Smart Condition Monitoring System

Francis Chapoterera
College of Engineering
Northeastern University
chapoterera.f@northeastern.edu

Abstract

Condition monitoring is the systematic approach to tracking the health of machines in real time, with the objective of predicting faults before they lead to failure. By continuously observing variables such as vibration, temperature, orientation, and sound, engineers can proactively detect emerging mechanical issues and intervene before severe damage occurs. This method underpins predictive maintenance strategies, which are preferred over reactive maintenance because they reduce downtime, increase safety, and optimize equipment longevity.

In particular, vibration monitoring is a cornerstone in the maintenance of rotating machinery, including motors, fans, compressors, and turbines due to the strong correlation between vibration patterns and mechanical faults like imbalance, misalignment, or wear. While traditional systems rely on expensive, wired sensors and dedicated analysis units, modern advances in embedded IoT platforms and machine learning now enable cost-effective wireless monitoring solutions using microcontrollers.

This project proposes a smart condition monitoring system tailored for industrial rotating machinery. It utilizes an ESP32-WROVER-KIT microcontroller, an MPU6050 inertial measurement unit (IMU), and an OV3660 digital camera sensor. The system performs real-time monitoring of vibration, orientation, and visual inspection through a combination of embedded data acquisition and cloud-based machine learning analytics. This dual-modality approach, vibration plus imaging, provides a more comprehensive diagnostic tool, capable of not only classifying fan health conditions but also capturing visual cues for fault verification.

The project journey began with a different IMU (MPU9250), but due to complexity in processing and malfunctioning hardware, the MPU6050 (GY-521 module) was chosen instead. Integrating the sensor involved soldering the pins manually for the first time, an essential learning moment in electronics assembly. Challenges like short circuits, misalignment, and unstable sensor readings taught key lessons about hardware reliability.

This journey was also a testbed for communication protocols. Early testing with MQTT revealed its complexity and unreliability in constrained environments. Thus, the system evolved to use a Wi-Fi + HTTP-based cloud architecture with ThingSpeak, a practical decision driven by performance, simplicity, and reliability.

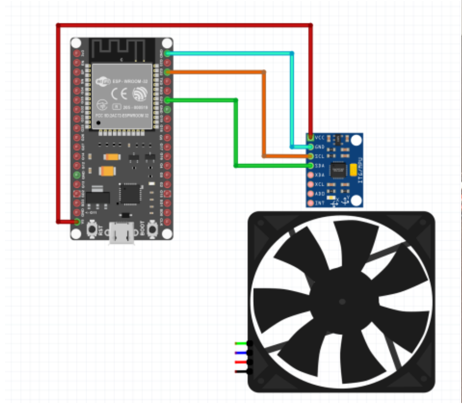
Key goals of the project include:

- Developing a low-cost, modular, and scalable monitoring platform.
- Enabling fault classification using a Subspace KNN ensemble classifier.
- Leveraging cloud services (ThingSpeak + MATLAB) for analytics and visualization.
- Integrating imaging to provide visual fault verification.

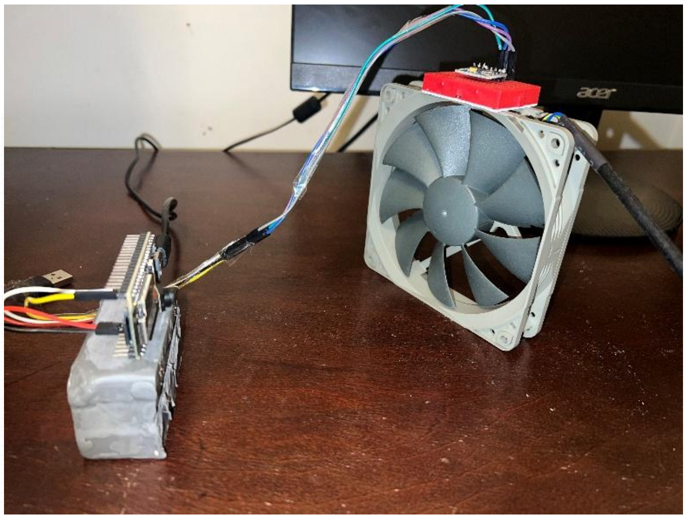
2. System Architecture and Design 2.1 Hardware Components

