Smart Spindle Fault Detection and Health Monitoring System with Wireless Data Transmission

Indian Institute of Technology Kanpur

Submitted by:

Chanda Bhavitha Sri Jainam Tated Humayun Ahmad

Mentor: Dr. Mohit Subhash Law
Department of Mechanical Engineering

Abstract:

This project aims to develop a smart spindle health monitoring system that automatically starts collecting vibration data at speeds determined by the user after a set warm-up routine. The NI CompactRIO 9040 platform is utilized to build the system, involving NI 9234 (for vibration acquisition), NI 9263 (for analog output to control the VFD), and NI 9375 (for digital I/O to monitor the spindle state). The system controls a variable frequency drive (VFD) that is connected to a high-speed spindle. A LabVIEW-based interface lets you start a warm-up cycle with just one click to stabilize the spindle thermally before analysis. An algorithm based on FFT keeps a watch on the vibration spectrum in real time and only logs data when peaks are detected at speeds specified by the user. The data that has been acquired is stored in .csv format for further use in diagnostics and prognostics. Designed to be generalized, the system can be adapted to different spindles and machines with minimal modification. In parallel, for non-critical applications, a compact & relatively inexpensive alternative based on the Raspberry Pi has been made. It uses an LM317-based constant current source to power an IEPE accelerometer. A capacitor-resistor network for AC coupling and signal conditioning. The conditioned signal is digitized using an MCP3008 ADC and processed using Python scripts on the Raspberry Pi. The system has features that other commercial systems don't have, like automatic triggering, thermal stabilization, and real-time diagnostics. Its ease of deployment and adaptability across various platforms make it a strong candidate for commercialization and real-world implementation in predictive maintenance applications.

Introduction:

In CNC machines, the spindle is a critical and highly exposed component responsible for essential operations such as cutting, drilling, and milling. A failure in the spindle can disrupt the entire production process, leading to costly downtime and significant financial loss. Consequently, modern manufacturing facilities are moving toward smart spindle systems that support real-time diagnostics and prognostics to detect early signs of mechanical faults.

One of the most reliable indicators of spindle health is vibration acceleration data measured at different RPMs. This data helps identify issues such as bearing wear, imbalance, or misalignment faults that often remain undetected during routine inspections. Capturing and analyzing this data allows for subtle anomalies in spindle behavior to be detected early, supporting the shift from reactive to predictive maintenance.

While high-end commercial spindle monitoring solutions exist, they are often prohibitively expensive or inflexible. There is a growing need for customizable, cost-effective systems that can offer similar diagnostic functionality without the complexity or price tag of proprietary tools. Our project addresses this gap by introducing a solution tailored for both high-performance industrial environments and budget-constrained settings, thereby democratizing access to predictive spindle health monitoring.

Before transitioning to cost-effective boards, we initially prototyped the system using the NI CompactRIO 9040. This setup integrated an IEPE accelerometer with high-speed analog input (NI 9234), analog output for VFD control (NI 9263), and digital I/O (NI 9375). The key work was the implementation of a spindle warm-up routine, triggered data acquisition, and automated FFT peak logging all seamlessly controlled through a single-click LabVIEW interface. Such a high level of automation is uncommon in standard off-the-shelf spindle monitoring systems.

To complement this, a Raspberry Pi-based low-cost variant was developed. It uses an LM317-based constant current source, AC coupled signal conditioning, and an MCP3008 ADC to digitize signals for analysis using Python. This variant prioritizes affordability and accessibility, making it a scalable solution for small factories, academic research labs, and pilot production setups.

Our hardware offers:

- Lower cost, especially with the Raspberry Pi variant, while maintaining diagnostic relevance
- Greater flexibility and customization, using open hardware/software platforms
- Integrated warm-up and automated triggering, which are rarely built into commercial systems

Methodology:

