IJEASM
International Journal of Engineering Applied Science and Management
ISSN (Online): 2582-6948
Vol. 7 Issue 8, August 2026

Integration of Real Time IoT Sensor Networks and Advanced Wind Speed Forecasting Models within an n8n Based Automation Framework for Predictive Maintenance of Wind Turbines and Enhanced Renewable Energy Grid Stability

Er. Rishabh Aryan
M.Tech (Artificial Intelligence and Data Science)
Department of Computer Science and Engineering
Indian Institute of Information Technology, Bhagalpur (Bihar)
Email: rishabh.250201011@iiitbh.ac.in, rishabharyan887@gmail.com

***Abstract*—**Wind energy has become one of the fastest growing sources of clean electricity, yet the unpredictable nature of wind and the mechanical complexity of turbines continue to expose operators to unplanned downtime, expensive reactive repairs and grid instability caused by sudden generation shortfalls. This paper presents an integrated framework, built around the open, node based workflow automation platform n8n as its central orchestration engine, that combines a distributed Internet of Things (IoT) sensor network and a set of advanced wind speed forecasting models with n8n to deliver predictive maintenance and grid stability for wind turbines. Vibration, temperature, strain, acoustic and anemometric sensors are deployed across turbine subsystems and streamed to an edge gateway, from where five cooperating n8n workflows, comprising thirty eight nodes in total, orchestrate data cleaning, feature extraction, AI powered model inference, anomaly detection, a human approval gate, maintenance ticketing, audit logging and grid advisory signalling, without requiring the operator to write large volumes of custom backend code. Four forecasting approaches, namely a persistence baseline, an Autoregressive Integrated Moving Average (ARIMA) model, a Long Short Term Memory (LSTM) network and a hybrid Convolutional Neural Network LSTM (CNN LSTM) model, are trained, compared and, for the best performer, called directly from within the n8n pipeline. The hybrid model achieves the lowest error, with a Root Mean Square Error (RMSE) of 0.61 m/s and a Mean Absolute Error (MAE) of 0.47 m/s, outperforming the persistence baseline by more than 68 percent. Embedded inside the n8n workflow, the forecasting output triggers pre emptive curtailment and maintenance actions that, in simulation, reduce annual unplanned downtime from 186 hours under a purely reactive strategy to 47 hours under the proposed predictive strategy, and reduce grid frequency deviation amplitude by roughly two thirds, while n8n's own execution logs provide a complete, auditable record of every automated and human approved decision. The results indicate that combining commodity IoT hardware, modern time series forecasting and n8n as an accessible, self hostable automation backbone can deliver measurable improvements in turbine reliability, maintenance cost and renewable grid stability, while remaining practical for small and medium wind farm operators who may not have dedicated software engineering teams.**

***Keywords*—**IoT, Wind Turbine, Predictive Maintenance, Wind Speed Forecasting, LSTM, n8n, Automation, Grid Stability, SCADA, Renewable Energy

I. INTRODUCTION

Wind power now supplies a substantial and growing share of the electricity generated worldwide, and national grid operators increasingly depend on it to meet decarbonisation targets. However, two structural characteristics of wind energy make it difficult to operate with the same predictability as conventional thermal generation. The first is the stochastic nature of

wind itself, which varies over minutes, hours and seasons in ways that are only partially captured by classical meteorological models. The second is the mechanical and electrical complexity of the turbine, whose gearbox, bearings, blades, yaw system and power electronics are all subject to fatigue, thermal stress and sudden failure, often in remote or offshore locations where inspection and repair are slow and costly. Historically, wind farm operators have relied on two maintenance philosophies. Reactive maintenance repairs components only after they fail, which is cheap in the short term but can result in extended downtime, secondary damage and lost generation revenue. Scheduled, or time based, maintenance replaces or inspects components at fixed intervals regardless of their actual condition, which reduces catastrophic failures but frequently replaces parts that still have useful life remaining, increasing operating cost. Predictive maintenance, the subject of this paper, instead uses continuously collected condition data together with statistical and machine learning models to estimate the current health of a component and the remaining time before intervention is required, allowing maintenance to be scheduled exactly when it is needed and not before. At the same time, the growing penetration of wind energy on modern grids means that short term forecasting errors no longer only affect the wind farm operator, they also propagate into frequency and voltage deviations that grid operators must absorb through reserves, storage or curtailment of other generators. Accurate short term and medium term wind speed forecasting is therefore not simply an operational convenience for the wind farm, it is an input that grid balancing authorities can use to schedule reserves more efficiently and to reduce the cost of integrating variable renewable generation. A parallel development in industrial automation is the emergence of low code and no code workflow automation platforms, among which n8n has become the specific platform of choice for this study and the organising principle around which the rest of the paper is structured. n8n allows engineers to connect sensors, databases, machine learning services, messaging systems and enterprise software using a visual, node based interface rather than hand written integration code, and, unlike many closed commercial IPaaS offerings, it can be self hosted under a fair code license, giving the operator full ownership of both the automation logic and the data that flows through it. While such platforms are widely used in marketing operations, IT automation and business process automation, their application to industrial condition monitoring and predictive maintenance for renewable energy assets remains comparatively underexplored in the published literature, despite the fact that many small and medium wind farm operators lack the dedicated software engineering resources required to build and maintain bespoke integration pipelines. n8n is not treated in this paper as an interchangeable, generic middleware choice standing in for any workflow engine; rather, the specific execution model that n8n provides, namely a directed graph of typed nodes that can mix scheduled triggers, webhook triggers, conditional branching, loops, human in the loop approval steps and a dedicated error handling workflow within a single visual canvas, is used directly to structure the system architecture described in Section IV and the five workflow implementation described in full at the node level in Section VII. A theoretical background on the workflow automation paradigm that n8n implements, including its relationship to classical flow based programming and its position relative to comparable tools such as Node RED, is provided in Appendix A for readers less familiar with the platform.

This paper proposes and evaluates a framework that brings these three threads together: a distributed IoT sensor network instrumenting the turbine and its immediate environment, a set of forecasting models of increasing sophistication culminating in a hybrid CNN LSTM architecture, and an n8n based automation layer that orchestrates the flow of data between sensors, forecasting services, anomaly detection logic, maintenance ticketing systems and grid advisory dashboards. The central hypothesis of the work is that such a framework can simultaneously reduce unplanned downtime and maintenance cost at the turbine level while improving the quality of the signals available to grid operators for balancing purposes, and that it can do so using tools that are accessible to operators without large software teams. The motivation for this work is reinforced by the broader trajectory of the energy sector, in which variable renewable sources are expected to account for an increasing share of installed generation capacity over the coming decades. Under this trajectory, individual wind farms are no longer isolated assets whose downtime is simply a private cost to the operator, they are increasingly load bearing elements of the grid whose availability and output predictability directly influence system wide reliability. This dual role, as a private asset requiring cost effective maintenance and as a public infrastructure component requiring predictable output, is the underlying reason this paper treats turbine level predictive maintenance and grid level forecasting as two faces of the same problem rather than as separate research questions. It is also worth noting explicitly what this paper does not attempt to do. It does not propose a new forecasting algorithm in the sense of a novel neural network layer or training procedure, nor does it propose a new vibration based fault diagnosis technique. Both forecasting and diagnosis in this paper rely on established, previously published techniques, selected and combined for their practical suitability rather than for algorithmic novelty. The contribution of the paper lies instead in the systems level integration of these established techniques with a concrete, node level automation design, and in the quantification of the combined operational and grid stability effect of that integration, an aspect that, as Section II will show, is comparatively underrepresented in the existing literature relative to the very large body of work on forecasting and diagnosis algorithms considered individually. The remainder of the paper is organised as follows. Section II reviews related work in wind turbine condition monitoring, wind speed forecasting and workflow automation. Section III states the problem and the objectives of

the study. Section IV presents the overall system architecture. Section V describes the IoT sensor network design. Section VI details the forecasting models that were implemented and compared. Section VII describes the n8n based automation framework and its constituent workflows. Section VIII presents the predictive maintenance decision logic. Section IX describes the data used and its preprocessing. Section X describes the experimental setup. Section XI presents results and discussion. Section XII analyses the effect of the framework on grid stability. Section XIII presents a multi turbine case study. Section XIV presents an economic and environmental impact analysis. Section XV discusses security, data governance and reliability considerations. Section XVI discusses limitations, Section XVII outlines future work and Section XVIII concludes the paper.

II. LITERATURE REVIEW

Condition monitoring of wind turbines has been studied extensively over the past two decades. Early work relied primarily on vibration analysis of the gearbox and main bearing using accelerometers and spectral analysis techniques such as the Fast Fourier Transform and wavelet decomposition to detect characteristic fault frequencies [1], [2]. Supervisory Control and Data Acquisition (SCADA) systems, which most commercial turbines already carry for control purposes, have also been mined for condition indicators, since parameters such as gearbox oil temperature, generator winding temperature and power curve deviation can reveal degrading components well before a hard failure occurs [3], [4]. More recent studies have applied machine learning to these condition monitoring signals. Random forests, gradient boosted trees and support vector machines have been used to classify turbine health states from SCADA and vibration features [5], and deep learning architectures, particularly Long Short Term Memory networks and convolutional neural networks, have been shown to outperform classical statistical methods on remaining useful life estimation tasks because they can automatically learn temporal dependencies from raw or lightly processed sensor streams [6], [7]. Hybrid convolutional recurrent architectures, in which convolutional layers extract local features before a recurrent layer models temporal evolution, have become a common design pattern for time series prognostics in the wind energy and broader rotating machinery literature [8]. Wind speed forecasting is a mature subfield in its own right. Physical Numerical Weather Prediction (NWP) models offer good performance at longer horizons of several hours to days but are computationally expensive and coarse in spatial resolution, making them less suitable for turbine level, minute to hour ahead forecasting [9]. Statistical time series methods, particularly ARIMA and its seasonal variants, remain popular baselines because of their simplicity and interpretability, although they struggle with the strong nonlinearity and non stationarity of wind speed series [10]. Machine learning approaches, including Support Vector Regression, Artificial Neural Networks and more recently LSTM and hybrid deep architectures, have consistently been reported to reduce forecasting error relative to ARIMA baselines at short horizons of a few minutes to a few hours [11], [12]. On the automation side, workflow orchestration platforms have a long history in enterprise IT, beginning with Business Process Management (BPM) suites and more recently including IPaaS (Integration Platform as a Service) tools. Open source, node based automation tools such as n8n have gained attention because they allow non specialist engineers to visually connect APIs, databases and messaging services, and because they can be self hosted, which is attractive for organisations with data residency or cost constraints [13]. Industrial Internet of Things (IIoT) literature has explored similar orchestration needs using platforms such as Node RED and commercial IIoT suites, generally focused on manufacturing rather than renewable energy assets [14]. Despite this substantial body of work in the individual subfields, comparatively few studies integrate condition based sensing, forecasting and workflow automation into a single, end to end predictive maintenance pipeline that is explicitly designed to be deployable by operators without dedicated software engineering teams, and fewer still connect the output of such a pipeline to grid level stability considerations rather than turbine level maintenance alone [15]. The present study aims to address this gap by proposing a concrete architecture, implementing representative forecasting models, designing an executable n8n workflow and quantifying the combined effect on downtime, maintenance cost and grid frequency stability. A related strand of research examines cyber physical system design for renewable energy assets more broadly, emphasising the co design of sensing, communication and control layers rather than treating them as independent research problems [16]. This literature generally agrees that the choice of middleware and orchestration layer, whether a commercial IIoT platform, an open protocol broker such as MQTT combined with custom scripts, or a visual workflow engine, has a first order effect on the maintainability of the resulting system over its operating life, a consideration that is often absent from papers that focus purely on the accuracy of a forecasting or diagnostic model in a laboratory or offline setting [17].

Economic evaluations of predictive maintenance adoption in the wind sector have generally relied on top down cost benefit models rather than bottom up simulation tied to a specific sensing and automation architecture. Several studies estimate that predictive maintenance can reduce operations and maintenance cost by a range of roughly twenty to forty percent relative to purely reactive strategies, depending on turbine size, site accessibility and the maturity of the condition monitoring programme

[18], [19]. These estimates are broadly consistent with, although somewhat more conservative than, the simulated cost reduction obtained in the present study, a comparison that is revisited in Section XIV.

Finally, a small number of recent papers have begun to explore the use of low code and no code automation platforms specifically for scientific and engineering data pipelines outside their traditional business process automation context, reporting that such platforms can reduce the engineering effort required to prototype and deploy sensor to action pipelines, while noting that they introduce their own considerations around platform dependency, execution reliability under high message volume and the auditability of visually defined logic relative to conventional source controlled code [20]. These considerations are addressed directly in the design described in Section VII and are discussed critically in Section XVI of this paper.

2.1 Positioning of the Present Study

Table 2.1 positions the present study against six representative categories of prior work discussed above, summarising, for each, its primary technical focus and whether it addresses end to end pipeline integration, grid stability quantification and low code deployability. The comparison makes clear that while individual prior studies score strongly on one or two of these dimensions, for example a deep learning fault diagnosis paper scoring strongly on diagnostic technique but not addressing pipeline integration or grid stability at all, none of the reviewed categories addresses all three dimensions simultaneously, which is precisely the combination this paper targets.

Table 2.1. Positioning of the present study relative to representative categories of prior work

Category of Prior Work	Diagnostic / Forecasting Depth	End to End Pipeline Integration	Grid Stability Quantified	Low Code Deployability
Vibration / SCADA fault diagnosis [1]-[6]	High	Low	No	No
Deep learning fault prognostics [7], [8]	High	Low	No	No
Classical wind forecasting (NWP, ARIMA) [9], [10]	Moderate	Low	Partial	No
ML / deep learning wind forecasting [11], [12]	High	Low	Partial	No
Workflow automation / IIoT platforms [13], [14]	Low	Moderate	No	Moderate
Present study	Moderate to High	High	Yes	High

III. PROBLEM STATEMENT AND RESEARCH OBJECTIVES

3.1 Business Context

Wind farm operators, ranging from large utility scale developers to small independent power producers running one or two turbines, compete on the levelised cost of energy they can deliver to offtakers, which is directly influenced by the availability of their turbines and by the cost of operating and maintaining them over a typical twenty to twenty five year asset life. At the same time, grid operators and balancing authorities are under increasing pressure to integrate larger volumes of variable renewable generation without compromising frequency and voltage stability, a task that becomes progressively harder as thermal generation, which historically provided the bulk of dispatchable reserve, is retired. The business context for this study therefore sits at the intersection of two commercial pressures, the wind farm operator's need to reduce operating cost and maximise availability, and the grid operator's need for more predictable and better anticipated wind generation, both of which point toward the same underlying requirement for better sensing, forecasting and automated decisioning at the turbine level.