- ESP32-WROVER-KIT: Dual-core microcontroller with 4MB PSRAM for image and data buffering.
- MPU6050 (GY-521): 3-axis accelerometer and gyroscope connected via I2C.
- OV3660 Camera: 3MP JPEG-capable module connected over DVP interface.
- Breadboard and jumper wires for prototyping.
- Noctua NF-P12 redux-1700 PWM, High Performance Cooling Fan, 4-Pin, 1700 RPM (120mm).
- Clay block for mounting ESP32 on a flat and stable surface.

Wiring was optimized by increasing the MPU6050 cable length, which minimized mechanical stress and reduced airflow-induced tension that could interfere with readings. Additionally, placement of the ESP32 module was strategically chosen to avoid turbulence from the fan. A clay block was used to elevate the ESP32 and ensure airflow was not reflected onto the sensor.



Fritzing Diagram of the System Wiring



Actual Fan Monitoring Setup on Desk with Clay Block and Breadboard

Wiring Table

Component	**Pin Name / Function**	**ESP32-WROVER-KIT Pin**	**Description**
MPU6050 (GY-521)	SDA (I2C Data)	GPIO 32	I2C data line for accelerometer/gyroscope
	SCL (I2C Clock)	GPIO 33	I2C clock line
	VCC	3.3V	Power supply
	GND	GND	Ground
OV3660 Camera	XCLK	GPIO 21	External clock input
	SIOD (SDA - SCCB)	GPIO 26	Camera I2C data line (SCCB)
	SIOC (SCL - SCCB)	GPIO 27	Camera I2C clock line (SCCB)

	D0	GPIO 4	Camera data line D0
	D1	GPIO 5	Camera data line D1
	D2	GPIO 18	Camera data line D2
	D3	GPIO 19	Camera data line D3
	D4	GPIO 36	Camera data line D4
	D5	GPIO 39	Camera data line D5
	D6	GPIO 34	Camera data line D6
	D7	GPIO 35	Camera data line D7
	VSYNC	GPIO 25	Frame synchronization
	HREF	GPIO 23	Horizontal reference signal
	PCLK	GPIO 22	Pixel clock
	PWDN	Not connected	Not used (tied low internally)
	RESET	Not connected	Not used (handled in software)
Wi-Fi	N/A	Internal	Connects to network using SSID Pulsar

