International Research Journal of Education and Technology (IR
ISSN: 2581-7795, Impact Factor 5.007
Peer-Reviewed | Open Access | International Journal
Volume: 09 | Issue: 07 | July 2026 | www.irjweb.com

PYTHON-BASED SMART FACTORY MONITORING DASHBOARD FOR PREDICTIVE MAINTENANCE

[1]Ms. S. Narmatha, [2]Ms.S.Lithika
[1]Assistant Professor, Department of Mathematics, Sri Krishna Arts and Science College, Kuniamuthur, Coimbatore, Tamil Nadu, India
[2]Student, III BSc Mathematics, Department of Mathematics, Sri Krishna Arts and Science College, Kuniamuthur, Coimbatore, Tamil Nadu, India

****

Abstract - *This paper presents a Python-Based Smart Factory Monitoring Dashboard for predictive maintenance. The proposed system monitors three important machine parameters: temperature, conveyor material flow, and machine vibration.. Newton's Law of Cooling and exponential decay models were used to derive analytical solutions. Python is used to generate and process simulated machine data, compare operating conditions with predefined threshold values, generate alerts, display machine status, and provide basic future-risk predictions and maintenance recommendations. NumPy is used for numerical calculations, Pandas for data organization, and Matplotlib for graphical visualization. The dashboard identifies overheating, low material flow, and bearing failure risk, while also simulating email alerts.*

Key Words: Python, Smart Factory, Predictive Maintenance, Machine Monitoring, Industry 4.0, IoT, Data Analytics

1. INTRODUCTION

1.1 Background

Industry 4.0 is transforming traditional manufacturing into smart manufacturing by integrating digital technologies such as automation, Industrial Internet of Things (IIoT), data analytics, artificial intelligence (AI), and predictive maintenance. Modern industries require continuous monitoring of machine conditions to reduce downtime, improve operational efficiency, and ensure product quality.

Unexpected machine failures can lead to production losses, increased maintenance costs, and safety hazards. Therefore, industries are adopting predictive maintenance systems that continuously monitor machine parameters and generate alerts before failures occur.

This project develops a Python-Based Smart Factory Monitoring Dashboard that monitors three important machine parameters: temperature, conveyor material flow, and machine vibration. The dashboard analyses machine conditions, generates alerts, predicts future risks, and recommends maintenance actions.

1.2 Project Overview

The monitoring system begins by generating machine operating data. Python analyses the data, compares it with threshold values, generates warning messages, and displays the results on a dashboard for operators. The same workflow can be adapted to real sensor data in modern manufacturing industries.

2. PROBLEM STATEMENT

2.1 Machine Overheating

Industrial machines produce heat while operating. If the temperature exceeds the safe operating limit, machine components may fail. In the proposed model, a temperature above 80°C is treated as an overheating condition, while values above 70°C indicate a warning.

Practical Importance: Continuous temperature monitoring can prevent machine damage and improve equipment life.

2.2 Conveyor Material Flow

Low conveyor material flow can interrupt production and reduce manufacturing efficiency. The proposed system generates a low-material alert when the flow falls below 150 units. Early detection helps operators refill materials before production stops.

2.3 Machine Vibration

Abnormal vibration may indicate bearing wear, shaft misalignment, or other mechanical faults. In the model, vibration above 9 is treated as a bearing failure risk, while values above 8 indicate a warning. Monitoring vibration supports early fault detection and reduces unexpected machine failures.

International Research Journal of Education and Technology (IR
ISSN: 2581-7795, Impact Factor 5.007
Peer-Reviewed | Open Access | International Journal
Volume: 09 | Issue: 07 | July 2026 | www.irjweb.com

3. MATHEMATICAL MODELING

3.1 Temperature Monitoring

$$T(t) = 30 + 60e^{-0.15t}$$

The temperature model is represented by an exponential expression describing machine cooling over time:

```
time = np.array([0,2,4,6,8,10])
temperature = 30 + 60*np.exp(-0.15*time)
print(temperature)
```

The model estimates machine temperature variation at different time intervals and provides data for threshold-based monitoring.