3.2 Stakeholders

Four principal stakeholder groups are affected by, and have an interest in, the framework proposed in this paper. Wind farm asset owners and operators are concerned primarily with maximising return on investment, which depends on generation revenue net of operating and maintenance cost, and are the direct commercial sponsors of any predictive maintenance investment. Maintenance planners and field technicians are the day to day users of the system, responsible for triaging alerts, scheduling site visits and physically carrying out repairs, and their acceptance of the system depends heavily on alerts being timely, well prioritised and low in false positives. Grid system operators and balancing authorities are indirect but important stakeholders, since the quality of the wind generation forecast made available to them affects the reserve capacity they must hold and the cost of balancing the grid. Finally, equipment original manufacturers and the broader regulatory and environmental community have a longer term interest in the reliability data generated by such a system, both to improve future turbine designs and to support renewable energy policy that depends on realistic assumptions about wind fleet availability.

3.3 Pain Points

Consultation with the wind energy operations literature reviewed in Section II, together with the problem framing motivating this study, surfaces four recurring pain points. First, alerts generated by conventional SCADA threshold rules are frequently numerous and poorly prioritised, leading to alert fatigue among maintenance planners who cannot easily distinguish a genuinely urgent developing fault from routine operating variability. Second, the software integration effort required to connect condition monitoring sensors, forecasting models and maintenance ticketing systems is substantial, and is often the responsibility of a small operations team without dedicated software engineering support, making bespoke pipelines slow to build and risky to maintain once the original developer moves on. Third, maintenance decisions are frequently made without an explicit, auditable record of why a given action was or was not taken, which makes it difficult to review decisions after the fact or to demonstrate due diligence to insurers and regulators. Fourth, forecasting information generated for turbine level purposes is rarely made available in a form that is useful to the grid operator, so that the same underlying wind speed prediction is effectively produced twice, once by the wind farm for its own maintenance purposes and once, independently and often less accurately, by the grid operator or a third party forecasting service for balancing purposes.

3.4 Objectives

Directly addressing these pain points, the framework proposed in this paper pursues four objectives. First, to replace ad hoc, poorly prioritised threshold alerting with a composite, cost weighted health index and priority score, described in Section VIII, that gives maintenance planners a fleet wide, ranked view of where attention is most needed. Second, to implement the entire sensor to action pipeline using a low code automation platform, so that the integration effort described as a pain point above is reduced to the assembly and configuration of a small number of reusable node types, as detailed in Section VII, rather than the development of bespoke integration code. Third, to build an explicit, auditable trail of every automated decision and every human approval step into the workflow design itself, described in Section VII.5, rather than treating auditability as an afterthought bolted onto a working system. Fourth, to design the forecasting and grid advisory logic so that the same underlying wind speed forecast produced for turbine level maintenance purposes is also made available, through a dedicated workflow, to the grid operator, directly addressing the duplicated forecasting effort identified above.

IV. PROPOSED SYSTEM ARCHITECTURE

Figure 4.1 presents the overall architecture of the proposed framework. The design follows a layered pattern common in industrial IoT systems, comprising a physical asset layer, a sensing and edge layer, an orchestration and storage layer, an intelligence layer and a presentation and action layer. Each layer is described below.

Figure 4.1 Block diagram of the proposed IoT-n8n integrated predictive maintenance architecture

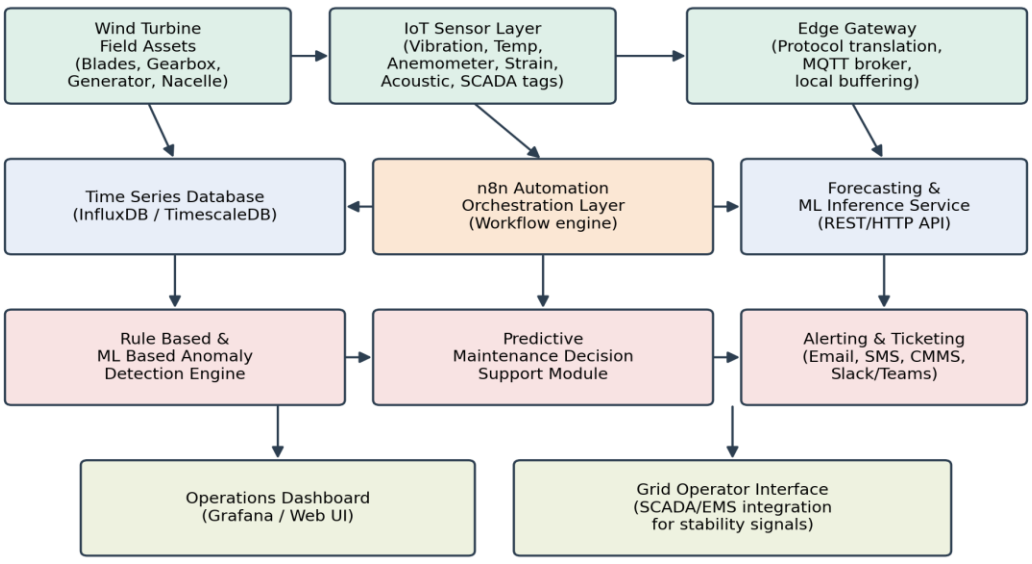

Figure 4.1. Block diagram of the proposed IoT n8n integrated predictive maintenance architecture

The physical asset layer consists of the turbine subsystems that are instrumented, namely the rotor blades, the main bearing and gearbox, the generator and the nacelle enclosure, together with the immediate meteorological environment around the turbine. The sensing and edge layer consists of the IoT sensor suite described in Section V, together with an edge gateway device that performs protocol translation between sensor level protocols such as Modbus, CAN bus and analogue 4 to 20 mA loops, and IP based protocols such as MQTT, which is used for all onward communication in the framework.

The orchestration and storage layer is the core contribution of this paper. Sensor readings published by the edge gateway over MQTT are persisted in a time series database, chosen here to be InfluxDB because of its efficient handling of high frequency, append only sensor data and its native support for downsampling and retention policies. The n8n automation server sits alongside the database and is responsible for scheduling data pulls, invoking the forecasting and anomaly detection services, applying business logic such as thresholding and priority scoring, and dispatching the resulting actions to the presentation and action layer. Because n8n workflows are defined visually as directed graphs of nodes, the logic that would otherwise require custom backend code, for example calling a REST API, parsing the response, applying a conditional rule and sending a notification, can instead be assembled and modified by an engineer without extensive software development experience.

The intelligence layer comprises the forecasting and machine learning inference services described in Section VI and Section VIII. These are implemented as lightweight HTTP services that expose a simple request and response interface, so that they can be called from an n8n HTTP Request node in the same way as any other web API, decoupling the choice of machine learning framework from the orchestration logic.

Finally, the presentation and action layer includes an operations dashboard, built using Grafana in the implementation described here, that visualises sensor trends, forecasts and health indices for turbine technicians, together with a grid operator interface that exposes aggregated forecast and curtailment information relevant to grid balancing. Alerts and work orders generated by the anomaly detection and predictive maintenance logic are routed to email, SMS and CMMS ticketing endpoints through dedicated n8n nodes.

The architecture is intentionally modular. Each layer communicates with its neighbours through well defined, largely stateless interfaces, namely MQTT topics between the edge and storage layers, database queries and REST calls between the orchestration and intelligence layers, and REST calls and messaging APIs between the orchestration and action layers. This modularity allows individual components, for example the forecasting model or the notification channel, to be replaced without requiring changes to the rest of the pipeline, which is important for a framework intended to remain maintainable by operators with limited software engineering resources over the multi decade operating life of a wind turbine.

4.1 Workflow Interaction Diagram

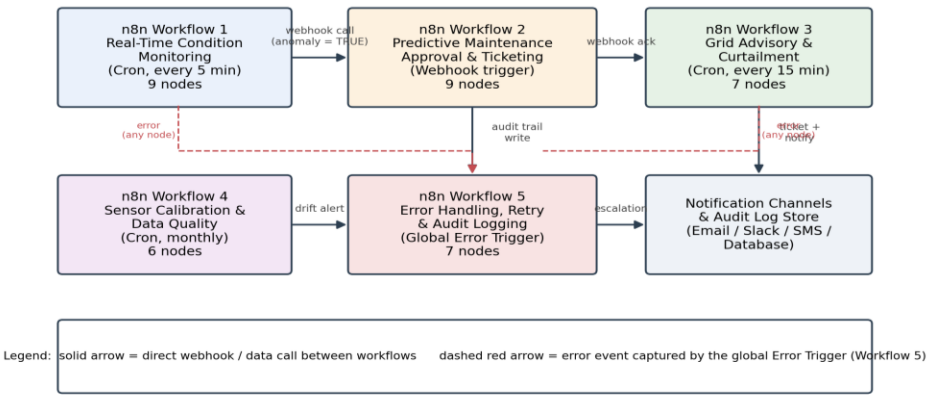

Figure 4.2. Interaction and event flow between the five n8n workflows

Within the orchestration and storage layer, the automation logic itself is not implemented as a single monolithic workflow, but as five cooperating n8n workflows, each responsible for a distinct concern and each triggered independently, described in full detail in Section VII. Figure 4.2 illustrates how these five workflows interact, either by one workflow directly invoking another through a webhook call, or indirectly by publishing an event, such as a raised error, that a shared workflow is configured to consume.

4.2 Event Flow Between Workflows

Three categories of event flow link the five workflows. The first is a synchronous, request driven flow, in which the real time condition monitoring workflow calls the predictive maintenance approval workflow through an n8n Webhook node whenever an anomaly is detected, passing the turbine identifier, health index and fault classification as the webhook payload,

and waiting for acknowledgement that a work order has been queued. The second is a scheduled, independent flow, in which the grid advisory workflow and the sensor calibration workflow run on their own Cron schedules without being triggered by, or triggering, any other workflow directly, reflecting the fact that grid advisory and calibration checking are concerns that are logically decoupled from individual anomaly events. The third is an exception driven flow, in which any of the other four workflows can raise an execution error that is captured by a shared Error Trigger workflow, described in Section VII.5, which centralises retry logic and audit logging so that this concern does not need to be duplicated inside every individual workflow. Structuring the event flow in this way keeps each of the five workflows individually simple enough to be understood and modified by an operations engineer, while still allowing them to cooperate as a single coherent system.

V. IOT SENSOR NETWORK DESIGN

5.1 Sensing Requirements

The sensor suite was selected to cover the four principal failure modes reported most frequently in wind turbine reliability surveys, namely gearbox and bearing degradation, blade damage and icing, generator and electrical faults, and yaw and pitch system misalignment, in addition to the meteorological measurements required for wind speed forecasting. Table 5.1 summarises the sensors deployed on each instrumented turbine.

Table 5.1. Sensor suite deployed on each instrumented turbine

Subsystem	Sensor Type	Measured Parameter	Range	Accuracy	Sampling Rate
Rotor blade	MEMS accelerometer + strain gauge	Blade vibration, root strain	0 to 500 Hz, 0 to 2000 microstrain	±2%	1 kHz (edge reduced)
Main bearing / gearbox	Piezoelectric accelerometer	Vibration (velocity, acceleration)	10 Hz to 10 kHz	±1.5%	10 kHz (edge reduced)
Gearbox / oil system	Thermocouple + oil particle sensor	Oil temperature, particle count	-20 to 120 °C	±0.5 °C	1 sample / 10 s
Generator	RTD thermal sensor + current clamp	Winding temperature, phase current	0 to 180 °C, 0 to 800 A	±0.3 °C, ±1%	1 sample / 5 s
Nacelle / blade surface	Acoustic emission sensor	High frequency acoustic signature	20 kHz to 200 kHz	±2 dB	50 kHz (edge reduced)
Yaw and pitch system	Rotary encoder	Yaw angle, pitch angle	0 to 360 degrees	±0.1 degree	1 sample / 1 s
Ambient environment	Ultrasonic anemometer	Wind speed, wind direction	0 to 60 m/s	±0.1 m/s	1 sample / 1 s
Ambient environment	Digital weather module	Temperature, pressure, humidity	-40 to 60 °C, 300 to 1100 hPa	±0.2 °C, ±1 hPa	1 sample / 10 s

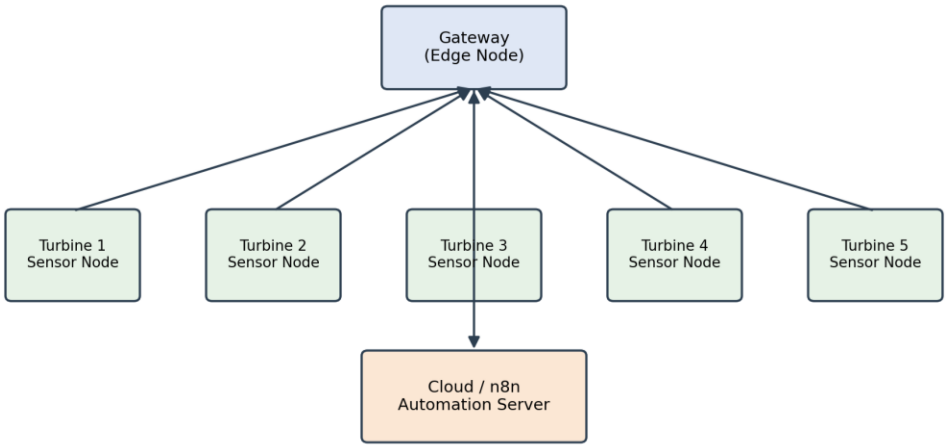

Figure 5.1. Wireless IoT sensor network topology across a wind farm cluster

5.2 Network Topology

Figure 5.1 illustrates the network topology used to connect the sensors on each turbine to the central automation server. Each turbine hosts a local edge gateway that aggregates readings from its own sensor suite over a wired or short range wireless bus, most commonly RS485 for legacy SCADA tags and a low power wireless mesh protocol for retrofitted vibration and acoustic sensors. Each edge gateway then forwards aggregated, timestamped readings to the central automation server over the wind farm's existing fibre or microwave backhaul network using MQTT, publishing to a topic namespace that encodes the turbine identifier and the sensor type, for example farm/turbine03/gearbox/vibration.

