Diagnostics for Periodically Operated Actuators

by

Łukasz Marek Huchel

B.S., Silesian University of Technology (2013)
M.S., Masdar Institute of Science and Technology (2015)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2021

© Massachusetts Institute of Technology 2021. All rights reserved.
Diagnostics for Periodically Operated Actuators

by

Łukasz Marek Huchel

Submitted to the Department of Electrical Engineering and Computer Science on May 18, 2021, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

Increasing constraints on quality, reliability and minimum downtime require the revision of existing maintenance approaches. Preventive maintenance, or even more reactive maintenance, require information about a system’s condition in order to enable predictive maintenance approach. Condition monitoring requires efficient sensing and data processing for extraction of condition-related signal features. Advances in both connectivity and embedded systems enable a wide range of possibilities in the field of condition monitoring.

This thesis develops signal processing tools and hardware solutions optimized for, but not limited to, diagnostics of periodically operated actuators. These actuators are mechanical or electromechanical systems that experience non-uniform loads during an operating cycle. The platform presented in this thesis serves state-of-the-art vibration and acoustic measurements and combines the quality of high-end acquisition systems with the portability of IoT devices. This allows for temporary field installations and monitoring of critical industrial equipment. Cyclostationary analysis enables diagnostics based on signals with strong random components by extracting modulation signatures otherwise unattainable by conventional time or frequency domain analysis, as demonstrated with applications to diaphragm pumps and cutting tools. An extension to the Integrated-Electronics-Piezoelectric (IEPE) industry standard for vibration measurements stretches the applications to a wide range of measurands like temperature, pressure or mechanical strain. These stretched capabilities enable a more unified sensing strategy and decrease complexity of the condition monitoring systems; thus, it further supports miniaturization and on-the-edge applications.

Thesis Supervisor: Steven B. Leeb
Title: Professor of Electrical Engineering and Computer Science
Acknowledgments

First of all, I would like to thank my thesis committee, prof. Steven Leeb, prof. James Kirtley and prof. Jan Helsen, for their valuable comments and help. I want to thank my advisor prof. Steven Leeb for all the time and effort he spent guiding me during the Ph.D. journey. You were always a great support and a source of advice in academic, as well as non-academic, matters which I will remember for a long time. It was a great pleasure for me to work in your lab and learn from you during last four years.

Without support and patience of my wife Anna last four years would have been much more difficult. I would have never written this thesis without support of my family. My parents have always supported me and provided conditions which made it possible for me to be in my current position. I will be always grateful for that. I would like to thank my grandparents and brother for giving me excellent family and great memories. All of that was an immense help in my life and I love you all.

From starting my academic journey in Poland to finishing my Ph.D. I have met many great people that have inspired me and helped to become a better engineer. All of the colleague students and staff in the LEES (GEM) laboratory at MIT were friendly on a personal level and valuable with engineering advice. I want to express my gratitude to Peter Lindahl, Daisy Green, Spencer Shabshab, Manuel Gutierrez, Eric Ponce, Erik Saathoff, Adedayo Aderibole, Andre Aboulian, Kahyun Lee, Coast Guard officers (Thomas, Devin, Andrew, Brian, Joey), Mike Ranjram, Colm O’Rourke, Krishan Kant Bhalla, Sajjad Mohammadi, Mohammad Qasim, Alexander Jurkov, Anas Al Bastami, Tony Zhang, Jessica Boles, Mike Po-Hsu Huang, David Otten, Yiou He, Aaron Langham, Rakesh Kumar, Kevin Kircher and everyone that I was lucky to work with. I also want to acknowledge my friends from Poland, colleagues I met during my studies at the Silesian University of Technology and Masdar Institute of Science and Technology.

I want to acknowledge Thomas Krause for the work we did together. It was a pleasure to work with you; I am sure that you will further contribute and improve
some of the ideas and hardware presented in this thesis. Industrial insight provided by Tomasz Ługowski was very helpful, especially with the CNC milling machine setup.

Thank you,
Łukasz Huchel
## Contents

1 Introduction 23
  1.1 Background and Problem Statement ............................... 23
  1.2 Thesis Outline ......................................................... 28

2 Signal Processing for Periodically Operated Actuators 31
  2.1 Cepstral Analysis ...................................................... 35
    2.1.1 Generalized Principle of Superposition ....................... 36
    2.1.2 Cepstrum ........................................................... 38
    2.1.3 Applications of Cepstral Analysis ............................ 40
    2.1.4 Application of Cepstral Analysis for Diagnostics and Fault Detection .................................................. 43
    2.1.5 Properties of Cepstrum .......................................... 46
  2.2 Spectral Correlation Analysis ....................................... 48
    2.2.1 Cyclostationary Analysis ......................................... 49
      2.2.1.1 Empirical Approach ...................................... 49
      2.2.1.2 Probabilistic Approach .................................. 58
    2.2.2 Properties of Spectral Correlation ............................ 64

3 Diagnostics for Periodically Operated Actuators 71
  3.1 Diagnostics for Diaphragm Pumps ................................... 71
    3.1.1 Diaphragm Pump Experiment .................................... 74
    3.1.2 Vibration Based Approach .................................... 76
      3.1.2.1 Valve Clogging ........................................... 78
5.1.1 Switched Capacitor Filter ............................................ 134
5.1.2 Frequency Detection ................................................... 137
5.2 Hardware Support for IEPE Transducer ............................... 139
  5.2.1 Processing Pipeline ............................................... 140
  5.2.2 Sampling and Filtering ......................................... 141
  5.2.3 Time-frequency Conversion ..................................... 142
  5.2.4 Frequency to Resistance Conversion ............................ 144
  5.2.5 Resistance to Temperature Conversion ......................... 146
  5.2.6 Temperature Conversion ....................................... 148
  5.2.7 Pipeline Performance ........................................... 148
  5.2.8 Memory ............................................................. 149
  5.2.9 Display ............................................................ 149
  5.2.10 Device Utilization .............................................. 151
  5.2.11 Sensor Resolution .............................................. 151
5.3 Acquisition Hardware .................................................. 153
  5.3.1 Analog Front-End ............................................... 154
    5.3.1.1 Current Source and Coupling ............................ 154
    5.3.1.2 Single-Ended Amplifier .................................. 160
    5.3.1.3 Single-Ended to Differential Converter and Filter .... 164
    5.3.1.4 Analog Inputs ............................................. 169
    5.3.1.5 Overall Transmission Path ................................ 171
  5.3.2 Analog-to-Digital Converter ................................... 178
  5.3.3 Wireless Platform .............................................. 183
  5.3.4 Wired Platform .................................................. 193

6 Software and Firmware Support ........................................ 197
  6.1 Firmware and Software for the Wireless Platform .................. 197
    6.1.1 Embedded Device Firmware .................................. 198
    6.1.2 Client Software .............................................. 207
  6.2 Firmware and Software for the Wired Platform ..................... 208
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Typical maintenance approaches.</td>
<td>25</td>
</tr>
<tr>
<td>1-2</td>
<td>Sensing and processing approach trade-off.</td>
<td>27</td>
</tr>
<tr>
<td>1-3</td>
<td>Example application fields.</td>
<td>28</td>
</tr>
<tr>
<td>2-1</td>
<td>Classification of signals.</td>
<td>32</td>
</tr>
<tr>
<td>2-2</td>
<td>Generalized linear filter block diagram.</td>
<td>37</td>
</tr>
<tr>
<td>2-3</td>
<td>Implementation of invertible system T.</td>
<td>37</td>
</tr>
<tr>
<td>2-4</td>
<td>Sample signal with undistorted echo.</td>
<td>41</td>
</tr>
<tr>
<td>2-5</td>
<td>The complex cepstrum for the sample signal.</td>
<td>42</td>
</tr>
<tr>
<td>2-6</td>
<td>The sample signal with echo and recovered wavelet.</td>
<td>42</td>
</tr>
<tr>
<td>2-7</td>
<td>Vibration signal of a diaphragm pump.</td>
<td>44</td>
</tr>
<tr>
<td>2-8</td>
<td>Magnitude spectrum of a diaphragm pump vibration signal.</td>
<td>45</td>
</tr>
<tr>
<td>2-9</td>
<td>Logarithm of the magnitude spectrum of the vibration signal.</td>
<td>45</td>
</tr>
<tr>
<td>2-10</td>
<td>Power cepstrum of the vibration signal.</td>
<td>46</td>
</tr>
<tr>
<td>2-11</td>
<td>Power cepstrum of a CNC milling machine vibration signal.</td>
<td>47</td>
</tr>
<tr>
<td>2-12</td>
<td>Average deviation as a function of SNR.</td>
<td>48</td>
</tr>
<tr>
<td>2-13</td>
<td>Synchronously-averaged first-order cyclostationary signal.</td>
<td>51</td>
</tr>
<tr>
<td>2-14</td>
<td>Time-variant autocorrelation of second-order cyclostationary signal.</td>
<td>52</td>
</tr>
<tr>
<td>2-15</td>
<td>The power spectral density of the deterministic signal with suppressed carrier.</td>
<td>65</td>
</tr>
<tr>
<td>2-16</td>
<td>The spectral coherence map of the deterministic signal with suppressed carrier.</td>
<td>66</td>
</tr>
<tr>
<td>2-17</td>
<td>The power spectral density of the deterministic signal with full carrier.</td>
<td>66</td>
</tr>
</tbody>
</table>
2-18 The spectral coherence map of the deterministic signal with full carrier. 67
2-19 The power spectral density of the random signal. 68
2-20 The spectral coherence map of the random signal. 68
2-21 The spectral coherence map of vibration data of a milling machine. 69
2-22 Vibration data of a milling machine. 69
2-23 The spectral coherence map of vibration data of a diaphragm pump. 70
2-24 Vibration data of a diaphragm pump. 70

3-1 IEPE accelerometer data during an impulse response. 74
3-2 Experimental setup with diaphragm pump. 76
3-3 Increasing clogging, presented starting from the left side. 77
3-4 Fixture for diaphragm damaging. 77
3-5 The spectral coherence map for pump vibration signal. 78
3-6 Average flow as a function of the clogging stage. 79
3-7 The MES zoomed in at fundamental cyclic frequency. 79
3-8 The magnitude spectrum of IEPE accelerometer reading. 80
3-9 Reverse emulation of the clogging process. 81
3-10 The MES during the reverse clogging experiment. 82
3-11 The resonant peak during the reverse clogging experiment. 82
3-12 The MES for good and damaged diaphragms. 83
3-13 The MES evaluated at the third harmonic. 83
3-14 The magnitude spectrum for different diaphragms. 84
3-15 Comparison of current magnitude spectra, valve clogging. 85
3-16 Comparison of current magnitude spectra, diaphragm degradation. 86
3-17 The view of milling machine experimental setup. 89
3-18 The spectral coherence map for sample vibration data. 90
3-19 Artificial end mill degradation. 91
3-20 The simulated signal for the artificial degradation experiment. 92
3-21 The MES for simulated data. 92
3-22 The vibration signal for the artificial degradation experiment. 93

12
3-23 The MES for the artificial degradation experiment. ................. 94
3-24 Comparison of the average deviation for power cepstrum and MES. 95
3-25 Good and bad 8mm carbide end mills. ............................. 96
3-26 The illustration of ADOC and RDOC. ............................... 97
3-27 Cut 2 comparison for 8 mm end mills. .............................. 98
3-28 Non-optimal cutting configuration. ................................. 101
3-29 Tool damage with non-optimal cutting parameters. ............... 102
4-1 High-level diagram of a typical current loop. ........................ 105
4-2 Typical IEPE Interface. .............................................. 105
4-3 Piezoelectric material circuit models: a) charge source, b) voltage source. 108
4-4 The charge amplifier .................................................. 109
4-5 The voltage amplifier ................................................... 113
4-6 Structure of the proposed IEPE transducer: a) input stage, b) differential amplifier, c) power stage, d) oscillator stage. .......... 115
4-7 The oscillator stage. .................................................... 116
4-8 The transient response of the oscillator. ............................ 118
4-9 The oscillator frequency as a function of input voltage. ........... 119
4-10 The proposed IEPE transducer and custom aluminium case. .... 120
4-11 The temperature measurement comparison. .......................... 123
4-12 The strain measurement. ............................................. 124
4-13 Temperature as a function of oscillator frequency. ............... 125
4-14 IEPE microphone: a) large-signal model, b) small-signal model. 127
4-15 The IEPE microphone with custom aluminum case. .............. 129
4-16 Microphone total harmonic distortion comparison. ............... 129
4-17 The IEPE microphone sensitivity plot. ................................ 130
4-18 Data acquisition device (left) with accelerometer (top right). ... 131
4-19 The proposed sensing strategy in an example scenario. .......... 131
5-1 A controllable notch filter. ........................................... 135
5-2 High-level block diagram of switched capacitor filter. .......... 135
| 5-3 | Hysteresis comparator circuit | 137 |
| 5-4 | Sample current signal and magnitude spectrum of the filtered signal | 139 |
| 5-5 | High-level block diagram of the processing pipeline | 141 |
| 5-6 | The magnitude frequency response of the band-pass filter | 142 |
| 5-7 | Sample output of a 1024 long FFT | 143 |
| 5-8 | The graphical interface | 150 |
| 5-9 | Block diagram of the VGA text controller | 150 |
| 5-10 | The effect of lack of synchronization between RAM and ROM memories | 151 |
| 5-11 | View of the device utilization report | 152 |
| 5-12 | View of the power analysis report | 152 |
| 5-13 | Temperature as a function of the oscillator frequency | 153 |
| 5-14 | Resistance as a function of the oscillator frequency | 153 |
| 5-15 | Analog front-end block diagram | 154 |
| 5-16 | XTR111 diagram | 155 |
| 5-17 | XTR111-related noise | 156 |
| 5-18 | Circuit diagram of custom current source | 157 |
| 5-19 | Sample acoustic signal for current source performance comparison | 158 |
| 5-20 | Sample vibration signal for current source performance comparison | 159 |
| 5-21 | AC coupling network | 159 |
| 5-22 | Circuit diagram of the non-inverting amplifier | 161 |
| 5-23 | Circuit diagram for noise analysis of the non-inverting amplifier | 162 |
| 5-24 | Output voltage noise spectral density for the analytical model and SPICE simulation of the non-inverting amplifier | 163 |
| 5-25 | Circuit diagram of the fully-differential amplifier | 164 |
| 5-26 | Frequency response of the fully-differential amplifier | 165 |
| 5-27 | Circuit diagram of the extended fully-differential amplifier | 166 |
| 5-28 | Frequency response of the fully-differential amplifier with additional pole | 167 |
| 5-29 | Circuit diagram for noise analysis of the fully-differential amplifier | 167 |
| 5-30 | Simplified circuit diagram for noise analysis | 168 |
5-31 Output voltage noise spectral density for the analytical model and
SPICE simulation of the fully-differential amplifier. ................. 170
5-32 Input type selection switches. ........................................... 170
5-33 Analog front-end for the voltage inputs. ............................ 171
5-34 A 3 Vₚ input signal. ....................................................... 172
5-35 Scaled and biased 3 Vₚ input signal. ................................. 173
5-36 Signal at the output of the non-inverting amplifier. ............... 173
5-37 Signals at the outputs of the fully-differential amplifier. ......... 174
5-38 Differential signal at the output of the fully-differential amplifier. . 174
5-39 Gain selection for the IEPE channel. ................................. 176
5-40 Output voltage noise spectral density for the analytical model and
SPICE simulation of the overall circuit. ................................. 177
5-41 Sampling capacitor buffer. .............................................. 180
5-42 High-level diagram of the embedded platform. ..................... 185
5-43 Views of both PCBs stacked together. ............................... 186
5-44 Custom machined aluminium case. .................................... 186
5-45 Custom case without cover. .......................................... 187
5-46 IEPE front-end U.FL connectors. ...................................... 187
5-47 Analog inputs connector. .............................................. 188
5-48 Power configuration headers. ......................................... 189
5-49 Programming headers. .................................................. 190
5-50 Wi-Fi antenna selection. .............................................. 190
5-51 External USB port. ...................................................... 191
5-52 Recommended µSD card. ............................................... 191
5-53 Block diagram of the on-board voltage levels........................ 192
5-54 Auxiliary connector H9. ............................................... 193
5-55 Solid-state relays output connector. ................................. 194
5-56 Developed hardware in a power monitor. ............................ 194
5-57 Views of the developed hardware. .................................... 195
5-58 Device with the external SDRAM adapter board. ..................... 196
| 6-1 | Pseudo-code of the non-RTOS application loop. | 199 |
| 6-2 | Role swap between devices. | 202 |
| 6-3 | Block diagram of the on-board voltage levels, early revisions. | 203 |
| 6-4 | Disturbance on the 3.3 V rail. | 203 |
| 6-5 | Disturbance in the acquired data. | 204 |
| 6-6 | Disturbance in the acquired data, final revision. | 204 |
| 6-7 | Trigger signal input pin. | 206 |
| 6-8 | Pseudo-code of the TCP client software. | 208 |
| 6-9 | Block diagram of the RTOS-based firmware. | 209 |
| 6-10 | Code execution profile. | 211 |
| 6-11 | Code execution profile, RTOS impact. | 212 |
| A-1 | Help menu in the command line. | 229 |
| A-2 | Command line error message. | 230 |
| A-3 | Help menu in the serial interface. | 231 |
| A-4 | Deleting/downloading multiple files. | 231 |
| A-5 | Deleting/downloading above 99 files. | 232 |
| A-6 | Tested SD card. | 232 |
| A-7 | The CC31XXEMUBOOST-CC3100 programming connection. | 233 |
List of Tables

3.1 Setup Details, Diaphragm Pump ........................................ 75
3.2 Setup Details, Milling Machine .......................................... 90
3.3 Optimal Cutting Parameters .............................................. 97
3.4 The MES Table for 6 mm Tool ........................................... 99
3.5 The MES Table for 8 mm Tool ........................................... 99
3.6 The MES Table for ER16 Collet and 8 mm Tool ....................... 100

5.1 Switched Capacitor Filter Circuit Parameters ......................... 136
5.2 Hysteresis Comparator Circuit Parameters ............................ 138
5.3 Module Latency .............................................................. 144
5.4 Module Latency, Frequency Conversion ................................. 146
5.5 Module Latency, Resistance Conversion ................................. 148
5.6 Fixed Point Performance, Temperature ................................. 149
5.7 Fixed Point Performance, Sensed Voltage .............................. 149
5.8 Sensitivity Data ............................................................. 152
5.9 Modulation Range .......................................................... 153
5.10 Current Source Circuit Parameters ...................................... 157
5.11 Coupling Network Circuit Parameters .................................. 160
5.12 Non-inverting Amplifier Circuit Parameters ......................... 161
5.13 Non-Inverting Amplifier Noise Gains ................................... 162
5.14 Fully-Differential Amplifier Circuit Parameters ..................... 166
5.15 Fully-Differential Amplifier Noise Gains ............................. 168
5.16 Analog Input Circuit Parameters ...................................... 172
5.17 External Capacitor $C_{ext}$ ........................................ 181
5.18 Effective Source Resistance ...................................... 182
5.19 Amplifier-Converter Interface .................................... 183
5.20 Auxiliary Functions ............................................. 193

6.1 Interface Functions ............................................. 200
6.2 Firmware Performance .......................................... 211

A.1 Revision History .............................................. 233

C.1 IEPE Transducer Circuit Parameters ......................... 328
C.2 IEPE Microphone Circuit Parameters ......................... 328
Acronyms

ADC  Analog-to-digital converter.

ADOC Axial depth of cut.

AP Access point.

ASCII American Standard Code for Information Interchange.

AXI Advanced eXtensible Interface.

BRAM Block random access memory.

CMRR Common-mode rejection ratio.

CNC Computer numerical control.

CNN Convolutional neural network.

DAC Digital-to-analog converter.

EMI Electromagnetic interference.

FFT Fast Fourier Transform.

FIR Finite impulse response.

FPGA Field programmable gate arrays.

FSI Fluid-structure interaction.
HEM  High-efficiency milling.

i.i.d.  Independent and identically distributed.

IEPE  Integrated Electronics Piezo-Electric.

IoT  Internet of things.

LTI  Linear time-invariant.

MES  Mean envelope spectrum.

MESbi  Mean envelope spectrum-based indicator.

MRR  Material removal rate.

PoE  Power over Ethernet.

PTFE  Polytetrafluoroethylene.

PV  Process variable.

RAM  Random access memory.

RDOC  Radial depth of cut.

ROM  Read-only memory.

RTOS  Real-time operating system.

SINAD  Signal-to-noise and distortion.

SNR  Signal-to-noise ratio.

SPL  Sound pressure level.

STA  Station.

TCM  Tool condition monitoring.
**TCP** Transmission control protocol.

**TFTP** Trivial file transfer protocol.

**UDP** User datagram protocol.

**VGA** Video Graphics Array.

**WSS** Wide-sense stationary.
Chapter 1

Introduction

Mission-critical machinery and associated test and maintenance equipment are not always designed to speed repairs and minimize down-time. Life-cycle cost analysis and maintenance planning are not new ideas. However, we stand in the dawn of new sensing and communication technologies. Are we applying these technologies with best efficacy and return-on-investment for mission critical installations? Are sensors revealing or, rather, a source of new maintenance problems? What are the best ways to process, present, and store data to provide the fastest decision assistance? Do our procedures and tools empower or hinder operators in speeding completion of maintenance and minimizing cost?

1.1 Background and Problem Statement

Profiles for machinery maintenance costs arise from three essentially different engineering management strategies:

- Critical maintenance can occur on a scheduled or routine basis to attempt to ensure system availability, at the possible cost of an early “repair”.

- Maintenance can occur when a system breaks, essentially deferring costs to a “day of reckoning” when the system is essentially guaranteed to be unavailable and therefore not mission capable.
Operators can attempt to optimize maintenance costs and system availability by working to predict needed repairs before systems become casualties.

The third option is appealing in the sense that, at least intellectually, it sounds thrifty. We work and think, and, if efficient and effective, achieve a desired degree of mission capability with well-reasoned expenses. However, all three approaches have both obvious and also subtle trade-offs between cost and mission availability. There is not necessarily a single correct approach, as the attractiveness of each philosophy will vary with the facility, the mission, and the expense in all resources (time, labor, training, budget, convenience, availability, etc.) inherent in the operation of different systems. Even a specific analysis for a particular installation of machinery will vary with environment, budget, operator skill, and operating duty cycle, and may therefore change, possibly suddenly.

