WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

AI with Digital Twin for Predictive Maintenance in Industrial Motors and Pumps

Dr. Baby Kalpana Y[1], Sanjayram M[2], Sarveswaran V A[3], Sanjaysree T[4], Priadharshan A K[5]

[1]Faculty Guide, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.
Email: babykalpanacse@siet.ac.in
[2-5] Student(s), Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.
Email:2)sanjayramm25cs@srishakthi.ac.in, 3)priadharsanak25cs@srishakthi.ac.in, 4)sarveswaranva25cs@srishakthi.ac.in, 5)sanjaysreet25cs@srishakthi.ac.in

ABSTRACT

This paper presents a software prototype of an industrial digital twin for a coupled induction motor and centrifugal pump assembly. The system combines an interactive Three.js model, a Python FastAPI service, scikit-learn classifiers for machine-level condition assessment, and an explainable rule-based layer for component-level diagnosis. Separate motor and pump classifiers label the operating condition as Normal, Warning or Critical; component rules combine telemetry symptoms into severity, supporting evidence, a confidence label and a maintenance recommendation, which are linked to selectable 3D components. Telemetry is produced either by a built-in coupled simulator with fault-injection scenarios or by an ESP32 firmware module executed in the Wokwi simulator, which posts JSON over HTTP. The classifiers are trained on synthetic datasets and all telemetry in the reported work is simulated, so no physical-sensor validation and no real-world predictive accuracy is claimed. The paper documents the architecture, the ESP32 data path with its signal-conditioning equations, and the dashboard behaviour observed under a pump Warning condition, and it identifies the signal-range and unit discrepancies between the firmware and the dashboard. The work is presented as a reproducible demonstration of an architecture and interface, not as a validated industrial digital twin.

Keywords: *digital twin, condition monitoring, centrifugal pump, induction motor, machine learning, explainable diagnostics, fault injection, Three.js, FastAPI, ESP32, cyber-physical system.*

1. INTRODUCTION

Induction motors and centrifugal pumps form a common electromechanical pairing in industrial fluid handling. Their characteristic failure modes, such as bearing wear, winding overheating, shaft or rotor eccentricity, impeller wear, cavitation and casing or seal distress, tend to appear first in a small set of measurable quantities: temperature, vibration, current, speed, pressure and flow. Condition-based maintenance uses such measurements to decide when intervention is justified, and reviews of machinery diagnostics and prognostics organise the task into data acquisition, data processing and maintenance decision-making [5], [6].

A digital twin extends monitoring by maintaining a virtual counterpart of a physical asset that is updated from asset data [1], [2]. Surveys of industrial digital twins emphasise the integration of the physical and cyber spaces, data fusion and modelling [2], [3], and cyber-physical architectures describe a progression from sensing and data-to-information conversion towards cognition and configuration [4]. The term is applied to systems of very different maturity. The prototype reported here sits at the lower end of that range: it maintains a visual and diagnostic counterpart of a motor-pump skid that is fed by simulated telemetry, and it contains no calibrated physics model, no feedback to a physical asset and no validation against a real machine.

Two practical shortcomings motivate the work. First, a machine-level classifier reports that a motor or pump is abnormal but not which component is implicated or why, and a dashboard of numbers gives no spatial context for deciding where to inspect. Second, the motor and pump are mechanically and hydraulically coupled: a hydraulic fault such as dry running changes the load seen by the motor, and motor thermal or vibration stress may force the pump to be derated. These interactions are reflected in the coupled scenarios of the prototype (Section V-E).

The Industrial Pump and Motor Digital Twin combines (i) a Python FastAPI service that ingests telemetry and exposes the aggregated twin state, (ii) separate scikit-learn classifiers for the motor and the pump, (iii) an explainable rule-based layer that maps telemetry symptoms to individual components, and (iv) a Three.js browser model in which components can be selected and inspected beside their diagnostic status (Fig. 1). The prototype separates what it demonstrates from what it does not: the classifiers are trained on synthetic data, all telemetry is simulated, and no physical machine was instrumented.

The contributions are:

1. A Two-level diagnostic architecture in which machine-level ML classification and component-level rule-based diagnosis are separate stages, cross-checked by a threshold-based safety guardrail;
2. An Interactive 3D interface that ties component selection to diagnostic status, severity, evidence and recommendation;
3. A REST-driven coupled fault-injection mechanism with core and component-level scenarios;
4. A documented ESP32/Wokwi telemetry module, including its signal-conditioning equations and an explicit account of which signals are sensed and which are simulated; and
5. An analysis of the prototype's limits, including mismatches between the firmware and dashboard signal ranges.

The remainder of the paper states the problem and objectives (Section II), reviews related work (III), describes the architecture (IV), methodology (V), 3D visualization (VI) and implementation (VII), reports the verifiable results (VIII), and closes with discussion, limitations, future work and conclusions (IX–XII).

2. PROBLEM STATEMENT AND OBJECTIVES

A. Problem Statement

P1: Localisation. Machine-level condition labels do not identify the component responsible for an abnormal state. **P2: Opacity.** A classifier output without evidence cannot be audited by a maintenance engineer. **P3: Coupling.** Treating the pump and motor as independent monitored objects ignores the load and thermal interactions between them. **P4: Data availability.** Labelled fault data from industrial machines are often scarce and risky to produce, which motivates simulated faults for development, at the price that any model trained on them has not been shown to transfer to real equipment.