a) With NI LabVIEW

1. Hardware Platform Setup
   - Use a **CompactRIO (cRIO 9040)** as the core controller.
   - Insert the following **C-Series modules** into the cRIO chassis:
     - **NI 9234** – for **vibration signal acquisition** from IEPE sensors.
     - **NI 9263** – for **analog voltage output** to control spindle speed.
     - **NI 9375** – for **digital I/O** to control relays (spindle ON/OFF, direction).
   - Power the system using a **24 V DC supply**, which also feeds external relays and voltage converters.
2. Vibration Signal Acquisition (NI 9234)
   - Connect an **industrial-grade IEPE vibration sensor** to the **NI 9234** using BNC connectors.
   - The NI 9234 provides **built-in constant current excitation**, eliminating the need for external biasing or capacitors.
   - The vibration data is **acquired directly by the FPGA** in the cRIO and passed on for further processing.
3. Digital Relay Control (NI 9375)
   - Use **channel D0** of the NI 9375 to **trigger Relay 1** (spindle ON/OFF).
   - Use **channel D1** to **trigger Relay 2** (motor direction: forward/reverse).
   - Power the relays using a **12 V buck converter** (stepping down from 24 V supply).
   - Relays interface with the **Delta VFD (Variable Frequency Drive)** through control pins **M0, M1, M2**, enabling isolated control over high-power lines.
   - Add **four 1 kΩ resistors** between I/O lines and relay inputs on a breadboard for signal safety and to avoid false triggering.
4. Analog Speed Control (NI 9263)
   - Use **channel A0** of the NI 9263 to send a **control voltage signal** to the **AVI (Analog Voltage Input)** of the VFD.
   - Tie **NI 9263's GND** to the **VFD's AGND** to maintain signal integrity.
   - This setup allows **precise spindle speed control** via voltage-level commands.
5. FPGA-Based Acquisition and Control Logic
   - Develop **FPGA.vi** to run on the cRIO's FPGA:
     - Implement **canned startup/warm-up sequences** for the spindle.
     - Acquire **high-speed vibration data** from NI 9234.
     - Store vibration data in a **FIFO buffer** for handoff to the RT layer.
6. Real-Time Data Processing and Logging
   - Develop **RT.vi** to run on the cRIO's **real-time processor**:
     - Read vibration data from the FPGA FIFO.
     - Perform **Fast Fourier Transform (FFT)** for frequency-domain analysis.
     - Apply **threshold-based peak detection** logic.
     - Save relevant datasets in **TDMS format** for offline analysis or reporting.
7. User Interface and System Control
   - Create a **LabVIEW GUI** (front panel) hosted on a **PC connected via Ethernet**.
   - Interface provides:
     - **Real-time waveform plots** (time and frequency domain).
     - **System health/status indicators**.
     - **Buttons for warm-up cycles**, **motor direction control**, and **log review**.
8. Final Outcome
   - The system provides:
     - **High-speed, high-resolution vibration monitoring** using IEPE sensors.
     - **Real-time FFT and thresholding** using the cRIO's RT processor.

- **Safe and isolated control** of spindle motor via relays and analog voltage.
- A **user-friendly PC-based GUI** for diagnostics and control.