System Description

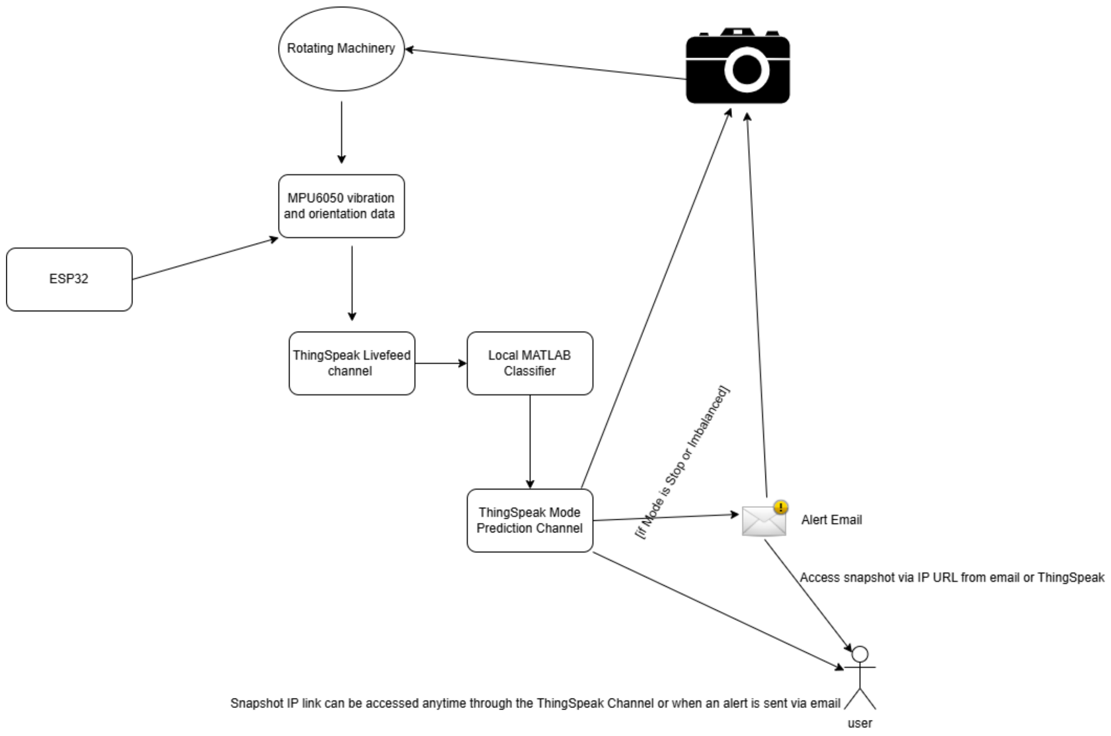
The Smart Condition Monitoring System integrates an ESP32-WROVER-KIT, an MPU6050 inertial measurement unit (IMU), and an OV3660 camera sensor to monitor the operational status of rotating machinery in real time. The MPU6050 is wired to the ESP32 via I2C, with the SDA and SCL lines connected to GPIO 32 and GPIO 33, respectively. This setup enables acquisition of tri-axis acceleration and angular data necessary for calculating the pitch and roll of the machinery.

The camera module, an OV3660, connects to the ESP32 through a dedicated parallel DVP interface. Data lines D0–D7 are mapped to GPIOs 4, 5, 18, 19, 36, 39, 34, and 35, respectively. Control signals such as XCLK, VSYNC, HREF, and PCLK are assigned to GPIOs 21, 25, 23, and 22. The SIOD and SIOC lines for camera configuration use GPIOs 26 and 27, forming an SCCB/I2C-like bus.

The system continuously reads real-time vibration and orientation data from the MPU6050 every 10 seconds. These readings are transmitted via HTTP POST to a ThingSpeak live data feed channel. Simultaneously, the ESP32 periodically polls a prediction channel that contains the most recent mode classification (Normal, Imbalanced, or Stop), which is generated by a MATLAB-based classifier running locally and uploaded to ThingSpeak.

The camera does not capture images automatically. Instead, it serves a real-time snapshot over an onboard HTTP server, which users can access by visiting the ESP32's IP address (http://10.0.0.131/). This IP address can be seen on the mode prediction Thingspeak channel or it can also be seen on the alert emails sent if the mode is “stop” or “imbalanced” This allows for flexible, on-demand visual inspection, either proactively through a monitoring dashboard or reactively when an email alert is triggered by an abnormal mode prediction.

Together, this modular setup creates a scalable, low-cost smart monitoring system capable of detecting early signs of mechanical faults in rotating machinery, while offering both sensor-based diagnostics and visual validation through embedded web connectivity.