3.2 Conveyor Monitoring

```
flow = np.array([200,185,171,156,149,140])
print(flow)
```

The array represents conveyor material flow at different time intervals and is used to detect material shortages.

3.3 Machine Vibration

```
vibration = np.array([10,8.1,6.7,5.5,4.2,3.7])
print(vibration)
```

The vibration values are used to identify abnormal machine behaviour and bearing failure risk.

4. PYTHON IMPLEMENTATION

4.1 Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

NumPy performs numerical calculations, Pandas organizes the machine data, and Matplotlib creates graphs for visual monitoring.

4.2 Monitoring and Alert Logic

```
# Current values
current_temp = temperature[0]
current_flow = flow[-1]
current_vibration = vibration[0]

# Temperature status
if current_temp > 80:
    temp_status = "OVERHEATING"
elif current_temp > 70:
    temp_status = "WARNING"
else:
    temp_status = "NORMAL"

# Conveyor status
if current_flow < 150:
    flow_status = "LOW MATERIAL"
elif current_flow < 170:
    flow_status = "WARNING"
else:
    flow_status = "NORMAL"

# Vibration status
if current_vibration > 9:
    vibration_status = "BEARING FAILURE RISK"
elif current_vibration > 8:
    vibration_status = "WARNING"
else:
    vibration_status = "NORMAL"
```

The program compares current machine values with threshold limits and assigns a corresponding operating status. This provides a simple rule-based foundation for predictive maintenance.

4.3 Alerts and Prediction

```
if current_temp > 80:
    print("Check Cooling System")
if current_flow < 150:
    print("Refill Conveyor Material")
if current_vibration > 9:
    print("Inspect Bearing")

future_temp = current_temp + 5
if future_temp > 80:
    print("Future Overheating Risk")
if current_vibration > 9:
    print("Bearing Failure Predicted")
```

The dashboard generates maintenance recommendations and basic future-risk messages. Email alert simulation is also included for temperature and bearing-failure conditions.

5. DASHBOARD DESIGN

The dashboard presents temperature, conveyor material flow, and vibration in a single monitoring interface. For the simulated current values, the system reports temperature of 90°C as OVERHEATING, material flow of 140 kg as LOW MATERIAL, and vibration of 10 as BEARING FAILURE RISK.

The active alerts are Check Cooling System, Refill Conveyor Material, and Inspect Bearing. The prediction module reports Future Overheating Risk and Bearing Failure Predicted. These outputs support rapid operator decision-making.

6. REAL-LIFE IMPLEMENTATION

In a real manufacturing environment, temperature, flow, and vibration sensors can provide live machine data to the Python monitoring system through PLCs or IoT devices. The workflow is: Sensor Data → Python Dashboard → Condition Analysis → Alert → Maintenance Team or Production Team.

The current project uses simulated data. The simulated values can be replaced with live sensor measurements to extend the system toward a real-time Industry 4.0 monitoring solution.

7. BENEFITS

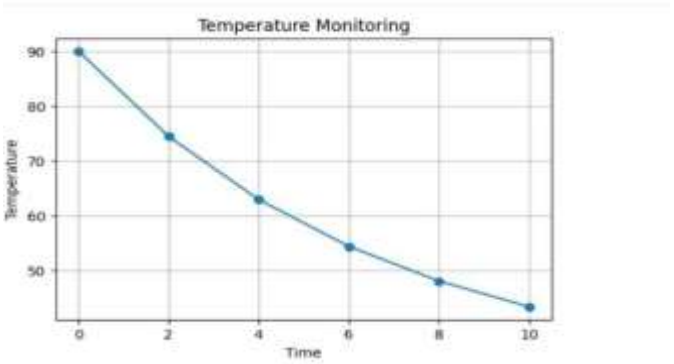
- Reduces machine downtime.
- Detects faults early.