Complicating matters, we stand in the dawn of a technological change, where we expect electronics and communication systems to provide us with unprecedented levels of warning and situational awareness. These technological changes potentially introduce transformative effects for control and maintenance. Gathering useful information, however, may require the installation of an expensive and intrusive array of sensors. Can this burden be reduced? If not, can sensors and data be interpreted in different and new ways to squeeze more value from a monitoring installation? Time spent early in a system design implementing useful sensing and data reduction strategies for maintenance decision support may pay vast benefits for system availability and life-cycle cost. This thesis demonstrates new hardware and signal processing algorithms for in situ vibration monitoring. The approach and applications illustrated here, both in the laboratory and in “real-world” applications, are intended to suggest possibilities beyond vibration monitoring. The intent of this work is, in part, to illustrate considerations for designing or retrofitting a system to minimize downtime, maximize the availability of actionable information, and reduce the expense and background maintenance associated with a sensor array.

Maintenance approaches can be classified into three main groups: reactive maintenance, preventive maintenance and predictive maintenance, see Fig. 1-1. Reactive
maintenance responds to equipment failure. Preventive maintenance is typically implemented as periodic equipment inspections, commonly with established inspection routine. The goal of preventive maintenance is to minimize unexpected failures, downtime and associated loss. Maintenance intervals are designed based on the average life cycle of the equipment and do not account for the actual state of a particular piece of equipment. Alternately, predictive maintenance seeks information to estimate when service is actually required. As the name suggests, the idea is to perform a prediction based on the estimated condition of the equipment under study. A key ingredient of predictive maintenance is the ability to assess the condition of the equipment. This ability is described as condition monitoring [1]; therefore, predictive maintenance is sometimes called condition-based maintenance.

Electromechanical actuators with periodically varying loads serve as workhorse components in many industrial and commercial processes [2], [3], [4], [5]. Compressors for HVAC and high pressure air service, machine tools with periodic stamping or cutting and reciprocating pumps all experience periodic load modulation throughout a mechanical operating cycle [6], [7]. Another class of mechanical systems that present periodic modulation in signals associated with their operation are turbines and ship propellers [8]. Periodic modulations appear in many signals associated with these actuators and machines, including vibration, electrical current, speed and torque [7], [9]. Analysis of these periodic variations are an under-exploited opportunity for diagnostics. Signal processing techniques and measurement hardware can be tailored to exploit periodic signal features in order to enhance measurement accuracy [10].
and the possibilities for fault detection [11], [12], [13], [14]. Despite the deterministic nature of the periodic operation, signals of interest may also contain strong random components. For instance, propellers of underwater vessels may generate bubbles due to cavitation induced by the rotating propeller blades. Acoustic emission related to cavitation is random in nature, and thus, the random signal is modulated by the rotating propeller, resulting in an amplitude modulated random signal [15]. As a result, the signal processing approach should be tailored not only to work with various types of modulations but also with random signals that the modulation is applied to.

Non-uniform loading is not the only condition that causes periodic characteristics in the signals associated with operation of the machine under study. For instance, discrete faults in a radially loaded rolling bearing, e.g., inner race fault, can cause amplitude modulation of the vibration signal due to the non-uniform load zone [16]. Thus, the concept of periodic operation is much more general and can be applied to analysis of a wide class of systems.

In order to enable diagnostics and condition-based maintenance, appropriate sensing and analysis methods are indispensable. Relatively simple metrics like RMS value, peak value or mean value are sometimes sufficient to provide meaningful input for predictive maintenance. Sensing technologies with low bandwidth and sensitivity are relatively inexpensive. Unfortunately, a combination of low-bandwidth hardware and simple signal processing approach does not provide solution to all diagnostics problems. High-fidelity signal processing and high-bandwidth sensing solutions may be necessary in some instances; especially if expensive or critical equipment is considered. A pictorial representation of trade-offs between sensing technologies and signal processing approaches is presented in Fig. 1-2. The comparison of sensing technologies, left plot in Fig. 1-2, presents trade-off between cost and performance. In case of vibration measurements, a low-cost and low bandwidth MEMS sensor is compared with a piezoelectric accelerometer. The right plot in Fig. 1-2 provides pictorial comparison between low-complexity and low-fidelity methods with high-complexity and high-fidelity methods. In general, some sort of combination of sensing and processing approach is necessary for any diagnostics or fault detection mechanism [17].
Work presented in this thesis provides both hardware and signal processing tools for condition-monitoring, it focuses mostly on periodically operated systems. Signal processing and hardware are built to utilize characteristics of signals associated with such systems. Proposed solutions are tested on two industrial devices, i.e., a diaphragm pump [18] and a CNC milling machine [19]. However, there is a wide range of other applications, such as the examples shown in Fig. 1-3. Hardware and signal processing described in this thesis are tested on machinery in the US Coast Guard and Navy ships to help improve reliability and minimize equipment downtime. Ships present very complex systems from the monitoring point of view with an abundance of devices operating with periodic loads, ranging from main propulsion to power generators and small auxiliary pumps. Effective and practical condition monitoring systems, involving both hardware and software solutions, are critical in marine applications. Reliability of operation is only one of the benefits, while effective condition monitoring can be beneficial in minimizing acoustic signature of a vessel [20, 21].

The proposed signal processing approach utilizes properties of cyclostationary signals and enables diagnostics based on signals that have strong random components. Ultimately, a metric is proposed to simplify the decision process up-stream in the maintenance algorithm. A new sensing solution that is compatible with industrial standards is presented in this thesis. This new sensing solution unifies and extends industrial interface dedicated for vibration measurements to a wide range of measurands like strain, temperature or pressure [22]. Two versions of acquisition platform were developed to support high-quality vibration and acoustic measurements.
developed hardware supports wired and wireless connectivity to create IoT platforms performing state-of-the-art measurements. An analog front-end of both platforms is compatible with IEPE accelerometers, a *de-facto* industry standard for high-quality vibration measurements [23].

### 1.2 Thesis Outline

This thesis is organised as follows. Chapter 2 introduces signal processing tools that are valuable for condition monitoring of periodically operated actuators. Details of cepstral and spectral correlation analyses are provided with example applications and properties. Performance in the presence of additive noise is discussed and compared between methods. Spectral correlation and cyclostationary properties of many signals associated with periodically operated actuators enable metrics used in the following chapters to perform diagnostics. Chapter 3 discusses proposed signal processing approach. Applications to diagnostics of diaphragm pump and cutting tools of a milling machine are shown with experimental results. Chapter 4 is devoted to the extension of the IEPE interface. This extension enables wide range of measurements over IEPE interface, a robust and simple interface dedicated for vibration
measurements, which can now be shared between multiple sensor types. This not only simplifies device management but also allows a wider range of measurements using existing instrumentation. Chapter 5 presents custom hardware developed to support presented diagnostic techniques; analog, digital and discrete-time solutions are discussed in details. A switched capacitor notch filter is a discrete-time system allowing to adjust filter center frequency in real-time. In case of grid-connected applications, the frequency content of interest can be extracted despite the non-stationary nature of grid frequency. An analog front-end for IEPE transducer input signal is a critical component in the developed acquisition system. Low-noise and precise signal conversion is indispensable to perform high-quality analog-to-digital conversion. Thus, gain selection and noise analysis are presented with analytical approach and simulation verification. Finally, the digital system enables low-level data acquisition and file management, as well as, it enables user interaction to provide practical platform for both laboratory and in-field applications. Chapter 6 presents firmware enabling the functionalities of the digital system. Software solutions used by the user to interact with the embedded hardware and discussed as well in Chapter 6.
Chapter 2

Signal Processing for Periodically Operated Actuators

The ability to detect meaningful signatures in the acquired signals is a necessary requirement for any condition monitoring system. A signal itself is defined as any function conveying information [24]. Meaningful information is a feature in any signal representation that provides insight into the system’s state or fault occurrence. Methods used to extract information should take into account the impact of speed or load profiles that vary during operation of the system. Additionally, signals in different domains, i.e., electrical or mechanical signals, can have different properties and make separate analysis methods optimal from the diagnostics point of view. To elaborate on this observation, the most common classes of signals are presented in Fig. 2-1. Each class of signal has specific properties in particular representation, e.g., frequency or time domain representation. Faults or gradual degradation tend to alter features in those representations, and thus help to diagnose the system under study [25]. It can be seen in Fig. 2-1 that on the highest level signals are classified as either stationary or non-stationary. It is important to note that this classification has nothing to do with the definition of stationarity for random processes. It rather distinguishes signals with statistical properties constant in time (either random or deterministic in nature) from signals with time-variant properties.

A random signal is considered to be particular realization of a random process
and it’s statistical properties can be described by the ensemble averages that characterize random process. For time-invariant case, the probability density function (or probability mass function in a discrete case) of random variables that comprise random process do not depend on time and their joint distributions are independent of a shift of the time origin. Thus, the statistical properties of a stationary random process are time-invariant, e.g., first order and second order moments, or depend only on time difference, e.g., autocorrelation function. Oftentimes, random processes are not necessarily described by time-invariant probability density functions, however, they do present time-invariant first order and second order moments, as well as, an autocorrelation function that is dependent only on time difference [26]. Such processes are denoted as WSS. Contrary to the WSS case, strictly stationary random processes are described by time-invariant probability density functions. Stationary random signals in the mechanical domain can be often associated with steady-state phenomena that are hard to model and do not exhibit additive (discrete) components in the frequency domain, e.g., acoustic signal associated with turbulent flow or cavitation [16]. Another subclass of stationary signals in Fig. 2-1 contains deterministic signals, i.e., signals that can be modeled analytically and predicted as a function of time. Stationary deterministic signals can be categorized as either periodic or quasi-periodic. Both categories have discrete (additive) frequency components. In case of the quasi-periodic signals, not all of the frequency components are integer multiples
of a fundamental frequency component, contrary to periodic signals which can be described with the Fourier series. Thus, quasi-periodic signals do not resemble any periodic structure in the time domain.

The class of non-stationary signals can be divided into two categories: transient and continuous signals. The category of non-stationary continuous signals refers typically to random signals. The lack of stationarity in the continuous case can be twofold. Random signals with periodic statistical properties are denoted as cyclostationary, while other non-stationary signals have continuously time-varying statistical properties. Depending on which statistics are periodic authors define different orders of cyclostationarity [5]. The second category of the non-stationary class consists of transient signals, i.e., energy signals. Energy signals are characterized by a finite integral of instantaneous power, in other words, those signals have finite energy. Neither stationary nor cyclostationary signals can be energy signals.

Different types of signals provide different signal processing opportunities. In general, all signal types can be represented in time, frequency or joint time-frequency domains. For instance, non-stationary transient signals are often represented in time-frequency domain. Signal processing tools often used for time-frequency analysis are Wavelet transform, short-time Fourier transform or Wigner-Villa distribution [27]. The Hilbert transform can be used to calculate the instantaneous frequency of a signal as a function of time. Transient signals can be recorded, for example, in case of impulse response tests [28] or spin-down transients [29]. In case of an impulse test or spin-down, system response can be associated with a homogeneous response; response parameters like natural frequency or damping are valuable sources of information about the system condition.

As previously described, continuously varying non-stationary random signals cannot be characterized by time-invariant statistics. Additionally, due to the random nature there are no additive frequency components or clear time domain structure. From the condition monitoring point of view, such signals are not very meaningful. However, one can consider to divide a signal into sections and evaluate statistics, for example to estimate moments of the underlying distribution describing the ran-
dom process [30]. In order to treat signal in this way, ergodicity has to be assumed in each data segment, i.e., time averages in each segment are estimates of ensemble averages [26]. A more rigorous mathematical apparatus is available for a particular class of non-stationary random signals, cyclostationary random signals. Despite the non-stationary nature, statistical properties of such signals are time-variant but periodic. This turns out to be a very good model for variety of mechanical and electromechanical systems [5] because rotating or reciprocating machines operate periodically. There is an abundance of rotating devices like centrifugal pumps, turbines, propellers, milling machines, lathes and many more. Similarly, reciprocating devices like compressors, diaphragm pumps, piston pumps or internal combustion engines are workhorses of many industrial processes. In the idealized case, with perfectly uniform load across the operating period, a rotating machine may not experience any kind of modulation affecting electrical signals (like current) or mechanical signals (like vibration or acoustic signals). However, most of the rotating or reciprocating devices experience some kind of non-uniform load during the operating period. For example, strong non-uniform load is experienced by compressors, internal combustion engines or positive displacement pumps [7]. More subtle load variations are experienced by lathes or milling machines operating with high flute-count tools. All of those devices experience so-called torsional vibrations [16], i.e., variations in instantaneous rotational speed of the shaft. In case of a periodic load driven by an induction motor, the frequency of operation is altered depending on the changes in slip; thus, it is an example of frequency modulation [16]. In case of a synchronous machine drive operating in synchronism, a periodic load introduces phase modulation on the rotor angular position. Vibration of a gearbox operating under periodic load is a practical example of amplitude modulation. Variable load alters the deflection of interacting teeth and affects the vibration amplitude around tooth-meshing frequencies. In all of those cases, modulation in the form of amplitude, phase or frequency modulation will be present. In case of deterministic signals being modulated, modulation will be visible in the frequency domain as sidebands around carrier frequency being modulated. Different sidebands properties arise for different modulation types but periodic
spacing between carrier and sidebands components is preserved [25]. In case of random signals being modulated, the same reasoning can be applied, however, due to the continuous spectral content no sidebands can be distinguished. In other words, carrier frequencies are continuously distributed in the frequency domain which masks all of the potential modulation effects. In practice, both random and deterministic parts are present in the signals associated with rotating or reciprocating machines.

This chapter provides signal processing tools for analysis of signals typical for operation of periodically operating actuators. The actuator term defines any kind of mechanical or electromechanical system that experiences non-uniform load during the cycle of operation. Non-uniform loading is not the only condition that causes periodic characteristics in the signals associated with the device under study. For instance, discrete faults in the radially loaded rolling bearing, e.g., inner race fault, can cause amplitude modulation of the bearing-induced vibration signal due to the non-uniform load zone [16]. Similarly, propellers of underwater vessels generates bubbles due to cavitation induced by the propeller blades; acoustic emission associated with this phenomenon is random in nature. As a ship propagates, the random signal is modulated by the rotating propeller. Thus, the concept of periodic operation is more general. Many time and frequency domain methods can be applied to analyze signals associated with periodically operating actuators. However, certain properties are greatly exposed by the methods presented hereunder. This chapter is organised as follows. Section 2.1 provides mathematical details and application examples of cepstrum. Section 2.2 provides mathematical details and application examples of spectral correlation function.

## 2.1 Cepstral Analysis

This section introduces the concept of cepstrum. Three most common definitions of cepstrum are explained, namely: complex, real and power cepstra. The concepts of homomorphic deconvolution and generalized principle of superposition are introduced with mathematical background and application examples. Lastly, properties
of cepstrum and practical limitations are discussed.

### 2.1.1 Generalized Principle of Superposition

Conventionally, the principle of superposition can be applied to systems that can be represented by linear transformations between vector spaces. In order to apply a principle of superposition, a vector space needs to have two operations defined, namely vector addition and scalar multiplication, see (2.1).

\[
T[c_1 v_1 + c_2 v_2] = c_1 T[v_1] + c_2 T[v_2],
\]

(2.1)

where \( T \) is a linear transformation, \( c_1 \) and \( c_2 \) are scalar multipliers. Without loss of generality (2.1) can be rewritten in (2.2) by denoting vector addition and scalar multiplication on the input vector space as \( \circ \) and \( \triangleleft \), respectively. Similarly, vector addition and scalar multiplication on the output vector space are denoted as \( \bullet \) and \( \triangleright \).

\[
T[(c_1 \triangleleft v_1) \circ (c_2 \triangleleft v_2)] = (c_1 \triangleright T[v_1]) \bullet (c_2 \triangleright T[v_2])
\]

(2.2)

Definitions of \( \circ \), \( \triangleleft \), \( \bullet \) and \( \triangleright \) operations are not unique. In general, many systems that are nonlinear can be represented in the form of (2.2), and thus can be treated as a linear, in the generalized sense, transformation \( T \) between input and output vector spaces [31]. With \( \circ \) and \( \bullet \) defined as addition, \( \triangleleft \) and \( \triangleright \) defined as scalar multiplication (2.2) becomes a principle of superposition as applies to linear systems. A system that satisfies (2.2) is described as a homomorphic system [31] and the property presented in (2.2) is a generalized principle of superposition.

A particular class of homomorphic systems is of special interest in the fields of diagnostics and speech processing, i.e., a class of systems for which \( \circ \) and \( \bullet \) operations are convolution, see (2.3). In other words, the homomorphic system in (2.3) satisfies (generalized) principle of superposition under the operation of convolution.

\[
T[v_1(t) \otimes v_2(t)] = T[v_1(t)] \otimes T[v_2(t)]
\]

(2.3)
Formulation of (2.3) allows for separation of signals $v_1(t)$ and $v_2(t)$ with respect to the operation that was used to combine them, i.e., convolution. In a similar way, linear filters are designed to separate signals which have been combined in an additive manner [32]. A filtering action which targets separation of signals that were combined in a non-additive manner is termed as a generalized linear filtering. A graphical representation of the concept of generalized linear filter is shown in Fig. 2-2. The LTI system in Fig. 2-2 represents a linear filter action and system $T$ that is invertible. An invertible system $T$ is a key to explaining concept of generalized filters. The realization of system $T$ is presented in Fig. 2-3 where $FT$ and $FT^{-1}$ denote Fourier transform and inverse Fourier transform, respectively.

\[ T[v_1(t) \ast v_2(t)] = T[v_1(t)] + T[v_2(t)], \]  
(2.4)

where \( T[v_x(t)] = \mathcal{F}^{-1}\{ln(\mathcal{F}\{v_x(t)\})}\}. The LTI system in Fig. 2-2 preserves the additive nature of the signal in (2.4). Inverse of the system $T$ has the same structure, however, the complex logarithm is replaced by an exponentiation operation. Thus, signal $V_{n2}(f)$ at node 2 of the inverse system block due to input signal $v_1'(t) + v_2'(t)$ is given in (2.5).

\[ V_{n2}(f) = B^V_1(f) \times B^V_2(f), \]  
(2.5)
where $B = e$ is the exponentiation base. The last operation in the inverse system corresponds to the inverse Fourier transform. Thus, the frequency domain multiplication included in (2.5) corresponds to the convolution operation at the output of the inverse system and has the following form: $\mathcal{F}^{-1}\{B^{V'_1(f)}\} \circ \mathcal{F}^{-1}\{B^{V'_2(f)}\}$. Taking into consideration the definition of the homomorphic system in (2.3), it can be concluded that system in Fig. 2-2 is homomorphic and satisfies generalized principle of superposition.

### 2.1.2 Cepstrum

The signal at the output of system in Fig. 2-3 is commonly referred to as a complex cepstrum [26]. Considering an input signal $v(t)$ with Fourier transform $V(f)$ and properties of the complex logarithm, the complex cepstrum $\hat{v}(t)$ is expressed in (2.6).

$$\hat{v}(t) = \mathcal{F}^{-1}\{\ln(|V(f)|) + j\theta(f)\}, \quad (2.6)$$

where $\theta(f)$ is a phase spectrum of $v(t)$. Complex cepstrum has an interesting property of isolating the system response and excitation that were combined with the convolution operation. This property enables homomorphic deconvolution of analyzed signals, as it is presented in the further subsection discussing applications of cepstrum.

Complex cepstra that are causal signals are of special interest. By definition complex cepstrum $\hat{v}(t)$ is the inverse Fourier transform of $\ln(V(f))$. Thus, $\hat{V}(f) = \ln(V(f))$ at locations of $V(f)$ zeros diverges to negative infinity and at locations of $V(f)$ poles diverges to infinity. In order for the complex cepstrum to exist and be a causal signal, $V(f)$ needs to have all have poles and zeros in the left half-plane (or inside the unit circle in discrete-time representation); thus, $v(t)$ must be a minimum-phase signal. Another useful property of causal signals is that they can be reconstructed from the even part of the signal.

$$\hat{v}(t) = 2u(t)\hat{v}_{ev}(t), \quad (2.7)$$
where \( \hat{v}_{ev}(t) \) is the even part of signal, \( \hat{v}_{ev}(t) \) and \( u(t) \) are defined as follows:

\[
\begin{align*}
    u(t) &= \begin{cases} 
    1 & t > 0 \\
    0.5 & t = 0 \\
    0 & t < 0 
  \end{cases} \\
    \hat{v}_{ev}(t) &= \frac{\hat{v}(t) + \hat{v}(-t)}{2}
  \end{align*}
\] (2.9)

For real-valued signals, the even part of the signal can be associated with the real part of the Fourier transform [26]. Thus, for complex cepstrum \( \hat{v}(t) \) with real part of Fourier transform \( ln(|V(f)|) \) it can be written:

\[
\hat{v}(t) = 2u(t)\mathcal{F}^{-1}\{ln(|V(f)|)\},
\] (2.10)

It can be summarized that only the magnitude of the spectrum is needed. Another definition of cepstrum is the real cepstrum \( \hat{v}_r(t) \) defined in (2.11). By inspection of (2.10) and (2.11), it can be stated that for a minimum-phase signal real cepstrum and complex cepstrum are identical, except for a scaling factor.

\[
\hat{v}_r(t) = \mathcal{F}^{-1}\{ln(|V(f)|)\},
\] (2.11)

Even if the input signal is not minimum-phase, there is a particular relation between real and complex cepstra for real-valued inputs. As previously stated, for real-valued signals the even part of the signal can be associated with the real part of the Fourier transform. Remembering that \( ln(|V(f)|) \) is a real part of the complex cepstrum, it can be stated that (for real-valued inputs) real cepstrum \( \hat{v}_r(t) \) corresponds to even part of complex cepstrum \( \hat{v}(t) \). It should be emphasised that despite the name of complex cepstrum, signal \( \hat{v}(t) \) is real-valued for real-valued inputs [16]. The complex cepstrum can be complex-valued signal when (2.6) is applied to the analytic signal, i.e., (2.6) is applied to scaled spectrum \( V(f) \) with zero negative frequency components. Such cepstrum implementation is often denoted as analytic cepstrum [16] to distinguish it
from the complex cepstrum.

Another form of cepstrum often seen in the literature is power cepstrum \( \hat{v}_p(t) \) [33, 34]. It was originally defined in the field of signal processing related to seismic signals with echoes. Definition of power cepstrum is provided in (2.12); although, some authors [34] introduce power cepstrum as a squared expression (2.12).

\[
\hat{v}_p(t) = \mathcal{F}^{-1}\{\ln(|V(f)|^2)\}, \tag{2.12}
\]

Power cepstrum \( \hat{v}_p(t) \) can be related to real cepstrum \( \hat{v}_r(t) \) using logarithm properties: \( \hat{v}_p(t) = 2\hat{v}_r(t) \). It is interesting to note that originally authors in [33] used Fourier transform instead of inverse Fourier transform to evaluate power cepstrum.