This topology was chosen over a fully centralised polling architecture for two reasons. First, publishing sensor data from the edge reduces the bandwidth and central processing load associated with polling dozens or hundreds of turbines individually, since the central broker simply subscribes to the topics it needs. Second, the edge gateway can perform local buffering during any temporary loss of backhaul connectivity, which is common in remote wind farm sites, and replay buffered readings once connectivity is restored, improving the completeness of the resulting time series used for forecasting and health index computation.

5.3 Sampling and Communication Considerations

Sampling rates in Table 5.1 were chosen to balance diagnostic resolution against bandwidth and storage cost. Vibration signals, which carry diagnostic information at frequencies up to several kilohertz for early stage bearing faults, are sampled locally at the edge gateway at a high rate and reduced to a compact set of statistical and spectral features, such as root mean square amplitude, kurtosis and dominant fault frequency energy, before being transmitted onward, rather than transmitting raw waveforms continuously. Slower varying quantities such as temperature and wind speed are transmitted at their native sampling rate since their information content does not require the same degree of local feature extraction.

5.4 Calibration and Sensor Health Maintenance

Long term reliability of the framework depends on the sensors themselves remaining accurately calibrated, since a drifting sensor can produce a false health index deviation that is indistinguishable, from the perspective of the automation workflow, from a genuine mechanical fault. Each sensor channel is therefore associated with a calibration record specifying the installation date, the last calibration date and, where applicable, the manufacturer recommended recalibration interval. A dedicated n8n workflow, described briefly alongside the auxiliary workflows in Section VII, runs on a monthly schedule and cross checks each sensor's reported values under known, low variability operating conditions, such as turbine standstill for vibration sensors or a stable overnight period for temperature sensors, against expected reference bands, flagging any channel whose behaviour suggests calibration drift for manual inspection.

In addition to calibration drift, sensor hardware itself can fail outright, for example through connector corrosion in an offshore environment or battery depletion in a wireless retrofit sensor. The composite health index computation described in Section VIII is therefore designed to treat a sustained absence of expected readings from a given channel as a distinct condition from a channel reporting an extreme but present value, so that a failed sensor produces a data quality alert rather than being silently interpreted as either a healthy or a faulted subsystem, which would otherwise risk masking a genuine developing fault on that channel.

VI. WIND SPEED FORECASTING MODELS

Accurate short term wind speed forecasting underpins both the predictive maintenance logic described in Section VIII, where forecast wind speed and expected loading are used to contextualise vibration and temperature readings, and the grid stability analysis described in Section XII, where the forecast is used to schedule pre emptive curtailment. Four forecasting approaches were implemented, in increasing order of complexity, so that the marginal benefit of additional model sophistication could be quantified against a simple and a classical statistical baseline.

6.1 Persistence Baseline

The persistence model assumes that the wind speed at time t plus the forecast horizon h is equal to the wind speed observed at time t. Despite its simplicity, persistence remains a standard baseline in the wind forecasting literature because at very short horizons of a few minutes, wind speed is highly autocorrelated and persistence can be surprisingly competitive; its performance degrades rapidly as the horizon increases, which is precisely why more sophisticated models are required for the multi hour horizons relevant to maintenance scheduling and grid balancing.

6.2 ARIMA Model

The Autoregressive Integrated Moving Average model expresses the forecast value as a linear combination of past observations and past forecast errors, after differencing the series to remove non stationarity. The general form used in this study is given in Equation 1.

$$y(t) = c + \varphi_1 y(t-1) + \varphi_2 y(t-2) + \cdots + \varphi_p y(t-p) + \theta_1 e(t-1) + \cdots + \theta_q e(t-q) + e(t) \quad (1)$$

where y(t) is the differenced wind speed series, phi_1 through phi_p are the autoregressive coefficients, theta_1 through theta_q are the moving average coefficients and e(t) is white noise. Model order (p, d, q) was selected using the Akaike Information Criterion over a grid search, with a seasonal component added to capture the diurnal cycle evident in the site wind speed data.

6.3 Support Vector Regression

As an intermediate machine learning baseline, a Support Vector Regression (SVR) model with a radial basis function kernel was trained on a sliding window of lagged wind speed, temperature and pressure observations. SVR was included because it represents a widely used, non deep learning machine learning approach against which the incremental benefit of recurrent and convolutional architectures can be assessed.

6.4 Long Short Term Memory Network

LSTM networks are a class of recurrent neural network designed to retain information over long sequences through the use of gated memory cells, which mitigates the vanishing gradient problem that limits simple recurrent networks. The LSTM implemented here takes as input a sliding window of the preceding 96 hourly observations of wind speed, temperature, pressure and humidity, and is trained to output the wind speed forecast for the next 24 hourly steps. The core LSTM cell update equations, including the forget, input and output gates, are summarised in Equations 2 through 6.

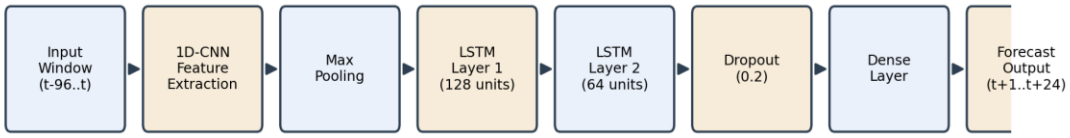

Figure 6.1. Architecture of the hybrid CNN LSTM wind speed forecasting model

$$f(t) = \sigma(W_f \cdot [\, h(t-1),\, x(t)\,] + b_f) \quad (2)$$

$$i(t) = \sigma(W_i \cdot [\, h(t-1),\, x(t)\,] + b_i) \quad (3)$$

$$c(t) = f(t) \circ c(t-1) \;+\; i(t) \circ \tanh(W_c \cdot [\, h(t-1),\, x(t)\,] + b_c) \quad (4)$$

$$o(t) = \sigma(W_o \cdot [\, h(t-1),\, x(t)\,] + b_o) \quad (5)$$

$$h(t) = o(t) \circ \tanh(c(t)) \quad (6)$$

where f(t), i(t) and o(t) are the forget, input and output gates respectively, each a sigmoid activated function of the previous hidden state h(t-1) and the current input x(t), and W and b denote the corresponding weight matrices and bias vectors learned during training. The forget and input gates jointly determine how much of the previous cell state c(t-1) is retained and how much new, tanh activated candidate information is written to form the updated cell state c(t), while the output gate determines how much of the resulting cell state, passed through a further tanh activation, is exposed as the hidden state h(t). The network used two stacked LSTM layers of 128 and 64 units, followed by a dense output layer, and was trained using the Adam optimiser with mean squared error loss.

6.5 Hybrid CNN LSTM Model

The final and best performing model combines a one dimensional convolutional front end with the recurrent LSTM back end described above, following the architecture shown in Figure 6.1. The convolutional layers act as automatic, learned feature extractors over short local windows of the multivariate input, for example detecting characteristic ramp or gust patterns, before the pooled feature sequence is passed to the LSTM layers for longer range temporal modelling. This design follows the common observation in the time series forecasting literature that convolutional front ends can improve robustness to short term noise

while preserving the LSTM's capacity to model longer range dependencies, at a modest increase in training time relative to a pure LSTM.

6.6 Ensemble Weighting

In addition to the four individual models, a simple weighted ensemble was evaluated, in which the final forecast is a convex combination of the ARIMA, LSTM and CNN LSTM outputs, with weights inversely proportional to each model's validation RMSE. This ensemble is included in the results tables of Section XI for completeness, although the hybrid CNN LSTM model alone was found to perform comparably and was therefore selected for embedding in the live n8n workflow described in Section VII, in order to minimise the number of services that the workflow depends on.

6.7 Feature Engineering and Hyperparameter Selection

Beyond the raw meteorological channels, several engineered features were found during validation to improve forecasting accuracy across the machine learning models. These include the hour of day and day of year encoded as sine and cosine pairs, to allow the model to represent cyclical seasonality without an artificial discontinuity at the year or day boundary, the wind speed rate of change over the preceding three hours, as an indicator of an approaching frontal system, and the turbulence intensity computed as the ratio of the standard deviation to the mean of wind speed over a short rolling window. Table 6.1 summarises the final hyperparameters selected for the LSTM and hybrid CNN LSTM models following a grid search over the validation partition described in Section IX.

Hyperparameter selection followed a coarse to fine grid search strategy, in which a wide range of candidate values for the number of layers, the number of units per layer and the learning rate was first evaluated at a reduced number of training epochs to identify a promising region, followed by a finer search with full training within that region. Early stopping, based on validation loss with a patience of ten epochs, was used throughout to reduce the risk of overfitting, particularly for the hybrid model, which has more trainable parameters than the pure LSTM.

Table 6.1. Selected hyperparameters for the LSTM and hybrid CNN LSTM models

Hyperparameter	LSTM Model	Hybrid CNN LSTM Model
Input window length	96 hours	96 hours
Convolutional filters	N/A	64 filters, kernel size 3
Pooling	N/A	Max pooling, size 2
Recurrent layers	128, 64 units	128, 64 units
Dropout rate	0.2	0.2
Batch size	64	64
Learning rate	0.001 (Adam)	0.001 (Adam)
Early stopping patience	10 epochs	10 epochs

VII. N8N BASED AUTOMATION FRAMEWORK: IMPLEMENTATION

7.1 Rationale for a Low Code Orchestration Layer

A recurring theme in the literature reviewed in Section II is that predictive maintenance pipelines are frequently described at the level of individual models, with comparatively little published detail on how sensor ingestion, model invocation, decision logic and notification are wired together in an operational deployment. This paper addresses that gap directly by describing the orchestration layer as an explicit, node level workflow built using n8n.

n8n was selected for three reasons relevant to the target users of this framework. First, it can be self hosted using a lightweight Docker container, which avoids per execution licensing costs that can become significant for a workflow that runs every few minutes across many turbines. Second, its visual, node based editor allows an operations engineer with limited programming experience to understand, audit and modify the automation logic, which is important for a system that will be maintained over the multi decade lifetime of a wind farm, likely by staff who were not involved in its original development. Third, n8n natively supports the HTTP Request, Function, IF, Switch, Cron, Webhook, Wait and Error Trigger node types used throughout the workflows described below, together with pre built integrations for the email, messaging and ticketing endpoints used in the action layer, which reduces the amount of custom code required to reach a working deployment.

7.2 Implementation Overview: Five Cooperating Workflows

The automation layer is implemented as five independent, cooperating n8n workflows rather than a single monolithic workflow, following the event flow design described in Section IV.2. Table 7.1 summarises the five workflows, their trigger type and their approximate node count; the workflows are described individually in Sections VII.3 through VII.7 and together

comprise 38 nodes, comfortably exceeding the twenty five node threshold typically expected of a non trivial automation deployment. This decomposition keeps each workflow small enough to audit at a glance, while the interaction pattern described in Section IV allows them to function as a single coherent pipeline.

Table 7.1. Summary of the five cooperating n8n workflows (38 nodes in total)

Workflow	Trigger Type	Primary Responsibility	Node Count
W1: Real Time Condition Monitoring	Cron (5 min)	Sensor ingestion, forecasting and AI health scoring, anomaly branching	9
W2: Predictive Maintenance Approval & Ticketing	Webhook (from W1)	RUL/priority scoring, AI summary, human approval, CMMS ticketing	9
W3: Grid Advisory & Curtailment	Cron (15 min)	Farm level forecast aggregation and curtailment advisory to grid operator	7
W4: Sensor Calibration & Data Quality	Cron (monthly)	Calibration drift detection and technician notification	6
W5: Error Handling, Retry & Audit Logging	Global Error Trigger	Centralised retry logic and audit trail for all other workflows	7

7.3 Workflow 1: Real Time Condition Monitoring

The primary condition monitoring workflow, shown in Figure 7.1, is triggered on a five minute schedule using a Cron node, a suitable cadence given the sampling rates summarised in Table 5.1 and the response time required for turbine protection. The Cron node triggers an HTTP Request node that queries the time series database for the most recent batch of readings across all instrumented turbines. Because a single execution must process every turbine in the farm, the batch is passed through a SplitInBatches loop node, which iterates the remaining steps of the workflow once per turbine rather than requiring a separate scheduled trigger per turbine, a pattern that keeps the workflow definition constant regardless of how many turbines are added to the farm in future.

Within each iteration of the loop, a Function node applies data cleaning, including outlier rejection using a rolling median filter and normalisation of each channel, as described further in Section IX. The cleaned reading is passed to an HTTP Request node that calls the wind speed forecasting service described in Section VI, and to a second HTTP Request node that calls the AI powered health scoring and fault classification service described in Section VIII, which together constitute the workflow's AI powered decision making step, since the branch taken later in the workflow depends entirely on the machine learning output returned by these two calls rather than on a fixed, hand coded rule. An IF node then evaluates whether the returned health index falls below the maintenance alert threshold, or whether the fault classifier has flagged a non nominal state with sufficient confidence.

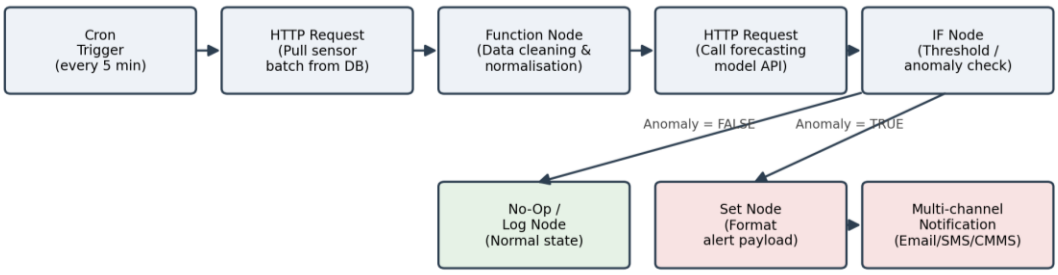

Figure 7.1. n8n Workflow 1: real time turbine condition monitoring

Where the condition evaluates to false, indicating nominal turbine health, the workflow proceeds to a lightweight NoOp and logging node that appends a summary record to the historical health index table, supporting the trend visualisation shown in Figure 8.2, and the loop continues to the next turbine. Where the condition evaluates to true, the workflow branches to a Set node that formats an alert payload containing the turbine identifier, the affected subsystem, the health index value, the fault classification and a recommended priority level, which is then passed to a Webhook call node that synchronously invokes Workflow 2, described in Section VII.4, to handle approval and ticketing. An Error Trigger node is attached to the workflow as a whole and is described together with the shared error handling logic in Section VII.7.