B. Objectives

The objectives are: (O1) acquire motor and pump telemetry from a built-in simulator and from an ESP32 module; (O2) classify motor and pump condition separately as Normal, Warning or Critical using scikit-learn models; (O3) produce explainable component-level diagnoses with severity, evidence, confidence label and recommendation; (O4) simulate coupled fault scenarios with a REST interface; (O5) present the state in an interactive 3D model whose components can be selected and inspected; and (O6) state clearly which parts of the system are simulated, synthetic or unverified.

C. Scope

The scope is a single motor-pump skid with three condition classes, evaluated through simulated telemetry. Prognostics (remaining useful life), closed-loop control and deployment on a physical machine are outside the scope.

3. EXISTING APPROACHES AND RELATED WORK

A. Condition-Based Maintenance and Diagnostics

Jardine *et al.* review machinery diagnostics and prognostics for condition-based maintenance, covering data acquisition, signal processing and decision support [5]. Lei *et al.* survey machinery health prognostics from data acquisition through health-indicator construction and health-stage division to remaining-useful-life prediction [6]. Fixed-threshold alarming remains the simplest form of monitoring and is easy to interpret, but a single signal threshold does not localise a fault. Vibration severity is commonly evaluated from band-limited vibration measurements on the machine structure, for which general guidance is given in ISO 20816-1 [9].

B. Digital Twins

Grieves and Vickers describe the digital twin as a virtual counterpart of a physical product maintained through its lifecycle [1]. Tao *et al.* review digital twins in industry and identify data fusion, modelling and the cyber-physical connection as central themes [2], and Fuller *et al.* discuss enabling technologies, challenges and open research questions [3]. Lee *et al.* describe a cyber-physical systems architecture for Industry 4.0 manufacturing in staged levels from connection to configuration [4]. These sources motivate the architecture of Section IV; this paper does not claim to extend them.

C. Machine Learning for Condition Classification

Supervised classifiers learn a mapping from features to condition labels; scikit-learn offers such estimators through a uniform programming interface [7], [12]. Their value depends on the representativeness and label quality of the training data.

When training data are synthetic, the model learns the assumptions embedded in the generator, which is why this paper does not treat synthetic-data accuracy as evidence of field performance.

D. Explainability and Rule-Based Diagnosis

Barredo Arrieta *et al.* distinguish models that are transparent by design from post-hoc explanation of opaque models [8]. Rule-based diagnosis belongs to the first group: the conclusion and the evidence that triggered it can be read directly. Its weakness is that thresholds and symptom combinations encode expert assumptions that must themselves be validated.

E. Enabling Technologies

Three.js is a JavaScript library for rendering 3D scenes in the browser [11]; FastAPI is a Python framework for building web APIs [10]; Wokwi simulates ESP32 boards and a virtual Wi-Fi network [13]; PlatformIO and the Arduino-ESP32 core provide the firmware build chain [14], [15]. These are software documentation sources rather than research results and are used here only to describe the tools.

F. Positioning

Table I compares typical characteristics of common approaches with the prototype. It is a qualitative positioning, not a benchmark; the prototype's entries describe its intended behaviour as observed in the interface and have not been validated on physical equipment.

TABLE I

QUALITATIVE COMPARISON OF APPROACHES

Approach	**Localises to components**	**Explains output**	**Models coupling**
Fixed-threshold alarms	No	Yes (threshold)	No
Stand-alone ML classifier	Typically no	Typically limited	Typically no
Model-based twin	Model-dependent	Model-dependent	Possible
This prototype	Rule-based, per component	Evidence text, confidence label	Simulated scenarios only

4. PROPOSED SYSTEM ARCHITECTURE

The system is organised in layers (Table II). Telemetry enters through a FastAPI service, is assessed at machine level by the scikit-learn classifiers and a threshold guardrail, is interpreted at component level by rules, and is presented in a browser dashboard that includes the 3D model. The header strip of the dashboard (Fig. 1) exposes the twin state: the overall twin state, the last ingested timestamp, the API gateway endpoint (GET /twin), the telemetry source (displayed as SIMULATED in the captures) and the status of the predictive engine ("scikit-learn ML + component rules").

TABLE II ARCHITECTURE LAYERS

Layer	**Technology**	**Role**
Telemetry sources	ESP32 firmware (Wokwi); built-in coupled simulator	Supply motor and pump telemetry
Ingestion and state API	Python, FastAPI	POST /telemetry, POST/telemetry/simulate, GET/twin
Machine-level assessment	scikit-learn	Motor and pump Normal / Warning / Critical with confidence
Safety guardrail	Threshold comparison	Cross-checks ML class against fixed thresholds
Component diagnostics	Rule-based logic	Severity, evidence, confidence label, recommendation per component
Presentation	Browser dashboard, Three.js	3D model, subsystem panels, scenario controls

The data flow is: (1) telemetry is produced by the ESP32 module or by a scenario injected through POST /telemetry/simulate; (2) the backend records the sample and its timestamp; (3) the machine-level models classify the motor and pump separately; (4) the guardrail compares key signals with fixed thresholds and reports whether it agrees with the ML class; (5) the component rules evaluate symptom combinations and emit diagnoses; and (6) the dashboard retrieves the aggregated state and renders panels and the 3D scene.

Three design principles follow from the objectives. *Separation of levels:* statistical classification and rule-based explanation

are different stages so that each can be inspected and tested on its own. *Deterministic floor:* a threshold guardrail runs beside the ML model so that an obviously out-of-range value does not depend solely on a learned decision. *Declared provenance:* the telemetry source is displayed, and the simulator carries an on-screen disclaimer.

5. METHODOLOGY

A. Telemetry Acquisition

The monitored parameters are temperature, vibration (RMS), current and rotational speed for the motor, and housing temperature, vibration (RMS), current, rotational speed, discharge pressure and flow rate for the pump. Table III relates each dashboard quantity to its source in the ESP32 module. The module is a signal-path demonstrator: only temperature and acceleration come from simulated sensor parts, current and flow come from potentiometers, and speed and pressure are computed constants or functions.