Input signal and cepstrum are both defined in the time domain. In order to distinguish them, authors in [33] introduced term "cepstrum" as a paraphrase to "spectrum". The domain in which cepstrum is defined is commonly denoted as quefrecy, as a paraphrase to "frequency", despite having unit of time. Idea behind the term "cepstrum" is explained in details in [33]. Inspection of (2.6), (2.11) and (2.12) reveals that complex cepstrum requires a continuous phase function \( \theta(f) \), i.e., phase function must be unwrapped. Thus, using real or power cepstra results in simpler implementation; although, only complex cepstrum can be used if the original signal is to be recovered, e.g., in applications related to echo cancellation [35].

### 2.1.3 Applications of Cepstral Analysis

Echo cancelation was one of the prime motivations for the cepstral analysis [33]. It found multiple applications in the fields such as seismography, speech and sonar data signal processing. In principle, a signal \( x(t) \) with an undistorted echo can be modeled as follows:

\[
x(t) = x'(t) + \alpha x'(t - \tau), \tag{2.13}
\]

where \( \alpha \) is the attenuation factor and \( \tau \) is the delay. Using complex logarithm and power series expansion it can be shown [34] that the complex cepstrum of \( x(t) \) can
be expressed as in (2.14).

\[ \hat{x}(t) = \hat{x}'(t) + \alpha \delta(t - \tau) - \frac{\alpha^2}{2} \delta(t - 2\tau) + \frac{\alpha^3}{3} \delta(t - 3\tau) + ..., \]  

(2.14)

where \( \hat{x}'(t) \) is the complex cepstrum of \( x'(t) \). It can be seen that periodic pulses appear in the quefrency domain with spacing \( \tau \) of the echo arrival time. Additionally, if \( \hat{x}'(t) \) and the periodic pulses do not overlap in the quefrency domain, it is possible to recover fundamental wavelet \( x'(t) \) (signal without echo). Recovering of the fundamental wavelet can be achieved by liftering (paraphrase of "filtering" [33]) the high-quefrency content.

Consider the example presented in Fig. 2-4. The waveform in Fig. 2-4 can be modeled with (2.13) and corresponding complex cepstrum is presented in Fig. 2-5. By removing higher-quefrency components the original wavelet can be recovered. Fig. 2-5 presents original signal and recovered wavelet.

![Figure 2-4: Sample signal with undistorted echo.](image)

Homomorphic deconvolution makes cepstrum very attractive tool in the field of speech processing [36]. The source-filter model (2.15) is a common way to model speech \( s(t) \) [32].

\[ s(t) = x(t) \otimes v(t), \]  

(2.15)

where \( x(t) \) denotes excitation (including glottal pulse characteristic) and \( h(t) \) denotes
the overall vocal tract filter response. Using (2.4) the cepstrum of speech $\hat{s}(t)$ can be expressed as follows:

$$\hat{s}(t) = \hat{x}(t) + \hat{v}(t), \quad (2.16)$$

where $x(t)$ and $v(t)$ denote excitation and filter response cepstra, respectively. If $\hat{x}(t)$ and $\hat{v}(t)$ do not overlap in the quefrency domain, it is possible to separate effects of each component in (2.16) and convert back to time domain representation. This provides a very powerful tool in multiple problems in the field of speech processing.
2.1.4 Application of Cepstral Analysis for Diagnostics and Fault Detection

In section 2.1.3 applications of cepstrum to echo cancellation and speech processing were discussed. The key property used in the case of speech processing is homomorphic deconvolution. It allows separation of the signals of interest in the quefrency domain; in case of complex cepstrum, the signal of interest can potentially be recovered in the time domain. For a wide class of mechanical and electromechanical systems variety of phenomena can be modeled as modulation [25], either amplitude, frequency or phase modulation. Despite different characteristics of sidebands for each type of modulation [37], regular spacing between spectral components is typical. To illustrate the effect of amplitude modulation in the quefrency domain the following signal is considered:

\[ x(t) = v(t) \times (m(t) + 1) = v(t) \times m'(t), \]  

(2.17)

where \( m(t) \) is a low frequency modulation signal and \( v(t) \) is a carrier signal. Frequency domain representation of (2.17) corresponds to \( V(f) \circledast M'(f) \), i.e., convolution of Fourier transforms of \( v(t) \) and \( m'(t) \). Based on (2.6) the logarithm is applied to the frequency domain representation of (2.17). Application of logarithm does not provide a simple and useful representation, as in the case of the speech model, where multiplication in the frequency domain is converted to addition, see (2.16). Despite that, cepstrum is still a valuable tool for analysis of modulated signals.

To illustrate this property, a vibration signal of a diaphragm pump is considered [18]. The time domain representation of a vibration signal acquired at the pump’s head in the axis of diaphragm operation is presented in Fig. 2-7. The pump is operated by a two pole pair induction machine fed by a 60 Hz voltage source. Taking into account a 10:1 gear ratio, pumping action is expected in the vicinity of 3 Hz (depending on a particular slip value). The spacing of bursts in Fig. 2-7 is approximately \( \Delta T \approx 176 \text{ ms} \), i.e., it corresponds to each half cycle of pump operation. Thus, the fundamental frequency of operation equals to \( f \approx 2.83 \text{ Hz} \). The mag-
Figure 2-7: Vibration signal of a diaphragm pump.

The magnitude frequency spectrum is presented in Fig. 2-8. It can be seen that 2.83 Hz spacing between frequency components is dominant. This spacing is a consequence of a strong modulation happening at 2.83 Hz, as well as, many frequency components located at multiples of 2.83 Hz, e.g., gear meshing frequency of the internal gearbox. In order to evaluate cepstrum, the natural logarithm operation is applied to the frequency domain representation. In the field of detection and diagnostics, power cepstrum (2.12) is very common. First of all, it does not require phase unwrapping. Second of all, in this application, reconstruction back to the time domain is often not necessary. Fig. 2-9 presents the logarithm evaluated over squared magnitude spectrum, i.e., \( \ln(|X(f)|^2) \). It can be seen that the relative magnitude difference of frequency components is smaller in Fig. 2-9 than in Fig. 2-8 (when interpreted in a linear scale). The last operation necessary to evaluate power cepstrum is the inverse Fourier transform (2.12). In general, the inverse transform can be applied to signals with linear or logarithmic scale representation; in this case, it provides the time domain representation of the signal that has the spectrum presented in Fig. 2-9. Whereas the magnitude spectrum in Fig. 2-8 has strong evidence of modulation, the natural logarithm of the magnitude spectrum in Fig. 2-9 is more typical for periodic signals where fundamental frequency \( f_1 \) and its harmonics \( f_n \) are of similar magnitudes. Thus, the inverse transform of the logarithm magnitude spectrum is expected
to have time domain structure with evident periodicity of $T = \frac{1}{f_1}$ s. Power cepstrum of a vibration signal of the diaphragm pump is presented in Fig. 2-10. Indeed, it can be seen that the periodic nature of power cepstrum is dominant. The fundamental quefrency component is located at $\frac{1}{f_1} = \frac{1}{2.83} \approx 0.35$ s and higher order rahmonic components (paraphrase of "harmonic" components) are located at integer multiples of the fundamental component. The ordinate in Fig. 2-10 is scaled in NeperVolts, i.e., $1 \, NpV = ln(\frac{E}{1V})$

Based on the example, it can be concluded that cepstrum provides a valuable method to evaluate effects of families of harmonics with sidebands. This is a powerful
tool for diagnostics of rotating machinery and wide range of electromechanical systems where multiple harmonic contents are modulated simultaneously [16]; analyzing separate frequency components is not practical and case-dependent. Monitoring of cepstral components over time may be a valuable tool in diagnostics and fault detection. Depending on the fault type, energy at the existing quefrequency component is subject to change, or a new quefency component can appear, e.g., due to the new modulation component.

### 2.1.5 Properties of Cepstrum

In previous sections, motivation and mathematical background for cepstrum were introduced. In this section, practical implications of cepstrum for diagnostics and fault detection are discussed. One advantage of cepstrum is its low sensitivity to sensor placement [38]. In order to show this, a signal $x(t)$ is modeled as a convolution of excitation $e(t)$ and system response $h(t)$, i.e., $x(t) = e(t) \ast h(t)$. Due to the cepstrum properties, the excitation and the system response can often be separated in the quefrequency domain. Additionally, the system response function $h(t)$ is dependent on a transmission path between source and sensor location. In many applications, excitation signal can be successfully liftered and decoupled from the impact of transmission path; thus, minimizing sensitivity to sensor location.
Although the logarithm operation included in the definition of cepstrum enables the homomorphic deconvolution, it can affect the performance of cepstrum in the low SNR scenario. As explained in section 2.1.4, the logarithm operation diminishes the magnitude difference between spectral components. Unfortunately, the logarithm operation applies as well to undesirable signal components, increasing their relative importance. Thus, the signal in the quefrency domain can be affected in the low-SNR scenarios.

To illustrate the impact in the low-SNR scenarios, a vibration signal of a CNC machine was recorded and an additive white Gaussian noise signal was added to simulate various SNR levels. A sample power cepstrum of vibration signal with no added noise is presented in Fig. 2-11. The CNC was operated at 2150 RPM; thus, a strong family of harmonic content is expected at multiples of 36 Hz. A white Gaussian noise signal with variance designed according to (2.18) is used in the analysis. By inspection of (2.18), the deviation from the case with no added noise is expected to approach zero with increasing SNR, e.g., case with infinite SNR becomes the reference case.

![Figure 2-11: Power cepstrum of a CNC milling machine vibration signal.](image)

\[ \sigma^2 = \frac{P_{\text{sig}}}{10^{\frac{SNR}{10}}} \]  

(2.18)
where $\sigma^2$ is the noise variance, $P_{\text{sig}}$ is the average power of the reference vibration data. Average deviation for the lowest six dominant quefrency components is presented in Fig. 2-12. For SNR values below -6 dB, the deviation for cepstrum becomes excessively big, in fact the quefrency components cannot be correctly distinguished from the noise floor.

![Figure 2-12: Average deviation as a function of SNR.](image)

Cepstral analysis has several advantages from the implementation point of view. Forward and inverse discrete Fourier transforms can be efficiently implemented by the means of FFT algorithm. FFT algorithms of size up to $2^{16}$ can be efficiently calculated by embedded system or FPGA, it depends on the sampling frequency and algorithm details. Additionally, the logarithm operation has an efficient implementation in both fixed and floating point arithmetic [39, 40]. Especially power and real cepsrtas are efficient to implement because phase unwrapping can be avoided.

### 2.2 Spectral Correlation Analysis

Applications and properties of cepstral analysis revealed high potential for diagnostics, however, low-SNR scenarios may require different signal processing approach. Frequency distribution of modulation content and superior performance with signals exhibiting stationary random content make spectral correlation an interesting signal
processing tool. This section introduces concepts of cyclostationarity and spectral correlation. Spectral correlation is presented twofold, using empirical (time average) and probabilistic (ensemble average) approach. Properties of cyclostationary signals and spectral correlation are discussed in details with application examples.

2.2.1 Cyclostationary Analysis

The concept of cyclostationary processes is critical to understand and fully appreciate benefits of spectral correlation. Cyclostationary random process is a part of a broader class of non-stationary random processes, see signals classification in Fig. 2-1. A random signal is considered to be a particular realization of the underlying random process, thus, signals classification in Fig. 2-1 is applicable. Periodicity in the moments of distributions that describe the random process distinguishes cyclostationary process from a generic non-stationary random process [41]. Similarly, behaviour of synchronous time averages can reveal periodicity in the statistical properties of a cyclostationary signal [42]. Following this explanation, the analysis of cyclostationarity can be introduced twofold, either using probabilistic or empirical approach. Independently of the chosen approach, the theory of cyclostationary signals is indispensable to study random data arising from periodic phenomena [8].

2.2.1.1 Empirical Approach

In order to characterize quantities associated with a cyclostationary signal, necessary definitions can be introduced. The time-variant mean \( \hat{M}_x^N(t) \) and time-variant autocorrelation function \( \hat{R}_x^N(t, \tau) \) are defined below. The time-variant mean is commonly referred to as a synchronous average, also known as a superposed epoch analysis [41].

\[
\hat{M}_x^N(t) = \frac{1}{2N+1} \sum_{i=-N}^{N} x(t + i \cdot T_0)
\] (2.19)
\[
\hat{R}_x^N(t, \tau) = \frac{1}{2N + 1} \sum_{i=-N}^{N} [x(t + \frac{\tau}{2} + i \cdot T_0) \cdot x(t - \frac{\tau}{2} + i \cdot T_0)]
\]  

(2.20)

When the averaging period goes to infinity, the limit time-variant mean \( \hat{M}_x(t) \) and limit time-variant autocorrelation function \( \hat{R}_x(t, \tau) \) can be defined.

\[
\hat{M}_x(t) = \lim_{N \to +\infty} \hat{M}_x^N(t)
\]

(2.21)

\[
\hat{R}_x(t, \tau) = \lim_{N \to +\infty} \hat{R}_x^N(t, \tau)
\]

(2.22)

A signal exhibiting periodic limit time-variant mean, \( \hat{M}_x(t) = \hat{M}_x(t + T_0) \), is called first-order periodic [5]. A signal with periodic components in the limit time-variant autocorrelation function is defined as second-order periodic, also known as second-order cyclostationary [5].

It is common to think about the mean of a signal as a constant in a given data set. Especially in the field of electrical engineering, the mean is often associated with a DC quantity or an offset value. However, (2.19) provides a way to generalize this definition. Many physical phenomena can be modeled using (2.19). Annual weather parameters, like temperature, can be modeled as cyclically-varying. The mean temperature in each season is essentially the time-variant mean; random temperature changes contribute to fluctuations around the mean. Another physical example is the vibration signal of rotating machinery. The time-variant mean could be associated with the component of the vibration signal that corresponds to the shaft unbalance, i.e., at a given rotational speed this component is described by the kinematic model and is periodic with the period of rotation and it’s harmonics. This can be summarized as a macro-phenomenon. Fluctuations around the mean component can be associated, for example, with bearing operation. This can be summarized as a micro-phenomenon which is often considered as random in nature and hard to model analytically. A numerical example of a first-order cyclostationary signal with periodic
time-variant mean in presented in Fig. 2-13. Lines in Fig. 2-13 correspond to different averaging periods in the synchronous average formula (2.19). With $N \to +\infty$ the outcome of (2.19) would correspond to the periodic component of the considered signal.

The example of rotating machinery can be extended to second-order cyclostationary signals. For example, in case of a non-uniform load being applied to the rotating machine, the effective modulation is applied to the random part of the signal associated with machine operation. The numerical example consists of random signal (a realization of a stationary random process with Gaussian probability density function, unity variance, zero mean and white spectral content) $X_t$ amplitude modulated by a low-frequency signal $m(t) = 1 + \cos(2\pi f_1 t)$. In this scenario there will be periodicity in the time-variant autocorrelation function (2.20), see Fig. 2-14. Above examples show how time averages in (2.19)-(2.20) provide statistics that are periodic [30], i.e., cyclic in time.

![Figure 2-13: Synchronously-averaged first-order cyclostationary signal.](image)

First-order cyclostationarity can be analyzed with conventional signal processing tools because periodicity in the signal is evident. In case of a second-order cyclostationary signal, Fourier series expansion can be applied to characterize the periodicity of the autocorrelation function associated with the signal. The Fourier series synthesis
The equation can be written as follows:

\[
\hat{R}_x(t, \tau) = \sum_{m=-\infty}^{+\infty} \hat{R}_{x}^{m}\tau_0(x(\tau) e^{j2\pi m/\tau_0 t},
\]

(2.23)

where \( \hat{R}_{x}^{m}\tau_0\) is the \(m^{th}\) Fourier coefficient also known as a limit cyclic autocorrelation function. Analyzing (2.20)-(2.23) it can be concluded that \( \hat{R}_{x}^{m}\tau_0\) describes periodic components of the lag-product time-series \(x(t+\frac{\tau}{2}) \cdot x(t-\frac{\tau}{2})\). Consequently, the Fourier series analysis formula can be applied to \(x(t+\frac{\tau}{2}) \cdot x(t-\frac{\tau}{2})\) in order to define \( \hat{R}_{x}^{m}\tau_0\).

\[
\hat{R}_{x}^{m}\tau_0(\tau) = \lim_{T \to +\infty} \frac{1}{T} \int_{\frac{T}{2}}^{\frac{T}{2}} x(t+\frac{\tau}{2}) \cdot x^*(t-\frac{\tau}{2}) e^{-j2\pi m/T_0 t} dt
\]

(2.24)

The * operator denotes complex conjugate and is introduced to maintain the energy \(E\) interpretation, \(E = \hat{R}_x^0(\tau = 0)\), for complex signals. Having defined the cyclic autocorrelation function in (2.24), the periodogram-correlogram relation can be used to obtain the power spectral density \(\hat{S}_{x}^{\alpha}(f)\) known as a limit cyclic spectrum [41]. Time lag variable \(\tau\) is used to perform frequency \(f\) decomposition as shown in (2.25).

\[
\hat{S}_{x}^{\alpha}(f) = \mathcal{F}\{\hat{R}_x^\alpha(\tau)\},
\]

(2.25)
where $\alpha = \frac{m}{T_0}$ and $\mathcal{F}\{\}$ denotes Fourier transform. When $\alpha = 0$, limit cyclic spectrum $\hat{S}_x^0(f)$ becomes a conventional power spectral density $\hat{S}_x(f)$. It is important to note that (2.25) exists if and only if limit cyclic autocorrelation function exists, meaning that (2.24) converges.

Equations (2.20)-(2.25) provide motivation of cyclostationary analysis in terms of frequency decomposition of the time-variant autocorrelation function. However, the cyclic spectrum can have an alternative interpretation in terms of spectral correlation [43]. The cross-spectral analysis and statistical spectral analysis tools can be applied to show the alternative interpretation. Firstly, the cross-correlation function of two signals $u(t)$ and $v(t)$ is considered in (2.28).

\[
\hat{R}_{uv}(t, \tau)_T = \int_{t_0}^{t_0 + \frac{T}{2} + |\tau|} u(w) \cdot v^*(w)dw
\]  

(2.28)

Applying a change of variables, $t' = w + \tau/2$, and taking into account finite-length $T$ of data (2.28) is written in the form of (2.29). Normalization with factor of $\frac{1}{T}$ is included in (2.29), this allows to work with power signals, i.e., signals with non-finite energy as $T \to \infty$.

\[
\hat{R}_{uv}(t, \tau)_T = \frac{1}{T} \int_{-\frac{T}{2} + |\tau|}^{\frac{T}{2}} u(t + t') \cdot v^*(t + t' - \tau)dt'
\]  

(2.29)

Once again, based on the periodogram-correlogram relation the time-variant power spectral density $\hat{S}_{uv}(t, f)$ can be calculated as shown in (2.30), technically it is a cross-spectral density of $u(t)$ and $v(t)$.

\[
\hat{S}_{uv}(t, f)_T = \mathcal{F}(\hat{R}_{uv}(t, \tau)_T)
\]  

(2.30)

For now, the reduction in the integration limit in (2.29) due to the lag variable $\tau$ is
neglected. Equation (2.30) can be simplified further:

\[
\hat{S}_{uv}(t, f) = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} u(t + t') \cdot v^\ast(t + t' - \tau) \cdot e^{-j2\pi f\tau} dt' d\tau
\] (2.31)

Using the definition of the Fourier transform of \(v(t)\) and assuming real-valued signal \(v(t)\) it can be written that:

\[
V^\ast(f) = \int_{-\infty}^{+\infty} v(-\tau)e^{j2\pi f(-\tau))} d\tau
\] (2.32)

With (2.32), the shift property of the Fourier transform and change of variables, (2.31) is presented in the simplified form of (2.33).

\[
\hat{S}_{uv}(t, f) = \frac{1}{T} U(t, f) \cdot V^\ast(t, f)
\] (2.33)

In general, as \(T \to \infty\) in (2.30) the underlying windowing function becomes infinitely wide, see (2.34). In case of the rectangular window, Fourier transform of the window function converges to a Dirac impulse. Consequently, the obtained power spectral density should have no interference due to the frequency domain convolution with the frequency response of the windowing function.

\[
U(t, f) = \mathcal{F}(u(t)) = \mathcal{F}(u'(t) \cdot w(t)) = U'(f) \ast W(f),
\] (2.34)

where \(u(t)\) is the windowed data segment, \(w(t)\) is the windowing function with the Fourier transform \(W(f)\), \(u'(t)\) is the original data. However, with the presence of random content in the signal the limit of cross-spectral density \(\lim_{T \to \infty}[\hat{S}_{uv}(t, f)]\) does not exist in general [41]. In other words, the estimation process of power spectral density introduces random variations in the spectrum and increasing the observation time does not resolve this limitation [26]. It can be shown that for a purely random signal two frequency components are completely uncorrelated, independently of how close they are to each other [42]. As a remedy to this limitation, statistical spectrum analysis can be applied [41]. A common approach to obtain a statistical spectrum is
time averaging, also known in the form of periodogram averaging [26] as discussed below.