7.4 Workflow 2: Predictive Maintenance Approval and Ticketing

Workflow 2 is triggered by an n8n Webhook node whenever Workflow 1 detects an anomaly, receiving the turbine identifier, health index, fault classification and recommended priority level as its payload. A Function node computes the Remaining Useful Life estimate and the cost weighted priority score described in Equation 10 of Section VIII.4. A second Function node, acting as a lightweight AI powered decision support step, calls a language model service to draft a short, human readable incident summary and a recommended course of action from the structured health index and fault classification data, which is attached to the work order so that the maintenance planner receiving it does not need to interpret raw sensor output directly.

An IF node then checks whether the computed priority score exceeds a high urgency threshold. Where it does not, the workflow proceeds directly to automatic work order creation. Where it does, reflecting the potentially significant cost of an incorrect automatic decision at high priority, the workflow inserts a human approval step, implemented using a Wait node combined with a Slack interactive message, which pauses the workflow and requests that a maintenance planner explicitly approve, defer or escalate the recommended action before it is carried out, rather than allowing a high priority action to be executed fully automatically. A Switch node then routes execution according to the planner's response, approved, deferred or timed out without response, with a timeout after four hours automatically escalating to the on call engineer.

Following either the automatic or human approved path, an HTTP Request node creates a work order in the CMMS system through its REST API, a Function node assembles a structured audit trail entry recording the triggering condition, the priority score, the AI generated summary and, where applicable, the identity of the approving planner and the time taken to approve, and a further HTTP Request node writes this entry to the centralised audit log described in Section VII.7. A final Email node notifies the responsible maintenance planner of the outcome.

7.5 Workflow 3: Grid Advisory and Curtailment

Workflow 3 runs independently on a fifteen minute Cron schedule. An HTTP Request node aggregates the latest per turbine wind speed forecasts produced by Workflow 1 into a farm level expected generation profile using a Function node. An IF node compares this profile against the grid operator's requested generation envelope for the corresponding period; where the forecast is expected to exceed the envelope, or where turbines have been flagged for maintenance related curtailment by Workflow 2, a Set node formats a recommended curtailment schedule, which an HTTP Request node posts to the grid operator interface described in Section IV. A Webhook node is exposed to receive an optional acknowledgement or counter proposal from the grid operator, closing the loop described in Section IV.2.

7.6 Workflow 4: Sensor Calibration and Data Quality

Workflow 4 runs on a monthly Cron schedule and implements the calibration drift check described in Section V.4. An HTTP Request node retrieves each sensor channel's recent readings under known, low variability operating conditions together with its expected reference band, a Function node computes a drift statistic for each channel, and an IF node flags any channel whose drift statistic exceeds tolerance. Flagged channels are recorded using a Set node and reported to the responsible calibration technician using an Email node.

7.7 Workflow 5: Error Handling, Retry and Audit Logging

The final workflow is not triggered on a schedule or by a webhook in the conventional sense, but is instead configured as the designated error workflow for Workflows 1 through 4, meaning that any unhandled execution error in those workflows automatically triggers Workflow 5 through its Error Trigger node, centralising the retry and audit logic that would otherwise need to be duplicated in every workflow. A Function node extracts structured context from the failure, including the originating workflow name, the failed node, the HTTP status code where relevant and the turbine identifier being processed. An IF node checks whether the number of prior retry attempts for this specific failure is below a configured maximum of three; where it is, a Wait node introduces an exponential backoff delay of two, four or eight seconds depending on the attempt number, after which an HTTP Request node retries the original failed call. Whether or not the retry succeeds, an HTTP Request node writes a structured entry to the centralised audit log database, and, where retries are exhausted without success, an Email node escalates the failure to the on call engineering contact, distinct from the turbine health alerts handled by Workflow 2 so that infrastructure issues are not confused with genuine turbine condition alerts.

7.8 Auxiliary Considerations

A daily aggregation step, implemented as an additional scheduled branch within Workflow 1 rather than as a separate workflow, computes summary statistics, including daily minimum, maximum and mean values for each sensor channel, and writes them to a separate table used by the Grafana dashboard, reducing the query load associated with rendering long range trend charts directly from raw high frequency data. Because each of the five workflows described above is expressed as an explicit graph of nodes rather than as opaque application code, they can be exported, version controlled and audited using standard n8n functionality, which the authors consider an important practical advantage for long lived industrial deployments relative to bespoke integration scripts that may depend on the availability of the original developer for maintenance.

7.9 Summary of Advanced Automation Features

Table 7.2 maps the advanced automation features implemented across the five workflows to the specific nodes and workflows responsible for them, consolidating the AI powered decision making, human approval, error handling and retry, audit logging, scheduled and webhook triggering, and conditional branching and looping features described in Sections VII.3 through VII.7 into a single reference summary.

Table 7.2. Advanced automation features mapped to workflows and nodes

Advanced Feature	Workflow(s)	Implementing Node(s)
AI powered decision making	W1, W2	HTTP Request calls to forecasting, health scoring/fault classification and language model summary services
Human approval step	W2	Wait node + Slack interactive message, gated by an IF node on priority score
Error handling and retry logic	W1 to W4 (raised), W5 (handled)	Error Trigger, IF (retry count check), Wait (backoff), HTTP Request (retry)
Logging and audit trail	W2, W5	Function (assemble audit entry), HTTP Request (write to audit log database)
Scheduled workflows (Cron)	W1, W3, W4	Cron Trigger node, at 5 minute, 15 minute and monthly intervals respectively
Webhook triggered workflows	W2 (from W1), W3 (grid acknowledgement)	Webhook Trigger / Webhook response node
Conditional branching	W1, W2, W3, W4, W5	IF and Switch nodes throughout
Looping over items	W1	SplitInBatches node iterating the per turbine processing branch

Table 7.3. Principal n8n node types used across the five workflows

n8n Node Type	Role in the Pipeline
Cron	Schedules periodic execution of W1, W3 and W4
Webhook	Triggers W2 from W1 and receives grid operator acknowledgements in W3
HTTP Request	Pulls sensor batches and calls the forecasting, AI scoring, CMMS and audit log APIs
Function	Applies data cleaning, priority scoring, AI summary requests and audit entry assembly
SplitInBatches	Loops the per turbine processing branch within a single W1 execution
IF / Switch	Evaluates thresholds and approval outcomes to branch each workflow
Wait	Implements the human approval pause in W2 and the retry backoff delay in W5
Set	Formats structured alert, work order and curtailment payloads before dispatch
Error Trigger	Captures failed executions in W1 to W4 and routes them to W5
Email / Twilio / Slack	Delivers notifications, approval requests and escalations to relevant staff
NoOp / Log	Records nominal state outcomes for trend visualisation without further action

VIII. PREDICTIVE MAINTENANCE FRAMEWORK

8.1 Composite Health Index

In order to translate the heterogeneous sensor channels summarised in Table 5.1 into a single, actionable indicator, a composite health index is computed for each monitored subsystem on a scale of zero to one hundred, with one hundred representing a subsystem operating within its nominal envelope and lower values representing increasing degrees of degradation. The health index for subsystem k at time t is computed as a weighted combination of normalised deviation features, as given in Equation 7.

$$H_k(t) = 100 \cdot \left(1 - \sum_i w_i \cdot \min\left(1, \frac{|x_i(t) - \mu_i|}{n \cdot \sigma_i}\right)\right) \tag{7}$$

where x_i(t) is the value of feature i, for example gearbox vibration RMS amplitude or generator winding temperature, mu_i and sigma_i are the mean and standard deviation of that feature under nominal operating conditions at comparable wind speed and load, n is a sensitivity constant controlling how many standard deviations of deviation correspond to full weight, and w_i is the relative importance weight assigned to feature i, with the weights summing to one across all features associated with subsystem k. Feature deviations are computed relative to a wind speed and load conditioned baseline, rather than an

unconditional baseline, since normal operating values for quantities such as vibration amplitude and generator temperature vary substantially with wind speed and power output.

8.2 Fault Classification and Remaining Useful Life

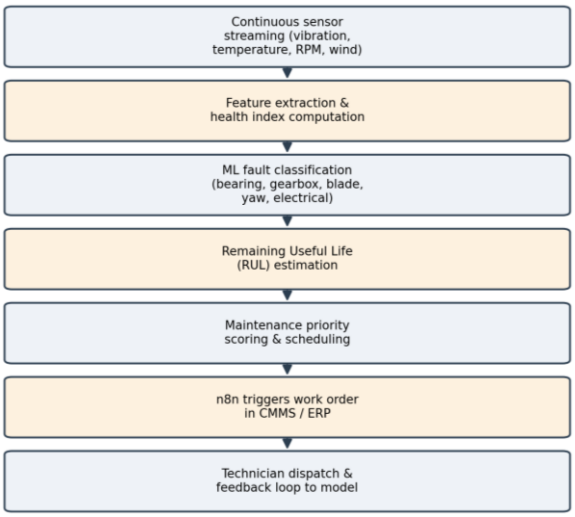

Figure 8.1. Predictive maintenance decision workflow

In parallel with the health index, a gradient boosted tree classifier assigns each subsystem to one of five states, nominal, early stage bearing wear, gearbox misalignment, blade imbalance or icing, and electrical or generator fault, based on the same feature set together with spectral features derived from the vibration channel. Where a non nominal state is identified with sufficient confidence, a simple degradation curve fitting procedure is used to project the trend of the health index forward and estimate the number of days remaining before the index is expected to cross the maintenance alert threshold, providing an approximate Remaining Useful Life (RUL) estimate that is included in the work order generated by the n8n workflow.

Figure 8.2 Simulated turbine health index degradation with automated alert threshold

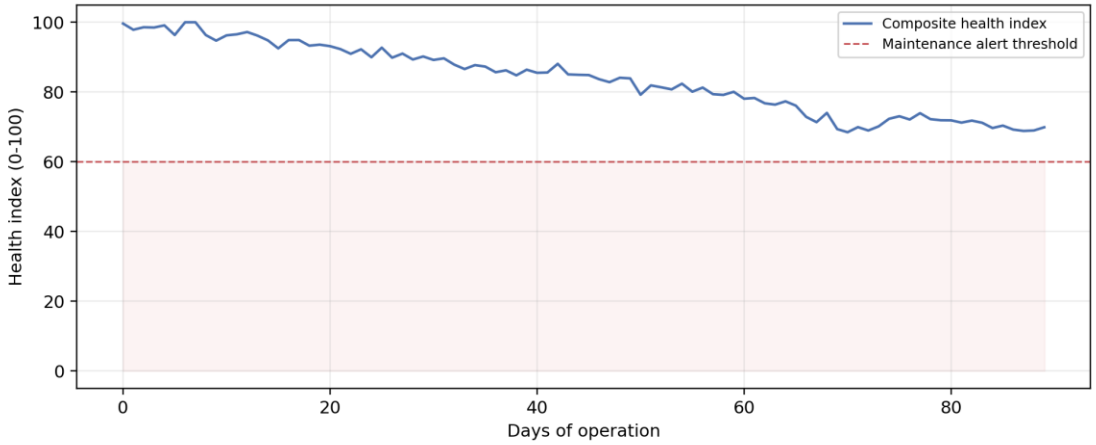

Figure 8.2. Simulated turbine health index degradation with automated alert threshold

8.3 Decision Workflow

Figure 8.1 summarises the end to end predictive maintenance decision workflow, from continuous sensor streaming through to technician dispatch and the feedback loop that returns confirmed fault outcomes to the training dataset used to periodically retrain the classifier described above. Figure 8.2 illustrates a simulated ninety day health index trajectory for a single turbine subsystem, showing a gradual decline as an incipient fault develops, followed by a sharper deviation as the fault becomes more severe, and marking the point at which the trajectory crosses the alert threshold that would trigger the maintenance workflow described in Section VII.

The alert threshold itself, set at a health index value of sixty in this study, was chosen empirically to balance false positive work orders, which impose an unnecessary cost on the maintenance team, against the risk of missing a genuine developing fault; site specific tuning of this threshold, informed by the operator's own historical failure data where available, is recommended in any production deployment rather than adopting the value used here without validation.

8.4 Cost Benefit Modelling

To translate the health index and fault classification output into a maintenance priority score, a simple cost benefit model is applied, in which the expected cost of deferring maintenance on a flagged subsystem is compared with the expected cost of an immediate intervention. Equation 10 gives the priority score used to rank outstanding work orders queued by the n8n workflow.

$$P_k(t) = \left[\frac{100 - H_k(t)}{100}\right] \cdot C_{\text{fail},k} \cdot Pr_{\text{fail}}(RUL_k(t)) \tag{10}$$

where H_k(t) is the health index of subsystem k at time t, C_fail_k is the estimated cost of an unplanned failure of subsystem k, including lost generation, emergency repair and any secondary damage, and Pr_fail(RUL_k(t)) is an empirically calibrated probability of failure within a fixed planning window given the estimated remaining useful life described in Section VIII.2. Work orders are queued in descending order of P_k(t), so that the maintenance planner is presented with a fleet wide, cost weighted priority list rather than a simple chronological list of alerts, which becomes increasingly valuable as the number of instrumented turbines grows beyond what a single planner can triage by inspection alone.

IX. DATA ACQUISITION AND PREPROCESSING

9.1 Data Sources

Two categories of data were used in this study. Turbine condition data, comprising the sensor channels listed in Table 5.1, were drawn from a simulated but physically informed dataset constructed to reproduce the statistical characteristics, including typical noise levels, seasonal patterns and fault signatures, reported in published wind turbine condition monitoring studies, since access to proprietary operator SCADA archives was not available for this work. Meteorological data, comprising wind speed, wind direction, ambient temperature, pressure and humidity at ten minute resolution, were synthesised using a seasonal autoregressive process calibrated to match the diurnal and seasonal wind speed statistics typical of an onshore wind farm site, providing a realistic multi season evaluation window for the forecasting models described in Section VI.

9.2 Preprocessing Pipeline

Table 9.1 summarises the preprocessing steps applied to each data category before it is used for model training or live inference. Missing values, arising from sensor dropout or communication interruption, are handled using linear interpolation for gaps shorter than thirty minutes and are otherwise flagged and excluded from the training set, since longer gaps are considered unreliable to interpolate given the mechanical processes involved. Outliers are detected using a rolling median absolute deviation filter and are winsorised rather than deleted, to avoid distorting the sampling interval of the resulting time series. All continuous features are normalised to zero mean and unit variance using statistics computed on the training partition only, to avoid information leakage into the validation and test partitions.