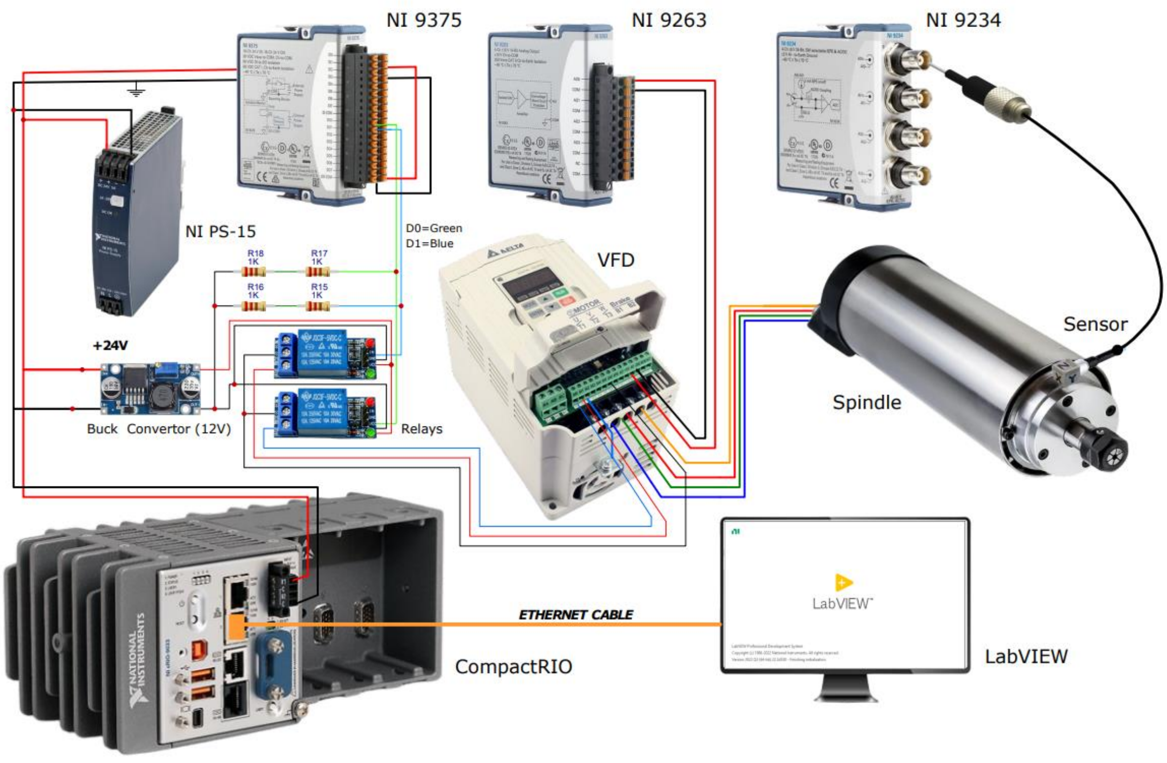

Figure 1: LabVIEW Data Acquisition and Automated Triggering setup

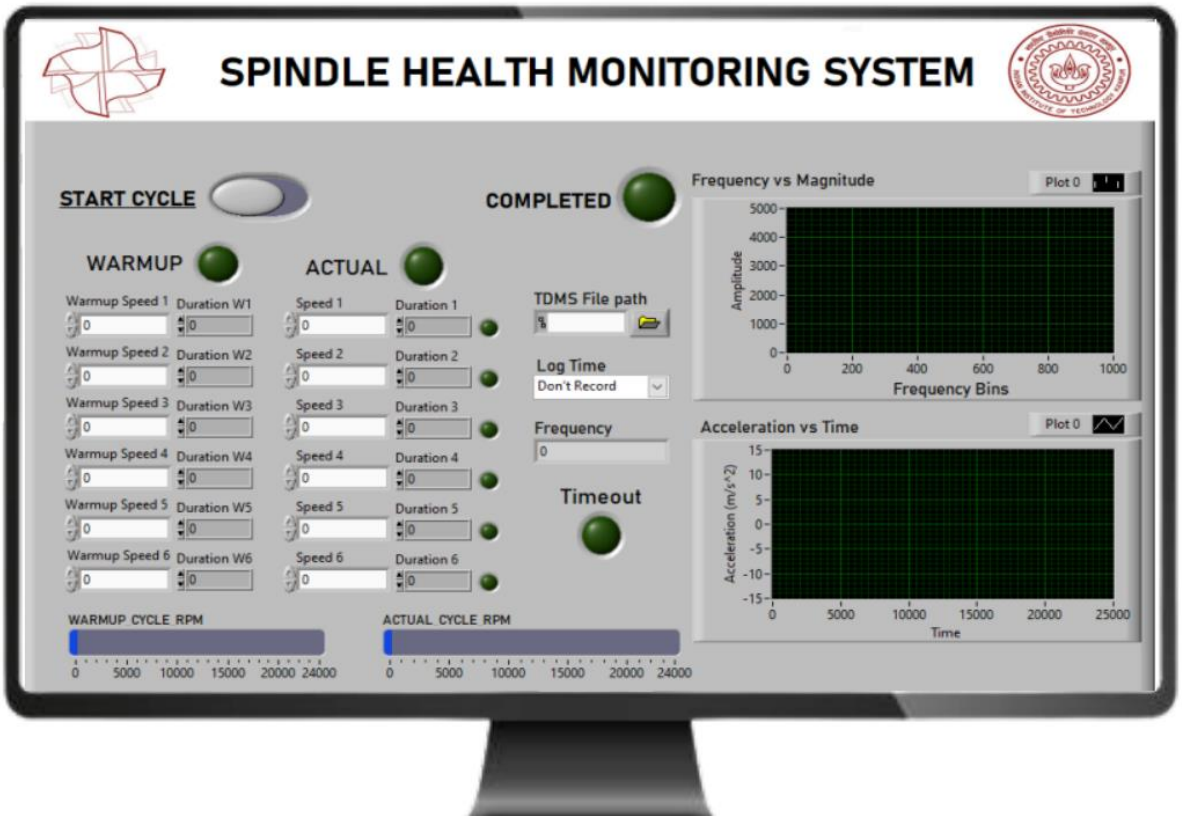

Figure 2: User Interface for LabVIEW setup

b) With Raspberry Pi

Step-by-Step Process for Vibration Signal Acquisition System:

1. Constant Current Source for IEPE Sensor

- Configure an **LM317 voltage regulator** as a **constant current source**.
- Connect a **560 Ω resistor** between the **output and adjustment pins** of the LM317.
- This sets a current of approximately **2.2 mA** to excite the IEPE sensor.
- Use a **24 V DC power supply** to provide enough compliance voltage for sensor operation under varying loads.

2. AC Signal Extraction from Sensor Output

- The sensor output contains an **AC vibration signal** overlaid on a **DC bias (~10–12 V)**.
- Insert a **1 µF coupling capacitor** in the signal path to **block the DC component** and allow only the AC signal to pass.

3. Shifting Signal to ADC-Compatible Range

- After DC removal, the AC signal oscillates around **0 V**, which is not suitable for the MCP3208 ADC (0–3.3 V range).
- Create a **biasing network** using a **voltage divider**:
  - Connect two **1 kΩ resistors** in series between **3.3 V** and **GND**.
  - Tap the **midpoint (1.65 V)** to shift the AC signal to **oscillate around 1.65 V** (±1.65 V swing).
- Connect the output of the capacitor to this midpoint to superimpose the AC signal onto the DC bias.
- Add a **100 kΩ resistor to ground** at the same point to stabilize the voltage and prevent charge buildup.