Rotating Machinery
MPU6050 vibration and orientation data
ESP32
ThingSpeak Livefeed channel
Local MATLAB Classifier
ThingSpeak Mode Prediction Channel
[If Mode is Stop or Imbalanced]
Alert Email
Access snapshot via IP URL from email or ThingSpeak
Snapshot IP link can be accessed anytime through the ThingSpeak Channel or when an alert is sent via email
user

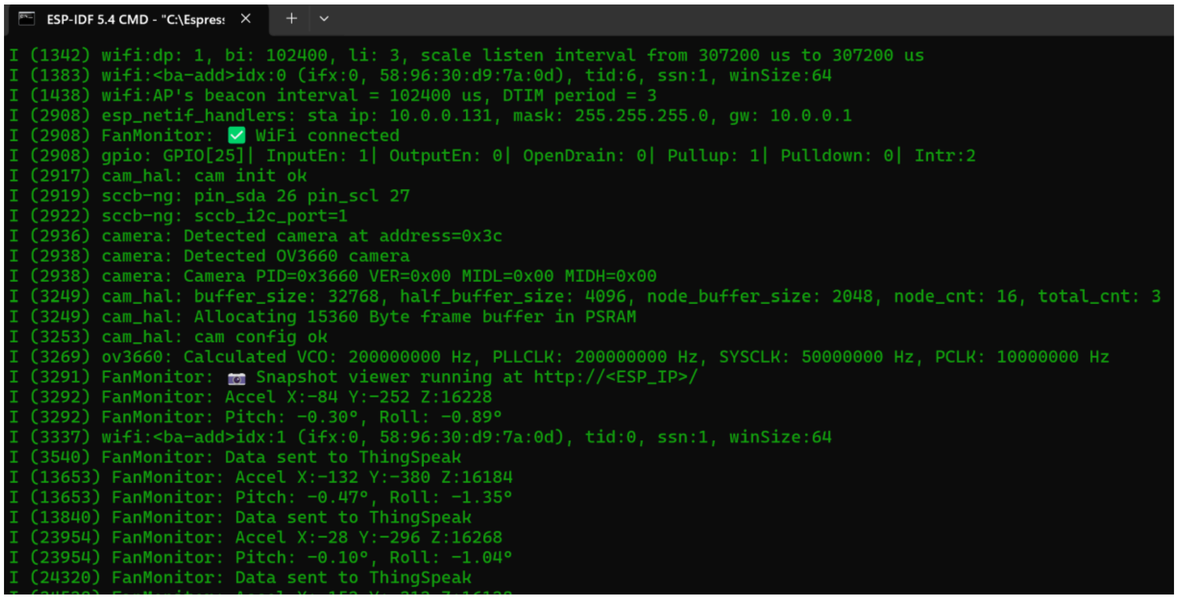
System Design

3. Software Stack Developed in ESP-IDF using FreeRTOS.

   Tasks include:

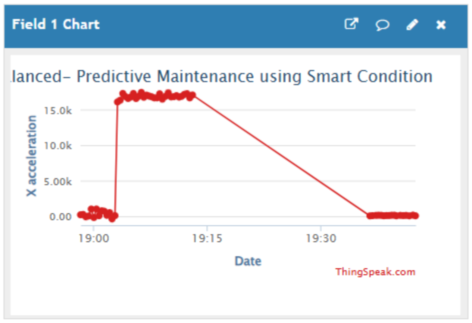
- Sensor Task: Reads IMU data every 20 ms.
- Camera Task: The system does not capture images automatically. Instead, the ESP32 hosts an onboard HTTP server that allows users to manually access a real-time snapshot by visiting the IP address (http://10.0.0.131/). This enables on-demand visual inspection of the machinery, either as part of routine checks via a dashboard or in response to an alert email indicating abnormal operation (e.g., Stop or Imbalanced mode). This design choice conserves system resources while still providing immediate access to visual context when needed.
- Wi-Fi Task: Handles cloud communication.