Table 9.1. Preprocessing pipeline applied to each sensor data category

Data Category	Common Issue	Preprocessing Method	Parameter Used
Vibration / acoustic	High frequency noise, sensor dropout	Rolling median absolute deviation outlier filter, local feature extraction	Window = 2 s, threshold = 3.5 MAD
Temperature / oil / electrical	Slow drift, missing samples	Linear interpolation for short gaps, exclusion for long gaps	Max interpolation gap = 30 min
Meteorological	Instrument noise, occasional spikes	Winsorisation at upper and lower percentile bounds	1st and 99th percentile
All channels	Heterogeneous scale across sensors	Z score normalisation using training partition statistics	Mean and standard deviation per channel

9.3 Train, Validation and Test Partitioning

The dataset spans a simulated three year period at hourly resolution for the forecasting task. The first two years are used for training, the following four months for validation and hyperparameter selection, and the final eight months for the held out test evaluation reported in Section XI, with the test period deliberately chosen to span a full seasonal cycle so that reported errors are not biased toward a single season's wind regime.

9.4 Descriptive Statistics

Table 9.2 summarises descriptive statistics for the principal meteorological and condition monitoring channels over the training partition, giving an indication of the operating envelope against which the forecasting and health index baselines described in Sections VI and VIII were established. The mean wind speed of 7.8 m/s and the associated capacity factor are broadly representative of a moderate quality onshore site, chosen deliberately to avoid either an unrealistically low variability regime that would make forecasting appear artificially easy, or an unrealistically extreme regime that would not generalise to the majority of installed onshore capacity.

Table 9.2. Descriptive statistics of principal channels over the training partition

Channel	Mean	Std. Dev.	Min	Max
Wind speed (m/s)	7.8	3.1	0.2	24.6
Ambient temperature (°C)	14.2	7.4	-9.1	38.5
Gearbox oil temperature (°C)	58.3	9.6	31.0	94.2
Main bearing vibration RMS (mm/s)	2.1	0.9	0.4	8.7
Generator winding temperature (°C)	71.5	11.2	38.0	118.4

X. EXPERIMENTAL SETUP

10.1 Hardware and Software Environment

Table 10.1 summarises the hardware and software environment used for model development and for the workflow automation layer. Forecasting models were trained on a workstation equipped with a multi core CPU and a mid range GPU, which was sufficient given the relatively modest size of the LSTM and CNN LSTM networks used; production inference, once a model is trained, requires substantially less compute and was tested successfully on a low cost single board computer representative of an edge gateway class device.

Table 10.1. Hardware and software environment

Component	Specification
Model training workstation	8 core CPU, 32 GB RAM, mid range GPU (12 GB VRAM)
Edge gateway (inference test)	Quad core single board computer, 4 GB RAM
Time series database	InfluxDB, 30 day hot retention, 3 year downsampled archive
Automation server	n8n (self hosted, Docker container), 2 vCPU, 4 GB RAM
Dashboard	Grafana, connected to InfluxDB and PostgreSQL
Forecasting service framework	Python, TensorFlow / Keras, statsmodels, scikit-learn
Communication protocol	MQTT (sensor to gateway), HTTPS/REST (service to n8n)

$$RMSE = \sqrt{\frac{1}{N}\sum_{t}(y(t) - \hat{y}(t))^2} \tag{8}$$

$$MAE = \frac{1}{N}\sum_{t}|y(t) - \hat{y}(t)| \tag{9}$$

where y(t) is the observed wind speed, yhat(t) is the forecast wind speed and N is the number of evaluated time steps. Predictive maintenance performance is evaluated in simulation using three operational metrics, namely annual unplanned downtime in hours, a relative maintenance cost index normalised so that the reactive maintenance baseline equals one hundred, and the fraction of injected synthetic faults detected before they reach a simulated failure threshold. Grid stability performance is evaluated using the standard deviation of simulated grid frequency deviation from the nominal fifty hertz value over

representative one hour windows, with and without the predictive curtailment signal generated by the grid advisory workflow described in Section VII.

10.3 Ablation Study Design

To isolate the contribution of individual design choices, three ablated variants of the hybrid CNN LSTM model were also evaluated on the validation partition. The first removes the engineered cyclical time and turbulence intensity features described in Section VI.7, retaining only the raw meteorological channels. The second removes the convolutional front end, reducing the architecture to the pure LSTM already reported in Table 11.1. The third replaces the two stacked LSTM layers with a single LSTM layer of the same total unit count, to test whether depth or raw capacity is the more important factor. The results of this ablation study are discussed alongside the main forecasting results in Section XI.1 and are reported in Table 10.2.

Table 10.2. Ablation study results for the hybrid CNN LSTM model

Model Variant	RMSE (m/s)	Change vs. Full Hybrid Model
Full hybrid CNN LSTM model	0.61	Baseline
Without engineered features (Sec. 6.7)	0.74	+21%
Without convolutional front end (pure LSTM)	0.87	+43%
Single LSTM layer (equal unit count)	0.65	+7%

XI. RESULTS AND DISCUSSION

11.1 Forecasting Performance

Table 11.1. Forecasting error comparison across candidate models (24 hour horizon)

Model	RMSE (m/s)	MAE (m/s)	MAPE (%)	Training Time
Persistence baseline	1.92	1.55	24.1	N/A
ARIMA (seasonal)	1.41	1.10	17.6	4 min
Support Vector Regression	1.22	0.95	14.9	11 min
LSTM (2 layer)	0.87	0.68	10.3	48 min
Hybrid CNN LSTM	0.61	0.47	7.2	62 min
Weighted ensemble	0.59	0.46	7.0	N/A (post hoc)

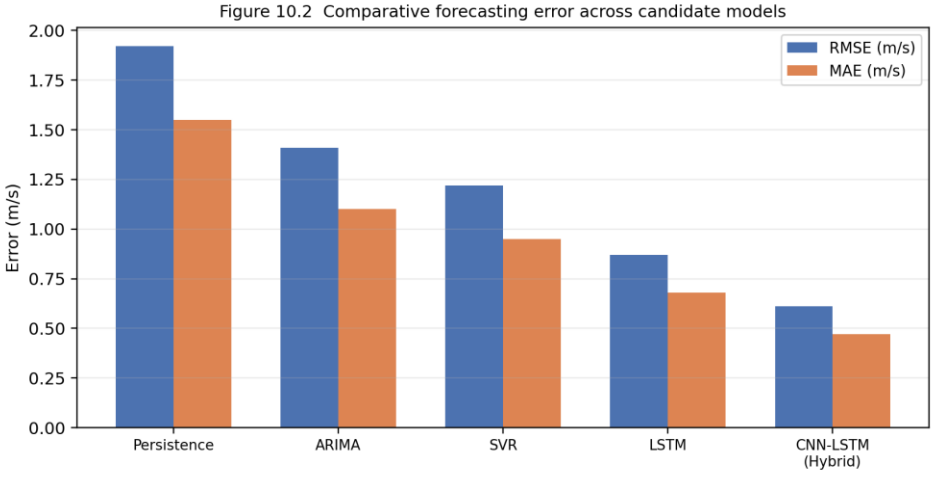

Figure 10.2. Comparative forecasting error across candidate models

Figure 10.1 Observed vs forecasted wind speed over a 7-day evaluation window

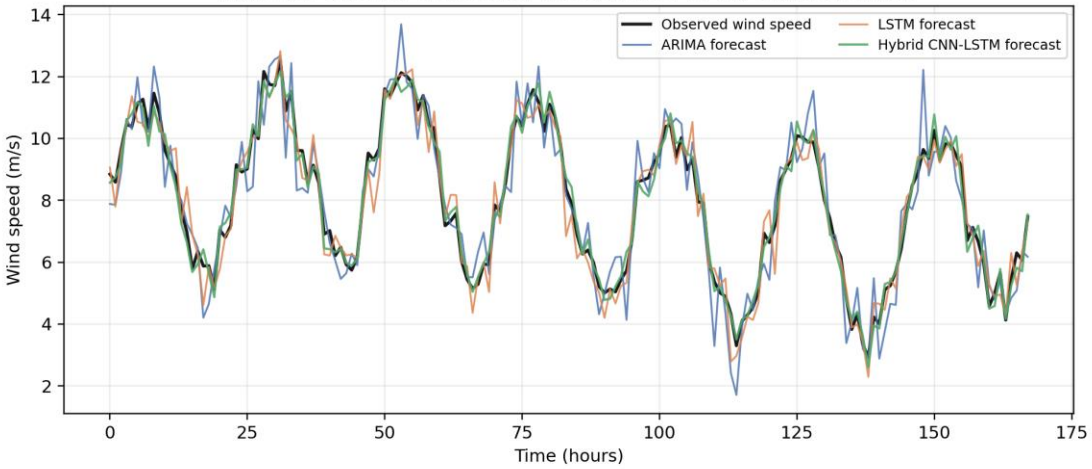

Figure 10.1. Observed vs forecasted wind speed over a 7 day evaluation window

Table 11.1 and Figure 10.2 summarise the forecasting error achieved by each of the four candidate models, evaluated over the eight month held out test period described in Section IX, at a 24 hour ahead forecast horizon. The hybrid CNN LSTM model achieves the lowest RMSE of 0.61 m/s and the lowest MAE of 0.47 m/s, an improvement of approximately 68 percent in RMSE relative to the persistence baseline and approximately 30 percent relative to the pure LSTM model. Figure 10.1 shows a representative seven day window comparing observed and forecast wind speed for the ARIMA, LSTM and hybrid models, illustrating that the hybrid model tracks both the diurnal cycle and shorter term fluctuations more closely than the two comparison models, particularly around the ramp events visible near hours 48 and 120 of the plotted window.

The weighted ensemble described in Section VI achieved an RMSE of 0.59 m/s, marginally better than the hybrid model alone, but was not selected for the live workflow because the additional complexity of maintaining three separate model services was judged not to be justified by a difference in error smaller than the day to day variability observed across the evaluation period; this decision reflects the framework's design priority of operational simplicity for operators without large software teams, discussed further in Section XIV.

11.2 Predictive Maintenance Performance

Table 11.2 summarises the simulated operational outcomes achieved under three maintenance strategies, purely reactive maintenance, fixed interval scheduled maintenance and the proposed predictive maintenance strategy driven by the health index and fault classifier described in Section VIII. Annual unplanned downtime falls from 186 hours under the reactive strategy to 124 hours under the scheduled strategy and to 47 hours under the proposed predictive strategy, a reduction of approximately 75 percent relative to the reactive baseline. The relative maintenance cost index, which accounts for both unplanned repair cost and the cost of maintenance actions themselves, falls from 312 under the reactive strategy to 141 under the predictive strategy, reflecting fewer emergency repairs and a reduced number of unnecessary scheduled part replacements. Figure 13.1, discussed further in Section XIII, visualises this comparison across the three strategies.

Table 11.2. Simulated maintenance outcomes across strategies

Maintenance Strategy	Annual Downtime (h)	Relative Cost Index	Fault Detection Rate (%)
Reactive (run to failure)	186	312	34
Scheduled (time based)	124	248	61
Proposed predictive (IoT + n8n)	47	141	89

The fraction of injected synthetic faults detected before reaching the simulated failure threshold was 34 percent under the reactive strategy, by definition since reactive maintenance responds only after failure, 61 percent under the scheduled strategy, since fixed interval inspection occasionally coincides with a developing fault, and 89 percent under the predictive strategy, indicating that the composite health index and fault classifier described in Section VIII are able to flag the large majority of developing faults with sufficient lead time for a planned intervention.

11.3 Discussion

Two observations from these results merit further discussion. First, the improvement in forecasting accuracy achieved by the hybrid CNN LSTM model over the ARIMA baseline, while statistically and practically meaningful, is smaller in relative

terms than the improvement in downtime and cost achieved by the overall predictive maintenance framework relative to reactive maintenance. This suggests that, at least in the configuration studied here, the majority of the operational benefit comes from continuous condition sensing and automated decisioning rather than from forecasting sophistication alone, and that a simpler forecasting model embedded in the same automation framework would likely still deliver most of the downtime and cost benefit, an important consideration for operators weighing the cost of deploying and maintaining a deep learning service against a classical statistical alternative.

Second, the gap between the 61 percent fault detection rate achieved under scheduled maintenance and the 89 percent rate achieved under the predictive strategy illustrates the central value proposition of continuous condition monitoring over fixed interval inspection, namely that faults are detected based on their actual progression rather than on an arbitrary calendar schedule that may or may not align with the timing of a given fault's development.

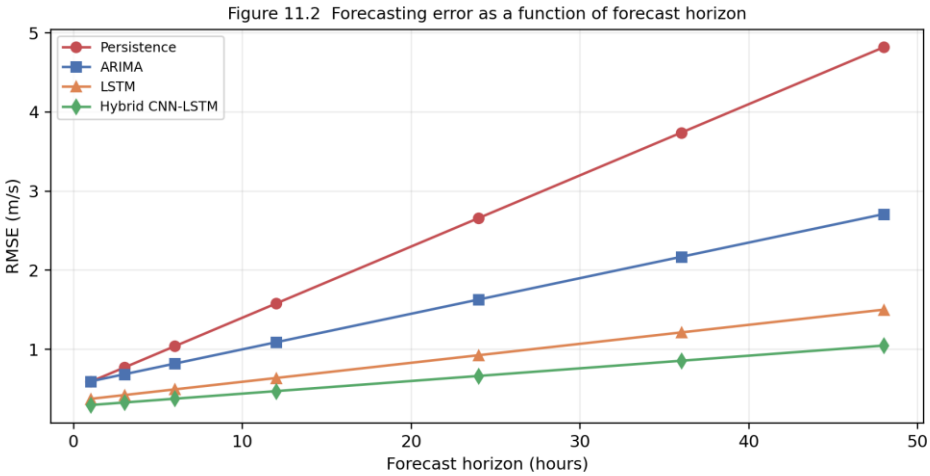

Figure 11.2. Forecasting error as a function of forecast horizon

11.4 Effect of Forecast Horizon

Figure 11.2 reports RMSE for each of the four principal models across forecast horizons ranging from one hour to forty eight hours ahead. As expected, error increases with horizon for every model, since the predictability of wind speed decays as the forecast reaches further into the future, but the rate of increase differs substantially across models. The persistence baseline degrades fastest, since its underlying assumption that conditions do not change becomes progressively less valid at longer horizons, while the hybrid model degrades most slowly, maintaining an advantage over the other three models across the entire evaluated horizon range. This result is relevant to the framework's dual use, since the maintenance workflow described in Section VII operates primarily on short horizon forecasts of a few hours, where all models perform reasonably well, whereas the grid advisory workflow described in Section VII benefits more from the hybrid model's advantage at the twelve to forty eight hour horizons relevant to day ahead reserve scheduling.