Available data can be divided into \( K \) segments. Discussion of segment selection and possible segment overlap is presented in [26]. The estimate of the power spectral density of each segment suffers from the same limitation. Random content of each data segment makes its corresponding power spectral density a random variable. Treating the spectral estimate of each segment as i.i.d. random variable, temporal averaging operation can be applied, see (2.35).

\[
\hat{S}_{uv}(f)_{T_k} = \frac{1}{K} \sum_{k=0}^{K-1} \hat{S}_{uv}(t_k, f)_{T_k} \tag{2.35}
\]

\[
T_k = T_0 \forall k \tag{2.36}
\]

Where \( K \) is the number of data segments to average over, \( t_k \) is a time index of the beginning of each segment of length \( T_k \). The variance of average of the i.i.d. random variables decreases with number of data segments, it can be expressed as below using properties of the i.i.d. random variables.

\[
\text{var}[\hat{S}_{uv}(f)] = \frac{1}{K} \text{var}[\hat{S}_{uv}(t_k, f)] \tag{2.37}
\]

However, for a fixed data length the segment length decreases as number of segments increases. It can be noticed in (2.34) that decreasing the window length will increase the bias of our estimate due to the underlying convolution [26], i.e., spectral content will be affected by the spectral characteristics of the windowing function, e.g., mainlobe width. The only way to increase the length of each segment and the number of segments is to increase the overall data length. In the limit, we can increase the length of each segment to infinity, as well as, achieve infinite number of data segments. Thus, reducing the variance to zero and achieving asymptotically unbiased estimator. The above discussion allows formulation (2.35) with the integration operator [41]. Final formulation is given in (2.38). In (2.35), data segments are typically overlapped by a fraction of length of windowing function, e.g., in the Welch approach. In (2.38), data
segments are continuously shifted for each time instant.

\[
\hat{S}_{uv}(f) = \lim_{T \to +\infty} \lim_{\Delta t \to +\infty} \frac{1}{\Delta t} \int_{-\Delta t/2}^{\Delta t/2} \hat{S}_{uv}(u, f) du,
\]  

(2.38)

where \( \Delta t \) is the available data length. In order to provide the desired spectral correlation interpretation, relation (2.39) will be proved, see (2.30) and (2.25).

\[
\hat{S}_{uv}(f) = \hat{S}_x^\alpha(f)
\]  

(2.39)

Previously omitted finite window length and lag variable \( \tau \) in the integration limits of (2.31) are included in the derivation. Equation (2.31) and new integration limits are substituted into (2.38).

\[
\hat{S}_{uv}(f) = \lim_{T \to +\infty} \lim_{\Delta t \to +\infty} \frac{1}{\Delta t} \frac{1}{T} \int_{-\Delta t/2}^{\Delta t/2} \int_{-\infty}^{+\infty} \int_{-T/2}^{T/2} u(t + u) \cdot v^*(t + u - \tau) e^{-j2\pi f \tau} du d\tau dt
\]  

(2.40)

Cross-correlation function \( \hat{R}_{uv}(t, \tau)_T \) defined in (2.29) for infinite data length is denoted as the limit cross-correlation function \( \hat{R}_{uv}(\tau) \) [41] and expressed in (2.41). With the change of integration order and limit cross-correlation function \( \hat{R}_{uv}(\tau) \) (2.40) is written in the form of (2.42).

\[
\hat{R}_{uv}(\tau) = \lim_{T \to +\infty} \frac{1}{T} \int_{-\infty}^{+\infty} \int_{-T}^{T} \hat{R}_{uv}(\tau) e^{-j2\pi f \tau} du d\tau
\]  

(2.41)

\[
\hat{S}_{uv}(f) = \lim_{T \to +\infty} \frac{1}{T} \int_{-\infty}^{+\infty} \int_{-|\tau|}^{\tau} \hat{R}_{uv}(\tau) e^{-j2\pi f \tau} d\tau
\]  

(2.42)

Because of the presence of lag variable \( \tau \) in the integration limit, integration over variable \( u \) has to happen first. It can be seen that time-invariant property of \( \hat{R}_{uv}(\tau) \) simplifies integration.

\[
\hat{S}_{uv}(f) = \lim_{T \to +\infty} \int_{-\infty}^{+\infty} (1 - |\tau|/T) \hat{R}_{uv}(\tau) e^{-j2\pi f \tau} d\tau
\]  

(2.43)
Multiplication in (2.43) corresponds to convolution in the frequency domain. Using the fact that \(1 - \frac{|\tau|}{T} = \frac{1}{T} \text{rect}(t/T) \otimes \text{rect}(t/T)\) (2.43) is rewritten in the form of (2.44).

\[
\hat{S}_{uv}(f) = \lim_{T \to +\infty} \left[ \mathcal{F}\{\hat{R}_{uv}(\tau)\} \otimes W(f) \right]
\]  

(2.44)

Where \(W(f)\) is the Fourier transform of \(\frac{1}{T} \text{rect}(t/T) \otimes \text{rect}(t/T)\):

\[
W(f) = \frac{1}{T} (T^2 \text{sinc}^2(fT))
\]  

(2.45)

In the limit \(T \to +\infty\) Fourier transform of \(\frac{1}{T} \text{rect}(t/T) \otimes \text{rect}(t/T)\) is non-negative for all frequencies, has unity area and non-zero value only at zero frequency. Thus, \(\frac{1}{T} \text{rect}(t/T) \otimes \text{rect}(t/T)\) converges in properties to Dirac delta function \(\delta(f)\). Additionally, (2.41) can be modified using a change of variables into the form of (2.46).

\[
\hat{R}_{uv}(\tau) = \lim_{T \to +\infty} \frac{1}{T} \int_{-T/2}^{T/2} u(t) v^*(t - \tau) dt = \lim_{T \to +\infty} \frac{1}{T} \int_{-T/2}^{T/2} u(t + \tau/2) v^*(t - \tau/2) dt
\]  

(2.46)

Using the definitions of \(u(t)\) and \(v(t)\), (2.44), (2.46) and the 'sifting' property of the Dirac delta function the proof of (2.39) is evident:

\[
\hat{S}_{uv}(f) = \hat{S}_x^\alpha(f) = \mathcal{F}\left\{ \lim_{T \to +\infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t + \tau/2) x^*(t - \tau/2) e^{-j2\pi\alpha t} dt \right\}
\]  

(2.47)

Final step is to show the spectral correlation interpretation of limit cyclic spectrum \(\hat{S}_x^\alpha(f)\). The shift property of the Fourier transform can be applied to (2.26) and (2.27) resulting in:

\[
U(t, f) = X(t, f + \frac{\alpha}{2})
\]  

(2.48)

\[
V(t, f) = X(t, f - \frac{\alpha}{2})
\]  

(2.49)

Substituting (2.48), (2.49) and (2.33) to (2.38) a final form of \(\hat{S}_x^\alpha(f)\) is achieved, limit
operators are omitted:

\[
\hat{S}_x^a(f) = \frac{1}{\Delta t} \int_{-\Delta t/2}^{\Delta t/2} \frac{1}{T} X(u, f + \frac{\alpha}{2}) \cdot X^*(u, f - \frac{\alpha}{2}) du
\]  

(2.50)

It can be concluded that an alternative interpretation of the limit cyclic spectrum arises as a temporal correlation between \(X(t, f + \frac{\alpha}{2})\) and \(X^*(t, f - \frac{\alpha}{2})\), where the separation in frequency equals to \(\alpha\). Thus, the cyclic spectrum \(\hat{S}_x^a(f)\) is often denoted as the spectral correlation function. Summarizing, the spectral correlation function can be seen as a spectral density of a temporal correlation between time-variant spectra. The frequency variable \(f\) is commonly referred to as a carrier frequency, while \(\alpha\) is commonly denoted as a cyclic or modulation frequency [5]. The derivation of \(\hat{S}_{uv}(f) = \mathcal{F}(\hat{R}_{uv}(\tau))\), (2.38)-(2.44), in case of zero cyclic frequency \(\alpha = 0\) is a proof to the Wiener theorem [42].

### 2.2.1.2 Probabilistic Approach

Based on the Wiener theorem it is possible to generate an estimate of power spectral density with time averaging [42]. However, in general, the time averaging technique cannot be applied to non-stationary processes, since time averaging removes time-varying effects. The derivation of the cyclic spectrum in the previous section was based on the empirical autocorrelation function, in this section the probabilistic approach is shown.

This section uses the probabilistic approach and some nomenclature should be clarified. In case of the empirical approach and time averages, derivations were based on random signals. In case of the probabilistic framework, the probability law that governs the random signal is of interest. The probability law describes the random process that models the random signal in the sense that the random signal is just a particular realization of the underlying random process. At each time instant \(t\), the ensemble of possible realizations forms a random variable. Thus, a random process can be viewed as a collection of random variables, one for each time \(t\). In the following derivations random variables will be denoted in bold, for example \(x(t)\), while
random processes will be denoted as a collection of random variables \( \{ x(t) \} \). Several definitions and properties are introduced below to motivate the derivation of the probabilistic approach. Empirical mean \( \hat{m}_x(t) \) and empirical autocorrelation function \( \hat{R}_x(t, \tau) \) for random signal \( x(t) \) are given in (2.51)-(2.52). The ensemble mean \( m_x(t) \) and autocorrelation function \( R_x(t, \tau) \) that characterize random process \( \{ x(t) \} \) are given in (2.53)-(2.54).

\[
\hat{m}_x(t) = \frac{1}{T} \int_{t-T/2}^{t+T/2} x(u) du
\]

\[
\hat{R}_x(t, \tau) = \frac{1}{T} \int_{t-T/2}^{t+T/2} x(u + \tau)x(u) du
\]

\[
m_x(t) = E[x(t)]
\]

\[
R_x(t, \tau) = E[x(t)x(t + \tau)],
\]

where \( E[...] \) denotes the expectation operator. For stationary or WSS processes first order moment, second order moment and the autocorrelation function are independent of time origin [26], thus, the following can be written:

\[
m_x = E[x(t)]
\]

\[
R_x(\tau) = E[x(t)x(t + \tau)]
\]

Additionally, the empirical mean and autocorrelation function can be rewritten assuming infinite estimation time.

\[
\hat{m}_x = \lim_{T \to +\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x(t) dt
\]

\[
\hat{R}_x(\tau) = \lim_{T \to +\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x(t + \tau)x(t) dt
\]

The property that guarantees duality between the empirical and probabilistic approach, i.e., between time and ensemble averages, is ergodicity [26]. For non-stationary processes another property can be defined, the property of regularity. Regular non-stationary process is characterized by the existence of \( \hat{m}_x \) and \( \hat{R}_x(\tau) \) [42]. Application
of the expectation operator to (2.58) is shown in (2.59).

\[
E[\hat{R}_x(\tau)] = < R_x > (\tau) = \lim_{T\to+\infty} \frac{1}{T} \int_{-T/2}^{+T/2} R_x(t, \tau) dt \tag{2.59}
\]

If \(< R_x > (\tau)\) is not zero for all \(\tau\), then \(\{x(t)\}\) is called asymptotically mean stationary process. In general, for a regular non-stationary process it cannot be claimed that \(R_x(t, \tau) = \hat{R}_x(\tau)\) (it is time-variant), however, for the class of non-stationary regular ergodic processes [42] it can written that:

\[
<R_x> (\tau) = \hat{R}_x(\tau) \tag{2.60}
\]

Taking into account (2.59), (2.60) it can be satisfied if and only if:

\[
E[\hat{R}_x(\tau)] = \hat{R}_x(\tau) \tag{2.61}
\]

Based on the above, it can be concluded that expected value of \(\hat{R}_x(\tau)\) has to be equal to \(\hat{R}_x(\tau)\), so the variance of \(\hat{R}_x(\tau)\) has to be zero.

Applying the expectation operator to periodogram-correlogram relation \(\hat{S}_x(f) = \mathcal{F}(\hat{R}_x(\tau))\) (2.62) can be written.

\[
E[\hat{S}_x(f)] = \mathcal{F}(E[\hat{R}_x(\tau)]) \tag{2.62}
\]

With the regularity assumption and (2.59):

\[
E[\hat{S}_x(f)] = \lim_{T\to+\infty} \frac{1}{T} \int_{-T/2}^{+T/2} \hat{S}_x(t, f) dt \tag{2.63}
\]

\[
\hat{S}_x(t, f) = \mathcal{F}(\hat{R}_x(t, \tau)) \tag{2.64}
\]

A random process \(\{x(t)\}\) can be described as first-order cyclostationary if its ensemble mean \(m_x(t)\) is periodic and as second-order cyclostationary if its ensemble
The autocorrelation function $R_x(t, \tau)$ is periodic.

$$m_x(t + T_0) = m_x(t) \quad (2.65)$$

$$R_x(t + T_0, \tau) = R_x(t, \tau) \quad (2.66)$$

Periodic autocorrelation function $R_x(t, \tau)$ can be represented using the Fourier series decomposition.

$$R_x(t, \tau) = \sum_{\alpha} R_\alpha^x(\tau)e^{j2\pi \alpha t} \quad (2.67)$$

$$R_\alpha^x(\tau) = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} R_x(t, \tau)e^{-j2\pi \alpha t} dt \quad (2.68)$$

where $\alpha$ is an integer multiple of $1/T_0$, $R_\alpha^x(\tau)$ is a cyclic autocorrelation function. A signal is called almost periodic if there are multiple periodic components that are not integer related: $1/T_0, 1/T_1, \ldots$. Following this definition, a random process is described as almost periodic if $R_\alpha^x(\tau)$ reveals more than one periodicity. Equation (2.68) is generalized to account for this by extending the period to infinity:

$$R_\alpha^x(\tau) = \lim_{T \to +\infty} \frac{1}{T} \int_{-T/2}^{T/2} R_x(t, \tau)e^{-j2\pi \alpha t} dt \quad (2.69)$$

It should be noticed that (2.69) is equal to (2.59) for $\alpha = 0$:

$$R_0^x(\tau) = \lim_{T \to +\infty} \frac{1}{T} \int_{-T/2}^{T/2} R_x(t, \tau) dt \quad (2.70)$$

To enable further derivation, (2.59) is rewritten in the form of (2.71).

$$\lim_{T \to +\infty} \frac{1}{T} \int_{-T/2}^{T/2} E[x(t + \tau)x(t)] dt =$$

$$\lim_{T \to +\infty} \frac{1}{T} \int_{-T/2}^{T/2} E[x(t + \tau/2)x(t - \tau/2)] dt \quad (2.71)$$

The cyclic nature in the frequency domain representation is shown using temporary random variables $u(t)$ and $v(t)$ defined in (2.72)-(2.73). Based on the shift property
of the Fourier transform, variables $u(t)$ and $v(t)$ are frequency shifted versions of the variable $x(t)$.

$$u(t) = x(t)e^{-j\pi\alpha t} \quad (2.72)$$

$$v(t) = x(t)e^{j\pi\alpha t} \quad (2.73)$$

The definition in (2.71) is applied to cross-correlation between $u(t)$ and $v(t)$. For the sake of generality, the complex conjugate * operator is included in the definition of correlation, see (2.74).

$$< R_{UV}(\tau) > = \lim_{T \to +\infty} \frac{1}{T} \int_{-T/2}^{+T/2} E[u(t + \tau/2)v^*(t - \tau/2)]dt \quad (2.74)$$

By substitution of definitions of $u(t)$ and $v(t)$ (2.75) can be written.

$$< R_{UV}(\tau) > = \lim_{T \to +\infty} \frac{1}{T} \int_{-T/2}^{+T/2} R_x(t, \tau)e^{-j2\pi\alpha t}dt \quad (2.75)$$

After comparing (2.68) with (2.75) it can be concluded that $< R_{UV}(\tau) > = R^\alpha_x(\tau)$.

This provides a clear interpretation of cyclic autocorrelation function $R^\alpha_x(\tau)$ as a temporal correlation between frequency shifted versions of random variable $x(t)$. Fourier transform can be applied to (2.75) in order to get the desired frequency distribution.

$$< S_{UV}(f) > = \mathcal{F}\{< R_{UV}(\tau) >\} = \mathcal{F}\{R^\alpha_x(\tau)\} \quad (2.76)$$

$$S^\alpha_x(f) = < S_{UV}(f) > \quad (2.77)$$

Where $< S_{UV}(f) >$ is a time-averaged probabilistic cross-spectral density, $S^\alpha_x(f)$ is called a cyclic spectrum or cyclic spectral density. It has a convenient representation as a bi-frequency map, where one axis is the cyclic frequency $\alpha$ and second axis is the carrier frequency $f$.

Similar to time average case, an alternative interpretation of the cyclic spectrum as a spectral correlation can be provided. In order to provide it, the concept of cycloergodicity of a random process has to be introduced. Cycloergodicity is a property
of a cyclostationary random process that enables the duality between ensemble averages and the time averages based on the individual member of the ensemble. For a cycloergodic process, the cyclic autocorrelation function can be expressed using time averages from previous section:

\[ \hat{R}_{uv}(t, \tau) = \frac{1}{T} \int_{t-T/2}^{t+T/2} u(w + \tau/2)v^*(w - \tau/2)dw \]  

(2.78)

\[ \hat{R}_{uv}(\tau) = \lim_{T \to +\infty} \hat{R}_{uv}(t, \tau)_T \]  

(2.79)

After revisiting (2.75) and (2.76), \(< S_{UV}(f) >\) can be written as shown in (2.80).

\[ < S_{UV}(f) > = \lim_{T \to +\infty} \frac{1}{T} \int_{-T/2}^{+T/2} S_{UV}(t, f)dt \]  

(2.80)

\[ S_{UV}(t, f) = \mathcal{F}\{R_{uv}(t, \tau)\}, \]  

(2.81)

where \(R_{uv}(t, f)\) is a time-variant ensemble cross-correlation function. With the assumption of cycloergodicity (2.80) can be rewritten as follows:

\[ < S_{UV}(f) > = \lim_{\Delta t \to +\infty} \frac{1}{\Delta t} \int_{-\Delta t/2}^{+\Delta t/2} \mathcal{F}\{\hat{R}_{uv}(t, \tau)_T\}dt, \]  

(2.82)

where \(\hat{R}_{uv}(t, \tau)_T\) is a finite-length empirical cross-correlation function, \(\Delta t\) is data length and \(T\) is a window length. Comparison of (2.82) with (2.33) and (2.30) allows to write \(< S_{UV}(f) >\) in the form of (2.83), limit operators are omitted.

\[ < S_{UV}(f) > = \frac{1}{\Delta t} \int_{t-\Delta t/2}^{t+\Delta t/2} \frac{1}{T} X(u, f + \frac{\alpha}{2}) \cdot X^*(u, f - \frac{\alpha}{2})du \]  

(2.83)

The spectral correlation interpretation is evident. The probabilistic approach with assumption of cycloergodicity verifies that the cyclic spectrum \(\hat{S}_{x}^{\alpha}(f)\) has an alternative interpretation as a spectral density of a temporal correlation between time-variant spectra \(X(u, f + \frac{\alpha}{2})\) and \(X(u, f - \frac{\alpha}{2})\).
2.2.2 Properties of Spectral Correlation

This section discusses properties of the cyclic spectrum and benefits of the spectral correlation interpretation developed in the previous section. Definitions in (2.50) and (2.83) provide temporal correlation values for spectral components shifted by \( \alpha \). For example, it is expected that the cyclic spectrum presents high values for frequency components corresponding to sidebands due to the modulation applied on the examined signal. Thus, cyclic spectrum is expected to be a valuable tool in analysis of modulation characteristics. Another feature of the cyclic spectrum is good noise immunity. For a stationary noise signal, i.e., a realization of a stationary random process, the time-variant autocorrelation function \( R_x(t, \tau) \) has no \( t \) dependence, \( R_x(t, \tau) = \text{const} \:\forall t \). Consequently, values of \( R_x^0(\tau) \), which are Fourier series coefficients of \( R_x(t, \tau) \), are zero for all frequencies other than DC, i.e., \( \alpha \neq 0 \). In other words, contributions to the cyclic spectrum due to stationary noise are limited only to the line \( \alpha = 0 \). Based on this observation and (2.50), the cyclic spectrum is immune to stationary noise [44]. In case of a conventional spectral analysis (\( \alpha = 0 \)), additive stationary noise can mask the potential signal of interest. From now on, terms of cyclic spectrum and spectral correlation (function) will be used interchangeably.

Different types of mechanical systems generate vibration signals at very different energy levels, but often the signals contain similar modulation features. It is of great interest to develop a condition indicator that is agnostic to energy scaling effects. A normalized indicator avoids false triggering due to varying energy levels, e.g., due to different loading conditions or system size. To address this, the spectral coherence coefficient \( \gamma_x^\alpha(f) \) is applied.

\[
\gamma_x^\alpha(f) = \frac{\hat{S}_x^\alpha(f)}{\sqrt{\hat{S}_x^0(f - \frac{\alpha}{2})\hat{S}_x^0(f + \frac{\alpha}{2})}}
\]

(2.84)

In general, \( \gamma_x^\alpha(f) \) is a complex quantity with magnitude \( |\gamma_x^\alpha(f)| < 1 \). The coefficient provides a measure of cyclostationarity.

To illustrate the properties of the spectral correlation consider the signal \( x(t) \):
\[ x(t) = \cos(2\pi ft)\cos(\pi \alpha t) = \frac{1}{4} \left( e^{j2\pi(f+\frac{\alpha}{2})t} + e^{j2\pi(f-\frac{\alpha}{2})t} + e^{-j2\pi(f+\frac{\alpha}{2})t} + e^{-j2\pi(f-\frac{\alpha}{2})t} \right), \]

(2.85)

where \( \frac{\alpha}{2} = 2 \) Hz and \( f = 400 \) Hz. Figs. 2-15 and 2-16 show the power spectral density and the spectral coherence map, respectively, for the signal in (2.85). The analyzed signal is an example of a double-sideband suppressed-carrier amplitude modulated signal. According to (2.85), the spacing of the frequency components in the frequency domain equals to \( \alpha = 4 \) Hz. As expected from (2.83), the temporal correlation of sequence \( X_{\Delta f}(t, f + \frac{\alpha}{2}) \) with sequence \( X_{\Delta f}(t, f - \frac{\alpha}{2}) \) reveals a component at \( f = 400 \) Hz and \( \alpha = 4 \) Hz on the spectral coherence map.

Figure 2-15: The power spectral density of the deterministic signal with suppressed carrier.

An example of double-sided full-carrier amplitude modulation is defined in (2.86).

\[ x(t) = \cos(2\pi ft)(1 + \cos(2\pi \alpha t)) = \frac{1}{4} \left( e^{j2\pi(f+\alpha)t} + e^{j2\pi(f-\alpha)t} + e^{-j2\pi(f+\alpha)t} + e^{-j2\pi(f-\alpha)t} \right) + \frac{1}{2} e^{-j2\pi ft} + \frac{1}{2} e^{j2\pi ft}, \]

(2.86)

where \( \alpha = 2 \) Hz and \( f = 400 \) Hz. Based on inspection of power spectral density of double-sided full-carrier amplitude modulation example, see Fig 2-17, it can be
concluded that spectral correlation should reveal high values for two values of cyclic frequency: 2 Hz and 4 Hz. It can be seen that the spectral coherence map in Fig. 2-18 indeed presents high correlation values at those frequencies. Spectral coherence value at cyclic frequency of 2 Hz presents a higher value due to the higher power of carrier frequency component in comparison with the power at sideband frequency components (spaced 4 Hz apart).

Next, consider a random signal that consists of a carrier modulated by a low
Figure 2-18: The spectral coherence map of the deterministic signal with full carrier.

frequency signal:

\[ x(t) = x_t \cos(2\pi f t), \]  

where \( x_t \) is a value of a random variable with Gaussian probability density function, zero mean and unity variance. Figs. 2-19 and 2-20 present the power spectral density and the spectral coherence map, respectively. It can be observed that the power spectral density fails to recognize the periodic variations in the signal of interest. The power spectral density is essentially wideband. However, the spectral coherence map gives a very clear indication of periodicity. The random nature of the carrier is manifested by presence of a vertical line across all carrier frequencies \( f \). It is a very important observation that can be used to interpret the signal of interest in real-world applications.