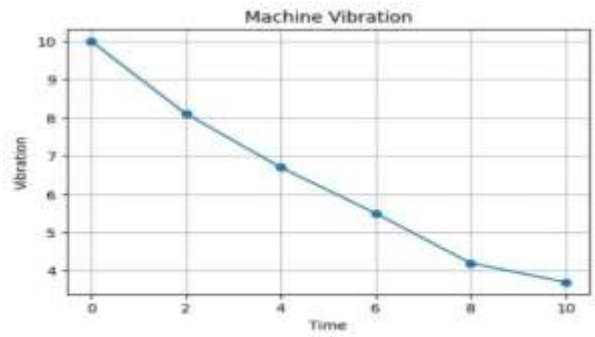
International Research Journal of Education and Technology (IR
‖ ISSN: 2581-7795, Impact Factor 5.007 ‖
‖ Peer-Reviewed | Open Access | International Journal ‖
‖ Volume: 09 | Issue: 07 | July 2026 | www.irjweb.com ‖

- Improves maintenance planning.
- Supports predictive maintenance.
- Increases production efficiency.
- Reduces maintenance costs.
- Improves operator awareness.
- Demonstrates Industry 4.0 concepts.
- Can be extended to real-time IoT systems.

8. RESULTS

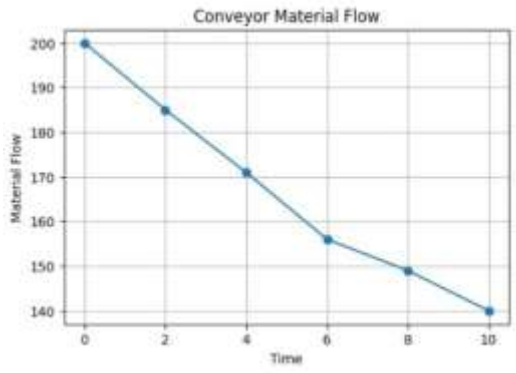
The developed Smart Factory Monitoring Dashboard successfully monitored machine temperature, conveyor material flow, and machine vibration. It generated threshold-based alerts, displayed machine status, simulated email notifications, demonstrated basic AI-based maintenance prediction, and produced graphical visualization of machine parameters. The results show the practical application of mathematical modelling and Python programming in manufacturing.





9. CONCLUSION

This project demonstrates how Python programming and mathematical modelling can be combined to develop a Smart Factory Monitoring Dashboard for predictive maintenance. The system monitors machine temperature, conveyor material flow, and vibration, analyses operating



conditions, generates warning alerts, and recommends maintenance actions.

Although simulated data are used in the present implementation, the same architecture can be integrated with real sensors and PLC systems in industrial environments. The project provides a foundation for Industry 4.0 applications and illustrates how data-driven monitoring can improve equipment reliability, reduce downtime, and enhance manufacturing productivity.

REFERENCES

1. **Boyce, W. E., & DiPrima, R. C. (2017).** Elementary Differential Equations and Boundary Value Problems (11th ed.). Wiley.

2. **Zill, D. G. (2018).** A First Course in Differential Equations with Modeling Applications (11th ed.). Cengage Learning.

3. **Edwards, C. H., & Penney, D. E. (2019).** Differential Equations and Boundary Value Problems: Computing and Modeling (6th ed.). Pearson.

4. **Simmons, G. F. (2016).** Differential Equations with Applications and Historical Notes (3rd ed.). CRC Press.

5. **Kreyszig, E. (2011).** Advanced Engineering Mathematics (10th ed.). Wiley.

6. **Strang, G. (2016).** Introduction to Applied Mathematics. Wellesley-Cambridge Press.

7. **Chapra, S. C. (2018).** Applied Numerical Methods with MATLAB for Engineers and Scientists (4th ed.). McGraw-Hill Education.

8. **Johansson, R. (2019).** Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib (2nd ed.). Apress.

9. **Langtangen, H. P. (2016).** A Primer on Scientific Programming with Python (5th ed.). Springer.

10. **Olver, P. J. (2014).** Introduction to Partial Differential Equations. Springer.