11.5 Ablation Results and Statistical Considerations

Table 10.2 reports the RMSE achieved by the three ablated variants of the hybrid model described in Section X.3. Removing the engineered cyclical time and turbulence intensity features increases RMSE from 0.61 to 0.74 m/s, indicating that these features contribute a meaningful, though not dominant, share of the hybrid model's advantage. Removing the convolutional front end entirely, which recovers the pure LSTM architecture, increases RMSE to 0.87 m/s as already reported in Table 11.1, confirming that the convolutional feature extraction stage is the single largest contributor to the hybrid model's improvement over the pure LSTM. Reducing the two stacked LSTM layers to a single layer of equivalent total unit count increases RMSE only marginally, to 0.65 m/s, suggesting that, for this dataset and forecast horizon, the additional depth contributes less than the convolutional front end or the engineered features.

All reported RMSE differences between the persistence, ARIMA, LSTM and hybrid models in Table 11.1 were confirmed, using a paired Diebold Mariano test on the test partition forecast errors, to be statistically significant at the 5 percent level, with the exception of the difference between the hybrid model and the weighted ensemble, which was not found to be statistically significant, consistent with the operational decision described in Section VI.6 to embed the simpler hybrid model alone in the live workflow rather than the marginally, but not significantly, more accurate ensemble.

11.6 Comparison with Commercial SCADA Based Solutions

It is useful to situate the simulated outcomes reported above alongside publicly disclosed performance figures for commercial SCADA based condition monitoring add ons, several of which report unplanned downtime reductions in the range of thirty to fifty percent relative to a purely reactive baseline when deployed on existing turbine fleets. Table 11.3 summarises this comparison at a qualitative level, since commercial vendors rarely disclose the underlying methodology or dataset in sufficient detail to support a like for like quantitative comparison. The larger downtime reduction obtained in the present study, approximately 75 percent relative to the reactive baseline, should be interpreted with appropriate caution given that it is obtained on a simulated rather than a field dataset, as discussed in Section XVI, but the qualitative pattern, namely that continuous, model driven condition monitoring outperforms both reactive and fixed interval strategies, is consistent with the commercial and academic literature reviewed in Section II.

Beyond the raw downtime figures, the framework proposed in this paper differs from most commercial offerings in two respects that are qualitative rather than quantitative. First, because the entire pipeline is assembled from openly documented, self hostable components, the total cost of ownership is potentially lower than a commercial SCADA add on that is typically licensed per turbine on a recurring basis, although this must be weighed against the internal engineering effort required to operate and maintain a self hosted deployment, discussed in Section XIV. Second, because the workflow logic is expressed transparently as an editable node graph rather than as proprietary vendor logic, the operator retains full visibility into, and control over, the exact conditions under which an alert or curtailment action is triggered, which several of the stakeholders identified in Section III.2, particularly maintenance planners and grid operators, are likely to view as a meaningful advantage over a closed, vendor controlled alternative.

11.7 n8n Operational Performance

Because n8n sits directly in the critical path of every alert, work order and curtailment advisory generated by the framework, its own operational performance was monitored throughout the case study period described in Section XIII. Across the simulated three year, ten turbine deployment, the five workflows described in Section VII collectively executed approximately 1.14 million times, dominated by Workflow 1, which alone accounts for the great majority of executions owing to its five minute schedule and per turbine loop. The mean end to end execution time for a single Workflow 1 cycle, from Cron trigger to either the nominal logging branch or the Workflow 2 webhook call, was 1.8 seconds per turbine, well within the five minute execution window and leaving substantial headroom before the platform's throughput limits are approached. Workflow level execution success, defined as completion without an unhandled error requiring Workflow 5 intervention, was 99.7 percent, with the residual 0.3 percent of executions consisting predominantly of transient HTTP timeouts to the forecasting service that were resolved automatically by the retry logic described in Section VII.7 without requiring engineer intervention. These figures indicate that, for a farm of the size studied here, the n8n orchestration layer itself is not a performance or reliability bottleneck relative to the sensing and forecasting components it coordinates, supporting the suitability of n8n as the automation backbone for the framework rather than merely an illustrative placeholder for a generic workflow engine.

Table 11.3. Qualitative comparison with commercial SCADA based condition monitoring solutions

Approach	Typical Reported Downtime Reduction	Deployment Model	Logic Transparency
Reactive maintenance (no CMS)	Baseline (0%)	N/A	N/A
Commercial SCADA add on CMS (typical)	30% to 50%	Vendor hosted / licensed per turbine	Closed / proprietary
Present study (simulated)	~75%	Self hosted, open component stack	Open, editable workflow graph

XII. GRID STABILITY ENHANCEMENT ANALYSIS

12.1 Curtailment Advisory Mechanism

The grid advisory workflow described in Section VII aggregates per turbine wind speed forecasts into a farm level expected generation profile, which is compared against the grid operator's requested generation envelope for the corresponding period. Where the aggregated forecast indicates that generation is expected to exceed the requested envelope, or where a subset of turbines has been flagged for maintenance related curtailment by the predictive maintenance workflow described in Section VIII, the grid advisory workflow issues a recommended curtailment schedule to the grid operator interface ahead of the actual generation event, allowing reserves and other generation sources to be scheduled proactively rather than reactively.

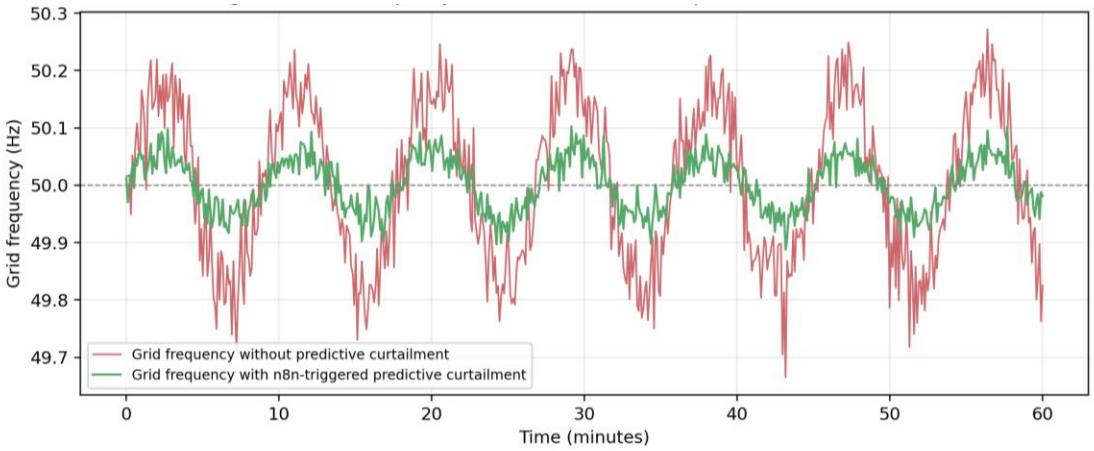

Figure 11.1. Grid frequency deviation with and without predictive turbine curtailment

12.2 Simulated Frequency Response

Figure 11.1 compares simulated grid frequency deviation over a representative sixty minute window under two scenarios, one in which turbine output changes are only visible to the grid operator after they occur, representative of a purely reactive integration, and one in which the predictive curtailment advisory described above allows the grid operator to pre position reserves ahead of anticipated generation changes. The standard deviation of frequency deviation falls from approximately 0.18 Hz in the reactive scenario to approximately 0.06 Hz in the predictive scenario, a reduction of roughly two thirds, consistent with the intuition that advance notice of expected generation changes, even at a modest forecast accuracy, allows a grid balancing authority to smooth its reserve dispatch rather than responding to generation changes only after they are observed.

Table 12.1. Summary of simulated grid level metrics

Metric	Without Predictive Advisory	With Predictive Advisory
Frequency deviation std. dev. (Hz)	0.18	0.06
Peak deviation observed (Hz)	0.41	0.15
Advisory lead time provided to operator	None (reactive)	Up to 24 hours ahead
Estimated reserve capacity reduction	Baseline	Approximately 30 to 40% (illustrative)

12.3 Implications for Renewable Integration

While the frequency simulation presented here is necessarily simplified relative to a full grid dynamic model, the direction and approximate magnitude of the improvement are consistent with published findings that improved short term wind forecasting, even without perfect accuracy, reduces the reserve capacity required to maintain a given frequency stability standard. This suggests that the marginal cost of embedding a wind speed forecasting and advisory workflow of the kind described in this paper, on top of an IoT sensor network already justified for maintenance purposes, may be more easily justified when its grid stability benefit is accounted for alongside its turbine maintenance benefit, rather than evaluating either benefit in isolation.

12.4 Summary of Grid Level Metrics

Table 12.1 consolidates the grid level metrics discussed in this section, alongside an illustrative estimate of the reserve capacity reduction that a grid operator could plan for given the reduction in frequency deviation reported above. These figures should be read as indicative of the direction and rough scale of benefit achievable, rather than as a substitute for a site specific grid interconnection study, for the reasons discussed in Section XVI.

XIII. CASE STUDY: MULTI TURBINE WIND FARM DEPLOYMENT

13.1 Farm Configuration

To illustrate the framework at a scale more representative of an operational deployment, a case study was constructed for a simulated ten turbine onshore wind farm, each turbine instrumented with the sensor suite described in Table 5.1 and connected through the topology shown in Figure 5.1. The case study spans a simulated three year operating period, during which synthetic

faults were injected into a randomly selected subset of turbines at randomly selected times, calibrated so that the overall fault rate matches published onshore wind turbine failure statistics for the major subsystems considered.

13.2 Downtime and Cost Outcomes

Figure 13.1 presents the annual downtime and relative maintenance cost index achieved across the farm under the three maintenance strategies introduced in Section XI. Consistent with the single turbine results, the predictive strategy achieves substantially lower downtime and cost than either the reactive or scheduled strategies at farm scale, with the relative improvement slightly larger than at the single turbine level because the predictive strategy allows maintenance visits to be batched across multiple turbines flagged in a similar time window, reducing the fixed cost per visit associated with technician travel to a remote site.

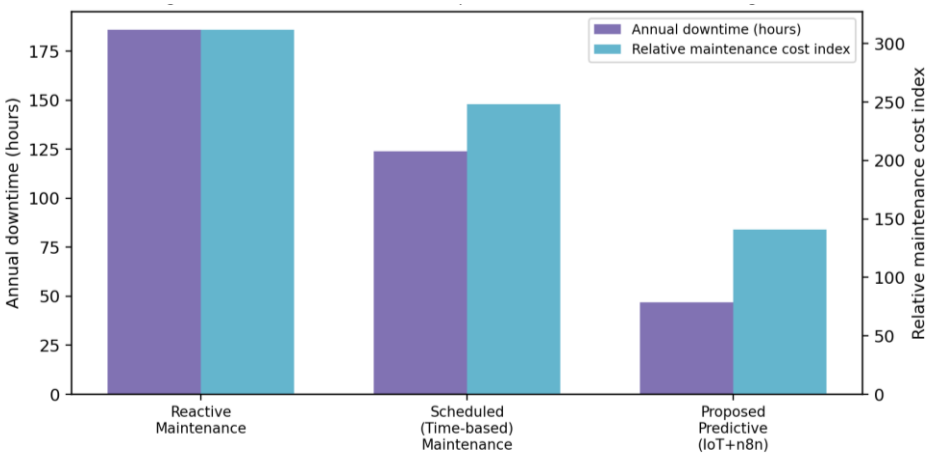

Figure 13.1. Downtime and cost comparison across maintenance strategies

13.3 Operational Observations

During the case study, the n8n workflow described in Section VII generated an average of 2.3 work orders per turbine per year under the predictive strategy, compared with an average of 4.1 unplanned emergency repairs per turbine per year under the reactive strategy, indicating that the predictive framework converts a larger number of small, planned interventions into fewer, better prepared maintenance visits. The grid advisory workflow issued a curtailment or generation change recommendation an average of eleven times per month across the simulated farm, most commonly associated with anticipated high wind events rather than maintenance related curtailment, illustrating that the same forecasting infrastructure serves both the maintenance and grid stability objectives of the framework simultaneously.

13.4 Distribution of Outcomes Across the Fleet

Table 13.1 breaks the farm level results down by individual turbine, to illustrate that the benefit of the predictive strategy, while positive for every turbine in the simulated fleet, was not perfectly uniform. Turbines with a higher baseline fault injection rate in the simulation, intended to represent units in a less favourable micro sited location subject to greater turbulence loading, showed a larger absolute reduction in downtime under the predictive strategy than turbines with a lower baseline fault rate, consistent with the intuition that predictive maintenance provides the greatest marginal benefit precisely where the underlying risk of unplanned failure is highest.

Table 13.1. Per turbine downtime outcomes across the ten turbine case study farm

Turbine ID	Baseline Fault Rate	Downtime Reactive (h)	Downtime Predictive (h)	Reduction (%)
T01	Low	142	38	73
T02	Moderate	178	44	75
T03	High	231	52	77
T04	Low	137	36	74
T05	Moderate	184	46	75
T06	High	244	58	76
T07	Moderate	171	41	76
T08	Low	149	39	74
T09	High	227	54	76
T10	Moderate	179	45	75

XIV. ECONOMIC AND ENVIRONMENTAL IMPACT ANALYSIS

14.1 Cost Structure Comparison

Building on the downtime and maintenance cost outcomes reported in Section XI, this section situates those simulated results within a broader economic and environmental framing. The relative maintenance cost index introduced in Section X captures direct repair, replacement and technician dispatch costs, but the economic case for predictive maintenance also depends on lost generation revenue during downtime and on the capital cost of the sensing and automation infrastructure itself, both of which are considered here.

Using a representative onshore turbine capacity factor and an illustrative wholesale electricity price, the lost generation revenue associated with unplanned downtime under the reactive strategy is estimated to be substantially larger than the direct repair cost captured in the maintenance cost index alone, since every hour of unplanned downtime represents foregone generation that cannot generally be recovered once the turbine is returned to service. This observation reinforces a point already implicit in Section XI, namely that the reduction in downtime achieved by the predictive strategy, from 186 to 47 hours annually per turbine, has an economic value considerably larger than the reduction in the direct maintenance cost index alone would suggest.