TABLE III

DASHBOARD QUANTITIES AND ESP32 SOURCES

Quantity	**Dashboard unit**	**ESP32 module source**
Temperature (motor, pump)	°C	DS18B20, GPIO 4; one reading sent to both machines
Vibration (motor, pump)	mm/s (RMS)	MPU6050 dynamic acceleration in m/sZ, Eq. (1); one value sent to both machines
Current	A	Potentiometer, GPIO 34, 0–10 A, Eq.(2)
Speed	RPM	Simulated, 1450 or 0, Eq. (4)
Discharge pressure	bar	Simulated from flow, Eq. (5)
Flow rate	m3/h	Potentiometer, GPIO 35, 0–15 L/min, converted, Eq. (3)

The pump panel of Fig. 2 displays housing temperature, vibration, discharge pressure and flow rate; the JSON payload additionally carries pump current and speed. The firmware conditions the signals as follows. Dynamic acceleration is taken as the deviation of the acceleration magnitude from the gravity baseline:

$a\text{dyn} = |\sqrt{(a\text{x}Z + a\text{y}Z + a\text{z}Z)} - g|,\ g = 9.81\ \text{m/sZ}$ (1)

The source code notes that this is an approximation and not a calibrated industrial vibration RMS measurement. The 12-bit

ADC counts N_{34} and $N35$ are scaled linearly to current in amperes and flow in litres per minute, and flow is converted to cubic metres per hour in the payload (1 L/min = 0.06 m^3/h):

$$I = 10 \cdot N_{34} / 4095 \quad (2)$$

$$QL = 15 \cdot N35 / 4095,\ \ Q = 0.06\ QL \quad (3)$$

No speed or pressure sensor exists in the circuit, so both are simulated. With the Iverson bracket [·] equal to 1 when its condition holds and 0 otherwise:

$$n = 1450 \cdot [I > 0.2 \text{ and } QL > 0.2]\ \text{RPM} \quad (4)$$

$$p = [QL > 0.2] \cdot (1 + 2\ QL / 15)\ \text{bar} \quad (5)$$

Telemetry is sent as a JSON POST to http://host.wokwi.internal:8000/telemetry over the Wokwi-GUEST network. The payload has a motor object (temperature, vibration, current, rpm) and a pump object (temperature, vibration, current, rpm, pressure, flow_rate). A transmission is attempted when at least 1.5 s have elapsed since the previous attempt; because this test runs once per loop iteration and the loop ends with a 1 s delay, the effective period is about two loop iterations (roughly 2 s or more). Connection and response timeouts are 1.5 s each; if Wi-Fi is down the sample is skipped and a reconnect is requested.

B. Data Processing

On the device, each loop iteration also computes a local health indicator from four fixed thresholds, independently of the backend:

$H = \max(0, 100 - 20\ \Sigma\ ck),\ k = 1\ldots4$ (6)

where $c1 = 1$ if $T > 60$ °C, $c2 = 1$ if $a\text{dyn} > 5$ m/sZ, $c3 = 1$ if $I > 7$ A, $c4$

= 1 if $Q\text{L} < 4$ L/min, and 0 otherwise. The local status is NORMAL for $H \geq 70$, WARNING for $40 \leq H < 70$ and CRITICAL otherwise. Since at most four indicators can be active, H ranges from 20 to 100 and the zero clamp never operates: no exceedance gives 100 (NORMAL), one gives 80 (NORMAL),

two give 60 and three give 40 (both WARNING), and four give 20 (CRITICAL). The value is shown on the 128×64 OLED and the serial monitor (115200 baud) and is not part of the transmitted payload.

On the backend, ingestion by POST /telemetry is followed by assembly of per-machine inputs, model inference, guardrail comparison, component rules and aggregation into the state returned by GET /twin. The validation, unit handling, scaling and storage performed in these steps are not described .

C. Machine-Learning Condition Assessment

Two separate supervised classifiers, one for the motor and one for the pump, are implemented with scikit-learn [7], [12]. Each maps the feature vector of its machine to a class ŷ in {Normal, Warning, Critical}:

$$\hat{y}m = fm(\mathbf{x}m),\ \hat{y}p = fp(\mathbf{x}p) \qquad (7)$$

The monitored parameters suggest four motor inputs and six pump inputs, but the exact feature vectors, scaling and learning algorithm have not been confirmed (Table IV). Separate models are used because the machines differ in signals and failure modes; in particular, only the pump has hydraulic quantities. Each prediction is shown with a confidence percentage (92 % for the motor and 100 % for the pump in Fig. 2). How this value is computed is unconfirmed, and it must not be read as a calibrated probability of correctness. Each subsystem also shows a health index (98.5 % and 70.0 % in Fig. 2) against a displayed target of >80 %; its formula is unconfirmed and it is treated as a display heuristic.

Synthetic training data. The classifiers are trained on synthetic datasets generated during development. Synthetic data allow the three classes to be populated deliberately but encode the designer's assumptions about normal and faulty telemetry. If labels are assigned by the same threshold logic used to generate the samples, high held-out accuracy mainly shows that the classifier has reproduced that logic. Accuracy, precision or recall measured on such data therefore says nothing about performance on a physical pump or motor, and no such figure is reported here.

Safety guardrail. Alongside each ML output the interface prints a guardrail message. For the pump in Fig. 2 it reads that the ML model and the guardrail are concordant (Warning) because the temperature of 78.5 °C is at or above the displayed warning threshold of 65.0 °C. The guardrail is therefore a fixed-threshold check that reports agreement or disagreement with the learned class. How a disagreement is resolved is not shown in the supplied materials [].