4. Buffering the Signal Using an Op-Amp

- Use a **TL072 op-amp** configured as a **voltage follower (buffer)**:
  - This provides **high input impedance** and **low output impedance**.
  - Prevents loading effects from the ADC that might distort the signal.
- Power the TL072 using a **±9 V dual supply** to ensure proper linear operation (only the positive half of the swing is needed).

5. ADC Signal Acquisition

- Connect the **buffered signal output** to **channel CH0** of the **MCP3208** (10-bit ADC).
- Power the MCP3208 using **3.3 V** from the Raspberry Pi (for both **VDD and VREF**).
- Wire the SPI interface to the Raspberry Pi:
  - DOUT → GPIO9 (MISO)
  - DIN → GPIO10 (MOSI)
  - CLK → GPIO11 (SCLK)
  - CS/SHDN → GPIO8 (CE0)

6. Data Collection and Processing via Raspberry Pi

- Use **Python scripts** on the Raspberry Pi to:

    - Initiate **SPI communication**.
    - Collect and digitize data from MCP3208.
    - Perform **real-time processing** such as **threshold detection** and **FFT analysis**.
- The Pi also handles **data logging** and allows for **system expansion** through additional GPIO-controlled features.

7. Final Outcome:

- This setup forms a **low-cost, professional-grade vibration sensing system** using:
    - Passive and active signal conditioning.
    - Raspberry Pi + MCP3208 as the digital processing and interface layer.
- It is ideal for **educational, experimental**, and even **industrial predictive maintenance**.

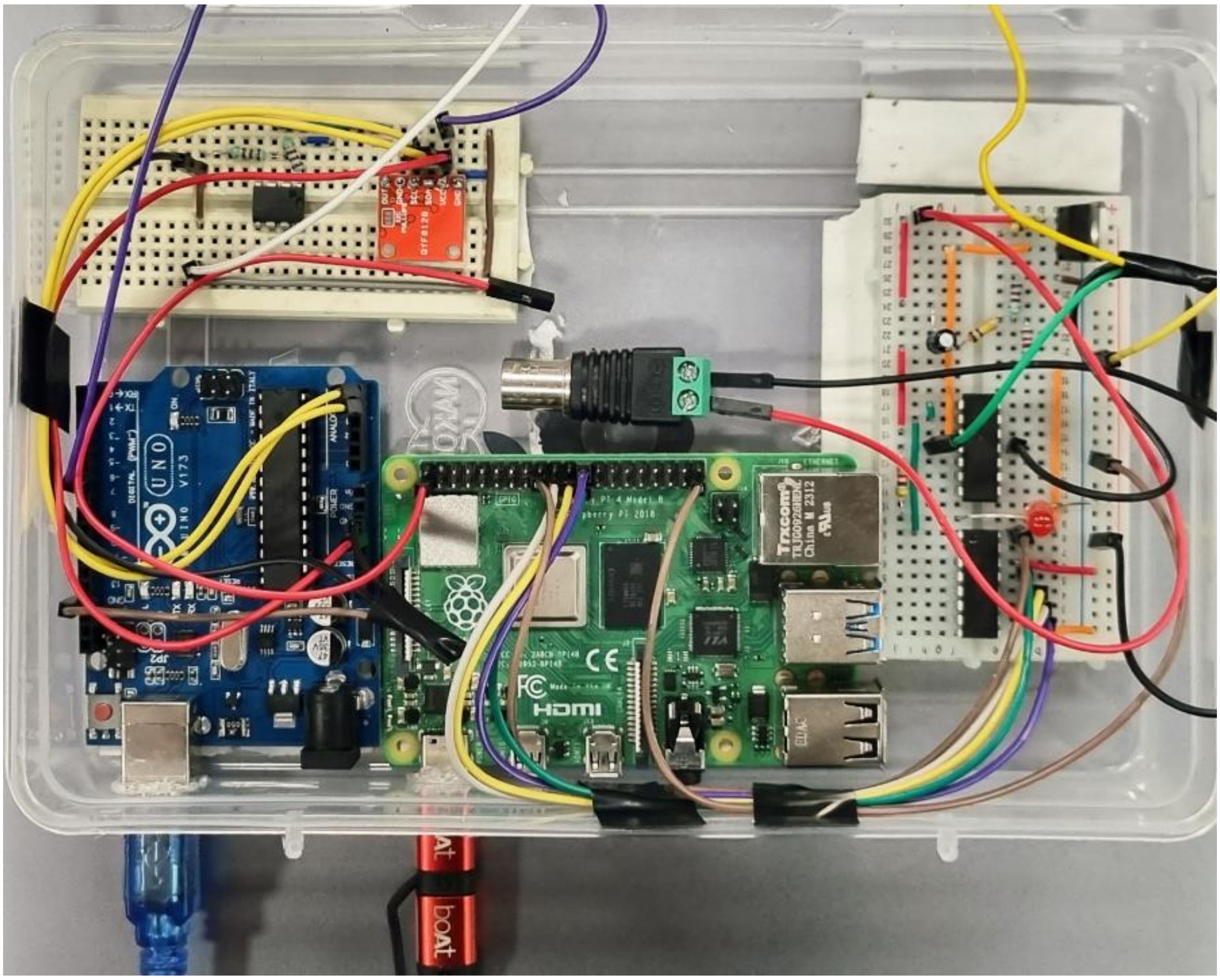
Figure 3: Raspberry Pi Triggering and Data Acquisition setup

