



# MedTwin AI: An IoT-Enabled Patient Digital Twin Framework for Personalized Health Monitoring and AI-Based Anomaly Detection

**Aryan Mali, Mashood Khan, Rudra Khairavle, Prof. Satish Kuchiwale\***

Department of Computer Engineering, Smt. Indira Gandhi College of Engineering (SIGCE), Ghansoli, Navi Mumbai, Maharashtra, India

\*Mentor, Department of Computer Engineering, Smt. Indira Gandhi College of Engineering (SIGCE), Ghansoli, Navi Mumbai, Maharashtra, India

## ABSTRACT

Remote patient monitoring can bring together wearable sensors, connected devices, and computational analysis to show health observations gathered outside clinics and hospitals. A stream of sensor readings, or an alert triggered by a threshold, is not by itself a patient digital twin, and a prototype that has not been validated must never be treated as a diagnostic device. This paper describes MedTwin AI, a proposed framework based on the Internet of Things (IoT) that connects timestamped physiological and motion observations to a structured virtual patient state, keeps recent history together with data-quality metadata, and supports review of anomalies through transparent baselines and candidate machine-learning methods. The planned prototype pairs an ESP32-class controller with a MAX30102 pulse-oximetry and heart-rate sensor and an MPU6050 inertial sensor, while further sensors remain optional and subject to verification. The study plan keeps offline analysis of public datasets distinct from later hardware testing. The candidate datasets, PhysioNet Wearable Device Dataset and PAMAP2, carry labels and populations that do not represent clinical emergencies. The paper sets out an architecture, a data schema, an evaluation protocol, implementation milestones, and privacy safeguards. At this stage no physical prototype, trained model, or experimental result is claimed. The intended contribution is a reproducible evaluation framework for testing whether a patient-specific or context-aware baseline offers value beyond fixed-rule monitoring on a clearly defined task.

Keywords: Digital Twin, Internet of Things, Remote Patient Monitoring, Anomaly Detection, Wearable Sensing, Personalized Baseline, Smart Healthcare

## 1. Introduction

Remote patient monitoring (RPM) relies on connected devices and information systems to gather health-related observations away from hospitals and clinics. Wearable sensors record physiological and movement signals, and networked services then store, display, and analyse them. Artificial intelligence (AI) can reveal patterns that no single rule would capture, yet how useful it is depends on the quality of the data, the representativeness of the evaluation, and the care taken in reading its output (Shaik et al., 2023; Verma et al., 2022). A screen that merely shows live readings is a dashboard. Calling a system a digital twin is a stronger claim: it implies that a structured model of one individual is held in software and revised each time new observations come in.

Healthcare digital twins vary widely in sophistication, ranging from simple data-linked models to elaborate ones tied to physiological processes or clinical workflows (Katsoulakis et al., 2024; El-Warrak & de Farias, 2024). The label should therefore not be applied casually to any sensor display. Here, the patient twin means a software-held state made up of the most recent valid sensor readings, their timestamps, short-term history, signal-quality indicators, data freshness, and baseline summaries once enough data have accumulated. This representation is intentionally limited and open to audit, and it does not attempt to simulate human physiology.

MedTwin AI is put forward as a student-level engineering framework that ties wearable observations to a timestamped patient state and to a workflow in which a human reviews unusual patterns. The smallest hardware set consists of an ESP32 development board, a MAX30102 optical pulse sensor, and an MPU6050 inertial measurement unit (IMU). ECG, temperature, GPS, display, buzzer, and emergency-button parts could be added later, but only once their purpose, compatibility, and safety consequences have been checked. The design leaves room for such extensions, and the paper keeps planned features clearly apart from those already built.

The main research question is this: on a clearly specified public wearable-data task, does a patient-specific or context-aware baseline reveal anything useful that a transparent fixed-rule baseline does not? The question is deliberately narrower than whether the system can detect medical emergencies. Public datasets allow offline study of algorithms, but they cannot on their own validate the proposed hardware, demonstrate clinical benefit, or show real-time behaviour from sensor to screen.