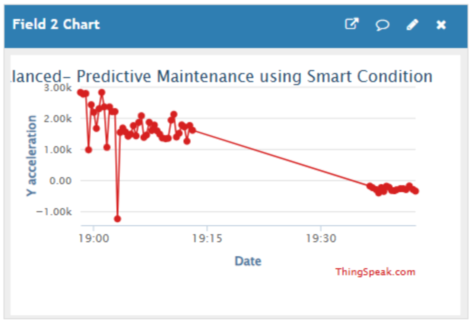
Data is sent to ThingSpeak in 3 separate channels (Normal, Imbalanced, Stop) for training and validation. A MATLAB-based classifier then runs in a loop, polling new data and writing predicted mode to a dedicated ThingSpeak prediction channel.

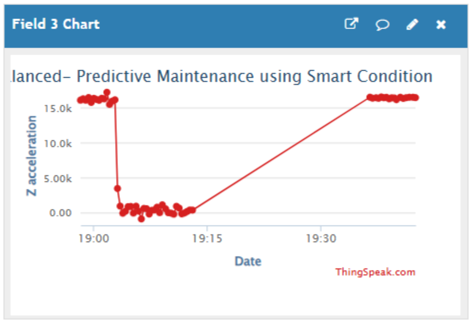


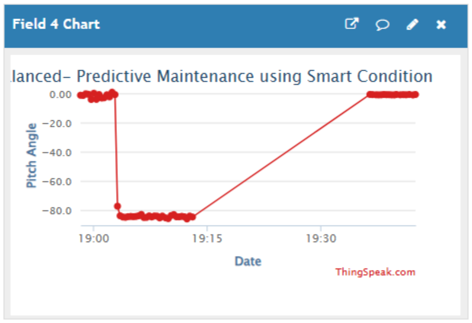
```
I (1342) wifi:dp: 1, bi: 102400, li: 3, scale listen interval from 307200 us to 307200 us
I (1383) wifi:<ba-add>idx:0 (ifx:0, 58:96:30:d9:7a:0d), tid:6, ssn:1, winSize:64
I (1438) wifi:AP's beacon interval = 102400 us, DTIM period = 3
I (2908) esp_netif_handlers: sta ip: 10.0.0.131, mask: 255.255.255.0, gw: 10.0.0.1
I (2908) FanMonitor: ✅ WiFi connected
I (2908) gpio: GPIO[25]| InputEn: 1| OutputEn: 0| OpenDrain: 0| Pullup: 1| Pulldown: 0| Intr:2
I (2917) cam_hal: cam init ok
I (2919) sccb-ng: pin_sda 26 pin_scl 27
I (2922) sccb-ng: sccb_i2c_port=1
I (2936) camera: Detected camera at address=0x3c
I (2938) camera: Detected OV3660 camera
I (2938) camera: Camera PID=0x3660 VER=0x00 MIDL=0x00 MIDH=0x00
I (3249) cam_hal: buffer_size: 32768, half_buffer_size: 4096, node_buffer_size: 2048, node_cnt: 16, total_cnt: 3
I (3249) cam_hal: Allocating 15360 Byte frame buffer in PSRAM
I (3253) cam_hal: cam config ok
I (3269) ov3660: Calculated VCO: 200000000 Hz, PLLCLK: 200000000 Hz, SYSCLK: 50000000 Hz, PCLK: 10000000 Hz
I (3291) FanMonitor: 📷 Snapshot viewer running at http://<ESP_IP>/
I (3292) FanMonitor: Accel X:-84 Y:-252 Z:16228
I (3292) FanMonitor: Pitch: -0.30°, Roll: -0.89°
I (3337) wifi:<ba-add>idx:1 (ifx:0, 58:96:30:d9:7a:0d), tid:0, ssn:1, winSize:64
I (3540) FanMonitor: Data sent to ThingSpeak
I (13653) FanMonitor: Accel X:-132 Y:-380 Z:16184
I (13653) FanMonitor: Pitch: -0.47°, Roll: -1.35°
I (13840) FanMonitor: Data sent to ThingSpeak
I (23954) FanMonitor: Accel X:-28 Y:-296 Z:16268
I (23954) FanMonitor: Pitch: -0.10°, Roll: -1.04°
I (24320) FanMonitor: Data sent to ThingSpeak
```