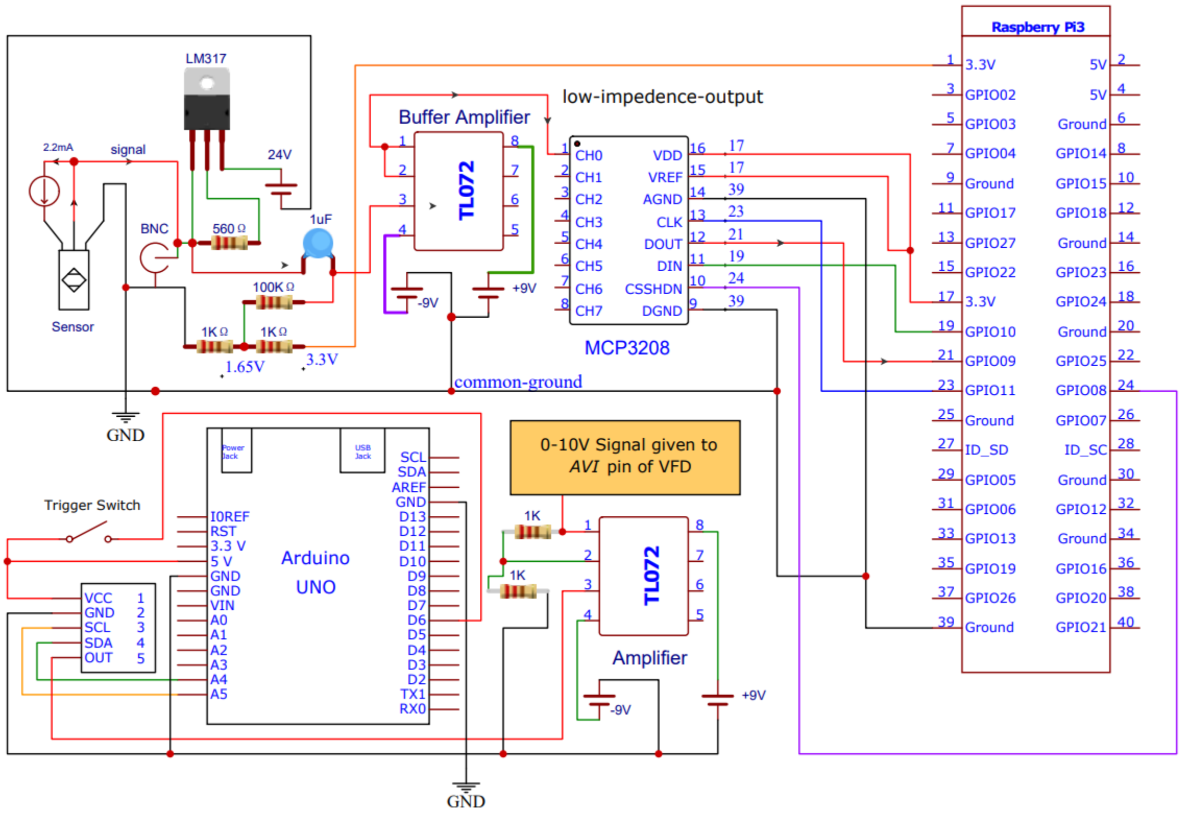

Figure 4: Circuit Diagram of the Raspberry Pi Triggering and DAQ setup

Results:

The developed spindle health monitoring system was successfully implemented and tested on a high-speed spindle setup. The NI CompactRIO-based system demonstrated precise control over spindle speed via the Variable Frequency Drive (VFD) and accurate vibration data acquisition using the NI 9234 module. The one-click warm-up functionality thermally stabilized the spindle before measurement, reducing temperature-induced signal distortions.

Real-time FFT analysis in the LabVIEW environment enabled immediate detection of spectral peaks related to common faults such as imbalance and bearing wear. Importantly, the system logged vibration data only at user-specified RPMs, optimizing storage and focusing analysis on critical operational zones. Across multiple spindle speeds, the collected vibration signals showed consistent behavior, with clearly observable amplitude increases and frequency shifts during induced fault conditions like imbalance or early-stage bearing damage. The system demonstrated high sensitivity, detecting even low-severity changes in vibration characteristics.