An example spectral coherence map of vibration data acquired during operation of milling machine is presented in Fig. 2-21. The ordinate corresponds to carrier frequency \( f \), while the abscissa corresponds to cyclic frequency \( \alpha \). In this experiment the spindle speed was set to 2150 RPM. It can be seen that spectral coherence is a discrete function of frequency \( \alpha \) and a continuous function of frequency \( f \). Based on the previous numerical experiment, the continuity of the spectral coherence in \( f \) reveals the presence of random content in a wide frequency range [5]. Cyclic components are present at multiples of the spindle frequency, with spacing \( \Delta \alpha = \frac{2150}{60} \approx 36 \text{ Hz} \). Time
domain representation of milling machine vibration data is given in Fig. 2-22.

Another example uses vibration data acquired during operation of a positive displacement diaphragm pump. A spectral coherence map is presented in Fig. 2-23. Distinct vertical lines at multiples of the pumping frequency reveal significant random content in the vibration signal. This is expected for structures with FSI, several rotating components supported with bearings and internal gear. The pump operates with a 4 – pole motor connected to a utility with a 60 Hz electrical frequency. The resulting motor mechanical shaft rotation near 30 Hz passes through a 10 : 1 internal gear ratio to produce a reciprocating motion with frequency in the vicinity of 3
Figure 2-21: The spectral coherence map of vibration data of a milling machine.

Figure 2-22: Vibration data of a milling machine.

Hz, depending on the loading condition and slip value. Detection of cyclostationary components provides a clear indication of the existing modulation frequencies and spectral locations of the carrier frequencies. Time domain representation of pump vibration data is given in Fig. 2-24.

Based on the numerical experiments and application examples, it can be seen that the spectral correlation provides a valuable signal processing tool for random and deterministic signals associated with the operation of periodically operated actuators. Immunity to stationary noise and normalization make spectral coherence a good candidate to develop diagnostic metrics. Performance comparison with cepstrum-based
techniques is given in Chapter 3.

Figure 2-23: The spectral coherence map of vibration data of a diaphragm pump.

Figure 2-24: Vibration data of a diaphragm pump.
Chapter 3

Diagnostics for Periodically Operated Actuators

In general, most of the electromechanical systems can be classified into two types: operating periodically and intermittently. Compressors, milling machines, pumps, internal combustion engines and propellers are examples of periodically operating devices; while device like circuit breaker, is an example of device operating intermittently. Work presented in this chapter is focused mostly on periodically operated systems. Proposed signal processing utilizes properties of cyclostationary signals and enables diagnostics based on signals that have strong random components. Proposed solutions are tested on two industrial devices, diaphragm pump [18] and CNC milling machine [19]; however, applications are not limited to those.

This chapter is organised as follows. Section 3.1 presents diagnostic approach for industrial diaphragm pumps. Section 3.2 presents diagnostic approach for cutting tools. Both sections provide experimental results with discussion.

3.1 Diagnostics for Diaphragm Pumps

Diaphragm pumps are an example of an industrially important, periodically operated actuator. These pumps are critical for medical, food, and petrochemical processes [45], [46] as they offer volumetric precision, high discharge pressures and superior
isolation of the working fluid. Diaphragm pumps also exhibit significant failure mechanisms. Two main challenges reported in the literature include diaphragm rupture and check valve failure [47], [48]. These two phenomena are related. For example, inlet valve clogging can lead to premature diaphragm damage. Physical rupture of the diaphragm can lead to contamination of the process liquids by hydraulic oil from the pump’s mechanism.

There are relatively few publications that examine the analytical description of diaphragm pump operation and associated degradation in practical environments [47], [49], e.g., in comparison to centrifugal pumps. Moreover, few references deal with diagnostics and condition monitoring of these positive displacement pumps [50], [51], [52]. Diaphragm pumps are susceptible to several crippling failure mechanisms and are often employed in harsh environments that induce regular degradation and eventual failure. A focus of this chapter will be fault detection for diaphragm pumps. The techniques developed here apply to other periodically cycled actuators, e.g., compressors and stamping machine tools.

New hardware and associated signal processing techniques that exploit the inherent periodic excitation in these types of loads will be demonstrated analyzing a LEWA EK-1 diaphragm pump [53]. Among other benefits, these signal processing techniques utilize periodicity to reduce the need for transmitting or storing large quantities of sampled data. Cyclically aware signal processing algorithms focus on signal features relevant to diagnostics and, with appropriate instrumentation hardware, are ideal for remote or embedded diagnostic systems [54].

Periodic operation colors or impacts the structure of many different physical signals associated with an actuator. For example, periodic variations may appear in torque, current, vibration, acoustic signature and other signals. Diagnostics indicators are often found in many or all of these signals. Some may be easier or more appropriate to measure in a given application. For example, where control requires current sensing, current measurements may be available without added expense. On the other hand, sealed or electrically isolated or installed systems may be more easily analyzed from mechanical measurements. The sections below describe two illustra-
tive approaches, one examining the cyclostationary components of a vibration signal and the second examining frequency components of a current signal associated with the induction machine driving the device under study. Alternating suction and discharge cycles create a non-uniform loading condition over a single operating cycle of a diaphragm pump. The pump’s head acceleration is influenced by these cyclic variations, which results in a strongly modulated vibration signal. Detection of variations and subtle periodicity in a signal can be challenging for conventional spectral analysis techniques [42]. For example, random content can interfere with the detection and isolation of the periodic part of a signal [44]. In [5], the authors show examples targeted to a diesel engine and a centrifugal pump, and highlights how inspection of the power spectral density can fail to distinguish hidden periodicity in the vibration signals. The concept of cyclostationarity can be used to expose the impact of load torque modulation on vibration measurements.

The first approach monitors the energy level at one of the mechanical resonances of the pump’s head structure. Vibration signal measured at the pump’s head is used to extract the desired information. Both diaphragm degradation, as well as inlet valve clogging, significantly impact the energy level at this resonant frequency. An impulse response analysis is able to identify the spectral location of the resonance. A sensitivity study for the location of the resonant frequency was performed to understand and minimize the influence of different mounting conditions of the diaphragm pump. An example of the acceleration signal acquired using the IEPE accelerometer during the decay experiment is shown in Fig. 3-1. The signal shown in Fig. 3-1 is the response of the LEWA EK-1 industrial pump. Vibration signal is used not only to monitor energy at the resonance frequency but also to extract cyclostationary characteristics using spectral correlation.

The second approach to fault detection and diagnostics uses current signature analysis associated with an induction machine driving the device under study [13], [7]. Experimental results confirm that the fundamental component of a current signal provides little diagnostic information. Moreover, it often masks subtle components that tend to be useful for diagnostic purposes. A custom adaptive discrete-time notch
Figure 3-1: IEPE accelerometer data during an impulse response.

filter is used to remove the fundamental component of the current signal and enhance opportunities for diagnostics.

As an estimator of the spectral correlation, the fast spectral correlation [55] is used in this work. The Matlab implementation of the fast spectral correlation is available in [43]. In order to mitigate scaling effects and allow for a reliable comparison between estimates of the spectral correlation, it is recommended to normalize the estimate [5, 12] and use spectral coherence. A new figure-of-merit is proposed for the spectral content by integrating over a carrier frequency range \( F \) for any particular choice of \( \alpha \). This figure-of-merit, the MES, summarizes the average impact of modulating frequency component \( \alpha \) on carrier frequencies \( f \) in the range of \( F \):

\[
S_x^{<f>}(\alpha) = \frac{1}{|F|} \int_F |\gamma_x(\alpha, f)| df
\]  

(3.1)

3.1.1 Diaphragm Pump Experiment

A custom data acquisition system was developed to permit focus on modulating signals most relevant for the device diagnostics. Validation of the diagnostic strategies was performed on a high pressure setup emulating a work environment of a diaphragm pump. The data acquisition board is equipped with a dedicated active filter for current signal processing. The adaptive discrete-time notch filter is implemented
with two stages of an LTC1060 switched capacitor filter IC. The bandwidth is fixed by the hardware configuration, while the center frequency is adapted. Firmware is responsible for tracking the fundamental frequency variations and updating the center frequency. This discrete-time, switched capacitor filter is used for fundamental frequency rejection in the current signal. The system continuously adapts the filter center frequency depending on the fundamental frequency of a power source. That is, as the utility supplying an induction machine or other motor under investigation varies in frequency, the filter system will automatically track to reject power line carrier frequency. After filtering, the remaining signal is amplified to exploit the full practical range of an ADC. This maximizes SNR [26], specifically targeting parts of the signal which contain diagnostic information. Details of the acquisition device and switched capacitor filter are provided in Chapter 5.

The experimental setup, focused on a LEWA EK-1 diaphragm pump, was constructed for this study. Fig. 3-2 presents a schematic diagram of the setup. Inlet and outlet pressure accumulators were used together with pressure regulators to create appropriate pressure levels at the input and output of the pump. The pump is driven by a 0.5 HP, 4-pole induction motor. It features an internal gearbox with a 10 : 1 gear ratio. The pump in the setup is equipped with a non-spring loaded ball-type check valve at both the inlet and outlet ports. The diaphragm installed during the experiments was manufactured from PTFE. Table 3.1 presents physical parameters used during experiments with the setup. Volumetric flow measurements were recorded by tracking the time necessary to displace a fixed amount of fluid [47].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output pressure, (psi)</td>
<td>60-185</td>
</tr>
<tr>
<td>Input pressure, (psi)</td>
<td>40</td>
</tr>
<tr>
<td>Flow, (l/min)</td>
<td>0-4</td>
</tr>
<tr>
<td>Temperature, (°C)</td>
<td>20</td>
</tr>
<tr>
<td>Viscosity, (cSt)</td>
<td>46(max)</td>
</tr>
</tbody>
</table>
The valve clogging phenomenon was emulated by gradually clogging the inlet check valve by means of a strong hydraulic sealant. Fig. 3-3 shows different stages in the clogging process, the check valve bracket and valve balls are visible. In order to analyze the results presented in this chapter, a figure-of-merit, the “clogging stage”, is introduced. The clogging stage is an empirical quantity that describes the percentage of the valve opening that has been clogged.

A special fixture was machined to permit accelerated diaphragm degradation consistent with that observed in the field. The fixture in Fig. 3-4 allows the application of force perpendicular to the membrane. The value of the force was empirically adjusted. The specific results presented in this chapter correspond to a membrane stretched with an 800 N force, which emulates damage that might be experienced by the diaphragm during a malfunction.

### 3.1.2 Vibration Based Approach

The performance of the proposed diagnostic techniques are demonstrated in this section. Acquired data was sampled at 21 kHz with a 24 bit resolution over a 20 second measurement interval during pump operation. The MES values were obtained
for 0 – 21 kHz range. All results were obtained with a working fluid consisting of food-grade hydraulic oil with a viscosity of 46 cSt. The inlet pressure was fixed at 275.79 kPa (40 psi), while the output pressure was kept at 1172.11 kPa (170 psi).

Fig. 3-5 presents the spectral coherence map for diaphragm pump vibration signal. Distinct vertical lines at multiples of the pumping frequency reveal significant random content in the vibration signal. This is expected for structures with FSI, several rotating components supported with bearings and internal gearbox. The pump operates with a 4-pole motor connected to a utility with a 60 Hz electrical frequency. The resulting motor mechanical shaft rotation near 30 Hz passes through a 10 : 1 internal gear ratio to produce a reciprocating motion with frequency in the vicinity of 3 Hz, depending on the loading condition and slip value. Detection of cyclostationary components provides clear indication of the existing modulation frequencies and spectral location of the carrier frequencies.
3.1.2.1 Valve Clogging

The first approach uses cyclostationary content of the vibration signal to detect the valve clogging phenomenon. Progressive clogging of the inlet valve leads to a decrease of the average flow. Obviously a decrease in the flow level is an intuitive method of detection often used in the field. However, a good diagnostic tool should allow for an early indication of failure even without a significant decrease of the average flow. Fig. 3-6 presents a relation between the clogging stage and the measured averaged flow. It can be seen that for 0.2 pu clogging stage, the flow rate decreases slightly. However, for the severe case the flow value is only 60% of the clean case. Fig. 3-7 relates the value of the MES with the clogging stage. It can be seen that the indicator increases with progressing clogging. For the clogging stage of 0.3 pu there is a clear increase in the MES, as compared with a clean case. Contr astingly, the average flow at the same stage is about 90% of the clean case. This is a promising performance from the condition monitoring point of view.

The second quantity used for the assessment of the valve condition is energy present at the resonant frequency. This is illustrated in Fig. 3-8. For low clogging stages the average flow value is essentially very similar. This means that, for a fixed amount of fluid and a smaller available check valve area, the average speed of the work fluid increases. The increase of the average speed results in higher magnitude values.
Figure 3-6: Average flow as a function of the clogging stage.

Figure 3-7: The MES zoomed in at fundamental cyclic frequency.

of spectral content near the resonance, exciting monotonically increasing vibration levels with clogging. For heavier clogs, e.g., a 0.7 pu clogging stage, flow volume is substantially impacted as the flow value drops by more than 40%. That is, as clogging becomes heavier, the trend in vibration energy will reverse.

A clear increase in the value of both the MES and energy at the resonant frequency can be observed for low clogging stages, i.e., for which the flow is maintained above 90% of its initial value. Clogging is a gradual process of filling openings in the valve bracket and early clogging stages can be reliably detected by both methods. Consequently, the non-monotonic behavior for severe clogging does not discount or
eliminate the value of either of the diagnostic indicators as a clog develops. Each result presented in Figs. 3-7 and 3-8 is the average of 10 data sets acquired independently.

In an industrial environment, clogging increases gradually depending on the solid fraction of the pumping fluid, its viscosity and other factors. Continuous operation of the experimental setup and addition of impurities in laboratory conditions is very difficult. Here, we emulate this situation by reversing the process. The hydraulic sealant used for clogging emulation presented good endurance in providing a clog. However, long operation at constant input and output pressures can lead to gradual removal of the sealing insert from the valve bracket. Starting with the clogging stage of approximately 0.3 pu, the pump was being operated until the desired clearing or removal of the sealant clog was achieved. This test technique facilitates recording measurements for different valve conditions without physical intervention into the pump’s mechanism, ensuring an operating cycle that is consistent with sustained use in an industrial environment. Fig. 3-9 shows gradual removal of the sealant clog over the course of an experimental cycle (compare with Fig. 3-3 to see the effect). Figs. 3-10 and 3-11 present the MES and peak of the magnitude spectrum in the vicinity of the resonant frequency, respectively. Data sets directly before and after the sealant removal were recorded. The measurements are presented in a chronological order as labeled in the legends. In Fig. 3-10 evolution of the cleaning process is visible. There
is a clear change in the MES value when comparing the data set labeled as Clean6 with the previous measurements. This is where significant weakening of the sealant structure happened. Further measurements Clean7-Clean9 present how this process evolved and led to the state presented in Fig. 3-9. Fig. 3-11 shows peak values of the magnitude spectrum at the resonant frequency, represented as a bar plot for clarity. As the sealant removal progresses, a decreasing trend in the peak energy at the resonant frequency can be observed. Both of the applied techniques are able to detect a change in the valve condition. Especially the MES-based approach presents a clean and valuable condition indicator.

3.1.2.2 Diaphragm Degradation

The second phenomenon targeted in this work is diaphragm degradation. The performance of two diaphragms was compared, a new PTFE diaphragm and one stressed as described in Section 3.1.1. Both the MES and energy at the resonant frequency approaches are tested. Fig. 3-12 presents the value of the MES for both diaphragms. Five independent measurement sets are presented for both cases. An increase in the
value of the MES at the fundamental cyclic frequency can be noticed for a damaged diaphragm case. However, it is not as pronounced as for the valve clogging problem. Inspection of Fig. 3-5 reveals that there is a significant energy at multiple harmonics of the fundamental cyclic frequency $\alpha$. Indeed, the third harmonic of $\alpha$ presents a good indication and allows for distinction between both diaphragms as shown in Fig. 3-13. Error bars represent the standard deviation across 5 independent measurement sets.

While there is no a priori reason to choose any particular harmonic as the diagnostic indicator, several practical approaches have been considered to develop a
diagnostic metric from the frequency data. First, a composite indicator can be found that sums content across many components. Second, choice of any particular component is not critical, as the diaphragm excites a broad set of modulations. In particular installations, it is conceivable that the metric is best determined by examining trends over a period of operating life.

Additionally, machine under study may offer an obvious choice of frequency for developing a diagnostic metric: a natural resonance of the mechanical structure, which amplifies the effect of the changes. Fig. 3-14 compares the magnitude spectra zoomed in at the resonant frequency. It can be seen that the condition of diaphragm
significantly contributes to the way the resonance is excited. The damaged diaphragm is substantially less efficient in pumping, and therefore provides less excitation at the resonant frequency. This is reflected in the figure as a reduced peak or energy content in the damaged condition in comparison to undamaged condition. The results in Fig. 3-14 are the averages of 5 measurement sets. Error bars represent the standard deviation of the magnitude of resonant frequency component.

3.1.3 Current Based Approach

A current signal can also be utilized to detect diaphragm degradation and valve clogging. The current signal used in this analysis is the output of the discrete-time notch filter on the data acquisition board.

High fidelity measurements of the filtered current allow for identification of sidebands around the fundamental frequency component. Change in energy of the sideband frequency components is utilized to monitor the device under study. Fig. 3-15 shows the fifth side-band of the fundamental 3 Hz modulation. Similarly to the previous section, the presented data is an average of five data sets. Error bars represent the standard deviation of the magnitude spectrum peak value. A clear indication of progressing clogging is observed. Consequently, it can be concluded that monitoring of the energy levels at this frequency could be used as a diagnostic tool. There is a
significant increase in the variance of the current sidebands for the clogged case as shown in Fig. 3-15.

A similar approach will detect diaphragm degradation. In this case the third side-band was identified as a strong signal for use as a diagnostic indicator. In general, the amplitudes of the side-bands will vary with the mechanical system or the electric machine and should be chosen to provide the largest signal for diagnostic detection. Fig. 3-16 presents the magnitude spectrum located around the third side-band. In this case, change in the average value of a given frequency component is even higher than for the valve clogging scenario. Current signal analysis in the frequency domain allows successful detection of the problems of interest.

![Figure 3-15: Comparison of current magnitude spectra, valve clogging.](image)

3.2 Diagnostics for Cutting Tools

Manufactured goods ranging from keys to ships are still primarily assembled with parts created by material removal with cutting, knurling, boring or drilling tools. Tool condition directly affects finish quality and safety, e.g., the likelihood of tool breakage or kickback [56]. Currently, human operators rely on heuristic techniques, visual inspection of the tools, bits, or machined parts, and experience, to decide when tools have reached the end of their life-cycle. Computer-based tool condition monitoring
systems provide the opportunity for more accurate condition diagnosis. These systems are increasingly important as machining processes transition to fully autonomous operation. Tool condition monitoring enables the Industry 4.0 revolution and the potential application of IoT devices in many manufacturing operations. This section introduces a robust condition metric based on spectral coherence in easily acquired vibration signals. Results are demonstrated using end mills from real manufacturing operations, however, the presented approach applies more generally to a wide range of cutting scenarios.

Monitoring systems that permit accurate fault detection and diagnostics with easy installation remote from the direct machining area would enhance safety, reliability, and flexibility in gathering actionable process information. References [57, 58, 59, 60] analyze the examination of acoustic and vibration signatures for TCM. These methods are useful for TCM, but they employ invasive measurement setups that are not always practical. For example, the setup presented in [57] requires an acoustic emission sensor to be mounted very close to the work piece. In [61] three accelerometers are mounted directly on the vice that clamps the work piece, a serious limitation in industrial environments that expect flood cooling, quick workpiece changes and highly automated procedures. Force and tool position measurements have been utilized in [56, 62] for the diagnosis of tool wear. Authors in [63] use image analysis for drilling

Figure 3-16: Comparison of current magnitude spectra, diaphragm degradation.
machine TCM. All of these methods may be stymied by the use of coolant or a requirement for distanced sensor installation from the cutting area.

Generally, approaches for cutting tool diagnostics fall into one of two categories. The first approach uses analytical signal processing algorithms to obtain condition metrics. The second approach identifies tool diagnostics as a classification problem and uses machine learning tools. Both time and frequency domain signal processing have been used for TCM [64, 60, 65]. Frequency domain signal processing is attractive because of the rich harmonic content and associated modulation components of measured signals in many machining operations. Many spectral or related decompositions have been applied, including various Fourier methods and also wavelet analysis [60], [66]. Short time or windowed decompositions like the short-time Fourier transform and the wavelet transform provide time localization [66], potentially valuable for analyzing non-stationary signals such as cutting scenarios with small work pieces or intermittent cutting. Cepstrum analysis presents great potential for the separation of system response and excitation [67], [31]. The properties of the complex logarithm make cepstrum-based TCM approaches less sensitive to sensor placement but more susceptible to noise [68]. The spectral correlation function can be used to explore the cyclostationary features of signals associated with milling. [61]. Periodic excitation at the cutting frequency, amplitude modulation due to tool symmetry and other phenomena related to material removal create multiple cyclostationary signals [5]. In [69], the authors apply cyclostationary analysis to monitor chatter in high speed milling. Reference [70] investigates the effect of force modulation of the boring bar in a lathe. The performance of cyclostationary analysis is compared with conventional spectral analysis. Application of spectral correlation function for TCM is described in [61]. Spectral correlation provides powerful opportunities for finding faults in periodic operations, e.g., the rotation of a mill cutter. However, these methods are usually applied to synchronously sampled data in order to align spectral frequency “bins” with the periodicity, e.g., rotation of the cutting tool. A quadrature encoder is attached, for example, to the machine spindle to synchronize sampling, a costly and invasive installation technique.
The classification approach to condition monitoring has grown in popularity with the increased interest in machine learning methods. Growth in computational power, data storage and interest for machine learning methods have contributed to this rise in popularity. For example, in [71], continuous wavelet transform scalograms of vibration signals from rotating machinery are fed to a CNN to produce good fault signatures and diagnostic indicators. In [72], a CNN and images of machining tools are used to diagnose tool wear. CNN has been used to predict important parameters such as remaining useful life of a mechanical tool [73]. These techniques require subject experts to train models and tune parameters. These methods produce heavy computation burdens in training and operation. It is always difficult or impossible to guarantee the response to situations that were not considered or “covered” by the training data.