Data being sent to ThingSpeak









Sample Data from Imbalanced Mode ThingSpeak Channel Showing Logged Values

4. Data Acquisition and Classification Results A total of 938 labeled samples were collected:

- Normal: 351
- Imbalanced: 231 (collected at different angles to simulate asymmetric imbalance)
- Stop: 356

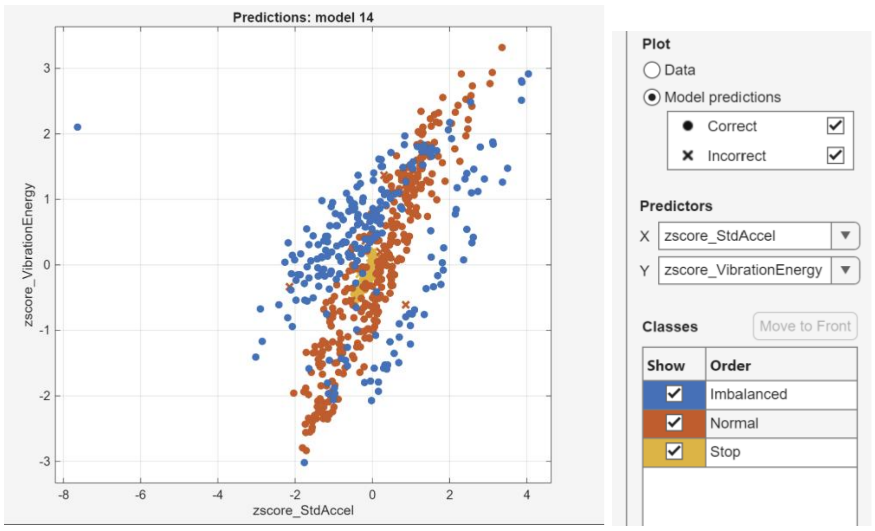
Selected Features:

- Zacceleration
- PitchAngle
- RollAngle
- StdAccel
- VibrationEnergy (z-score normalized)

Model Used: Subspace KNN Ensemble (Model 14)

- Learner: K-Nearest Neighbors
- Learners: 30
- Accuracy: 98.4%
- Precision (Macro): 98.5%
- F1 Score (Macro): 98.5%

Justification: This model offered the best trade-off between computational complexity and classification accuracy. It performed better than other ensemble methods, especially in distinguishing between Imbalanced and Normal modes. Additionally, scatter plots were used extensively to visualize feature separability.



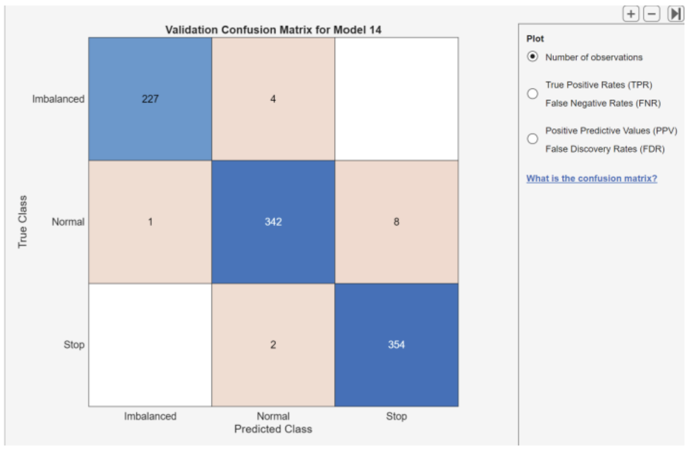
Plot
Data ( )
Model predictions (●)
● Correct ✓
× Incorrect ✓

