.

Benbouzid M, Berghout T, Sarma N, Djurović S, Wu Y, Ma X. Intelligent Condition Monitoring of Wind Power Systems: State of the Art Review. Energies. 2021;14(18):5967. https://doi.org/10.3390/en14185967

Ren Z, Verma AS, Li Y,Teuwen JJ, Jiang Z. Offshore wind turbine operations and maintenance: A state-of-the-art review. Renew. Sustain. Energy Rev. (2021); 144(110886). https://doi.org/10.1016/j.rser.2021.110886

Cao X, Xu Y, Wu Z, Qin X, Ye F. Data acquisition and management of wind farm using edge computing. Int. J. Grid Util. Comput. (2022); 13(2-3), 249–255, https://doi.org/10.1504/IJGUC.2022.124399.

Mwangi A, Fumagalli E, Gryning M, Gibescu M. Building Resilience for SDN-enabled IoT Networks in offshore renewable energy supply. 2023 IEEE 9th World Forum on Internet of Things (WF-IoT). DOI: 10.1109/WF-IOT58464.2023.10539483

Li Q, Deng Y, Liu X, Sun W, Li W, Li J, Liu Z. (2023). Autonomous smart grid fault detection. IEEE Communications Standards Magazine, 7(2), 40–47. https://doi.org/10.1109/MCOMSTD.0001.2200019

Fallah F, Ramezani A, Mehrizi-Sani A.Integrated fault diagnosis and control design for DER inverters using machine learning methods.Systems and Control (eess.SY). .2022. https://arxiv.org/abs/2202.09996.

Chanda D, Soltani NY. A Heterogeneous Graph-Based Multi-Task Learning for Fault Event Diagnosis in Smart Grid. IEEE Transactions on Power Systems.2025;40(2),1427-1438,doi: 10.1109/TPWRS.2024.3447533

Famitafreshi G, Afaqui MS, Melià-Seguí J. A Comprehensive Review on Energy Harvesting Integration in IoT Systems from MAC Layer Perspective: Challenges and Opportunities. Sensors. 2021; 21(9):3097. https://doi.org/10.3390/s21093097

Ng EY-K, Lim JT. Machine Learning on Fault Diagnosis in Wind Turbines. Fluids. 2022; 7(12):371. https://doi.org/10.3390/fluids7120371

Yamansavascilar B, Baktir AC, Ozgovde A, Ersoy C . Fault tolerance in SDN data plane considering network and application-based metrics. Journal of Network and Computer Applications. 2020;170(10). Doi:10.1016/j.jnca.2020.102780.

Upta P, Indhra-Om-Prabha M. (2021). A survey of application layer protocols for the Internet of Things. In the 2021 International Conference on Communication, Information and Computing Technology (ICCICT) (pp. 1–6). IEEE. https://doi.org/10.1109/ICCICT50803.2021.9510140.

Shahri E, Pedreiras P, Almeida L. A Scalable Real-Time SDN-Based MQTT Framework for Industrial Applications. IEEE Open Journal of the Industrial Electronics Society 2024. vol. 5, pp. 215-235. doi: 10.1109/OJIES.2024.3373232.

Kharchenko V, Tyurin S, Fesenko H, Goncharovskij O. (2021). The fault-tolerant Černý finite state machine: A concept and VHDL models. In 2021, the 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) (pp. 1163–1169). IEEE. https://doi.org/10.1109/IDAACS53288.2021.9660925

Jin X, Xu Z, Qiao W.Condition Monitoring of Wind Turbine Generators Using SCADA Data Analysis. IEEE Transactions on Sustainable Energy. 2021; 12(1), pp.202-210.doi: 10.1109/TSTE.2020.2989220.

Igoni L, Betti A, Tucci M, Crisostomi E. (2019). A scalable predictive maintenance model for detecting wind turbine component failures based on SCADA data. In 2019, IEEE Power & Energy Society General Meeting (PESGM) (pp. 1–5). IEEE. https://doi.org/10.1109/PESGM40551.2019.8973898.

Orozco R, Sheng S, Phillips C. (2018). Diagnostic models for wind turbine gearbox components using SCADA time series data. In 2018, the IEEE International Conference on Prognostics and Health Management (ICPHM) (pp. 1–9). IEEE https://doi.org/10.1109/ICPHM.2018.8448545.

Maldonado-Correa J, Martín-Martínez S, Artigao E, Gómez-Lázaro E. Using SCADA Data for Wind Turbine Condition Monitoring: A Systematic Literature Review. Energies. 2020; 13(12):3132. https://doi.org/10.3390/en13123132

Naimi JD, Shimjith SR, Arul AJ.Fault Detection and Isolation of a Pressurized Water Reactor Based on Neural Network and K-Nearest Neighbor.IEEE Access 2022;10, pp.17113-17121. doi: 10.1109/ACCESS.2022.3149772.