Four contributions are planned: (1) a modular architecture that keeps sensing, validation, communication, twin-state management, analytics, and presentation separate; (2) a defined policy for updating state and handling stale data; (3) a repeatable protocol for comparing a fixed-rule baseline with a

candidate statistical or machine-learning method; and (4) reporting guidelines that stop activity labels, stress annotations, or unsupervised scores from being passed off as medical diagnoses.

## 2. Related Work

### 2.1. Digital Twins in Healthcare

In their scoping review, Katsoulakis et al. (2024) found that digital twins for health cover many application areas and differ greatly in maturity. El-Warrak and de Farias (2024) likewise surveyed the literature and pointed to the inconsistent terminology and the variety of application patterns. Both reviews argue for a tightly defined scope in MedTwin AI. The first version of the twin is a synchronized, structured record of patient state, not a mechanistic model of the whole body, which would demand validated physiological models and clinical data beyond what this project has.

### 2.2. AI-Enabled Remote Monitoring

Shaik et al. (2023) reviewed AI-based remote patient monitoring, covering both its uses and its practical difficulties. A typical pipeline runs from sensors through connectivity and storage to analysis and a user-facing service, and faults can arise at any point: a sensor may sit badly on the body, a wireless link may lose messages, clocks may disagree, and a model may mistake an ordinary change of activity for something unusual. For this reason the proposed design keeps data-quality status and model output as two separate items of information.

### 2.3. IoT Architecture and Protocols

Verma et al. (2022) examined the IoT architectures and communication protocols used in healthcare monitoring. For a small prototype, Wi-Fi combined with MQTT or HTTPS is a sensible starting point, but the choice of protocol should be driven by the needs of the actual deployment and is not in itself a research contribution. Message identity, timestamps, units, reconnection behaviour, and access control matter at least as much as the transport used.

### 2.4. Anomaly Detection

Liu, Ting, and Zhou (2008) introduced Isolation Forest, which separates observations by random partitioning and yields an anomaly score. That score measures how unusual a sample is compared with the training data; it says nothing by itself about whether a person is ill. Statistical deviation measures, rolling baselines, and rule-based approaches are alternatives. Whichever is chosen should be justified by the features available, whether reference labels exist, and how easily false alarms can be explained.

### 2.5. Public Wearable Datasets

PAMAP2 provides time-series recordings of nine subjects performing 18 physical activities, captured with inertial sensors and a heart-rate monitor (Reiss, 2012). It suits activity-aware analysis but carries no labels for clinical deterioration. The PhysioNet Wearable Device Dataset holds physiological and motion signals recorded during acute stress and structured exercise (Hongn et al., 2025). Its annotations reflect the original study design and must not be recast as disease or emergency labels. Whether a dataset is suitable therefore turns on the exact question asked and on what its annotations actually mean.

Digital twins, wearable sensing, and anomaly detection are not new in themselves, so the opportunity lies elsewhere. MedTwin AI proposes to bring together data freshness, explicit state handling, context-aware baselines, and a transparent review workflow within a single testable engineering pipeline. Whether this combination actually improves monitoring can only be settled by building it and evaluating it repeatedly.

## 3. Problem Formulation and Objectives

A monitoring system can keep showing numbers without saying whether they are current, trustworthy, or out of the ordinary for that individual. Fixed thresholds are easy to explain, but they are only as good as the signal, the context, and the threshold itself. Adaptive methods can reflect individual patterns, yet they may overfit, raise too many alerts, or read a change in activity as an anomaly. Missing or outdated readings add another hazard, because a dashboard can display a believable value even after the sensor has stopped sending data.

Let  $x(t)$  be a feature vector computed from sensor observations at time  $t$ ,  $q(t)$  the associated data-quality metadata, and  $z(t)$  the virtual patient state being maintained. The state update is written  $z(t) = U(z(t-1), x(t), q(t))$ , where  $U$  takes in a new observation only if its timestamp and quality status pass defined checks. An anomaly module then produces a score  $a(t) = A(x(t), z(t), \theta)$ , in which  $\theta$  stands for parameters learned from training data. These expressions mark out the design boundary and do not imply that any such function has been implemented.