Predictors
X zscore_StdAccel
Y zscore_VibrationEnergy

Classes

Show	Order
🟦 ✓	Imbalanced
🟥 ✓	Normal
🟨 ✓	Stop

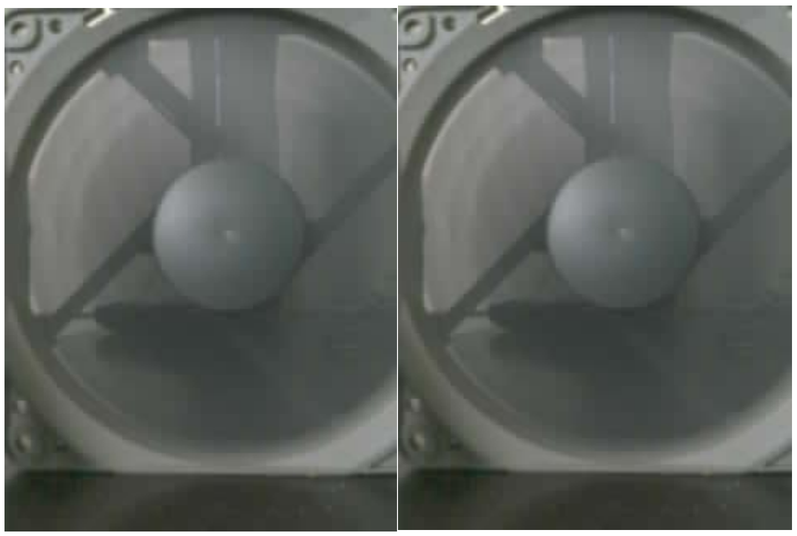
Scatter Plot Showing Mode Separation Based on Selected Features



Confusion Matrix of Final Model Predictions on Test Set

5. Engineering Challenges and Resolutions:

- Sensor Interference: Solved by extending wires, repositioning modules, and improving mounting stability.
- Camera Integration: Early JPEG errors were resolved by discarding the first frame post-initialization and using QVGA resolution for balance.
- Wi-Fi Reliability: Custom retry logic was added to ensure robust reconnection.
- Memory Management: PSRAM was used for camera buffers; optimized task scheduling avoided memory conflicts.
- Camera Framing: Field of view was tuned to focus only on the fan, minimizing irrelevant data. Distance from fan was also tested to avoid distortion and maintain image clarity.

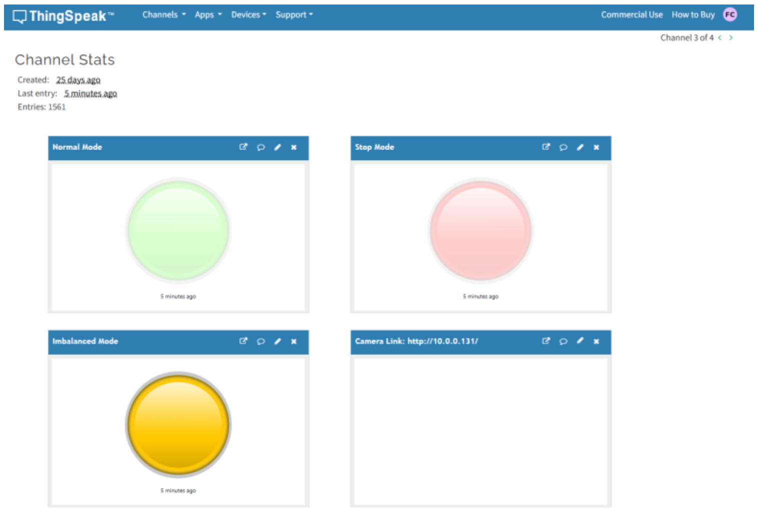


Sample Snapshot Taken by ESP32 Camera (OV3660) of the Spinning Fan

6. Communication and Control:

- Wi-Fi: Primary communication method for ThingSpeak API access.
- I2C: Used for MPU6050 communication.
- HTTP REST: Used to send vibration data to ThingSpeak.
- UART: Initially used in testing for debugging sensor readings before shifting to cloud communication.