This work proposes an instrumentation and measurement solution to condition monitoring for periodic operations like milling. The custom embedded hardware interfaces with common industrial accelerometers. The signal processing approach employs cyclostationary analysis tools to reveal hidden features of the vibration signal. Cyclostationary analysis is one of the analytical TCM methods that has presented high potential and gained acceptance [61, 69]. Despite the great characteristics of analytical methods, they are under-explored. A recent survey of milling TCM approaches [65] showed a disproportional interest in classification methods. As a result, cyclostationary based TCM lacks a normalized figure of merit and statistical verification. To address these shortcomings, a new normalized indicator is considered. The proposed tool condition indicator is demonstrated on a sample of end mills from an industrial facility. The impact of end mill mounting conditions and cutting parameters are also investigated.

Mechanical processes such as milling operations generate acoustic, vibration, and visual signals, all potentially useful for diagnostic indicators. Vibration sensing is a useful choice because of the availability of high bandwidth and high quality sensors. They provide signals that are directly useful for fault and wear detection. Vibration signals do not demand the computational burden of image analysis or suffer from
acoustic miscues. In this work, vibration signals drive the signal processing algorithm. However, other sensed signals, for example acoustic signals, could serve as inputs to the developed methods.

### 3.2.1 Cutting Tool Experiment

A vertical Bridgeport-style milling machine was used as a test bed. A CNC capability was necessary to achieve desired and repeatable cutting parameters. Three different types of cutting tools were used in the experiment: 4-flute 6 mm, 8 mm and 0.25 in carbide end mills. The acquisition device and sensor mounting points are marked in

![Image of milling machine experimental setup](image)

Figure 3-17: The view of milling machine experimental setup.

Fig. 3-17. The sensor is mounted on the bottom of the table. This sensor position is fixed for all of the experiments presented here. Note that this or a similar sensor location, isolated from the cutting action, is very practical for production environments. Experimental results confirm that the remote sensor location can provide data to successfully diagnose bit condition. Experimental setup details are provided in Table

89
3.2. Peripheral cuts with varying ADOC, RDOC and feed rate were considered. Two different tool mounting conditions were tested: R8 collets and ER16 collets.

Table 3.2: Setup Details, Milling Machine

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mill</td>
<td>ACER EVS-3VKH</td>
</tr>
<tr>
<td>Accelerometer</td>
<td>Wilcoxon 728T</td>
</tr>
<tr>
<td>Stock material</td>
<td>Aluminium</td>
</tr>
<tr>
<td>Stock grade</td>
<td>6061</td>
</tr>
<tr>
<td>6 mm end mill</td>
<td>WNT 54005060</td>
</tr>
<tr>
<td>8 mm end mill</td>
<td>WNT 54006080</td>
</tr>
</tbody>
</table>

3.2.2 Artificial Bit Degradation

To provide an illustrative application, as a demonstration, the MES was developed for vibration signal of 0.25 in end mills in a peripheral cut scenario. A spectral coherence map for a 0.25 in end mill is presented in Fig. 3-18. The ordinate corresponds to carrier frequency $f$, while the abscissa corresponds to cyclic frequency $\alpha$. In this experiment spindle speed was set to 2150 RPM. It can be seen that spectral coherence

![Spectral Coherence](image)

Figure 3-18: The spectral coherence map for sample vibration data.

is a discrete function of frequency $\alpha$, and a continuous function of frequency $f$. The
continuity of the spectral coherence in $f$ reveals the presence of random content in a wide frequency range [5]. Cyclic components are present at multiples of the spindle frequency, with spacing $\Delta \alpha = \frac{2150}{60} \approx 36$ Hz. Evaluation of the MES involves the integration of spectral coherence along frequency $f$, see (3.1), the direction of integration is marked in Fig. 3-18 with $\Delta f$. Integration will convert spectral coherence map into the 1D quantity of MES. However, as observed in Fig. 3-18, only a discrete set of frequencies $\alpha$ are meaningful, i.e., there are countable components due to cyclic operation. The MES will be represented with a bar plot.

To provide more intuition for the MES, a numerical experiment is conducted and compared with laboratory results. Four cases are considered: a new end mill and three degraded end mills. The degraded end mills are characterized as follows: one flute removed, two adjacent flutes removed and two opposite flutes removed. Fig. 3-19 presents an axial view of the of the considered end mills. The red outlines highlight the artificial degradation imposed on the tools. The simulated vibration signal for each scenario is presented in Fig. 3-20. The red envelope represents the amplitude modulation signal applied to the random carrier signal. Symmetry of the modulation signal in each case corresponds to engagement of the end mill cutting surface. The scaling of the signal is not important, the temporal features are of interest. The time period $T$ corresponds to the modulation period of the healthy four flute tool. Other

Figure 3-19: Artificial end mill degradation.
modulation periods are expressed in terms of $T$. The MES for each simulated case is presented in Fig. 3-21. The horizontal axis is modified from Hz to orders. The first order corresponds to an event that occurs once per spindle revolution, which rotates at a frequency of 36 Hz.

Figure 3-20: The simulated signal for the artificial degradation experiment.

Figure 3-21: The MES for simulated data.

Orders correspond to cyclic frequencies, i.e., frequencies of modulation. Therefore, an inherently related and easily observable quantity, the period of modulation, is analyzed. The fundamental period varies between cases (a)-(d) and has distinct
impact on the MES bar plots. For the healthy tool, the modulation period occurs four times per spindle rotation. Thus, the fundamental MES component in Fig. 3-21(a) appears at the fourth order. However, for one flute removed and two adjacent flutes removed, the modulation period occurs once per spindle rotation. In those cases, the fundamental MES components in Fig. 3-21(b)-(c) appear at the first order. For the tool with opposite flutes removed, Fig. 3-21(d), the modulation period occurs twice per spindle rotation. The non-sinusoidal nature of modulation contributes to higher order MES components at integer multiples of the fundamental MES order. The above discussion provides necessary insight for further analysis of laboratory results.

Laboratory measurements were conducted on one new and three artificiality degraded end mills. For each case of “damage”, the end mill flutes were hand ground on a bench grinder. Measured vibration signals for each of the cases (a)-(d) are presented in Fig. 3-22. Time domain interpretation of the signal content is difficult, i.e., the damage signatures in the measured signals are subtle and difficult to detect visually. In cases (c) and (d), some periodicity is visible as bursts in the vibration signal; however, any general modulation trend is hidden.

Figure 3-22: The vibration signal for the artificial degradation experiment.

The MES presents exceptional and intuitive insight. The MES plots for each case are given in Fig. 3-23. It can be seen in Fig. 3-23(a) that for the new tool, the
fourth order is dominant. However, contrary to the simulated case, the first order and its integer multiples are present. The lower orders appear due to imperfections in the original tool. The first order is dominant for cases (b) and (c), one flute ground and two adjacent flutes ground, respectively. This matches the observation from simulation. Even orders are dominant for case (d), two opposite flutes ground. However, there is a distinct dominance of sixth order when compared to the simulation data. Exact replication of relative magnitudes between the simulation and experiment is not expected. However, the trend of the MES indicator reveals the changes in degradation.

To further emphasize the benefits of the MES as a tool diagnostic indicator and justify the signal processing approach, a comparison with power cepstrum is developed. Cepstral analysis is a valuable tool in the diagnostic of mechanical systems and speech processing [31, 60, 68], as described in Chapter 2. For a given signal, the MES values of the first six orders contain important information. Similarly, for the cepstrum of a signal, the six lowest dominant quefrency components are considered to be informative [31]. The average deviation of the first six dominant MES values and six lowest dominant quefrency components of the same vibration data of a milling process corrupted with noise is considered. A white Gaussian noise with variance designed according to (3.2) is used in the analysis. By inspection of (3.2) the deviation
from the case with no added noise is expected to approach zero with increasing SNR, e.g., case with infinite SNR becomes the reference case. The results of the comparison are shown in Fig. 3-24. Each point in Fig. 3-24 corresponds to the average deviation of MES and power cepstrum quantities.

\[ \sigma^2 = \frac{P_{\text{sig}}}{10^{\text{SNR/10}}} \]  

(3.2)

where \( \sigma^2 \) is the noise variance, \( P_{\text{sig}} \) is the average power of the reference vibration signal. The average deviation of MES is uniformly better across the considered SNR range. For SNR values below -6 dB, the deviation for power cepstrum becomes excessively big, in fact the quefrency components cannot be distinguished from the noise floor. In this scenario, the MES continues to provide correct dominant components.

Figure 3-24: Comparison of the average deviation for power cepstrum and MES.

Simulation and experiments with intentionally damaged end mills, with flute surfaces marred to present a known pattern, demonstrate the utility of the MES for finding diagnostic indications in the cyclic and carrier frequencies. The next section extends this demonstration to end mills damaged by wear in a normal industrial environment.
3.2.3 Industrial Samples

The TCM system was tested with industrial end mill samples. Tools used in the experiment were retired from a CNC-based production line due to poor finish. A side by side comparison of new and worn samples is shown in Fig. 3-25. The distributed nature of the degradation is evident. Realistic wear is a complicated combination of multiple defects, including the simple cases from section 3.2.2. A useful condition indicator has to perform well in the complex realistic scenario. The vibration signatures of cuts with new and worn, 6 mm and 8 mm diameter industrial end mills were analyzed.

Figure 3-25: Good and bad 8mm carbide end mills.

In milling applications feed rates and cut parameters are set to achieve optimal chip load, which is recommended by tool manufacturers. There are two common milling strategies that aim to achieve optimal chip load, traditional milling and high efficiency milling. Traditional milling uses low ADOC, high RDOC and slower feed rates. Contrary to that, HEM uses low RDOC, high ADOC and higher feed rates [74]. The definitions of RDOC and ADOC are presented in Fig. 3-26 in a peripheral
For each end mill, four peripheral cuts were analyzed. The cut parameters are listed in Table 3.3. RDOC is expressed in terms of the percentage of the tool diameter. ADOC is expressed in terms of the percentage of tool cutting depth, which is typically twice the diameter. Feed rate is expressed in inches per minute. For example, Cut 1 with a 6 mm end mill results in the following values: 2.4 mm RDOC, 6 mm ADOC.

<table>
<thead>
<tr>
<th>Cut Number</th>
<th>Parameter</th>
<th>RDOC (%  ( \sigma ))</th>
<th>ADOC (% 2( \sigma ))</th>
<th>Feed Rate (IPM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut 1</td>
<td></td>
<td>40</td>
<td>50</td>
<td>15</td>
</tr>
<tr>
<td>Cut 2</td>
<td></td>
<td>5</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>Cut 3</td>
<td></td>
<td>10</td>
<td>100</td>
<td>30</td>
</tr>
<tr>
<td>Cut 4</td>
<td></td>
<td>5</td>
<td>100</td>
<td>45</td>
</tr>
</tbody>
</table>

The parameters were chosen to achieve almost optimal chip load for both 6 mm and 8 mm diameter end mills. The parameters of Cut 1 are representative of a traditional milling strategy. The parameters of Cut 4 are representative of a HEM strategy. The parameters of Cut 2 and Cut 3 are not prime candidates of either strategy, but all configurations maintain almost optimal chip load.
The MES bar plot for Cut 2 with an 8 mm tool is presented in Fig. 3-27. The height of the bars represents the average value of MES for each order. The error bars represent two standard deviations. The sample mean and sample variance were evaluated with a sample of 4 independent cuts for each tool. With the assumption of Gaussian distribution and treating sample mean and sample variance as distribution parameters, 95.4% of possible MES values are located within the error bars in Fig. 3-27. Even without this assumption, a valid bound is provided using Chebyshev’s inequality (3.3). This inequality can be applied to any distribution with defined mean and variance.

\[ P_r(|X - \mu| \geq 2\sigma) \leq \frac{1}{4} \]  

(3.3)

Using (3.3), and treating sample mean and sample variance as distribution parameters, 75% of possible MES values are located within the error bars in Fig. 3-27.

The MES provides a valuable condition indicator, however, a scalar indicator is of bigger interest. A normalized figure of merit was missing in previous cyclostationary based TCM literature. Simple threshold criteria can be applied on a single numeric value, MESbI:

\[ MESbI = \sum_{i=1}^{N} S_{<f}^<(i\Delta\alpha), \]  

(3.4)

where \( N \) is the total number of cyclic frequencies to be examined, \( \Delta\alpha \) is the frequency
spacing in the cyclic domain. For an idea of the relative MESbI variation between tests, the standard deviation of the MESbI is defined. Treating the MES values of each order as independent random variables, the standard deviation of the MESbI is defined in (3.5).

$$\sigma_{MESbI} = \sqrt{\sum_{i=1}^{N} \sigma^2(i\Delta\alpha)} \quad (3.5)$$

Of course, independence in this scenario cannot be guaranteed and is not necessarily expected. In general, a joint probability distribution is necessary for variance analysis. Additionally, without pair-wise independence a covariance matrix is necessary to characterize the problem. The definition in (3.5) does not survive rigorous mathematical scrutiny, however, it still provides useful indication of variability. Such information provides the potential user more insight. According to Fig. 3-27, very good separation of both cases can be achieved.

Tables 3.4 and 3.5 provide MESbI, $\sigma_{MESbI}$, and ratios of indicators for new and worn cases. There is a significant increase in the MESbI value for degraded end

<table>
<thead>
<tr>
<th>Cut Number</th>
<th>Worn</th>
<th>New</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESbI</td>
<td>$\sigma$</td>
<td>MESbI</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>Cut 1</td>
<td>2.76</td>
<td>1.76</td>
<td>0.08</td>
</tr>
<tr>
<td>Cut 2</td>
<td>3.12</td>
<td>2.05</td>
<td>0.18</td>
</tr>
<tr>
<td>Cut 3</td>
<td>2.84</td>
<td>1.96</td>
<td>0.06</td>
</tr>
<tr>
<td>Cut 4</td>
<td>3.21</td>
<td>2.08</td>
<td>0.08</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cut Number</th>
<th>Worn</th>
<th>New</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESbI</td>
<td>$\sigma$</td>
<td>MESbI</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>Cut 1</td>
<td>2.34</td>
<td>1.23</td>
<td>0.03</td>
</tr>
<tr>
<td>Cut 2</td>
<td>2.71</td>
<td>1.79</td>
<td>0.05</td>
</tr>
<tr>
<td>Cut 3</td>
<td>2.88</td>
<td>1.71</td>
<td>0.11</td>
</tr>
<tr>
<td>Cut 4</td>
<td>2.80</td>
<td>1.93</td>
<td>0.08</td>
</tr>
</tbody>
</table>

mills. A 90% increase in the MESbI value is seen for Cut 1 of the 8 mm end mills.
For 6 mm and 8 mm end mills, there is at least a 40% increase in the MESbI value from new to worn end mills.

The indicator proposed in this section is capable of diagnosing degraded end mills. In previous section MES was used for analysis of end mills with discrete and exaggerated damage. A single numeric MES indicator facilitates simpler analysis. The MESbI proved to be a good indicator for industrially-worn end mills with a complex combination of defects. Repetitiveness of separation between cases was validated with a sample of experiments.

### 3.2.4 Indicator Sensitivity

Results provided in section 3.2.3 validate the performance of the proposed indicator. There are multiple factors that can affect the indicator quality in an industrial environment. Stock and tool mounting conditions, general stiffness of the machine under study, cutting parameters and tool size can impact the overall machining process. Two major factors are addressed in this section: tool mounting conditions and non-optimal cutting parameters, i.e., cutting speed, feed rate, RDOC and ADOC.

#### 3.2.4.1 Tool Holder

There are multiple types of tool holder systems for end mills. A R8 tool holder system was used in the artificial degradation and industrial sample experiment. Cutting operations were repeated using an ER16 tool holder system. The MESbI values for traditional milling and 8 mm end mills are provided in Table 3.6.

<table>
<thead>
<tr>
<th>Cut Number</th>
<th>Worn MESbI</th>
<th>Worn σ</th>
<th>New MESbI</th>
<th>New σ</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut 1</td>
<td>2.03</td>
<td>0.04</td>
<td>1.20</td>
<td>0.04</td>
<td>1.69</td>
</tr>
<tr>
<td>Cut 2</td>
<td>2.87</td>
<td>0.06</td>
<td>2.07</td>
<td>0.03</td>
<td>1.39</td>
</tr>
</tbody>
</table>

Results in Table 3.6 validate the performance of the MESbI for an ER16 tool holder system. The proposed indicator performed well with two widely used tool
mounting solutions.

3.2.4.2 Non-optimal Chip Load

Optimal chip load parameters are often tabulated by tool manufacturers. For a RDOC less than the full radius of the tool, the feed rate must be adapted to avoid chip thinning [74]. The adjustment of the feed rate avoids rubbing and maintains desired MRR [74]. Although non-optimal cutting parameters are undesirable, the proposed indicator is verified in such a scenario. Fig. 3-28 presents the MES for six cutting scenarios of an 8 mm end mill. The first row presents analysis for cuts with traditional milling parameters. The second row presents data for cuts with HEM milling parameters. The last bar plot in the first row and the middle bar plot in the second row in Fig. 3-28 correspond to cuts with optimal chip load, Cut 1 and Cut 4 from Table 3.3. The other columns present data for cuts with non-optimal chip load. Presented results verify that the MES, and consequently the MESbI, perform well for non-optimal cutting configuration.

![Figure 3-28: Non-optimal cutting configuration.](image)

The failure to maintain optimal cutting parameters can cause pre-mature tool failure. A cutting scenario with RDOC of 10%, ADOC of 100% and 45 IPM feed rate resulted in the end mill breakage. The instant of tool damage is marked in the vibration data in Fig. 3-29.
Figure 3-29: Tool damage with non-optimal cutting parameters.
Chapter 4

Stretched IEPE Interface

Condition monitoring applications require various kinds of sensing to enable the underlying data processing. In case of mechanical measurements, vibration and acoustic signals are a very important class of signals used for condition monitoring. Operation of the machine under study can provide very valuable characteristics in the time or frequency (or both) domain representation of vibration and acoustic signals. However, high-fidelity and high-bandwidth acquisition of those signals is challenging. Robust solutions to interface with piezoelectric accelerometers and some microphones have been widely accepted in the industry [75]. Those standards could be extended to different types of measurands and allow for a more unified sensing strategy. There are several benefits of doing that; less dedicated front-ends for each sensor type, cabling management in the facility can be significantly simplified, etc. The advances in the embedded systems and rise of IoT devices are part of the "Fourth Industrial Revolution" and "Industry 4.0" [76, 77, 78]. To fully realize the potential of the IoT for monitoring and diagnostics, quality, not quantity, of data must be considered [56, 57, 59]. For example, a generic current loop vibration sensor may be easy to install but prove to be of little value if bandwidth, frequency limits, or sensitivity limits prevent the collection of useful information. Thus, it is important to provide sensing solutions that utilize the on-the-edge system capabilities but also preserve or even improve quality of the acquired data.

This chapter is organised as follows. Section 4.1 presents concept of the IEPE
interface and its advantages. Section 4.2 presents typical amplifier topologies to interface with piezoelectric transducers. Section 4.3 describes the concept of stretched IEPE interface and details of implementation, application examples are presented in Section 4.4. A cost-friendly solution for IEPE microphone and a new sensing strategy based on the stretched IEPE interface are discussed in Sections 4.5 and 4.6, respectively.

4.1 IEPE Interface

A current loop signal transmission is a workhorse for industrial process control [79]. Current loops allow for reliable transmission of PV data. The transmission of voltage signals over long two-wire channels is unreliable due significant voltage drops along the line and EMI introduced by harsh industrial environments. High noise immunity, long distance transmission and the availability of a shared power and signal path over the same two wires are some of the features that have made the 4-20 mA current loop a de facto industrial standard. Current loops do more than just PV transmission. A current signal is often used as a control signal. For example, current loops have gradually replaced the legacy 3-15 psi pneumatic control signal standard. Several modifications have been applied to the standard current loop, including, for instance, the Highway Addressable Remote Transducer Protocol (HART), which provides a means for digital communication over a two-wire current loop [80]. A wide range of industrial equipment is compatible with current loops due to the versatile nature of the system. A high-level view of a typical current loop is presented in Fig. 4-1.

Another significant application of current loops is to interface with the piezoelectric accelerometers. There are two types of piezoelectric accelerometers: charge-mode (charge-output) and voltage-mode (voltage-output) accelerometers [81]. Charge-mode sensors are characterized by flexible electrical parameters, a wide range of operating temperatures, wide dynamic range and compatibility with remote amplifiers [82]. Remote analog front-ends for charge-mode sensors are characterized by high impedance. Thus, charge-mode accelerometers are susceptible to noise and require
extra measures such as a low-noise cabling [83]. These limitations make the charge-mode architecture more common in laboratory conditions. In contrast, voltage-mode accelerometers have built-in electronics that make them less susceptible to noise [23]. Voltage-mode accelerometers commonly operate from a constant-current source over a two-wire interface such as a coaxial cable. The two-wire interface enables simultaneous power delivery to the accelerometer and analog signal transmission [82]. Voltage-mode accelerometers that operate with the two-wire interface are commonly denoted as the IEPE accelerometers. Due to the built-in electronics IEPE accelerometers do not require low-noise cabling [81]. Standard coaxial cables and connectors like BNC or SMA provide great performance at a reasonable cost for many applications. The mechanical features of coaxial cables and connectors provide good shielding for the whole transmission path. A typical IEPE interface is illustrated in Fig. 4-2. Several accelerator manufacturers use a proprietary name for the voltage-mode architec-

Figure 4-1: High-level diagram of a typical current loop.

Figure 4-2: Typical IEPE Interface.
ture, however, the name IEPE is commonly accepted to characterize the voltage-mode solutions. The IEPE interface is a *de facto* standard for high-quality industrial accelerometers and offers higher-fidelity vibration measurements compared to vibration measurements via a conventional 4-20 mA current loop [84]. There are relatively few IEPE solutions for other types of transducers. In particular, IEPE-compatible front-end electronics do not, unfortunately, allow for DC signal transmission.

Two types of amplifiers are employed in the IEPE accelerometer, either discrete FET and BJT amplifiers or operational amplifier based solutions. Typically, only the FET and BJT discrete solutions can benefit from the two-wire interface. Operational amplifier based solutions usually involve a three or four wire interface [81]. As seen in Fig. 4-2, the current source provides a DC current component and establishes a DC bias $V_{dc}$ across the impedance of the transducer. This DC bias directly impacts the dynamic range of the transducer [82]. The dynamic range is defined as the ratio between the largest and smallest signal values that can be measured. The built-in amplifier sets the sensitivity of the IEPE accelerometer. In the IEPE interface, DC bias is removed by an AC coupling capacitor. The AC component $V_{ac}$ of the signal is referred to a reference voltage $V_{ref}$, this allows for a unipolar power supply for the anti-aliasing filter and upstream analog front-end. The AC coupling does not limit the sensing performance, as long as, the effective high-pass corner frequency due to the AC coupling capacitor does not exceed transducer’s lowest frequency. The internal properties of a piezoelectric transducer limit available DC information anyway, so the piezoelectric transducer marries well with the IEPE hardware.