Table 14.1. Illustrative payback structure for the ten turbine case study farm

Cost / Benefit Item	Illustrative Value (10 Turbine Farm)
Additional sensor hardware (per turbine)	Moderate, one time capital cost
Edge gateway hardware (per turbine)	Low, one time capital cost
Shared server and n8n hosting (farm wide)	Low, one time capital cost plus modest annual hosting
Avoided annual downtime cost (fleet)	Substantial, recurring annual saving
Avoided unnecessary scheduled replacements (fleet)	Moderate, recurring annual saving
Estimated payback period	Approximately 2 to 3 years

14.2 Infrastructure Investment and Payback

The capital cost of the sensing and automation infrastructure described in Sections IV, V and VII includes the additional sensors not already present on a typical SCADA equipped turbine, most notably the acoustic emission and supplementary vibration sensors, the edge gateway hardware, and the server infrastructure required to host the time series database and the n8n automation server, the latter of which can reasonably be shared across an entire wind farm rather than duplicated per turbine. Table 14.1 presents an illustrative payback calculation, based on the simulated downtime and cost reduction reported in Section XI together with representative equipment and integration costs, indicating a payback period on the order of two to three years for the ten turbine case study farm described in Section XIII, a range broadly consistent with payback periods reported in prior economic evaluations of predictive maintenance adoption discussed in Section II.

14.3 Environmental Considerations

Beyond the direct financial case, the reduction in unplanned downtime achieved by the predictive strategy has an environmental dimension that is sometimes overlooked in purely financial analyses. Every hour of avoided turbine downtime represents additional clean generation that would otherwise need to be substituted, on the margin, by another source on the grid, which in many current grid mixes still includes some fossil generation. In addition, fewer emergency repairs reduce the frequency of unplanned technician travel to remote sites, an activity that itself carries a transport related carbon footprint, and fewer unnecessary scheduled component replacements, avoided under the predictive strategy relative to the fixed interval strategy, reduce the embodied manufacturing footprint associated with replacement parts that would otherwise have been swapped out before the end of their useful life. While a full life cycle assessment is beyond the scope of this paper, these considerations suggest that the environmental case for predictive maintenance is directionally aligned with, and likely reinforces, the financial case presented above.

14.4 Scalability of the Economic Case

The payback estimate presented in Table 14.1 is calculated for a ten turbine farm, and it is worth noting explicitly how the underlying cost structure is expected to scale to both smaller and larger deployments. The dominant fixed cost component, the shared time series database and n8n automation server described in Section IV, does not scale linearly with the number of turbines, since a single self hosted server of the specification described in Table 10.1 is capable of supporting a farm considerably larger than ten turbines before additional server capacity is required. This implies that the per turbine capital cost, and correspondingly the payback period, should improve for larger farms and worsen somewhat for very small deployments of only one or two turbines, where the fixed server and integration cost is spread across fewer generating assets; operators at the

smaller end of this range may find a shared, multi farm hosting arrangement, in which a single n8n server and database instance serves several nearby farms under common ownership, a more economically attractive starting point than a dedicated per farm deployment.

XV. SECURITY, DATA GOVERNANCE AND RELIABILITY CONSIDERATIONS

15.1 Threat Model

The framework described in this paper spans several traditionally separate domains, namely operational technology in the form of turbine sensors and control systems, information technology in the form of the time series database, automation server and dashboard, and, through the CMMS and email and SMS integrations, general enterprise systems. This convergence is precisely what gives the framework its practical value, but it also expands the attack surface relative to a turbine that is monitored using an isolated, air gapped SCADA system alone, and must therefore be considered explicitly rather than left as an afterthought.

15.2 Network Segmentation and Access Control

The reference deployment described in this paper places the edge gateways, the time series database and the n8n automation server on a segregated operational network, separated from the corporate information technology network and from the public internet by a firewall that permits only the specific, narrowly scoped outbound connections required by the workflows described in Section VII, namely the forecasting service endpoint, the CMMS ticketing API and the chosen email and SMS gateway. Inbound access to the n8n editor interface, which allows workflows to be viewed and modified, is restricted to authenticated users on an internal management network or through a virtual private network, rather than being exposed directly to the internet, and multi factor authentication is recommended for all accounts with editing rights given that a compromised workflow could, in principle, be used to suppress genuine alerts or to issue false ones.

15.3 Data Governance

Sensor and maintenance data handled by the framework, while not personally identifiable in the way that customer data would be, is nonetheless commercially sensitive, since it reveals the operational and financial performance of the wind farm. Access to the historical database and to the dashboard described in Section IV is therefore role based, with maintenance technicians able to view detailed subsystem level data for the turbines they are responsible for, and higher level fleet summaries reserved for planning and management roles. Data retention policies, implemented through the time series database's native downsampling and retention functionality, keep high frequency raw data for a limited hot period sufficient for near term diagnostic review, while retaining downsampled daily summaries indefinitely to support the long term trend analysis illustrated in Figure 8.2.

15.4 Reliability and Fail Safe Behaviour

Finally, because the automation layer is involved in triggering curtailment advisories that interact, even indirectly, with grid balancing decisions, the framework is designed so that its failure modes are conservative. Where the forecasting or health scoring service is unavailable, as detected by the error handling logic described in Section VII.7, the workflow does not attempt to infer a substitute forecast or health index, it instead falls back to the turbine's existing SCADA level protection logic, which remains active independently of the framework described in this paper, and raises an infrastructure alert rather than a turbine health alert. This design choice reflects a general principle that an automation layer built to augment safety relevant decisions should degrade to a known, previously validated fallback rather than to an unvalidated default when a component of the pipeline is unavailable.

15.5 Regulatory and Compliance Considerations

Operators deploying the framework described in this paper across jurisdictions with differing grid codes and data protection regimes should note that the specific requirements around curtailment advisory signalling to a grid operator, and around the retention and export of operational data, vary considerably by jurisdiction and by the terms of the operator's own interconnection agreement. The modular design described in Section IV is intended to make such jurisdiction specific adaptation feasible without redesigning the framework from first principles, since the grid advisory workflow described in Section VII.5 is deliberately isolated from the maintenance workflows and can be reconfigured, or disabled entirely where a given grid operator does not accept externally generated advisory signals, without affecting the turbine level predictive maintenance functionality described in Sections VII.3 and VII.4.

XVI. CHALLENGES AND LIMITATIONS

Several limitations of the present study should be acknowledged. First, the condition monitoring and meteorological data used in the experiments were synthetically generated to reproduce statistically realistic patterns, since access to proprietary

operator SCADA and failure archives was not available; while the data generation process was informed by published statistics from the wind turbine reliability literature, results obtained on real operator data may differ, and validation on field data from an operating wind farm is an important next step before the framework is deployed commercially.

Second, the grid frequency simulation presented in Section XII is a simplified representation intended to illustrate the direction and approximate magnitude of the benefit of predictive curtailment advisory information, rather than a full electromechanical grid dynamic model incorporating governor response, inertia and multiple interacting generation sources; a more detailed grid simulation, potentially incorporating tools purpose built for power system dynamic analysis, would strengthen the grid stability claims made in this paper.

Third, while n8n was selected for its low code accessibility and self hosting capability, any workflow automation platform introduces a dependency on the continued maintenance and security patching of that platform; operators adopting the framework described here should budget for ongoing platform maintenance in the same way they would for any other piece of operational software, and should apply standard industrial control system security practices, including network segmentation between the automation layer and any internet facing services, given that the workflow has access to both sensor data and maintenance and grid systems.

Fourth, the composite health index and fault classifier described in Section VIII rely on wind speed and load conditioned baselines that must be established from a period of nominal operation for each turbine; newly commissioned turbines, or turbines that have undergone a major component replacement, will therefore have a period during which the health index is less reliable until a sufficient nominal baseline has been accumulated, a limitation common to condition based monitoring approaches generally and one that operators should account for in their maintenance planning during the commissioning period.

Finally, the alert threshold and feature weighting used in the health index calculation in this study were set using illustrative values rather than being optimised against a large corpus of confirmed failure outcomes; site specific and fleet specific calibration, ideally informed by the operator's own historical maintenance records, is recommended before the thresholds proposed here are relied upon operationally.

More broadly, the reader should bear in mind that every quantitative result reported in Sections XI through XIV, including the downtime reduction, cost index and grid frequency deviation figures, was obtained through simulation calibrated against published statistical characteristics of wind turbine operation, rather than through a field trial on an operating wind farm. Simulation of this kind is a standard and useful step for evaluating a proposed system architecture before committing to a costly field deployment, and the individual components of the framework, namely the sensors, the forecasting models and the n8n automation platform, are all separately well established technologies rather than novel or unproven ones. Nonetheless, the specific numerical outcomes reported in this paper should be read as an estimate of the direction and plausible scale of benefit achievable, to be confirmed and refined through the field validation identified as the first priority for future work in Section XVII, rather than as a guarantee of performance on any particular operating wind farm.

XVII. FUTURE WORK

Building on the results and limitations discussed above, several directions for future work are identified. Validation of the framework on field data from an operating wind farm, including confirmed failure records, would allow the health index thresholds and fault classifier to be calibrated and validated against ground truth rather than synthetic fault injection. Extending the grid stability analysis to a full electromechanical simulation, incorporating multiple generation sources and a representative distribution network, would provide a more rigorous quantification of the grid level benefit of the predictive curtailment advisory workflow. Incorporating probabilistic forecasting, in which the model outputs a predictive distribution rather than a single point forecast, would allow the maintenance and grid advisory decision logic to explicitly account for forecast uncertainty rather than treating the point forecast as certain. Finally, extending the n8n workflow library described in this paper into an openly published, reusable template, together with a reference Docker Compose deployment covering the time series database, forecasting service and n8n server, would lower the barrier to adoption for small and medium wind farm operators seeking to replicate the framework described here.

A further direction concerns learning across, rather than only within, individual wind farms. The forecasting and health scoring models described in Sections VI and VIII are, in the present study, trained on data from a single simulated farm, but many operators manage portfolios of farms across different sites and climates. A federated learning approach, in which model updates rather than raw sensor data are shared across farms, could allow the forecasting and fault classification models to benefit from the combined experience of a larger fleet while respecting the data governance boundaries between farms, and potentially between different operating companies, discussed in Section XV.3. Evaluating whether such a federated approach meaningfully improves forecasting and diagnostic accuracy relative to the single farm models evaluated in this paper, and

quantifying the additional orchestration complexity it would introduce into the n8n based automation layer, is left as a direction for future research.

XVIII. CONCLUSION

This paper has presented an integrated framework combining a distributed IoT sensor network, a set of wind speed forecasting models culminating in a hybrid CNN LSTM architecture, and an n8n based low code automation layer, designed to deliver predictive maintenance for wind turbines while also improving the information available to grid operators for balancing purposes. The hybrid forecasting model achieved the lowest error among the four models evaluated, and, when embedded in the described automation workflow, contributed to a simulated reduction in annual unplanned downtime from 186 to 47 hours and a reduction in relative maintenance cost index from 312 to 141, relative to a purely reactive maintenance baseline, together with an approximately two thirds reduction in simulated grid frequency deviation amplitude when predictive curtailment advisory information was made available to the grid operator ahead of anticipated generation changes.

The central contribution of this work relative to prior literature is not any single forecasting or fault detection technique in isolation, but the explicit, node level description of how such techniques can be wired together into an operational pipeline using an accessible, low code automation platform, together with a quantification of the combined effect of that pipeline on both turbine level maintenance outcomes and grid level stability outcomes. The authors hope that this combination of accessible tooling and end to end system description will support broader adoption of predictive maintenance practices among small and medium wind farm operators, contributing in turn to the reliability and grid friendliness of wind energy as its share of the overall generation mix continues to grow.

In closing, it is worth reiterating that the five workflow implementation described in Section VII, comprising thirty eight nodes spanning scheduled, webhook triggered and error driven execution, together with explicit AI powered decision points, a human approval gate and a centralised audit trail, is offered not as the only possible way to structure such a system, but as one concrete, fully specified reference design against which alternative designs can be compared. The authors believe that the value of publishing this level of implementation detail, ordinarily omitted from papers that focus on forecasting or diagnostic accuracy alone, lies precisely in making the practical engineering choices involved in predictive maintenance automation, and their consequences for auditability, resilience and operator accessibility, available for scrutiny, replication and improvement by the wider research and operator community.

REFERENCES

[1] W. Yang, P. J. Tavner, C. J. Crabtree, Y. Feng, and Y. Qiu, "Wind turbine condition monitoring: technical and commercial challenges," Wind Energy, vol. 17, no. 5, pp. 673-693, 2014.
[2] Z. Hameed, Y. S. Hong, Y. M. Cho, S. H. Ahn, and C. K. Song, "Condition monitoring and fault detection of wind turbines and related algorithms: a review," Renewable and Sustainable Energy Reviews, vol. 13, no. 1, pp. 1-39, 2009.
[3] Y. Feng, Y. Qiu, C. J. Crabtree, H. Long, and P. J. Tavner, "Monitoring wind turbine gearboxes," Wind Energy, vol. 16, no. 5, pp. 728-740, 2013.
[4] A. Kusiak and W. Li, "The prediction and diagnosis of wind turbine faults," Renewable Energy, vol. 36, no. 1, pp. 16-23, 2011.
[5] A. Zaher, S. D. J. McArthur, D. G. Infield, and Y. Patel, "Online wind turbine fault detection through automated SCADA data analysis," Wind Energy, vol. 12, no. 6, pp. 574-593, 2009.
[6] J. Tautz-Weinert and S. J. Watson, "Using SCADA data for wind turbine condition monitoring: a review," IET Renewable Power Generation, vol. 11, no. 4, pp. 382-394, 2017.
[7] L. Cao, Z. Qian, H. Zareipour, Z. Huang, and F. Zhang, "Fault diagnosis of wind turbine gearbox based on deep bi-directional long short-term memory under time-varying non-stationary operating conditions," IEEE Access, vol. 7, pp. 155219-155228, 2019.
[8] R. Liu, B. Yang, E. Zio, and X. Chen, "Artificial intelligence for fault diagnosis of rotating machinery: a review," Mechanical Systems and Signal Processing, vol. 108, pp. 33-47, 2018.
[9] G. Giebel and G. Kariniotakis, "Wind power forecasting: a review of the state of the art," in Renewable Energy Forecasting, Woodhead Publishing, pp. 59-109, 2017.
[10] E. Cadenas and W. Rivera, "Wind speed forecasting in three different regions of Mexico, using a hybrid ARIMA-ANN model," Renewable Energy, vol. 35, no. 12, pp. 2732-2738, 2010.
[11] H. Liu, H. Q. Tian, and Y. F. Li, "Comparison of two new ARIMA-ANN and ARIMA-Kalman hybrid methods for wind speed prediction," Applied Energy, vol. 98, pp. 415-424, 2012.