While ThingSpeak supports image uploads, this is a paid feature. Instead, a snapshot link is displayed as a clickable URL on ThingSpeak that users can access to view a real-time photo served from the ESP32's onboard HTTP server.



Channel Stats

Created: 25 days ago
Last entry: 5 minutes ago
Entries: 1561

ThingSpeak Channel Showing Prediction and Snapshot Link Field

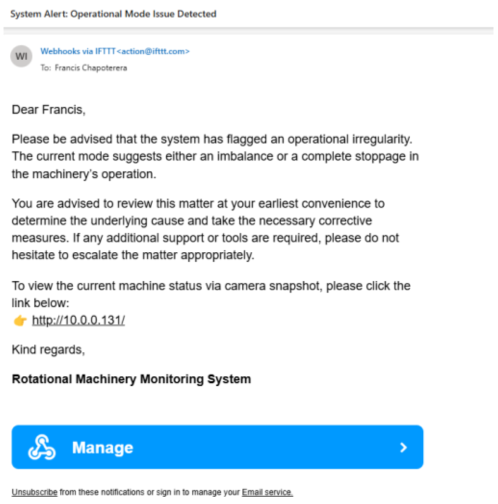
7. Alerting System and Remote Diagnostics:

   To enhance responsiveness and user awareness, an automated alerting mechanism was implemented using IFTTT (If This Then That). When the machine enters either the "Imbalanced" or "Stop" mode as predicted by the cloud-based classifier, a real-time email alert is triggered.

   Each email includes:

   A direct snapshot viewer URL (http://10.0.0.131/) that allows the user to remotely inspect the machinery via a live image served from the ESP32's onboard camera server.

   This setup enables remote fault verification by combining vibration-based prediction with visual confirmation, empowering operators to make faster, more informed decisions without physically inspecting the machine.



System Alert: Operational Mode Issue Detected

WI Webhooks via IFTTT <action@ifttt.com>
To: Francis Chapoterera

Dear Francis,

Please be advised that the system has flagged an operational irregularity. The current mode suggests either an imbalance or a complete stoppage in the machinery's operation.

You are advised to review this matter at your earliest convenience to determine the underlying cause and take the necessary corrective measures. If any additional support or tools are required, please do not hesitate to escalate the matter appropriately.

To view the current machine status via camera snapshot, please click the link below:
👉 http://10.0.0.131/

Kind regards,

Rotational Machinery Monitoring System

Unsubscribe from these notifications or sign in to manage your Email service.

8. Future Enhancements

- Local ML inference with TensorFlow Lite Micro.
- OTA firmware and model updates.
- Integration of thermal or acoustic sensors.
- Industrial accelerometer upgrade (e.g., ADXL355).
- Long-term data logging via microSD card.

Conclusion

This project served as a complete embedded systems journey, from hardware assembly and soldering to machine learning model deployment and web integration. Challenges in sensor communication, camera handling, memory constraints, and Wi-Fi stability were overcome through iterative testing and redesigns.

Ultimately, the project demonstrates a scalable, real-time predictive maintenance system using a Subspace KNN classifier with cloud analytics. It integrates sensor data, computer vision, and wireless communication in a way that is practical, extensible, and aligned with Industry 4.0. The decision to adopt HTTP over MQTT for simplicity and resilience, and the thoughtful positioning of components for accurate measurements, highlight a strong engineering design ethos throughout the work.

Appendices and References

- Scatter Plot (see attached)
- Confusion Matrix Image (attached)
- ESP32 Camera Snapshot (link available on ThingSpeak)
- MATLAB Classification Script (in submission folder)
- Reference Links:
    - processingmagazine.com
    - monitran.com
    - docs.espressif.com
    - thingspeak.mathworks.com
    - edgeimpulse.com
    - ez.analog.com
    - mdpi.com
    - sciencedirect.com