Crozier K, Baker Y, Du J, Mohammadi J, Li M. (2022). Data-driven contingency selection for fast security-constrained optimal power flow. In 2022, the 17th International Conference on Probabilistic Methods Applied to Power Systems (PMAPS) (pp.1–6).IEEE. https://doi.org/10.1109/PMAPS53380.2022.9810574

Sasikala G, Chandra YPS, Siva N, Vinesh AS.2021.Wind Turbine Fault Monitoring System Using MQTT.Journal of Physics: Conference Series, Volume (2040). International Conference on Physics and Energy 2021 (ICPAE 2021) 24 March (2021). Kancheepuram, India, DOI 10.1088/1742-6596/2040/1/012002

Vidal Y, Pozo F, Tutivén C. Wind Turbine Multi-Fault Detection and Classification Based on SCADA Data. Energies. 2018; 11(3018). https://doi.org/10.3390/en11113018

Sudhakar P, Kamble NK, Geetha K, Turukmane AV, Perli SB, Jayaraman P.Faulty diagnostics model for wind power plant application using AI. Measurement: Sensors.. 2023; 25(100621). https://doi.org/10.1016/j.measen.2022.100621.

Lee CY, Macrene EDC, Wind energy system fault classification and detection using deep convolutional neural network and particle swarm optimization-extreme gradient boosting.IET Energy system integration 2024;6(4),479-497. DOI: 10.1049/esi2.12144.

Nithya M, Nagarajan S, Navaseelan P. (2017). Fault detection of wind turbine system using neural networks. In 2017, IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR) (pp. 103–108). IEEE. https://doi.org/10.1109/TIAR.2017.8273694

Zhang D, Qian L, Mao B, Huang C, Huang B, and Si Y. A Data-Driven Design for Fault Detection of Wind Turbines Using Random Forests and XGBoost. IEEE Access. 2018, vol. 6, pp. 21020-21031, 2018, https://doi.org/10.1109/ACCESS.2018.2818678.

Jiang G, Xie P, He H, and Yan J. Wind Turbine Fault Detection Using a Denoising Autoencoder with Temporal Information.in IEEE/ASME Transactions on Mechatronics, vol. 23, no. 1, pp. 89-100, Feb. 2018, Doi: 10.1109/TMECH.2017.2759301.

Bouzekri A, Tayeb A, Mouloud D, Youcef M. Artificial Intelligence-Based Fault Tolerant Control Strategy in Wind Turbine Systems. International Journal of Renewable Energy Research-IJRER, Vol. 7, No. 2 (2017). https://www.ijrer.org/ijrer/index.php/ijrer/article/view/5555.

Sousa PHF, Nascimento NMM, Almeida JS, Rebouças-Filho PP, and Albuquerque VHC. Intelligent Incipient Fault Detection in Wind Turbines based on Industrial IoT Environment. Journal of Artificial Intelligence and Systems 2019, 1(1), 1–19. DOI: 10.33969/AIS.2019.11001.

Santolamazza A, Dadi D, Introna V. A Data-Mining Approach for Wind Turbine Fault Detection Based on SCADA Data Analysis Using Artificial Neural Networks. Energies 2021, 14, 1845. https://doi.org/10.3390/en14071845.

Zemali Z, Cherroun, NH, Hafaifa A, Iratni A, Obaid S, Alshammari, IC.Robust intelligent fault diagnosis strategy using Kalman observers and neuro-fuzzy systems for a wind turbine benchmark, Renewable Energy,2023, Volume 205,2023, Pages 873-898, ISSN 0960-1481, https://doi.org/10.1016/j.renene.2023.01.095.

Han T, Xie W, Pei Z. Semi-supervised adversarial discriminative learning approach for intelligent fault diagnosis of wind turbine. Information Sciences 2023, Volume 648,2023,119496, ISSN 0020-0255, https://doi.org/10.1016/j.ins.2023.119496.

Yu Y et al. Fault Management in Software-Defined Networking: A Survey. IEEE Communications Surveys & Tutorials 2019. vol. 21, no. 1, pp. 349-392, First quarter 2019, doi: 10.1109/COMST.2018.2868922.

Kavaz AG, Barutcu B. Fault Detection of Wind Turbine Sensors Using Artificial Neural Networks, Journal of Sensors, online-WILEY Library,19 December 2018. https://doi.org/10.1155/2018/5628429.

Morshed MJ and Fekih A. A Fault-Tolerant Control Paradigm for Microgrid-Connected Wind Energy Systems. IEEE Systems Journal,2018, vol. 12, no. 1, pp. 360-372, March 2018, doi: 10.1109/JSYST.2016.2531718.

Mehmood KT, Hussain MM. Dynamic Load Management in Modern Grid Systems Using an Intelligent SDN-Based Framework. Energies 2025, 18, 3001. https://doi.org/10.3390/en18123001.