TABLE IV

DOCUMENTATION STATUS OF THE ML CONFIGURATION

Item	**Status**
Learning algorithm (motor, pump)	Confirm algorithm from backend code
Feature vectors and scaling	Motor: 4 inputs; Pump: 6 inputs (verify)
Synthetic data generator, sample counts, class balance	Confirm implementation and sample counts
Labelling rule	Normal/Warning/Critical
Training / validation protocol	Confirm actual split
Confidence and health-index computation	Confirm implemented formula
Reported accuracy	Not reported (synthetic data)

D. Component-Level Diagnostics

Machine-level classes indicate that a machine is abnormal but not where to look. The component layer therefore applies explainable rules that combine several telemetry symptoms.

For each component c the output can be represented as

$$dc = (sc, Ec, \kappa c, rc) \qquad (8)$$

where sc is the severity (Normal, Warning or Critical), Ec the supporting evidence (the telemetry values that satisfied the rule), κc a categorical confidence label and rc a maintenance recommendation. This tuple abstracts the outputs visible in the interface; it is not a transcription of the code. The covered components are the motor housing and frame, stator windings, rotor assembly, drive shaft, drive-end and non-drive-end bearings, and the pump casing, impeller, mechanical seal, bearings, and shaft and related parts (Table V).

Fig. 1 shows one such diagnosis. The pump bearings are in Warning with the title "Pump Bearing Overheating & Mechanical Stress"; the evidence is a pump vibration of 6.20 mm/s together with a bearing housing temperature of 78.5 °C; the text lists bearing cage distress, lubrication degradation and excessive axial thrust as possible causes; the confidence label is High; and the panel header states that one warning condition was detected and that component inspection is recommended. The induction motor subsystem is shown as Normal. Because rules combine symptoms, a single out-of-range signal does not by itself implicate a component, and an injection scenario named "Motor Vibration Ambiguous" exercises the ambiguous case. The confidence label is categorical and is not a statistical probability. The thresholds and combination logic have not been compared with manufacturer limits or with ISO 20816-1 guidance [9] in the supplied materials, so they are treated as engineering heuristics.

TABLE V

COMPONENTS AND RELATED INJECTION SCENARIOS

Unit	Component	Related scenario in Fig. 2 (by name)
Motor	Housing and frame	None dedicated
Motor	Stator windings	Motor Stator Overheat (Critical)
Motor	Rotor assembly; drive shaft	Motor Rotor / Shaft Eccentricity (Warning)
Motor	Drive-end bearing	Motor DE Bearing Wear (Warning)
Motor	Non-drive-end bearing	None visible
Motor	Cause-ambiguous vibration	Motor Vibration Ambiguous (Warning)
Pump	Casing	Pump Deadhead / Casing Pressure (Critical)
Pump	Impeller	Pump Impeller Recirculation / Wear (Warning)
Pump	Mechanical seal	None visible
Pump	Bearings	Pump Bearing Thermal Distress (Critical)
Pump	Shaft and related parts	None visible

Association is by scenario name only; the component rules were not supplied. The scenario list in Fig. 2 is cut off at the lower edge, so further scenarios may exist.

E. Coupled Simulation and Fault Injection

Scenarios are injected through POST/telemetry/simulate. Four core coupled scenarios are visible in Fig. 2: *Reset Baseline (Normal BEP)*, *Motor Thermal/Vib Stress (Coupled Pump Derate)*, *Pump Dry-Run / Cavitation (Coupled Motor Unload)* and *Stop System (Idle Standby)*. Seven component-level scenarios follow (Table V), each labelled with its expected severity. The interface states that the injected scenarios represent idealised deterministic physics models for testing and review demonstration and are not physical sensor readings; this paper adopts the same position, and the header shows the telemetry source as SIMULATED.

The coupled scenarios follow the hydraulic relation between pump and motor loading. The hydraulic power delivered to the fluid is the product of volumetric flow Q and pressure rise Δp:

$$Ph = Q\,\Delta p \qquad (9)$$

Dry running or severe cavitation reduces the fluid actually delivered and so lowers the pump's power demand and the motor load, which is consistent with the scenario label "Coupled Motor Unload". Conversely, motor thermal or vibration stress may require the pump to be derated ("Coupled Pump Derate"). The numerical models used by the simulator for these effects are not part of the supplied materials [], and the scenario that produced the state in Figs. 1

and 2 is not recorded.

6. 3D DIGITAL TWIN VISUALIZATION

A. Interface and Observed Behaviour

The 3D panel is titled "Coupled Skid Assembly" with the subtitle "Interactive Physical Model • Click to Inspect" (Fig. 1). It shows the pump, a coupling with guard and the motor on a common baseplate above a ground grid. The pump is drawn in blue with a volute casing and flanged connections, the motor in cyan as a cylindrical frame with a terminal box, and the coupling guard in tan. The grid lines converge towards the horizon, which indicates a perspective projection.

Three view buttons (Coupled Skid, Motor View, Pump View) select the framing. Further controls toggle a cutaway ("Cutaway: OFF" in the capture), start rotation (Rotate) and restore the default view (Reset). A hint states the pointer mapping: left-button drag to orbit, scroll to zoom and right-button drag to pan. Beneath the canvas an inspector line names the selected component and its status ("Motor Housing & Frame [NORMAL]"), a badge shows that a warning is active, and a legend labelled "Click to Select in 3D" lists selectable components with colour-coded Normal, Warning and Critical markers and a textual status tag.