Piezoelectricity is understood as a linear electromechanical interaction between the mechanical and the electrical state in crystals [83]. Materials exhibiting piezoelectric properties serve as basic components of piezoelectric-based transducers. There are several piezoelectric materials commonly used in the sensing applications [85] and can be classified as: single crystals, piezoelectric ceramics, textures and thin films. Sensing materials can be natural and (or) synthetically generated. For instance, quartz ($SiO_2$) is probably the most common material used in piezoelectric transducers. It is a crystalline material that besides piezoelectric properties presents
several properties useful in the sensing applications. Quartz-based transducers do not present pyroelectric or ferroelectric properties what improves the SNR and response to the mechanical stress. Additionally, quartz components present good stability, i.e. aging properties, and high insulation resistance [28, 83]. As mentioned, quartz is the most common piezoelectric material in the sensing applications, however, acceleration transducers are commonly built with other materials. This is due to the need for small and light transducers, while transducers utilizing quartz tend to be bigger due to the size of the quartz crystal. Piezoelectric ceramic materials allow for transducer miniaturization [83]. Especially lead-zirconite-titanate mixed ceramics (PZT) gained popularity enabling high-sensitivity and low-noise sensing solutions [81].

Several models for piezoelectric materials were presented in the literature. The Mason’s model is one of the most common models for piezoelectric materials [86]. The Mason’s model uses a circuit model to represent an electro-mechanical coupling inherent to the piezoelectric materials. A different modeling approach is applied to model just the electrical properties of the piezoelectric material. The critical parameters in this case are: capacitance of the transducer $C_S$, insulation resistance $R_S$ and charge sensitivity, i.e. amount of charge generated as a response to the mechanical excitation. The circuit models can be found in [87, 85, 88, 28]. Fig. 4-3 presents two alternative forms of the model. The open circuit voltage in Fig. 4-3(b) is defined as $v(t) = \frac{q(t)}{C_s}$, where $q(t)$ is a charge response of the piezoelectric material. Due to the internal leakage of the piezoelectric material, the equivalent circuits shown in Fig. 4-3 have no excitation in steady state if the rate of change of charge is zero, i.e., at a static mechanical stress. The limitations of transducer bandwidth are modeled by the resistor and capacitor in the circuit model. Thus, the initial energy delivered to the RC circuit will be dissipated with a rate depending on the circuit time constant. Additionally, amplifiers interfacing piezoelectric components do not present infinite input impedance and the dissipation rate is even faster.

This chapter demonstrates an instrumentation and measurement strategy for increasing compatibility of the IEPE interface to a wider range of sensors. Custom hardware extends the IEPE interface to low-frequency and DC signal transmission via
frequency modulation. Following a description of the expanded hardware interface, applications for temperature and strain measurements are presented. Additionally, an inexpensive IEPE microphone is presented. The new sensor interface and microphone are combined with an embedded device to form a condition monitoring system.

4.2 Amplifiers for Piezoelectric Transducers

A piezoelectric material presents high impedance output and requires a high input impedance amplifier for proper processing. There are several different approaches towards classifying amplifier solutions and the nomenclature should be clarified. Two distinct amplifier topologies are: voltage amplifier and charge amplifier [81]. It should be mentioned that sometimes authors use a third term, an electrometer amplifier [83]. The electrometer amplifier has a positive gain which depends on the capacitance of the transducer and cabling; thus, it shares the same features with the aforementioned voltage amplifier, a term used by authors in [81, 88].

Yet another way to classify the piezoelectric transducers is based on the location of the front-end amplifier, such a classification was briefly introduced in the introduction of this chapter. The first option is a voltage-mode (or voltage-output) transducer. This term does not imply voltage amplifier topology being used but it describes transducers with built-in amplifiers. Both types of amplifiers, i.e. voltage and charge amplifiers, can be built-in into the casing of the voltage-mode transducer. The built-
in amplifier provides a low impedance output at the sensor terminals, i.e. internal electronics converts signal from high impedance input into a signal at low impedance output. This is very beneficial for signal transmission and enables high noise immunity with standard coaxial cables [89, 23]. The second solution is a charge-mode (or charge-output) transducer. In this case amplifier is connected with a cable to the remote transducer. A remote amplifier increases the operating temperature range because the amplifier circuitry does not have to be exposed to higher temperature. Another consequence of the built-in amplifier are fixed electrical parameters, while remote amplifier can have variable settings allowing for higher flexibility. A great discussion of transducers classification is given in chapter 1 of [81].

4.2.1 Charge Amplifier

Charge amplifiers are a common solution to interface with the piezoelectric materials [82]. The charge amplifier was first proposed by W. P. Kistler in 1950 [83]. The operational amplifier based charge amplifier is presented in Fig. 4-4. The gain of the amplifier is derived below and current expressions are provided in (4.1). The

![Figure 4-4: The charge amplifier](image)

AC analysis is performed with $V_{com}$ shorted to ground. In this case, voltage across amplifier inputs equals to $-v_i(t)$; thus, output can be written as $v_o(t) = -Av_i(t)$,
where $A$ is the open loop gain of the amplifier.

$$
\begin{align*}
   i_{C_f}(t) &= C_f \frac{d(v_o(t) - v_i(t))}{dt} \\
   i_q(t) &= \frac{dq(t)}{dt} \\
   i_{C_i}(t) &= -C_i \frac{d(v_i(t))}{dt}
\end{align*}
$$

(4.1)

$C_i$ in equation (4.1) denotes a parallel combination of $C_c$ and $C_s$. Applying the Kirchhoff current law at node $V_i$ and ignoring the feedback resistance $R_f$ and resistance $R_s$ (4.2) can be written.

$$
\frac{dq(t)}{dt} = \frac{dv_i(t)}{dt} (C_f(A + 1) + C_i).
$$

(4.2)

Applying the $v_o(t) = -Av_i(t)$ relation and integrating equation (4.2) (assuming initial conditions nullifying the integration constant) the following expression is achieved:

$$
v_o(t) = -\frac{q(t)}{C_f(1 + \frac{1}{A}) + \frac{C_i}{A}}.
$$

(4.3)

The output contribution from $V_{com}$ can be written as:

$$
V_o = V_{com}((1 + \frac{R_f}{R_s}) - \frac{R_f}{R_s}) = V_{com}.
$$

(4.4)

Thus, the total output voltage for the ideal amplifier, i.e., with infinite open loop gain and infinite bandwidth, can be written in the following form:

$$
v_o(t) = V_{com} - \frac{q(t)}{C_f}
$$

(4.5)

The Laplace domain expression is given in equation (4.6), a full feedback impedance and sensor model circuit components are considered. Equation (4.6) was derived with the assumption of the ideal amplifier, i.e. $V_i(s) = 0$, and current source $sQ(s)$.

$$
V_o(s) = \frac{V_{com}}{s} - Q(s) \frac{sR_f}{sC_fR_f + 1}
$$

(4.6)
It can be seen in (4.6) that feedback resistance introduces a high-pass action to the frequency response. This limits the static performance, however, stabilizes amplifier operation from output drift and saturation. In practical circuit a series resistor $R_s$ is added at the inverting input to protect from electrostatic discharges and provide high-frequency roll off in the circuit frequency response [88]. For very high resistance $R_s$ this additional pole is located at $\omega_p = \frac{1}{R_s(C_s+C_c)}$

It can be noticed that due to the feedback operation and high gain, ideally infinite, the presented amplifier maintains zero voltage between its inputs. This virtual short in the ideal case eliminates the influence of cable capacitance, which appears across amplifier inputs [90, 88]. The immunity to capacitance variations is not the sole benefit of utilizing charge amplifier. The only capacitance that determines the circuit response is a feedback capacitance; thus, the time constant of the circuit can be made really high by utilizing capacitors with very high insulation resistance. This minimizes the discharge rate and enables measurements of very low-frequency signals. In ideal case DC quantities could be measured. Unfortunately, there are several practical limitations that contribute to a so called drift of the amplifier [83], which lead to saturation of the amplifier output. The main reason for this is the input bias current that keeps charging the feedback capacitor. In order to mitigate this a parallel resistor is applied in the feedback line, however, this introduces a high-pass action, refer to (4.6), and only a quasi-static signals can be measured. Nevertheless, charge amplifier enables relatively low-frequency measurements and strongly contributed to popularization of the piezoelectric transducers since 1960s. According to [81], the IEPE transducers with charge amplifier are the most common solution on the market. Another point to consider is that the bandwidth of the amplifier is not infinite and at high frequencies the virtual short assumption becomes weaker. Therefore, cable and sensor capacitance will introduce a low-pass action to the frequency response.

The charge amplifier is utilized in both charge-output and IEPE transducers. The IEPE transducers are less sensitive to capacitance variations due to the built-in electronics. However, charge-mode transducers use remote amplifiers. The benefits of applying charge amplifier in this case are very high, cables of various types and lengths
can be employed without the loss of performance. Additionally, amplifier settings can be altered by the user. Those benefits do, however, come at a cost. Typically, a low-noise cabling is required to provide appropriate shielding and minimize the noise contribution from the cable. The triboelectric effect is the most commonly cited in the literature as a cause for special low-noise cabling [91]. The triboelectric effect causes charge fluctuations between interface wires when mechanical stress is applied to the cable. Mechanical stress may momentarily separate dielectric and wires what induces the triboelectric effect [91, 81]. This charge contribution cannot be distinguished from the charge contribution due to the piezoelectric effect. Thus, the SNR is decreased. A low-noise cable has a graphite lubricant embedded in the dielectric layer to minimize friction and triboelectricity [82], older solutions were utilizing oil lubrication [83]. Obviously, this significantly increases the cost per-channel of the whole monitoring system.

It should be noted that the "charge amplifier" term is a misnomer. As the analysis above revealed, charge amplifier does not amplify electric charge but rather converts it into voltage with a conversion ratio fixed by the circuit parameters, in particular by the feedback capacitance.

4.2.2 Voltage Amplifier

Voltage amplifiers are the second topology used with piezoelectric materials. This solution significantly differs from the previously described charge amplifier. The operational amplifier based voltage amplifier is presented in Fig. 4-5. The gain of the amplifier is given derived below in the Laplace domain. Due to the properties of ideal operational amplifier input voltage \( V_i(s) \) can be expressed in (4.7).

\[
V_i(s) = sQ(s) \frac{1}{ \frac{1}{R_s} + sC_s + sC_c + \frac{1}{R_b}}
\]  

(4.7)
With feedback network in Fig. 4-5 and non-inverting amplifier topology the output voltage is expressed as:

$$V_o(s) = V_i(s)(1 + \frac{R_f}{(sC_f R_f + 1) R_1}).$$  \hspace{1cm} (4.8)

After combining (4.7)-(4.8) and including response due to the voltage $V_{com}$ the output voltage can be expressed as shown in (4.9)

$$V_o(s) = \frac{V_{com}}{s} + Q(s) \frac{s R_a R_b}{s R_a R_b (C_s + C_c) + R_a + R_b} \cdot (1 + \frac{R_f}{(sC_f R_f + 1) R_1}).$$ \hspace{1cm} (4.9)

Analysis of (4.9) reveals two poles. A high-pass pole $\omega_{HP} = \frac{1}{R_a || R_b || C_a || C_c}$ and a low-pass pole $\omega_{LP} = \frac{1}{R_f C_f}$.

First of all, the output voltage depends on multiple circuit components. Both sensor and cable electrical parameters influence the gain expression and contribute in the form of the pole responsible for high-pass action. Dependence of the output voltage on the cable parameters requires calibration whenever cable is replaced. However, IEPE transducers with voltage amplifiers have fixed parameters, since piezoelectric material interfaces the amplifier directly in the casing. Feedback components define the pole responsible for low-pass action in the frequency response. Another important difference is a positive amplifier gain, whereas charge amplifier is an inverting amplifier.

The above analysis revealed that the cable capacitance is not shorted by the
amplifier action, as it was the case for charge amplifier. This results in a non-zero voltage across the cable capacitance $C_c$. Due to the very high impedance of the piezoelectric material this voltage is prone to noise. Thus, a very good shielding is required for remote amplifiers.

4.3 IEPE Transducer

The AC coupling does not limit the measurement of vibration or acoustic signals; however, it limits possible applications of the IEPE interface for measurements of quantities such as temperature, pressure or mechanical strain [83]. Conventionally, other two-wire interfaces can transmit DC measurements but lack the benefits provided by the IEPE interface. A two-wire interface that allows for high-fidelity DC and AC signal transmission, as well as power delivery over the same medium, is of great interest. To address this need, an extension of the IEPE interface is proposed in this chapter in a form of IEPE transducer. This extension enables new measurements to be acquired over the IEPE interface. The new sensing solution utilizes frequency modulation for signal transmission; thus, it presents high noise immunity [92]. Fig. 4-6a) presents possible input connections to the proposed IEPE transducer, i.e., single-ended voltage source or Wheatstone bridge configuration. Fig. 4-6b) shows a differential amplifier at the transducer input. Figs. 4-6c) and d) present the power stage and on-board oscillator, respectively.

4.3.1 Input Stage, Power Stage and Amplifier

Fig. 4-6a) presents two alternative ways of connecting the sensing element. A passive sensing element with measurand-dependent resistance is commonly used with the Wheatstone bridge configuration. A small resistance $R_i$ is added in series with the Wheatstone bridge to limit current consumption. The series resistance impacts the mapping between sensed voltage $V_s$ and resistance of the sensing element $R_s$. The mapping of $V_s$ to $R_s$ is given in (4.10).
\[
R_s = \frac{V_{cc}R_x^2 - V_s(3R_iR_x + 2R_x^2)}{V_s(R_i + 2R_x) + V_{cc}R_x} \tag{4.10}
\]

Sensing is not limited to the Wheatstone bridge configuration. Single-ended voltage fed sensing elements can be used as an input to the transducer. Fig. 4-6a) presents a TMP35 temperature sensor [93] providing voltage \(V_s\). In order to accommodate for a wide class of input signals, a differential input amplifier with high-input impedance is used. A differential amplifier with input buffers is presented in Fig. 4-6b). The amplifier converts a differential signal into a single-ended quantity without loading the input stage.

The power stage is shown in Fig. 4-6c). It is built around a bias network \(R_b\|C_b\) and a series regulator. The current source generates a DC bias across the bias network, which is used to generate a stable voltage \(V_{cc}\) for on-board electronics.

### 4.3.2 Oscillator Stage

The single-ended control signal, \(V_o\), at the output of the amplifier stage is used as a control input to the oscillator stage of the transducer. It controls the center frequency
of the voltage-controlled oscillator. There are several voltage-controlled oscillator solutions for high-frequency applications [92]. However, to limit the burden on the acquisition device, an audio frequency range relaxation oscillator is realized. High frequency modulation requires very high sampling frequencies or modulation and demodulation techniques, e.g., IQ modulation. Such a configuration would limit compatibility with many commercial IEPE acquisition devices; thus, is avoided.

The relaxation oscillator used in the proposed transducer is presented in Fig. 4-6d). It consists of an operational amplifier, comparator, one transistor and eight passive components. In order to achieve desired performance, the components around the negative feedback amplifier are designed as follows: $R_3 = R_4$ and $R_1 = 2R_2$. To reflect this and enable further derivations, the oscillator circuit is repeated in Fig. 4-7, together with necessary node labels and waveform of voltage at node $V_z$. The

![Oscillator Circuit](image)

Figure 4-7: The oscillator stage.

input operational amplifier operates with negative feedback and the voltage at node $V_z$ equals to $\frac{1}{2}V_o$, i.e., half the control input. To get the relation between input voltage and frequency of the oscillator, the capacitor current $i_{C_1}(t)$ is evaluated in both states of switch $Q_1$. The high gain of the amplifier and negative feedback lead to expressions
for $i_{C_1}^m(t)$ and $i_{C_1}^{off}(t)$ in terms of the voltage at node $V_x$.

$$
\begin{align*}
&i_{C_1}^m(t) = v_x(t) \frac{R_{ds} - 0.5R_1}{R_1(0.5R_1 + R_{ds})} \\
&i_{C_1}^{off}(t) = \frac{v_x(t)}{R_1},
\end{align*}
$$

(4.11)

where $R_{ds}$ is the drain-to-source resistance when MOSFET is turned on. If $R_{ds}$ is neglected, $i_{C_1}^m(t) = -i_{C_1}^{off}(t)$.

A virtual short assumption does not apply to the output stage comparator inputs because of the positive feedback. However, limit values of the voltage at node $V_c$ allow for calculation of the voltage swing across $C_1$, namely $\Delta V_z$. The limit values at node $V_c$ are marked in Fig. 4-7 and can be expressed in (4.12) with the assumption of $V_{cc}$ or ground potential at the output of comparator in saturation.

$$
\begin{align*}
&V_{ch} = V_{cc} \frac{R_3}{R_3 + R_3||R_4} \\
&V_{cl} = V_{cc} \frac{R_3||R_4}{R_3 + R_3||R_4}
\end{align*}
$$

(4.12)

The voltage swing at node $V_z$ is $\Delta V_z = V_{ch} - V_{cl}$. The static operating point is considered next, i.e., $v_o(t) = V_o$. Capacitor currents are static in each stage of operation and with $R_{ds}$ neglected are equal to $i_{C_1}^m(t) = -\frac{0.5V_o}{R_1}$ and $i_{C_1}^{off}(t) = \frac{0.5V_o}{R_1}$. The charge-voltage relationship of a capacitor is applied, which leads to $\Delta t_1$ and $\Delta t_2$ as expressed in (4.13). With a constant input signal, the AC component of the capacitor voltage equals the AC component at node $V_z$.

$$
\Delta t_1 = \Delta t_2 = \frac{\Delta V_z C_1}{I_{C_1}},
$$

(4.13)

where $I_{C_1}$ equals to $|\frac{0.5V_o}{R_1}|$. Based on (4.13), the frequency $F_{osc}$ of the output signal is equal to:

$$
F_{osc} = \frac{1}{2\Delta t_1}.
$$

(4.14)

The output signal drives the base of transistor $Q_2$ in Fig. 4-6c). The transistor connects a resistor in parallel with the biasing network $R_b||C_b$ and periodically modulates
4.3.3 Dynamic Performance

The relaxation oscillator operation may introduce a delay at transients. To quantify the delay, a maximum $\Delta t_{\text{max}}$ and minimum $\Delta t_{\text{min}}$ are evaluated with (4.13). In the realized circuit input signal $V_o$ can be between 0.05-3.25 V (with $V_{cc} = 3.3 \, V$) and at zero response it equals $V_{com}$, see Fig. 4-6b). The voltage swing $\Delta V_z$ across the feedback capacitor $C_1$ is equal to 1.775 V. Assuming maximum change of the control signal from 0.05 V to 3.25 V, the feedback capacitor current changes from 0.58 $\mu$A to 37.8 $\mu$A based on (4.11) ($R_1 = 43 \, k\Omega$ and $C_1 = 470 \, pF$). Assuming a transient in the input signal presented in Fig. 4-8 and following (4.13), the maximum period deviation $|\Delta t_{\text{max}} - \Delta t_{\text{min}}|$ is 1.4 ms. In the next cycle, the oscillator resumes the correct frequency. The extreme change of the control signal from 50 mV to 3.25 V corresponds to a step change across the full range of the measured quantity, see Fig. 4-8. In practice, all physical phenomena have limited bandwidth and a step change is very exaggerated. A more realistic bound on input signal bandwidth is considered. The input impedance $R_s$ is varied between 100 $\Omega$ to 70 k$\Omega$ while $R_i$ and $R_x$ in the bridge in Fig. 4-6a) are equal to 499 $\Omega$ and 1 k$\Omega$, respectively. With an amplifier stage of gain 0.5, the input voltage of the oscillator circuit ranges from 1.20 V to 2.24 V. Considering (4.11)-(4.14), the frequency range is given in Fig. 4-9. The oscillator frequency varies between 8.02 kHz to 14.93 kHz. Maximum frequency of the
input signal is constrained to less than approximately 10% of the minimum oscillator frequency. Thus, the input signal frequency content should not exceed 1 kHz. For a typical thermistor characteristic, the considered range of input resistance corresponds to a wide temperature range between $-55^\circ$C and $90^\circ$C [94]. The oscillator frequency is limited by the Nyquist rate of the IEPE acquisition device. Higher sampling frequency enables increased oscillator frequency and thus higher input signal bandwidth.

![Graph](image)

Figure 4-9: The oscillator frequency as a function of input voltage.

All four stages described above enable a wide range of measurements over the IEPE interface. The oscillator stage encodes the instantaneous value of the input signal, either from the Wheatstone bridge or the single-ended voltage source, into a frequency of the output waveform. The sensing configuration presented in Fig. 4-6a) enables two-wire measurements in scenarios that in general require at least three wires, i.e., $V_{cc}$, ground and output signal. Additionally, the conversion of sensing element properties into an electrical quantity, i.e., transduction, can happen very close to the system under study. The proposed transducer reduces the length of the connection between a high impedance sensing element and tranducing element. Consequently, the noise immunity is improved. The transducer output is transmitted with a standard coaxial cable and BNC or SMA connector technology, simplifying cable management. The transducer is displayed in Fig. 4-10. A stack of two printed circuit boards was used for a cylindrical transducer shape. The aluminum case provides external
protection and EMI shielding. Once assembled, only the input and SMA connectors are exposed.

Figure 4-10: The proposed IEPE transducer and custom aluminium case.

4.4 Applications of the Proposed IEPE Transducer

The proposed IEPE transducer was tested with several sensing elements. In this section, example applications with an NTC thermistor and a strain gauge are presented. Sensitivity analysis describes the transducer performance in more detail.

4.4.1 Sensitivity Analysis

Modern digital acquisition systems can handle a wide range of transducer non-linearity, however, a linear transducer response is desired [83]. Equation (4.13) and Fig. 4-9 show that the amplifier and oscillator stages present the desired linearity. Despite the linearity of the amplifier and oscillator stages, the mapping from sensing element signal to its resistance is non-linear, see (4.10). However, this is a feature of the Wheatstone bridge and not the proposed topology. If a voltage source sensing element is used, as shown in Fig. 4-6c), linearity is preserved.