The objectives are to: (1) specify a modular architecture running from sensor to dashboard; (2) define a state schema that holds values, timestamps, history, freshness, and quality flags; (3) audit a public dataset before settling on a task and a method; (4) compare a transparent baseline with a candidate personalized or statistical method using the same held-out data; (5) record false positives, false negatives, data loss, and reproducibility; and (6) identify the privacy, safety, and validation requirements that must be met before any real-world use.

Scope and assumptions: at the time of writing, the project exists only as a concept. Firmware, a physical prototype, a database, a dashboard, a trained model, and experimental results have not been built or verified. The architecture and evaluation described here are proposed work. No patients are recruited, no clinical intervention is performed, and no claim about diagnostic performance is made.

#### 4. Proposed System Architecture

The architecture comprises six logical layers: sensor acquisition; edge validation and timestamping; communication and storage; management of digital-twin state; anomaly analysis; and visualization with alerts that can be reviewed. Splitting the system this way makes each stage easier to test and prevents a sensor driver from being tied directly to a model or an interface. Fig. 1 outlines the proposed flow.

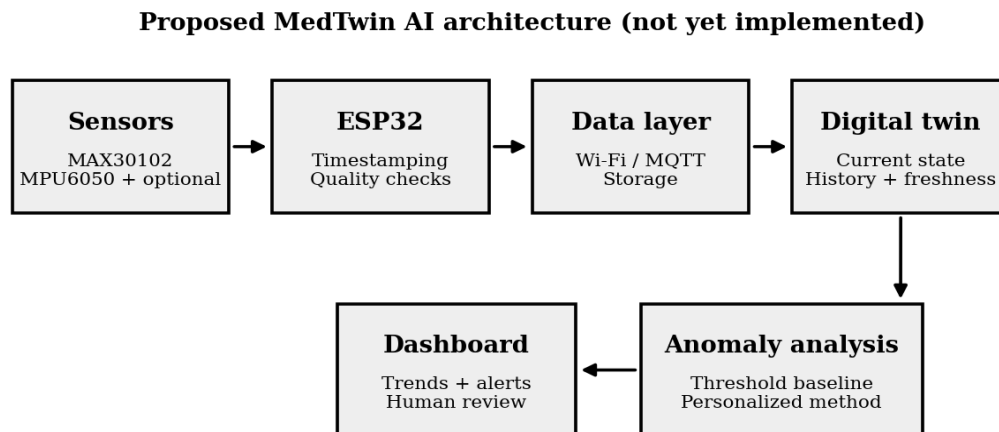


**Fig. 1 – Proposed MedTwin AI architecture; not yet implemented.**

##### 4.1. Sensor Acquisition

The MAX30102 is an optical sensor built for pulse oximetry and heart-rate measurement (Analog Devices, n.d.). What it reports are estimates, which can be disturbed by sensor placement, movement, perfusion, ambient conditions, and signal quality. The MPU6050 supplies accelerometer and gyroscope data from which motion features can be derived; detecting falls would still need a defined algorithm and validation against a suitable reference. The DHT22 senses ambient temperature and humidity and should not be presented as a body-temperature sensor. If body temperature is needed, both the sensor and the measurement site must be named and justified.

##### 4.2. Edge Device

An ESP32-class board is proposed to read the sensors, stamp each reading with a time, run basic validity checks, and transmit records over Wi-Fi (Espressif Systems, n.d.). The sampling rate and sensor settings must be documented for every channel. The device should tell a missing observation apart from a genuine zero, keep sensor error states, and never present an old value as if it were current. If the connection drops briefly, local buffering is an option.

##### 4.3. Communication and Storage

Structured messages can be sent to a backend using MQTT or HTTPS; the implementation should pick one route and record why. Every message should carry a device or session identifier, a timestamp, the signal name, a value or feature vector, a unit, a quality status, and a software or schema version. Raw observations should be stored apart from derived features and model scores so that an auditor can retrace exactly how any alert came about.

##### 4.4. Patient Twin State

The state manager holds, for each signal, the latest valid observation, recent history, sensor-status metadata, and baseline summaries wherever enough data exist. An incoming observation that fails quality checks must not replace a valid state. The state should openly show how old the data are. When the newest valid reading exceeds a configured age limit, the interface should display “stale” or “unavailable” and avoid suggesting that all is normal.