In the captured frame the selected component is in Normal status and the surfaces appear in their base colours. The frame therefore does not demonstrate how a Warning or Critical component is highlighted in the scene; a capture with a warning component selected should be added .

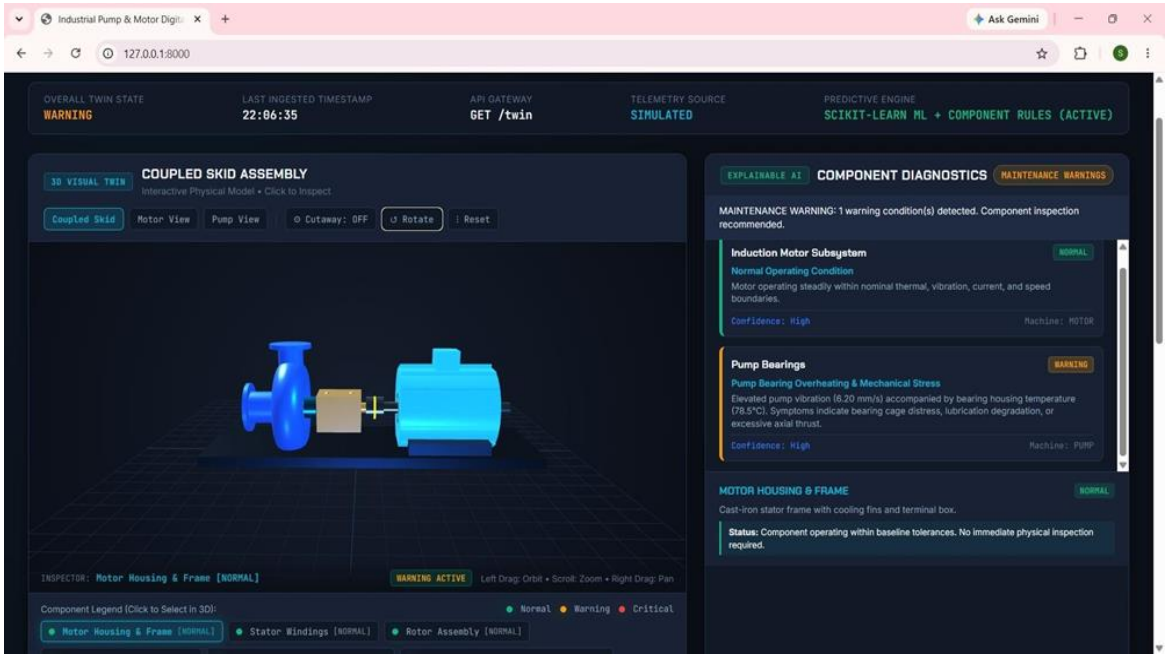


Fig. 1. Interactive 3D coupled pump-motor digital twin with component diagnostics. The panel shows the coupled skid model with view, cutaway, rotate and reset controls, the inspector line, the component legend and the explainable component-diagnostics cards (browser capture of the dashboard served at 127.0.0.1:8000, 29 October 2026; telemetry source: simulated).

B. Implementation Elements

The front-end source was not among the supplied materials. Table VI therefore separates what can be observed in the screenshots from the Three.js implementation details that the paper needs to document: scene, camera, renderer, geometry, materials, lighting, controls and picking. Three.js provides these building blocks for browser rendering [11]; the entries below name the items to be documented, not code that has been inspected.

TABLE VI

3D IMPLEMENTATION ELEMENTS AND EVIDENCE

Element	Evidence in supplied materials	Status
Library	Three.js, as stated by the authors	Implemented using Three.js
Scene graph	Legend lists per-component entries (Motor Housing & Frame, Stator Windings, Rotor Assembly)	Component hierarchy configured
Camera	Perspective foreshortening of the floor grid	Perspective camera configured
Renderer	Canvas inside the 3D panel	WebGL renderer configured
Geometry	Volute casing, cylindrical frame, terminal box, guard, shaft, baseplate	3D pump and motor components modelled
Materials	Distinct colours for pump, motor and guard; status colours in legend	Component colours and status colours assigned
Lighting	Shaded surfaces	Lighting and shading configured
Controls	Hint: orbit, zoom, pan	Orbit, zoom and pan controls implemented
Picking	“Click to Inspect”; inspector line	Component selection for inspection implemented
Cutaway	“Cutaway: OFF” toggle	Cutaway view toggle implemented
Status mapping	Legend tags such as [NORMAL]	Digital twin status mapped to component colours

C. Linking 3D Selection to Diagnostic Data

At the interface level Fig. 1 shows the coupling between selection and diagnostics. The panel headed “Explainable AI” lists one card per subsystem or component with a severity badge (NORMAL or WARNING), a diagnosis title, an explanatory sentence, a confidence label and a machine tag (MOTOR or PUMP). Below the cards, a detail panel for the selected component gives its name, a one-line description (“Cast-iron stator frame with cooling fins and terminal box.” for the motor housing) and a status statement (“Component operating within baseline tolerances. No immediate physical inspection required.”). The inspector line, the highlighted legend entry and the detail panel all show the same component, so selection state is reflected consistently across the three views.

Component status reaches these views through the aggregated twin state (GET /twin). The mapping between diagnostic component identifiers and scene objects, and the way a status change updates mesh appearance, are not described in the supplied materials . A well-defined mapping from component identifier to mesh group and status is the essential requirement for the design, because it keeps the diagnostic layer independent of the rendering layer.

7. SYSTEM IMPLEMENTATION

A. Software Stack

Table VII lists the stack. The firmware library versions were read from the library metadata in the supplied build tree. According to the firmware README, the backend is started from a sibling project directory with python backend/main.py and the dashboard is served locally on port 8000 (the address bar in Figs. 1 and 2 reads 127.0.0.1:8000).