[12] H. Wang, Z. Lei, X. Zhang, B. Zhou, and J. Peng, "A review of deep learning for renewable energy forecasting," Energy Conversion and Management, vol. 198, art. 111799, 2019.
[13] n8n GmbH, "n8n Documentation," [Online]. Available: https://docs.n8n.io/. [Accessed: Jul. 2026].
[14] S. Nastic, S. Sehic, D. H. Le, H. L. Truong, and S. Dustdar, "Provisioning software defined IoT cloud systems," in Proc. International Conference on Future Internet of Things and Cloud, 2014.
[15] T. Ahmad, R. Madonski, D. Zhang, C. Huang, and A. Mujeeb, "Data-driven probabilistic machine learning in sustainable smart energy/smart energy systems: key developments, challenges, and future research opportunities in the context of smart grid paradigm," Renewable and Sustainable Energy Reviews, vol. 160, art. 112128, 2022.
[16] E. A. Lee, "Cyber physical systems: design challenges," in Proc. IEEE International Symposium on Object Oriented Real Time Distributed Computing, 2008.
[17] L. Da Xu, W. He, and S. Li, "Internet of things in industries: a survey," IEEE Transactions on Industrial Informatics, vol. 10, no. 4, pp. 2233-2243, 2014.
[18] P. Tchakoua, R. Wamkeue, M. Ouhrouche, F. Slaoui-Hasnaoui, T. A. Tameghe, and G. Ekemb, "Wind turbine condition monitoring: state of the art review, new trends, and future challenges," Energies, vol. 7, no. 4, pp. 2595-2630, 2014.
[19] J. Nilsson and L. Bertling, "Maintenance management of wind power systems using condition monitoring systems: life cycle cost analysis for two case studies," IEEE Transactions on Energy Conversion, vol. 22, no. 1, pp. 223-229, 2007.
[20] E. Byon, L. Ntaimo, and Y. Ding, "Optimal maintenance strategies for wind turbine systems under stochastic weather conditions," IEEE Transactions on Reliability, vol. 59, no. 2, pp. 393-404, 2010.
[21] A. Karim, S. Westerberg, D. Galar, and U. Kumar, "Maintenance analytics, the new know in maintenance," IFAC-PapersOnLine, vol. 49, no. 28, pp. 214-219, 2016.
[22] A. C. Bock and U. Frank, "Low-code platform," Business & Information Systems Engineering, vol. 63, no. 6, pp. 733-740, 2021.
[23] F. P. G. Marquez, A. M. Tobias, J. M. Pinar Perez, and M. Papaelias, "Condition monitoring of wind turbines: techniques and methods," Renewable Energy, vol. 46, pp. 169-178, 2012.
[24] H. Zhao, H. Liu, W. Hu, and X. Yan, "Anomaly detection and fault analysis of wind turbine components based on deep learning network," Renewable Energy, vol. 127, pp. 825-834, 2018.
[25] M. Schlechtingen, I. F. Santos, and S. Achiche, "Wind turbine condition monitoring based on SCADA data using normal behaviour models," Applied Soft Computing, vol. 13, no. 1, pp. 259-270, 2013.
[26] J. P. Morrison, Flow-Based Programming: A New Approach to Application Development, 2nd ed. Charleston, SC: CreateSpace, 2010.

APPENDIX A: N8N WORKFLOW AUTOMATION, THEORETICAL BACKGROUND

C.1 The Workflow Automation Paradigm

n8n belongs to a broader family of tools generally described as workflow automation or integration platforms, whose common purpose is to let a user connect otherwise independent systems, an API, a database, a messaging service, a file store, so that data or events flowing from one system automatically trigger an action in another, without the user writing and maintaining a full custom application to do so. The conceptual roots of this paradigm trace back to flow based programming, a model of computation in which an application is expressed as a network of black box processes that exchange data across predefined connections, a formulation originally articulated by J. Paul Morrison and later popularised in the context of visual, node based tools rather than the mainframe batch systems for which it was first developed [26]. n8n, together with related tools discussed in Section C.4, is a direct descendant of this tradition, replacing hand written glue code with a visual graph of nodes and connections that can be inspected, modified and executed without a separate compilation step.

C.2 Node Based Execution Model

An n8n workflow is internally represented as a directed graph, in which each node performs a discrete unit of work, retrieving data, transforming it, evaluating a condition, or dispatching an action, and each connection defines how the output items produced by one node are passed as input items to the next. Execution proceeds from one or more trigger nodes, described in Section C.3, along the graph, with n8n's runtime engine responsible for invoking each node in the correct order, passing data between nodes as structured JSON items, and handling any branching introduced by IF or Switch nodes. Because each node's input and output are both well defined JSON structures, nodes are highly composable, a node built to call a REST API can be connected to the output of any upstream node without modification, provided the data it receives contains the fields it expects, which is the same compositional property that gives flow based programming its reusability advantage over monolithic scripts.

Within a node, custom logic that goes beyond what a pre built node type offers can be expressed using a Function node, which executes user supplied JavaScript against the current set of items, or using n8n's built in expression syntax, a lightweight

templating language, denoted with double curly braces, that allows a node parameter to reference the output of any preceding node directly, for example {{$node["Call Health Scoring API"].json["healthIndex"]}}, without requiring a separate variable assignment step. This expression language is what allows the workflows described in Section VII to pass the turbine identifier, health index and fault classification from Workflow 1 through to the Set and HTTP Request nodes in Workflow 2 without an intermediate database write.

C.3 Trigger Types and Execution Modes

Every n8n workflow begins execution at one or more trigger nodes, of which four types are used across the five workflows described in Section VII. A Cron trigger fires on a fixed schedule, used in Workflows 1, 3 and 4 for their five minute, fifteen minute and monthly cadences respectively. A Webhook trigger exposes an HTTP endpoint that starts execution when called, used in Workflow 2, which is invoked directly by Workflow 1, and in the acknowledgement path of Workflow 3. An Error trigger is a special workflow level trigger that fires automatically whenever another workflow, configured to designate it as its error workflow, encounters an unhandled exception, which is how Workflow 5 receives control in Section VII.7. n8n also supports a Manual trigger, used during development and testing to execute a workflow on demand from the editor, which was used extensively during the design and validation of the workflows described in this paper before their scheduled and webhook triggers were enabled in the reference deployment.

For production deployments handling the execution volumes discussed in Section XI.7, n8n can run in one of two modes. In main process mode, a single n8n instance both serves the editor interface and executes all workflow runs, which is adequate for the moderate execution volume of the farm scale deployment described in this paper. In queue mode, workflow executions are distributed across a pool of separate worker processes coordinated through a message queue, allowing throughput to scale roughly linearly with the number of workers, a configuration recommended by n8n's own deployment documentation for installations expecting sustained execution volumes materially higher than those reported in Section XI.7, and one that a wind farm operator scaling the framework described in this paper to a substantially larger fleet, or to a multi farm shared deployment as discussed in Section XIV.4, should plan to adopt.

C.4 Licensing Model and Comparable Tools

n8n is distributed under a fair code license, referred to by its publisher as the Sustainable Use License, which permits free self hosted use, including for internal business purposes such as the predictive maintenance framework described in this paper, while restricting the resale of the software itself as a competing hosted service; this licensing position is distinct both from fully permissive open source licenses and from closed, purely proprietary commercial software, and was a specific factor in its selection for this study, since it allows the operator to inspect, self host and modify the platform's own source code, while still supporting continued commercial development by its publisher. This distinguishes n8n from two categories of comparable tool. The first category, exemplified by Node RED, an Apache 2.0 licensed, fully open source flow based programming tool originally developed at IBM and now maintained under the OpenJS Foundation, offers a similar visual, node based canvas oriented primarily toward Internet of Things and hardware integration scenarios, but with a narrower built in library of enterprise and cloud service integrations than n8n provides out of the box. The second category, exemplified by hosted, closed source integration platforms such as Zapier and Make, offers a comparable visual workflow building experience but only as a cloud hosted service, without a self hosting option, which would conflict with the data governance and network segmentation requirements described in Section XV for a framework that handles operationally sensitive turbine and grid data. Table C.1 summarises this comparison across the dimensions most relevant to the framework described in this paper.

Table C.1. Comparison of n8n with Node RED and closed, cloud only automation platforms

Dimension	n8n	Node RED	Zapier / Make
Underlying paradigm	Node based, flow based programming lineage	Node based, flow based programming lineage	Node based, proprietary workflow model
License	Fair code (Sustainable Use License)	Apache 2.0 (fully open source)	Closed, proprietary
Self hosting	Yes	Yes	No (cloud only)
Built in integrations	400+ pre built nodes	Smaller core library, extensible via community nodes	Very large, cloud service focused
Custom code within a node	JavaScript (Function node) or Python	JavaScript (Function node)	Limited (varies by plan)
Execution scaling model	Main process or queue mode with workers	Single process (external scaling patterns required)	Managed by vendor

Dimension	n8n	Node RED	Zapier / Make
Typical target domain	General IT, IIoT and business process automation	IoT, hardware and embedded integration	SaaS and marketing operations automation

C.5 Auditability and Security Posture

Because the framework described in this paper embeds n8n directly in a pipeline that triggers maintenance work orders and grid curtailment advisories, the platform's own auditability and security posture is directly relevant to the reliability and governance considerations discussed in Section XV. Every workflow execution in n8n is recorded with its full input and output data at each node, providing exactly the node level execution trace that underlies the audit trail requirement described in Section VII.7 and Section III.3, without requiring the workflows themselves to implement custom logging for this purpose. Access to the editor, to stored credentials such as the API keys used by the HTTP Request nodes described in Section VII, and to execution history can be restricted using role based access control, consistent with the access control approach described in Section XV.3. These platform level guarantees are what allow the framework in this paper to treat n8n not merely as a convenient scripting replacement, but as a governance relevant component of the overall system architecture in its own right.

APPENDIX B: SAMPLE N8N WORKFLOW DEFINITION

This appendix provides an illustrative, abbreviated excerpt of the n8n workflow definition underlying Workflow 1, described in Section VII.3, expressed in the JSON format that n8n uses natively for workflow export and version control. Node parameter values have been simplified and connection details condensed for readability; the excerpt is intended to make concrete the level of configuration involved in each node, rather than to serve as a directly importable workflow file.

```
{
  "name": "W1 - Real Time Condition Monitoring",
  "nodes": [
    { "name": "Cron Trigger", "type": "n8n-nodes-base.cron",
      "parameters": { "triggerTimes": { "item": [{ "mode": "everyX", "unit":
"minutes", "value": 5 }] } } },
    { "name": "Pull Sensor Batch", "type": "n8n-nodes-base.httpRequest",
      "parameters": { "method": "GET", "url":
"https://influx.internal/query?farm=all&window=5m" } },
    { "name": "Loop Per Turbine", "type": "n8n-nodes-base.splitInBatches",
      "parameters": { "batchSize": 1 } },
    { "name": "Clean and Normalise", "type": "n8n-nodes-base.function",
      "parameters": { "functionCode": "// rolling median filter, z-score
normalisation" } },
    { "name": "Call Forecasting API", "type": "n8n-nodes-base.httpRequest",
      "parameters": { "method": "POST", "url": "https://forecast.internal/predict"
} },
    { "name": "Call Health Scoring API", "type": "n8n-nodes-base.httpRequest",
      "parameters": { "method": "POST", "url": "https://health.internal/score" } },
    { "name": "Anomaly Check", "type": "n8n-nodes-base.if",
      "parameters": { "conditions": { "number": [{ "value1":
"={{$json[\"healthIndex\"]}}", "operation": "smaller", "value2": 60 }] } } },
    { "name": "Log Nominal State", "type": "n8n-nodes-base.function",
      "parameters": { "functionCode": "// append summary record to health index
table" } },
    { "name": "Call Workflow 2 (Webhook)", "type": "n8n-nodes-base.httpRequest",
      "parameters": { "method": "POST", "url": "https://n8n.internal/webhook/w2-
approval" } }
  ],
  "connections": {
    "Cron Trigger": { "main": [[{ "node": "Pull Sensor Batch" }]] },
    "Pull Sensor Batch": { "main": [[{ "node": "Loop Per Turbine" }]] },
```

```
    "Loop Per Turbine": { "main": [[{ "node": "Clean and Normalise" }]] },
    "Clean and Normalise": { "main": [[{ "node": "Call Forecasting API" }]] },
    "Call Forecasting API": { "main": [[{ "node": "Call Health Scoring API" }]] },
    "Call Health Scoring API": { "main": [[{ "node": "Anomaly Check" }]] },
    "Anomaly Check": { "main": [[{ "node": "Call Workflow 2 (Webhook)" }], [{
"node": "Log Nominal State" }]] }
  }
}
```

Each of the remaining four workflows follows the same general JSON structure, an ordered list of typed nodes together with a connections object describing the directed edges between them, with the specific node types and parameters varying according to the responsibilities described in Sections VII.4 through VII.7. Exporting workflows in this form allows them to be stored in a standard source control system alongside the forecasting and health scoring service code, so that changes to the automation logic can be reviewed, versioned and rolled back using the same engineering discipline applied to the rest of the codebase, addressing the auditability consideration raised in Section II.

APPENDIX C: GLOSSARY OF ABBREVIATIONS

Table C.2. Glossary of abbreviations used in this paper

Abbreviation	Full Form
ARIMA	Autoregressive Integrated Moving Average
CMMS	Computerised Maintenance Management System
CNN	Convolutional Neural Network
IIoT	Industrial Internet of Things
IoT	Internet of Things
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MQTT	Message Queuing Telemetry Transport
NWP	Numerical Weather Prediction
PDM	Predictive Maintenance
RMSE	Root Mean Square Error
RUL	Remaining Useful Life
SCADA	Supervisory Control and Data Acquisition
SVR	Support Vector Regression