##### 4.5. Analytics and Presentation

A transparent rule-based baseline should be built first, after which a candidate statistical or machine-learning method can be assessed on a defined task. The interface should show measured values, data freshness, quality warnings, the anomaly score or flag, and a short explanation of which signal or feature

prompted the review. Alerts should read “unusual pattern—review required” and never “diagnosis” or “emergency confirmed.” Responsibility for interpretation stays with a human.

Because the design is modular, the dataset-analysis pipeline can be developed before any hardware exists, and the same feature and state definitions can later be tried on hardware observations. Offline data experiments and hardware integration tests nevertheless address different questions, so they must be reported separately.

## 5. Materials and Methods

### 5.1. Proposed Materials

The smallest prototype consists of an ESP32 development board, a MAX30102 pulse-rate/SpO<sub>2</sub> module, and an MPU6050 IMU. Possible additions are an ECG module, a suitable temperature sensor, an OLED display, a buzzer, an SOS button, a location module, and a battery, each to be included only when its role and safety implications are understood. The software may be built with Python, pandas, NumPy, scikit-learn, a lightweight database, and a dashboard framework. These are intended implementation choices and do not show that a system has been completed.

### 5.2. Observation Schema

Each record should contain a pseudonymous subject or session identifier, a timestamp, the sensor or channel name, a value or feature vector, a unit, the source device, a quality flag, and sampling and configuration metadata. Derived records should also state the feature-window length and the algorithm version. Time should be represented consistently, with the handling of time zones documented in the logs. The schema should separate, at a minimum, the states valid, missing, stale, out-of-range, and sensor-error.

### 5.3. Data-Quality Processing

Preprocessing starts with an audit of the dataset: its data dictionary, licence, subject and session structure, units, sampling rates, and annotations. Duplicate timestamps, implausible values, gaps in channels, and sensor errors should be counted. Any filtering or interpolation needs a justification for each signal, and interpolation must not span long gaps or hide a sensor failure. Whenever records are dropped, both the number removed and the reason should be reported.

#### Proposed offline experimental workflow

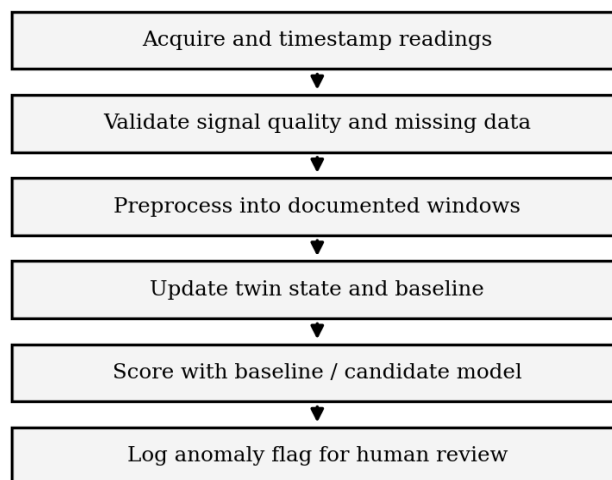


Fig. 2 – Proposed offline experimental workflow.

### 5.4. Dataset Selection

The main candidate is version 1.0.1 of the PhysioNet Wearable Device Dataset (Hongn et al., 2025), provided its signals and annotations are confirmed to support the chosen task. PAMAP2 may serve as a secondary dataset for activity-aware analysis of motion and heart rate (Reiss, 2012). A dataset should not be picked just because it contains physiological measurements; the task must fit what its labels describe. Where no valid ground truth for anomalies exists, the study should examine descriptive deviations or activity classification and refrain from reporting anomaly-detection accuracy.

### 5.5. *Windowing and Features*

For time-series work, continuous channels can be cut into fixed-length windows with a stated overlap. Candidate features include the mean, standard deviation, range, signal magnitude, and summaries sensitive to activity, depending on which channels are present. Window length should strike a balance between responsiveness and noise reduction and should be chosen on training and validation data. Feature scaling must use statistics from the training set alone, and nothing from held-out subjects or sessions may influence preprocessing or model selection.