In parallel, the Raspberry Pi-based version was also implemented and benchmarked. It utilized a custom signal conditioning circuit with an LM317 constant current source, AC coupling network, and a TL072 buffer amplifier for impedance matching. The MCP3008 ADC enabled 10-bit resolution sampling at frequencies up to 25.6 kHz. Python scripts managed SPI communication, live plotting, and conditional logging based on amplitude thresholds.

The following comparison illustrates the performance of both systems using a 159 Hz calibration signal with an acceleration amplitude of 10 m/s^2. The **LabVIEW** system's acceleration vs. time and FFT plots reveal a clean sinusoidal waveform and a sharp spectral peak at 159 Hz. Similarly, the **Raspberry Pi** setup accurately captured the same signal, with the FFT clearly indicating the target frequency. This validates the effectiveness of the low-cost system in detecting key frequency components with good signal fidelity.

While the Raspberry Pi system lacks the multi-channel capacity and processing power of the CompactRIO, it proves highly effective for single-sensor, cost-sensitive applications such as educational use, basic industrial diagnostics, or distributed sensor nodes. The strong correlation between both platforms in terms of frequency accuracy and waveform shape confirms the viability of our cost-effective solution for real-world spindle health monitoring.

Comparison of Vibration Signal Acquisition and FFT Output (159 Hz Calibration Signal):

NI LabVIEW System (Figure 5) vs. Raspberry Pi Setup (Figure 6)