TABLE VII SOFTWARE STACK

Part	**Technology (resolved version)**
Firmware	C++ with the Arduino framework; PlatformIO, platform espressif32, board esp32dev; serial monitor at 115200 baud
Firmware libraries	DallasTemperature 3.11.0; OneWire 2.3.8; Adafruit MPU6050 2.2.9; Adafruit Unified Sensor 1.1.15; Adafruit SSD1306 2.5.17; Adafruit GFX 1.12.6; Adafruit BusIO 1.17.4
Simulation	Wokwi (VS Code extension per README); Wi-Fi network Wokwi-GUEST
Backend	Python, FastAPI
Machine learning	scikit-learn
Front end	Browser dashboard, Three.js

B. ESP32 Firmware

The circuit (Table VIII) places the MPU6050 (address 0x68, per the README) and the SSD1306 OLED (0x3C) on the same IZC bus. At start-up the firmware initialises the serial port, the IZC bus, the temperature sensor, the MPU6050 and the display; if the MPU6050 or the OLED is not found it prints an error and halts in an endless loop. It then connects to Wi-Fi with a 15 s limit and continues with later retries if the connection fails. The main loop reads temperature, acceleration and the two ADC channels, derives the simulated quantities, sends telemetry when due, computes the local health indicator and refreshes the serial output and the OLED.

TABLE VIII

ESP32 CIRCUIT CONNECTIONS

Part	**Interface**	**Pins / address**	**Role**
DS18B20	OneWire	GPIO 4	Temperature
MPU6050	IZC	0x68; GPIO 21, 22	Acceleration
Potentiometer 1	ADC	GPIO 35	Simulated flow
Potentiometer 2	ADC	GPIO 34	Simulated current
SSD1306 128×64	IZC	0x3C; GPIO 21,22	Local display

C. Backend Interface

Three endpoints are evidenced by the firmware and the screenshots (Table IX). Their request and response schemas beyond the telemetry payload of Section V-A are not supplied

TABLE IX OBSERVED API ENDPOINTS

Endpoint	Method	Purpose
/telemetry	POST	Receives ESP32 JSON telemetry (firmware source)
/telemetry/simulate	POST	Injects a coupled or component scenario (Fig. 2)
/twin	GET	Returns the aggregated twin state (Fig. 1 header)

D. Build and Simulation

The firmware builds for the esp32dev target with PlatformIO and produces firmware.bin (1,220,656 bytes) and firmware.elf, which wokwi.toml references. The build outputs are newer than the last edit of src/main.cpp, and the binary contains the telemetry URL and network name strings, which indicates that it was built from the supplied source. The hostname host.wokwi.internal addresses the local machine through the Wokwi IoT Gateway, and the Wokwi documentation describes the Private Gateway as available to paying users only [13]; reproducing the ESP32-to-backend path therefore depends on that service.

8. RESULTS AND TESTING

A. Verification Scope

The supplied materials contain no automated test suite, no classification metrics, no latency or load measurements and no log of an end-to-end Wokwi session. This section therefore reports only properties that can be verified from the firmware source and build outputs and the behaviour observed in two dashboard captures. No accuracy figure is reported.

B. Observed Dashboard Behaviour

Table X and Fig. 2 give the state captured under a pump Warning condition, with the telemetry source displayed as SIMULATED.