### 5.6. *Data Splitting and Leakage Control*

When the aim is to generalize to people or sessions not seen before, splitting by subject or by session is preferable. Dividing neighbouring windows from one recording at random can overstate performance, since near-identical temporal patterns end up in both the training and test sets. The split method, random seed, exclusions, feature definitions, library versions, and model parameters should all be recorded. Under cross-validation, every learned preprocessing step must be fitted inside each training fold.

### 5.7. *Baseline and Candidate Method*

The baseline should be easy to understand and suited to the defined task. It can be a documented rule or a statistical boundary derived from training data, but medical thresholds must not be made up or borrowed from unrelated exercise data. The candidate can use rolling personal statistics or Isolation Forest (Liu et al., 2008), provided the feature space and dataset allow it. All baseline parameters must be chosen without looking at held-out test results. If an individual baseline is assessed, the least amount of earlier data needed to initialize it must be stated.

### 5.8. *Evaluation Metrics*

Where trustworthy labels exist for the task, precision, recall, F1-score, and a confusion matrix should be reported, along with class imbalance and, where suitable, variation between subjects or sessions. Precision is the share of flagged samples that agree with the reference label, and recall is the share of reference-labelled samples that were found. F1 combines the two but ignores clinical severity and the cost of a false alarm. If the task lacks meaningful labels, the report should present score distributions, case studies, and limitations instead of classification accuracy.

### 5.9. *System Metrics*

Once the hardware prototype is built, the following should be measured under a documented test schedule: delay from timestamp to dashboard, message delivery success, rate of missing readings, reconnection time, delay in updating state, and uptime. Controlled bench tests such as disconnecting a sensor, interrupting the network, repeating messages, and delaying packets should be used. An offline dataset cannot supply these metrics, and they must not be inferred from how well an algorithm performs.

---

## 6. *Digital-Twin State and Alert Logic*

A good state representation keeps both the measured value and the context needed to interpret it. A conceptual state record might include: (1) the current value; (2) the timestamp of the last valid sample; (3) recent history or a summary window; (4) a quality and availability flag; (5) the source sensor and unit; (6) a baseline summary; and (7) the latest anomaly score together with the model version. With this structure, “no unusual value was seen” can be told apart from “no trustworthy value is available.”

State updates should follow a fixed order. First, the structure and source of the incoming message are validated. Second, timestamp order, duplicate identity, range, and sensor status are checked. Third, the raw observation and its quality status are stored. Fourth, the current state is updated only if the sample meets the acceptance policy. Fifth, derived features and baseline summaries are updated under documented rules. Last, the anomaly output is computed and logged with a reference to the input window and model version. Rejected records should stay visible in diagnostic logs and not vanish without trace.

Alert handling should distinguish three situations: an unusual but valid observation, a data-quality problem, and a loss of connectivity. Each calls for different follow-up and so needs its own label. Repeated anomalies may be merged into one event to limit alert fatigue, but any such grouping rule should be documented and tested. Thresholds for stale data and for repeated alerts are engineering settings and not medical limits.

A patient-specific baseline needs safeguards as well. If it adapts continuously, a sustained abnormal pattern could slowly be absorbed into what counts as normal. A cautious design could update baselines only during accepted periods or through a recalibration process that has been reviewed. For this reason the paper treats baseline personalization as an experiment and does not assume it to be safer or better.

---

## 7. Implementation Plan and Reproducibility

The work is organized in stages so that each component can be shown and tested on its own. Stage 1 audits the dataset and defines the offline task. Stage 2 produces a reproducible preprocessing script that outputs versioned feature tables. Stage 3 implements a simple baseline and one candidate method. Stage 4 evaluates both on held-out subjects or sessions and analyses errors. Stage 5 brings up the sensors and timestamped logging on the ESP32. Stage 6 adds backend storage and a minimal state manager. Stage 7 integrates the dashboard and runs controlled network and sensor-failure tests.

The smallest demonstrable prototype should display a live or replayed observation, its timestamp, a quality and freshness indicator, a recent trend, and an anomaly flag that can be audited. Each alert should be traceable to the observation window, feature definition, and model version behind it. A replay mode helps with demonstrations because it makes scenarios repeatable, without suggesting that a public dataset is being streamed from real patients.