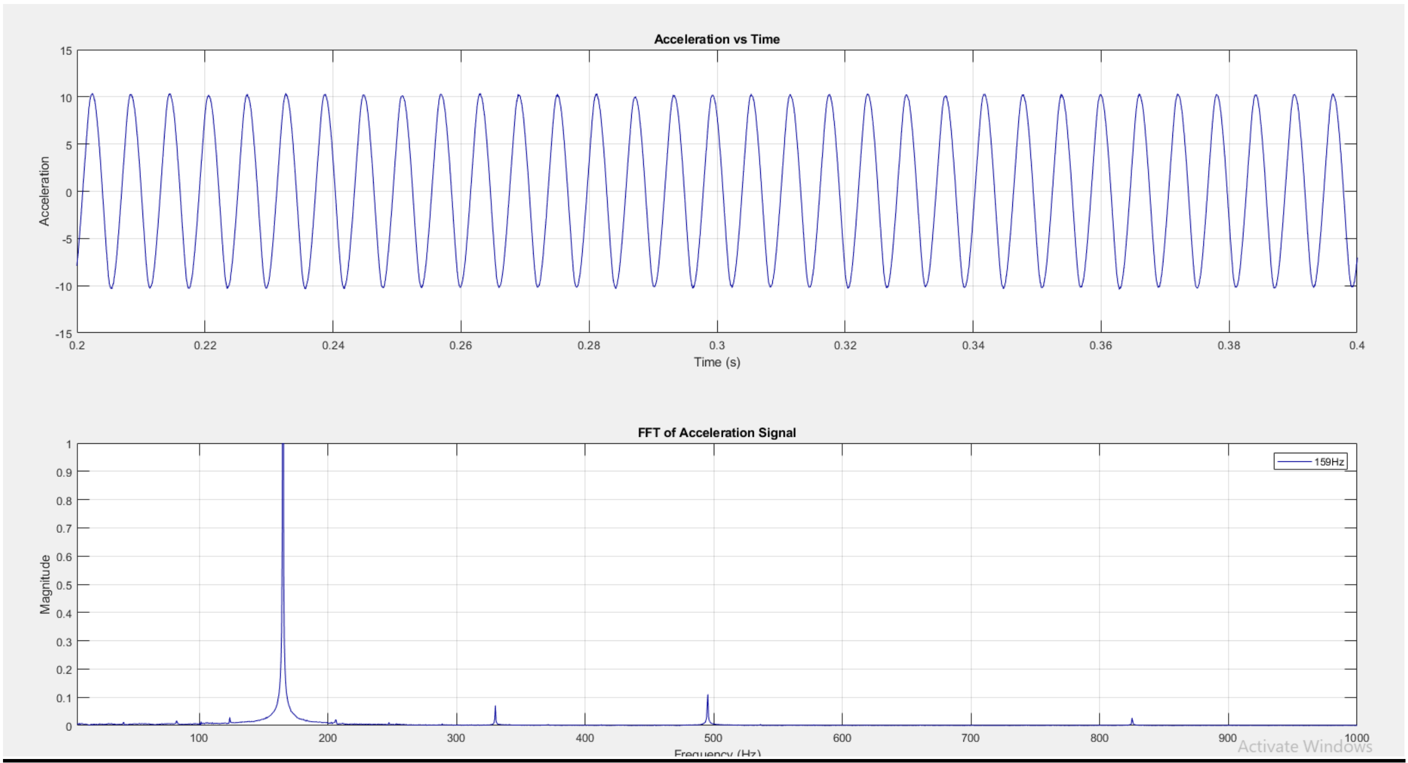

Figure 5

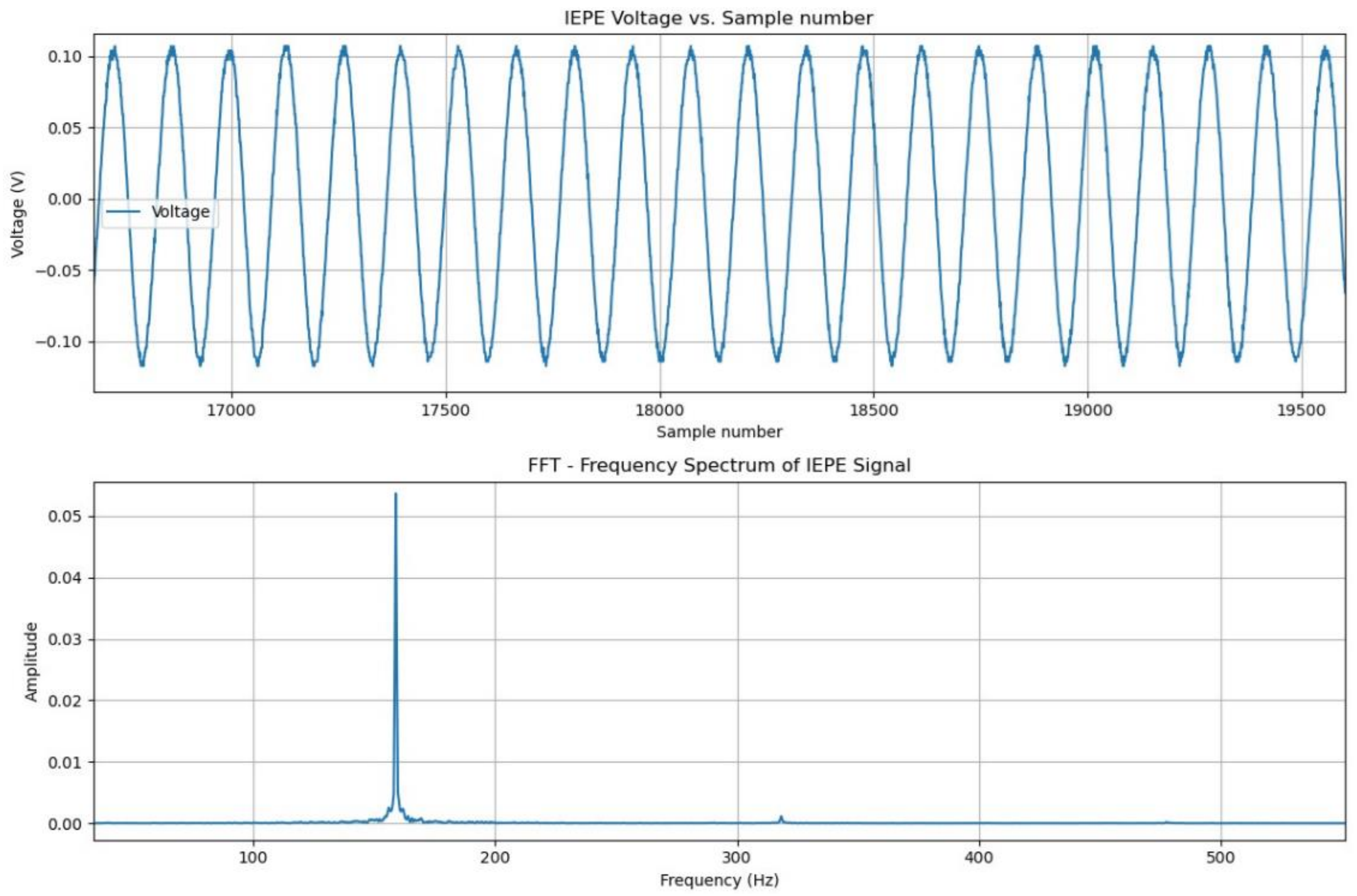

Figure 6

FFT Spectrum Indicating Misalignment

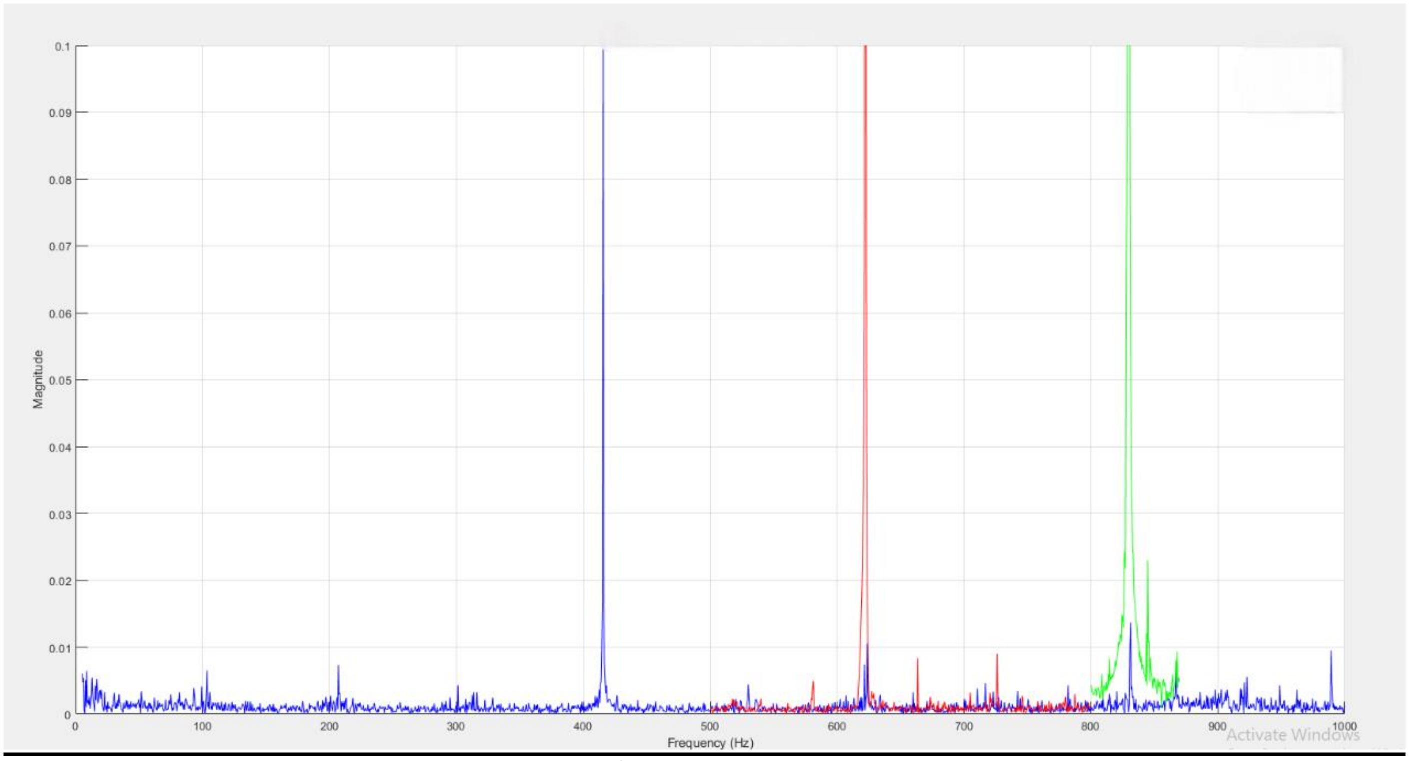

Figure 7: FFT plot for 200 Hz, 300 Hz and 400 Hz.

FFT spectra for 200 Hz, 300 Hz, and 400 Hz spindle speeds show dominant peaks at their second harmonics (2× RPM)—a key indicator of axial misalignment. The presence of these higher-order peaks suggests potential combined faults like looseness or coupling issues, demonstrating the system's capability for fault classification using frequency-domain analysis.

Conclusion:

This project successfully demonstrates a dual-platform spindle health monitoring system, combining the industrial-grade precision of NI CompactRIO with the affordability and simplicity of a Raspberry Pi-based solution. Together, these systems enable flexible deployment across a range of operational environments.

The CompactRIO setup leverages high-speed DAQ modules and LabVIEW-based real-time control to perform automated warm-up, RPM-specific logging, and multi-channel FFT analysis. Meanwhile, the Raspberry Pi system, built with a custom IEPE signal conditioning circuit and lightweight Python scripts, offers a cost-effective alternative for basic diagnostics.

A calibrated 159 Hz, 10 m/s² signal was used to validate the accuracy of both platforms, demonstrating strong correlation in signal fidelity. Beyond signal acquisition, the system performs frequency-domain transformation and **classifies spectral components** to identify fault signatures. By analysing harmonics and fault-specific frequency bands, the system can **detect combined defects such as misalignment, imbalance, and bearing wear**, enabling early and reliable fault classification.

This approach bridges the gap between high-end precision and accessible diagnostics. Its modularity and effectiveness make it highly adaptable for broader industrial applications, with strong potential for future extensions like cloud dashboards and machine learning–based automated diagnostics