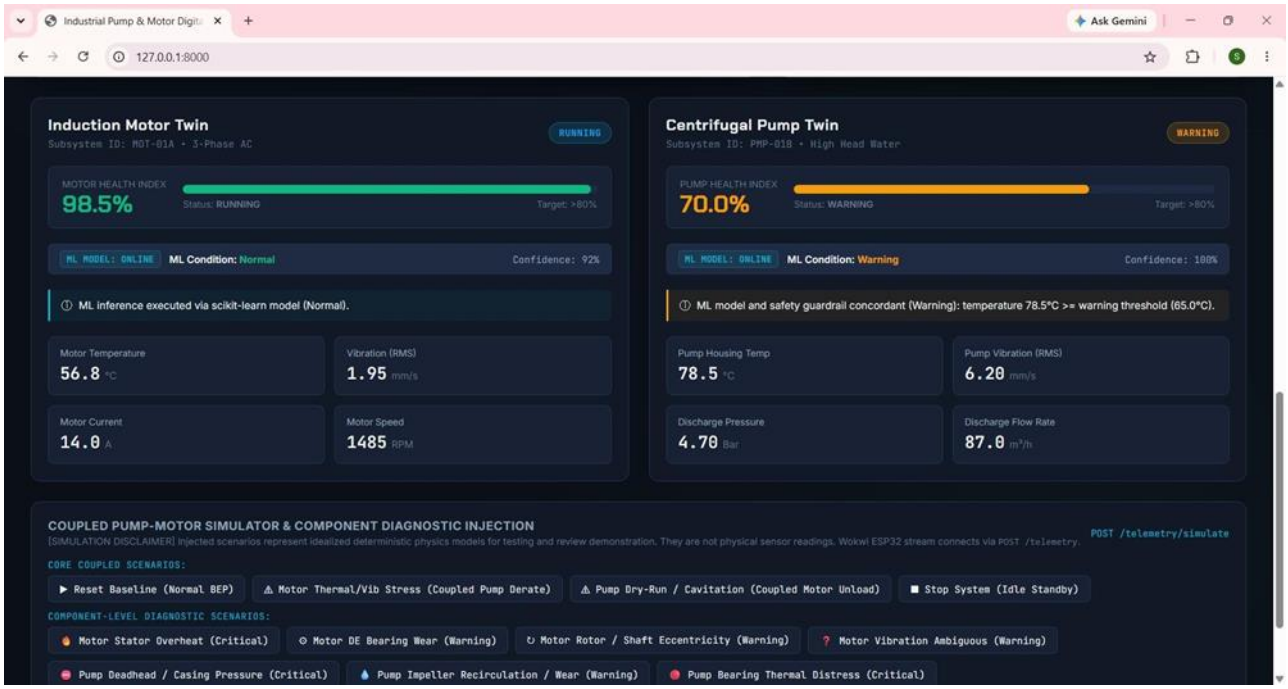


Fig. 2. Motor and pump monitoring panels showing telemetry values, machine-learning status and confidence, health indicators, the safety-guardrail message and the coupled and component-level fault-injection scenarios (browser capture of the dashboard served at 127.0.0.1:8000, 29 October 2026; telemetry source: simulated).

TABLE X

CAPTURED STATE SHOWN IN FIG. 2

Quantity	**Motor (MOT-01A)**	**Pump (PMP-01B)**
Temperature	56.8 °C	78.5 °C
Vibration (RMS)	1.95 mm/s	6.20 mm/s
Current / speed	14.0 A / 1485 RPM	Not displayed
Pressure / flow	—	4.70 bar / 87.0 m3/h
Health index	98.5 %	70.0 %
Subsystem status	RUNNING	WARNING
ML class (confidence)	Normal (92 %)	Warning (100 %)
Guardrail message	Inference via scikit-learn (Normal)	Concordant (Warning):78.5 ≥ 65.0 °C

Four observations follow. First, for the pump the ML class, the guardrail and the component layer agree: the classifier outputs Warning, the guardrail confirms it from the temperature threshold, the health index of 70.0 % lies below the >80 % target, and the component layer attributes the warning to the pump bearings from vibration and temperature (Fig. 1). Second, for the motor the ML class (Normal), the health index (98.5 %) and the component card (Induction Motor Subsystem, NORMAL) agree. Third, the overall twin state is WARNING, consistent with the worst subsystem state, although the aggregation rule is unconfirmed. Fourth, the agreement shows internal consistency only: the three assessments derive from the same inputs and are not independent evidence of correctness.

C. ESP32 Data-Path Analysis

The firmware's conditioning equations bound what the ESP32 module can send. Table XI compares those bounds with the values in Fig. 2. The module cannot reach the captured operating point: the motor current of 14.0 A exceeds the 10 A full scale of Eq. (2); the flow of 87.0 m3/h exceeds the 0.9 m3/h full scale of Eq. (3) by about two orders of magnitude; the pressure of 4.70 bar exceeds the 3 bar maximum of Eq. (5); and vibration is displayed as velocity in mm/s whereas the firmware sends an acceleration deviation in m/sZ. Either the backend rescales the ESP32 payload or the two telemetry sources use different operating envelopes and units; the supplied materials do not say which . In addition, the firmware sends one temperature and one vibration value for both machines, so the live stream cannot yet distinguish motor from pump.

TABLE XI

ESP32 OUTPUT BOUNDS VERSUS FIG. 2 VALUES

Quantity	**ESP32 module output**	**Fig. 2 value**
Current	0–10 A	14.0 A (motor)
Flow	0–0.9 m3/h (0–15 L/min)	87.0 m3/h
Pressure	0, or 1–3 bar	4.70 bar
Speed	0 or 1450 RPM	1485 RPM (motor)
Vibration	$\|\|a\| - g\|$ in m/sZ	1.95 and 6.20 mm/s
Temperature	One reading for both	56.8 °C and 78.5 °C

D. Items Not Verified

No end-to-end run from Wokwi through the backend to the dashboard is documented: the captures show the telemetry source as SIMULATED and no serial log or screenshot of a live ESP32 stream was supplied. ML performance, component-rule thresholds, 3D implementation internals, backend latency and persistence are likewise unverified.

9. DISCUSSION

Two-level diagnosis. Combining a learned machine-level classifier with explicit component rules and a threshold guardrail gives each stage a distinct job: the classifier supplies adaptable condition labels, the rules supply localisation and readable evidence, and the guardrail supplies a deterministic floor. Each stage needs its own validation, and agreement between them (Section VIII-B) should not be mistaken for it.

Declared provenance. The dashboard labels its telemetry source and carries a simulation disclaimer, and the firmware comments state that vibration is not a calibrated RMS value.

Carrying such labels into the paper prevents simulated behaviour from being read as measurement. A similar rule should cover absent data: a silent or zero-valued channel should not be displayed as healthy, and the backend's handling of missing telemetry is not described in the supplied materials.

Signal fidelity. The ESP32 module demonstrates the data path, not measurement quality. Dynamic acceleration derived from an accelerometer is not comparable with the mm/s RMS velocity that vibration severity guidance commonly uses [9]; a faithful chain would sample at a defined rate, remove the static component, filter, integrate to velocity and compute RMS over a stated bandwidth. Sharing one temperature and one vibration reading between both machines also prevents machine-specific diagnosis from the live stream.

Role of the 3D model. The scene is a navigation and communication layer: it gives spatial context for a diagnosis but adds no information beyond the diagnostic data. Its geometry is schematic, and Fig. 1 does not show a CAD-derived model.