For reproducibility, the following should be kept: dataset version, access date, licence, preprocessing configuration, feature-window definition, the train, validation, and test split, random seed, software environment, model parameters, and how the metrics were computed. A README should explain how to repeat the analysis. If code or models change after an experiment, the paper should name the new version and avoid mixing results across configurations.

No evidence of an implementation was present in the project materials reviewed for this paper, so the stages above are planned milestones. A later version should fill in the actual hardware revisions, wiring, firmware, backend, database schema, model configuration, and test conditions, but only after these have been verified.

---

## 8. Proposed Experimental Evaluation

### 8.1. Experiment 1 – Offline Data Study

The first experiment should address a single bounded question using one dataset and one clearly defined task. One example is testing whether a context-aware statistical method separates protocol-defined activity states more consistently than a fixed rule, on condition that the annotations justify that reading. Stress or exercise must not be relabelled as a medical emergency. If the dataset has no suitable reference labels, the experiment should be recast as descriptive anomaly scoring, and supervised classification metrics should not be reported.

The protocol should specify the dataset and its version, the channels, subjects, and sessions used, unit conversions, window size and overlap, the feature set, exclusions, the split strategy, baseline and candidate parameters, and the definitions of the metrics. Both methods must be tested on the same held-out records. The results table should list the task, method, test split, metric, and variability. Where possible, error analysis should examine false positives and false negatives by activity context, signal quality, and subject or session.

### 8.2. Experiment 2 – State Synchronization and Communication

Once the prototype is assembled, observations should be sent at a known interval and the messages expected compared with those the backend actually received. Controlled interruptions, delayed packets, duplicates, sensor disconnections, and reconnections should be introduced. The measurements are missing messages, update latency, detection of stale state, and recovery behaviour. Test duration, network conditions, number of trials, and the definition of each metric should be reported. These tests gauge engineering reliability and not clinical accuracy.

### 8.3. Experiment 3 – Sensor and Data Agreement

Where feasible and safe, the prototype's sensor output should be compared with a suitable reference device under a documented protocol, with the reference instrument, time alignment, participant conditions, and agreement metric all stated. Correlation by itself does not establish agreement. Any work involving human participants, health data, or clinical interpretation must comply with the relevant institutional review, consent, privacy, and safety requirements.

The comparison should be reported even when the personalized method does no better than the baseline. A neutral or negative outcome is still valuable because it shows the limits of the proposed integration. Claims such as "real-time accuracy," "early diagnosis," or "reduced mortality" should be avoided unless suitable evidence directly supports them.

---

## 9. Results and Discussion

This is a proposal-stage paper, so there are no experimental results. No model scores, dataset findings, sensor-agreement values, hardware latency figures, or clinical outcomes are reported. This is intentional, since filling the section with plausible-looking numbers that no experiment produced would mislead readers. The final experimental paper should replace this section with real tables, figures, uncertainty estimates where fitting, and a discussion based on those results.

The central issue will be whether the chosen personalized or context-aware method adds value over the fixed-rule baseline on the defined task. A cleaner separation of anomaly scores does not automatically mean better patient monitoring, as differences may stem from activity transitions, participant

characteristics, sensor placement, noise, or conditions specific to the dataset. False positives can erode trust in alerts, while false negatives can hide relevant patterns, and both should be examined in context.

Even strong offline performance would have limited external validity, since the research-grade wearables in public datasets differ from the planned ESP32, MAX30102, and MPU6050 setup. Sampling frequency, sensor placement, firmware, motion artefacts, and preprocessing can all alter model behaviour. The offline study and the hardware prototype should therefore be described as distinct evaluation stages, joined by a carefully reasoned bridge.

A realistic outcome of this work may be an auditable engineering demonstration and not a clinical prediction system. The benefit of a structured twin state is that it makes freshness, missing data, historical context, and model provenance visible. Whether that visibility helps users make better decisions is a separate question, requiring a defined user study and proper oversight.