10. LIMITATIONS

1. The classifiers are trained on synthetic data; no real-world accuracy, precision, recall or false-alarm rate is available.
2. All telemetry is simulated. In the ESP32 module, current and flow are potentiometer positions and speed and pressure are computed values; no physical machine was instrumented.
3. The vibration signal in the firmware is a gravity-subtracted acceleration magnitude, not a calibrated RMS velocity, and its units differ from the dashboard.
4. One temperature and one vibration reading are copied to both machines.
5. The firmware does not validate sensor readings (for example, a disconnected temperature sensor), and its ranges do not cover the dashboard scenarios.
6. Component rules and the guardrail thresholds are heuristic and have not been validated against manufacturer limits, standards or field data.
7. Telemetry is sent over plain, unauthenticated HTTP, which is acceptable only for a local simulation.
8. No latency, throughput, long-run stability or automated-test results are available.
9. The 3D model is schematic, and the highlighting of Warning and Critical components is not shown in the supplied captures.
10. The prototype is not a fully validated industrial digital twin; it has no prognostics and no feedback to a physical asset.

11. FUTURE WORK

The highest-priority step is validation on a physical or well-instrumented test rig, using real current transducers, pressure transducers, flow meters, a tachometer and separate temperature and vibration sensors for each machine. Vibration processing should be implemented as a documented chain to band-limited RMS velocity, and thresholds should be aligned with manufacturer limits and relevant standards [9]. ML evaluation should use held-out real data with per-class precision and recall, false-alarm burden and detection lead time, and the confidence output should be calibrated.

Remaining-useful-life estimation [6] and physics-based models could extend the diagnostics. System work includes secure transport and authentication, persistent time-series storage, an automated regression and load test suite, and a user study of the explanations. The 3D model could be derived from CAD geometry and could add component heat-map overlays.

Proposed validation protocol. For each class c of a held-out, independently labelled real dataset, precision and recall would be reported as

$$P_c = TP_c / (TP_c + FP_c), \quad R_c = TP_c / (TP_c + FN_c) \qquad (10)$$

together with the confusion matrix, the false-alarm rate per operating hour and the detection lead time between the first rule or model alert and a confirmed fault. Component-level diagnoses would be scored separately against fault labels assigned by independent maintenance reviewers, so that the ML layer and the rule layer are not credited for each other's results.

12. CONCLUSION

This paper described a prototype digital twin for a coupled induction motor and centrifugal pump that links simulated telemetry, machine-level ML classification, a threshold guardrail, explainable component-level rules and an interactive 3D model. Dashboard captures show a consistent Warning condition localised to the pump bearings, and the ESP32 firmware demonstrates a documented telemetry path with explicit equations. The results are limited to internal consistency: the classifiers rely on synthetic data, all telemetry is simulated, and the firmware and dashboard signal ranges and units do not match. The prototype is therefore an architecture and interface demonstration, and validation on real equipment is the necessary next step.

REFERENCES

Published research

[1] M. Grieves and J. Vickers, “Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems,” in Transdisciplinary Perspectives on Complex Systems, F.-J. Kahlen, S. Flumerfelt and A. Alves, Eds. Cham, Switzerland: Springer, 2017, pp. 85–113. doi:10.1007/978-3-319-38756-7_4.

2 F. Tao, H. Zhang, A. Liu and A. Y. C. Nee, “Digital twin in industry: State-of-the-art,” IEEE Trans. Ind. Informat., vol. 15, no. 4, pp. 2405–2415, Apr. 2019. doi:10.1109/TII.2018.2873186.

3 A. Fuller, Z. Fan, C. Day and C. Barlow, “Digital twin: Enabling technologies, challenges and open research,” IEEE Access, vol. 8, pp. 108952–108971, 2020. doi:10.1109/ACCESS.2020.2998358.

4 J. Lee, B. Bagheri and H.-A. Kao, “A cyber-physical systems architecture for Industry 4.0-based manufacturing systems,” Manufacturing Letters, vol. 3, pp. 18–23, 2015. doi:10.1016/j.mfglet.2014.12.001.

5 A. K. S. Jardine, D. Lin and D. Banjevic, “A review on machinery diagnostics and prognostics implementing condition-based maintenance,” Mech. Syst. Signal Process., vol. 20, no. 7, pp. 1483–1510, 2006. doi:10.1016/j.ymssp.2005.09.012.

6 Y. Lei, N. Li, L. Guo, N. Li, T. Yan and J. Lin, “Machinery health prognostics: A systematic review from data acquisition to RUL prediction,” Mech. Syst. Signal Process., vol. 104, pp. 799–834, 2018. doi:10.1016/j.ymssp.2017.11.016.

7 F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

8 A. Barredo Arrieta et al., “Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI,” Information Fusion, vol. 58, pp. 82–115, 2020. doi:10.1016/j.inffus.2019.12.012. Standards, official documentation and project artefacts

9 Mechanical vibration — Measurement and evaluation of machine vibration — Part 1: General guidelines, ISO 20816-1:2016, International Organization for Standardization, Geneva, 2016.

10 FastAPI contributors, “FastAPI documentation,” official documentation. https://fastapi.tiangolo.com/

11 Three.js authors, “Three.js documentation,” official documentation. https://threejs.org/docs/

12 scikit-learn developers, “scikit-learn user guide,” official documentation. https://scikit-learn.org/stable/user_guide.html

13 Wokwi, “ESP32 WiFi networking,” official documentation. https://docs.wokwi.com/guides/esp32-wifi

14PlatformIO, “PlatformIO documentation,” official documentation. https://docs.platformio.org/

15 Espressif Systems, “Arduino-ESP32 documentation,” official documentation.

https://docs.espressif.com/projects/arduino-esp32/en/latest/

16 Project artefacts: ESP32 telemetry firmware (src/main.cpp, platformio.ini, diagram.json, README.md) and dashboard captures of 29 October 2026, local engineering records