---

## 10. Limitations, Privacy, and Ethics

The most immediate limitation is that MedTwin AI has not yet been built. Public datasets may consist of healthy participants carrying out structured activities and not patients whose condition is worsening. Activity and stress annotations describe the design of the original studies and cannot serve as diagnostic labels. Moreover, the evidence at hand cannot show that the chosen sensors will give clinically reliable readings once assembled into the proposed prototype.

Health-related observations are sensitive. A deployed system should collect as little as possible, use pseudonymous identifiers, limit access by role, protect data in transit and at rest where supported, and set rules for retention and deletion. Logs should avoid unneeded personal information, and location collection should be optional and tied to a specific use case. Training and testing should rely on de-identified or properly governed data, and dataset licences and usage conditions must be respected.

Alerts should be framed as prompts for human review and not as treatment instructions or replacements for professional care. Sensor artefacts can trigger false alarms, and missing data can conceal real events. The prototype must not be relied on for emergency response or clinical decisions. Any later study with human subjects will need the appropriate approvals, informed consent where applicable, and a risk assessment. Clinical validation and regulatory assessment lie outside the present scope.

Bias and generalizability also need attention. A dataset with few subjects may not reflect differences in age, physiology, activity, skin characteristics, device fit, or co-existing conditions, and an algorithm trained on one device may not carry over to another. Future evaluations should describe the composition of their datasets and avoid broad claims about populations that were not represented.

---

## 11. Conclusion and Future Work

This paper puts forward MedTwin AI as a modular IoT framework for patient monitoring that connects timestamped wearable observations to a structured virtual patient state and to a limited anomaly-review workflow. The design stresses explicit data freshness, quality metadata, traceable state updates, and a clear separation between an anomaly score and a medical conclusion. It also lays out an evaluation plan that compares a transparent baseline with a candidate personalized or statistical method on suitable public data.

Until implementation and evaluation are finished, the contribution remains a proposal. The next steps are to examine the selected dataset and define a defensible task, build reproducible preprocessing and baseline methods, assemble and validate the sensor prototype, create a minimal state manager and dashboard, and run controlled communication tests. Any later paper should report real measurements, include error analysis, and keep dataset findings distinct from hardware performance. Clinical claims should stay out of scope unless backed by appropriate data, expert collaboration, and formal validation.

---

## REFERENCES

- Analog Devices. (n.d.). MAX30102: High-sensitivity pulse oximeter and heart-rate sensor for wearable health [Product documentation and datasheet]. <https://www.analog.com/en/products/MAX30102.html>
- El-Warrak, L., & de Farias, C. M. (2024). The state of the art of digital twins in health—A quick review of the literature. *Computers*, 13(9), Article 228. <https://doi.org/10.3390/computers13090228>
- Espressif Systems. (n.d.). ESP32 series datasheet. [https://www.espressif.com/sites/default/files/documentation/esp32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf)
- Hongn, A., Bosch, F., Prado, L. E., Ferrández, J. M., & Bonomini, M. P. (2025). Wearable physiological signals under acute stress and exercise conditions. *Scientific Data*, 12. <https://doi.org/10.1038/s41597-025-04845-9> (Dataset version 1.0.1: <https://doi.org/10.13026/he0v-tf17>)
- Katsoulakis, E., et al. (2024). Digital twins for health: A scoping review. *npj Digital Medicine*, 7, Article 77. <https://doi.org/10.1038/s41746-024-01073-0>
- Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *Proceedings of the 8th IEEE International Conference on Data Mining* (pp. 413–422). <https://doi.org/10.1109/ICDM.2008.17>

- 
- Reiss, A. (2012). PAMAP2 physical activity monitoring. UCI Machine Learning Repository. <https://doi.org/10.24432/C5NW2H>
- Shaik, T., Tao, X., Higgins, N., Li, L., Gururajan, R., Zhou, X., & Acharya, U. R. (2023). Remote patient monitoring using artificial intelligence: Current state, applications, and challenges. *WIREs Data Mining and Knowledge Discovery*, 13(2), e1485. <https://doi.org/10.1002/widm.1485>
- Verma, N., Singh, S., & Prasad, D. (2022). A review on existing IoT architecture and communication protocols used in healthcare monitoring system. *Journal of The Institution of Engineers (India): Series B*, 103, 245–257. <https://doi.org/10.1007/s40031-021-00632-3>