The frequency of the output signal is expressed in (4.15), see Fig. 4-6 for circuit details.

$$F_{osc} = \frac{V_{com} - V_{s} \left( \frac{R_1}{R_2} \right)}{2R_1 \cdot 2\Delta V_z C_1}$$

(4.15)

Sensitivity analysis based on (4.15) provides insight into oscillator frequency error...
due to variations in component values. Equation (4.16) provides expressions for the relative error of oscillator frequency.

\[
\begin{align*}
S_{C_1} &= \frac{C_1 \delta F_{osc}}{F_{osc} \delta C_1} = C_1 \Delta V_z C_1, \\
\left( \frac{1}{I^m_{C_1}} + \frac{1}{I^off_{C_1}} \right) \Delta V_z C_1^2 \left( \frac{1}{I^m_{C_1}} + \frac{1}{I^off_{C_1}} \right) &= -1 \\
S_{R_f} &= \frac{R_f \delta F_{osc}}{F_{osc} \delta R_f} = 1 \\
S_{R_g} &= \frac{R_g \delta F_{osc}}{F_{osc} \delta R_g} = -1
\end{align*}
\]

Equation (4.11) was derived under assumption of ideal matching between resistor values. In general, the voltage at node $V_x$ will deviate if the resistors in Fig. 4-7 are not matched. Component labels from Fig. 4-6d) are used in the formulas below.

\[
\begin{align*}
S_{R_1} &= \frac{R_3 R_4 R_1}{R_3 R_4 R_1 - R_3^2 R_2} - 2 \\
S_{R_2} &= -\frac{R_4 R_1 - R_3 R_2}{R_3 R_2} \\
S_{R_3} &= \frac{R_4 R_1 - 2R_3 R_2}{R_4 R_1 - R_3 R_2} - \frac{R_3}{R_3 + R_4} \\
S_{R_4} &= \frac{R_4 R_1}{R_4 R_1 - R_3 R_2} - \frac{R_3}{R_3 + 2R_4} \\
S_{R_5} &= -\frac{R_5 R_6 + R_5 R_7 + R_6 R_7}{R_5 R_7} \\
S_{R_6} &= -\frac{R_5 R_6 + R_5 R_7 + R_6 R_7}{R_7(R_6 + R_5)} \\
S_{R_7} &= \frac{R_5 R_6 + R_5 R_7 + R_6 R_7}{R_7(R_6 + R_5)}
\end{align*}
\]

When $R_3 = R_4$ and $R_2 = 0.5R_1$, the sensitivities have the following values: $S_{R_1} = 0$, $S_{R_2} = -1$, $S_{R_3} = -0.5$, and $S_{R_4} = 0.5$. Expressions for $S_{R_5}$, $S_{R_6}$ and $S_{R_7}$ depend on the particular circuit implementation, with the designed circuit parameters $S_{R_5}$, $S_{R_6}$ and $S_{R_7}$ have the following values: $S_{R_5} = -0.23$, $S_{R_6} = -0.23$, and $S_{R_7} = 0.4624$.

For uncorrelated errors, the total mean square error of the oscillator frequency due to the variations in component values is given in (4.18). High-precision and high thermal stability components should be used to minimize the total error, e.g., ceramic...
capacitors with C0G dielectric and 0.1% tolerance resistors. For 0.1% tolerance resistors and a 1% tolerance capacitor, the total mean square error of oscillator frequency is 1.02%. Thus, the frequency error for the oscillator in Fig. 4-7 is driven mostly by the deviation in capacitance value, $S_{F_{osc}} \approx S_{C_1}$. Commonly available C0G ceramic capacitors in the range of 500 pF are typically only offered with tolerance above 1%.

$$S_{F_{osc}} = (S_{C_1}^2 \delta_{C_1}^2 + S_{R_f}^2 \delta_{R_f}^2 + S_{R_g}^2 \delta_{R_g}^2 + \sum_{n=1}^{7} S_{R_n}^2 \delta_{R_n}^2)^{\frac{1}{2}} \quad (4.18)$$

The $\delta_X$ in (4.18) is a standard deviation of value of component $X$.

### 4.4.2 IEPE Temperature Transducer

Temperature sensing is an example application of the new sensing solution. A conventional NTC thermistor is used as a sensing element $R_s$, see Fig. 4-6a). The resistance of the sensing element varies between 40 $\Omega$ and 70 k$\Omega$ throughout its operating range. A TMP35 temperature sensor was used as a reference. Both the NTC thermistor and TMP35 sensor have similar packaging and dimensions [93, 94]; thus, the impact of different dynamics of the sensing elements is neglected in the comparison. Fig. 4-11 shows the reference measurement and the response of the IEPE temperature transducer. Data presented in Fig. 4-11 was acquired without calibration of the transducer, expressions (4.10)-(4.14) were used to convert frequency into the resistance of the sensing element.

A very good dynamic response can be seen in Fig. 4-11. The steady state response shows a good match with the reference temperature profile. The temperature deviation $\Delta T$ is equal to 1.5°C. This is less than the accuracy of TMP35 [93], which is $\pm 2^\circ C$.

### 4.4.3 IEPE Strain Gauge

Another application example is measurement with a strain gauge. A conventional Wheatstone bridge can be supplied by either a voltage, Fig. 4-6a), or current source.
The current source configuration should not be confused with the proposed transducer. There are a few market solutions offered as IEPE strain gauges, such as the model 740B02. Available solutions have a non-zero cut-off frequency in the low frequency range [95], i.e., the transducer is not able to measure static or very slowly varying phenomena. In the case of a strain gauge, this could be a static strain on the monitored structure. As demonstrated with temperature, the ability to perform static measurements is a key feature of the proposed solution. A strain gauge with 3.5 kΩ nominal resistance and gauge factor of two was used in the test. The strain gauge was attached to a subwoofer membrane to allow for a controllable excitation frequency. The response of the IEPE strain gauge to a 10 Hz excitation is presented in Fig. 4-12. The offset in the data in Fig. 4-12 is a combination of the static deformation of the strain gauge due to the natural curvature of the membrane, as well as the inherent DC offset of the acquisition channel.

4.4.4 Transducer Output Processing

The output of the proposed IEPE transducer is a continuous frequency-modulated voltage waveform. As described in previously, the accuracy of the transducer output is determined by component tolerances. However, the method used to process the transducer output can introduce finite measurement resolution. The output is processed
in real-time or recorded and processed offline. For example, in an offline analysis, the Hilbert transform is applied for instantaneous phase angle and frequency calculation. A frequency estimate is mapped to the measurement domain with (4.10)-(4.15). In such a configuration, resolution is not a concern. However, the Hilbert transform is non-causal and is not suited for real-time processing. In real-time applications, comment on processing resolution is necessary.

Real-time processing is investigated with temperature measurement used as an example. In a particular circuit realization, the transducer output has a fundamental frequency of 15.4 kHz for a sensing resistance of $R_s = 0 \Omega$ at the input stage. An output with a fundamental frequency of 8.02 kHz corresponds to a sensing resistance of $R_s = 70 \, k\Omega$. For the NTC thermistor described in [94], this range of resistance covers temperatures between -55°C and 125°C. The temperature, $T$, is plotted as a function of frequency, $F$, in Fig. 4-13. Due to the characteristic of the thermistor, the mapping is non-linear. A natural candidate for accurate real-time frequency estimation is the time-dependent discrete Fourier transform (DFT). However, a finite length DFT imposes limits on frequency resolution $\Delta F$. To provide a quantitative measure of the impact of DFT frequency resolution on temperature sensing, the DFT of length 32768 is considered for data sampled at a rate of 48 kHz. In this case, $\Delta F$ equals 1.46 Hz, i.e., the abscissa of Fig. 4-13 is discretized in increments of 1.46
Hz. Due to the non-linear nature of the characteristic in Fig. 4-13, the temperature resolution varies with frequency. From 10 kHz to 13 kHz, the temperature-frequency characteristic is well approximated as linear. Within this region and with the assumed DFT resolution $\Delta F = 1.46 \, \text{Hz}$, the temperature resolution is $\Delta T \approx 0.017 \, ^\circ\text{C}$.

Figure 4-13: Temperature as a function of oscillator frequency.

A detailed analysis revealed that the magnitude of all IEPE transducer sensitivity coefficients $|S|$ is less than or equal to one. The dominant component of output frequency error is due to the feedback capacitor in the oscillator stage. Two example applications of the proposed IEPE transducer were investigated. Both the thermistor and strain gauge measurements demonstrated successful extension of the IEPE interface. Table summarizing the transducer design is provided in Appendix C.

### 4.5 IEPE Microphone

Microphones are characterized by the principle of transduction. They convert acoustic waves into electrical signals. Microphones are used in settings ranging from musical performance to electromechanical system diagnostics [96, 97]. The most common types of microphones are: dynamic, piezoelectric, condenser, and electret microphones. Dynamic microphones use a moving coil for transduction, piezoelectric microphones use the piezoelectric effect, condenser and electret microphones use ca-
pactance variations. Each technology requires a specific amplifier topology [90]. For example, condenser microphones offer high sensitivity and good frequency response, however, they require a high voltage source to polarize the sensing element and thus have limited application in embedded devices. Sensing elements built with piezoelectric or electret materials do not require polarization circuitry and are conducive to compact sensing solutions [90].

Acoustic signals enable a wide range of diagnostics [98, 99]. A two-wire interface providing excellent shielding is beneficial in noisy industrial environments. IEPE microphones provide these benefits to the acquisition of acoustic signals. There are a few vendors offering IEPE microphones, some of the available solutions are referred to as constant current power (CCP) microphones [100].

The IEPE class of microphones is characterized by a very high cost. A typical price starts around $500 and much more expensive models are available. These high-end microphones provide excellent performance for critical applications, however, there is a wide range of monitoring applications where high cost is prohibitive. A much cheaper and still reliable IEPE microphone is of great interest. To the best of authors knowledge, an inexpensive IEPE microphone solution is lacking. Additionally, it is very hard to get detailed information on the built-in electronics of IEPE-compatible transducers. The sparse literature presents simplified schematics that do not allow for full comparison or reference [81]. A reference that discusses affordable current loop microphones is [101]. Although not a scientific reference, it highlights the need for an affordable IEPE microphone and discusses an example implementation.

A low-cost IEPE microphone that utilizes an electret capsule is presented in this section. It employs a current-fed common emitter amplifier with emitter resistance as a pre-amplifier stage. A small-signal analysis provides a gain expression that matches experimental results. The schematic of the proposed circuit is shown in Fig. 4-14a). A key benefit of this implementation is significantly lower cost when compared to available market solutions. The proposed microphone is not introduced to replace high-end and expensive market solutions, but is meant to enable monitoring for applications where very high instrumentation cost is prohibitive.
Figure 4-14: IEPE microphone: a) large-signal model, b) small-signal model.

The DC operating point is evaluated to calculate the transconductance parameter $g_m$, which is necessary for the small-signal model of the bipolar transistor. The transconductance parameter is expressed as $g_m = \frac{I_c}{V_T}$, where $I_c$ is the collector bias current and $V_T$ is the thermal voltage. To calculate the bias current, the load coupling capacitor $C_L$ in Fig. 4-14a) is open and electret capsule is modeled with resistance $R_{ds}$. At static condition, the microphone can be modeled with resistance because of the output JFET stage inside the electret capsule [90]. The formula for $I_c$ is given in (4.19). The bias current calculated with (4.19) matches the DC analysis of the circuit in the SPICE simulator.

$$I_c = \frac{R_{in}'I_s - V_{be}}{R_e + R_{in}'}$$

where $R_{in}'$ is the sum of $R_{ds}$ and $R_{in}$, $V_{be}$ is the base-emitter voltage in the active region of operation ($V_{be} \approx 0.7$ V), $\alpha$ is the bipolar transistor current gain parameter.

The small signal model of the BJT transistor allows for analytical evaluation of amplifier gain. The electret capsule with built-in JFET is considered in the design. The capsule output is modeled as a current source. The complete small signal circuit is presented in Fig. 4-14b). The employed electret capsule has a sensitivity parameter of $S_i = 8.08 \frac{\mu A}{Pa}$ [102, 103]. A 300 Hz harmonic source was used to generate 83 dB SPL measured at the capsule to validate the performance. The output signal, after rejecting DC bias voltage, presented a 45 mV peak value. The estimate of the peak output voltage $V_o$ based on the derived gain and input SPL is given in (4.20). The gain of the circuit is expressed as $G = \frac{V_o}{I_m}$, a detailed expression is provided in (4.21).
The designed circuit has a gain of 75 dB.

\[ V_o = S_i \cdot G \cdot SPL_{peak} = 19 \text{ mV}, \]  
(4.20)

where SPL of 83 dB is converted to 0.282 Pa and peak SPL value was calculated with a crest factor for sine wave. The electret capsule has a ±4 dB tolerance [102] and thus the output estimate can range up to 30 mV. Additionally, the microphone case alters SPL in comparison to the SPL meter used to set the experimental reference value of 83 dB. All considered, the estimated peak value closely matches the experimental results. The IEPE microphone is shown in Fig. 4-15.

\[ \frac{v_C}{i_m} = \frac{AR_o(1 - g_mR_1) + (R_1 + R_\pi)B}{C - R_oD - (R_1 + R_\pi)E}, \]  
(4.21)

where:

\[ A = R_1R_\pi + \frac{R_1^2R_\pi(1 + g_mR_\pi)}{a+c} \]

\[ B = \frac{R_1R_\pi(1 + g_mR_\pi)(g_mR_1R_o + R_1)}{a+c} \]

\[ C = b(R_1 + R_\pi) \]

\[ D = (1 - g_mR_1)(R_\pi + \frac{R_1a}{a+c}) \]

\[ E = (g_mR_1R_o + R_1)\frac{a}{a+c} \]

\[ a = (1 + g_mR_\pi)R_\pi + \frac{R_\pi + R_1}{R_o}R_\pi \]

\[ b = \frac{R_oR_L + R_oR_1 + R_LR_1}{R_L} \]

\[ c = \frac{R_\pi + R_1}{R_e}R_\pi \]

To verify the performance of the developed IEPE microphone, a THD comparison with a microphone that utilizes a transimpedance amplifier is presented. The transimpedance amplifier is a widely accepted solution to interface with pre-polarized
sensing elements [90, 103], such as an electret capsule. In the experiment, the SPL at the electret capsule was held constant at 72 dB for all considered frequencies. Both systems, i.e., IEPE and transimpedance amplifiers, worked with the same model of electret capsule. The THD for both types of amplifiers is presented in Fig. 4-16, the microphone with the transimpedance amplifier is denoted as a voltage microphone. To further characterize the IEPE microphone, a sensitivity plot is presented in Fig. 4-17. The sensitivity data is presented for two SPL values corresponding to 72 dB and the standard 94 dB.

The proposed circuitry provides good performance with a low-cost electret capsule. The IEPE microphone provided essentially the same THD as a common amplifier solution. The analytical expression for circuit gain closely matches the experimental results. Table summarizing the microphone design is provided in Appendix C.
4.6 A New Sensing Strategy

Condition monitoring systems provide information about the health of critical components. Reliable, modular and minimally intrusive condition monitoring systems are of interest to a variety of disciplines. Each measured signal plays a role in condition monitoring. Vibration, acoustic, temperature, strain, pressure and other types of signals all help to evaluate the operating status and condition of systems under study. The diversity of measurands leads to proprietary connectors, expensive hardware and interface-specific amplifiers. These constraints increase the complexity and cost of instrumentation for condition monitoring. The stretched abilities of the IEPE interface presented in Section 4.3 simplify the necessary hardware for condition monitoring. Paired with an acquisition device, the ensemble forms a compact, versatile and cost-friendly IEPE based monitoring solution. A custom wireless data acquisition device is presented in Fig. 4-18.

An example of the suggested monitoring solution is presented in Fig. 4-19, stretched capabilities of the IEPE interface allow to measure wide range of quantities, far beyond vibration and acoustic signals. Simplicity and compactness of the system are evident.
4.6.1 Acquisition device

Custom acquisition hardware is proposed as a component of a system level condition monitoring solution. The wireless device offers remote processing power and non-volatile memory for selective measurement and storage of raw time-series data. The custom instrumentation offers benefits in data acquisition, detection and non-intrusive sensing. There are four simultaneously sampled differential channels that are IEPE-compatible. Non-volatile memory and a real-time clock enable a lightweight data management system. Files are stored using the standard FAT file system to comply
with most modern operating systems. The device operates as a server, clients can check files in the remote memory, download files, delete files and ask for new data to be acquired. Detail description of the hardware is provided in Chapter 5.

4.6.2 Condition Monitoring System Solution

Small footprint, IEPE compatibility and wireless communication of the proposed device facilitate a general condition monitoring system. Measurements for predictive maintenance are taken throughout the life-cycle of equipment. In some cases, the most illustrative condition indicators arise from a measurement of a specific physical operation or phenomenon, e.g., the spin down of rotating machinery [104]. Measurements for predictive maintenance are classified as short duration, high priority. In contrast, measurands such as temperature and pressure are continually monitored and classified as long duration but low priority for predictive maintenance. The idea of measurement priority is central to the proposed condition monitoring system. Physical processes with different timescales introduce a natural measurement hierarchy. Natural order can be leveraged for device resource distribution. Variables with low predictive maintenance priority, such as temperature, are very valuable as low-latency alarm criteria, e.g., an overheating bearing can be directly signaled as abnormal. A high-level demonstration of the system with a motor-pump setup is given in Fig. 4-19. The wireless device, equipped with transducers, microphones, and accelerometers, can coordinate various measurements through a small number of IEPE channels. Natural timescales permit time sharing of the IEPE channels and increase effective channel count.
Chapter 5

Hardware Support

This chapter describes hardware that was developed to enable some of the diagnostic approaches described in the previous chapters. Section 5.1 introduces details of the switched capacitor filter and its performance. Section 5.2 presents an FPGA-based platform to enable real-time processing of signals from IEPE transducer described in Chapter 4 with details of fixed-point algorithm implementation. Section 5.3 describes the custom acquisition platform that was optimized for vibration measurements. Analog front-end and digital circuitry are discussed. Detailed noise analysis characterizes the analog part of the design. Schematics and details of the circuitry are provided in Appendix C.

5.1 Switched Capacitor Notch Filter

The condition monitoring approach presented in section 3.1 utilizes measurements of both electrical and mechanical quantities [18]. The sidebands of fundamental current component were used as a metric to distinguish damaged diaphragm from a new one, as well as clogged inlet valve from a clean one. The nature of the induction machine current, fed from a sinusoidal voltage source, limits the diagnostic information included in the fundamental component of current. For example, a motor connected to ideal 60 Hz grid draws a 60 Hz fundamental component of current that is responsible for torque production and power delivered to the machine. The RMS value of
current enables calculation of average power but is of little diagnostic importance in this configuration.

In [105] authors used an analog notch filter to filter out fundamental component of current and enhance induction machine slot harmonic detection for speed estimation purposes. Of course digital implementation of notch filter is possible, however, it requires the ADC to sample total current signal (fundamental component and remaining content) without saturation. In the application presented in [105] and for described condition monitoring system, fundamental component is of no information but it can be several orders of magnitude higher than components of interest. Thus, it is desirable to filter the dominant fundamental component prior to sampling. The filtered signal can be amplified to enhance content of interest and fully utilize the full-scale voltage of the ADC. This improves the SNR of the sampled data and better utilizes the available number of ADC bits [26]. Unfortunately, conventional analog filter has a fixed center frequency, and thus, its performance degrades when power source fundamental frequency deviates from nominal value or if it is variable in general, e.g., in case of variable-frequency drive. Variable-frequency drives are used in flow control of diaphragm pumps; by means of motor speed control pump’s average flow is adjusted.

5.1.1 Switched Capacitor Filter

In order to allow for variable center frequency the notch filter was realized with switched capacitor technology [106], see Fig. 5-1. Switched capacitor circuit is neither analog nor digital filter, it is a discrete-time filter due to the sampled nature but non-quantized amplitude [26].

A high-level diagram presenting the implementation of the switched capacitor filter is given in Fig. 5-2. Several building blocks are necessary to create an adjustable notch filter. A frequency detection stage is necessary for calculating the required switching frequency for filter block. Thus, a frequency-dependent signal is an input to a microcontroller adjusting the switching frequency based on the current input. Due to the discrete-di time nature of the switched capacitor filter, an anti-aliasing
filter is required in front of every filter stage. Additionally, an anti-aliasing filter is necessary at the output of the switched capacitor filter to remove the out-of-band frequency components before analog-to-digital conversion. The filter realization can incorporate an amplifier to enhance the signal of interest, i.e., slot harmonics or sidebands due to modulation.

The switched capacitor filter block was built around the LTC1060 universal dual filter building block from Linear Technology; an example application of the designed
hardware is presented in [106]. This dual filter building block consists of two 2\textsuperscript{nd} order filter blocks that can be cascaded together to achieve a 4\textsuperscript{th} order filter. Each block can generate different filter functions such as low-pass, band-pass, high-pass, notch and all-pass [107]. Filter block is designed to operate in mode 1b according to the datasheet designations [107]. It is recommended that for this mode of operation the circuit is designed such that the ratio of control signal frequency $f_{clk}$ to center frequency $f_0$ meets the following relationship:

$$500 \geq \frac{f_{clk}}{f_0} \geq 100$$

(5.1)

The 4\textsuperscript{th} order filter is realized by cascading two blocks building blocks of LTC1060. First stage with lower Q and second stage with higher Q are characterized in Table 5.1. The filter is designed with 100:1 prescaling of the control signal frequency. Experimental results revealed that filter provides best performance with input signal in the range of 1 V\textsubscript{pp}. Thus, hardware described in [106] is equipped with adjustable voltage divider. Additionally, a non-inverting amplifier at the output of the switched capacitor filter allows to match amplitude of signal of interest to the full-scale voltage of the ADC. The anti-aliasing stages are implemented as shown in Fig. 5-2, a simple first-order filter with buffer. Buffers are necessary to satisfy low source impedance requirement on the input pins of the LTC1060 [107]. The corner frequency $f_c$ is designed as:

$$f_c = \frac{1}{2\pi RC} = 2.12 \text{kHz},$$

(5.2)

where $R=75$ $\Omega$ and $C=1 \mu F$ are circuit parameters in Fig. 5-2. Designed filter limits frequency content above Nyquist frequency. Thus, the switching frequency $f_{clk}$ should never be below $2 \cdot f_c = 4.24$ $kHz$. In the designed circuit, the center frequency of the

<table>
<thead>
<tr>
<th>Stage</th>
<th>Q</th>
<th>$\frac{f_{clk}}{f_0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.6783</td>
<td>139</td>
</tr>
<tr>
<td>2</td>
<td>6.8450</td>
<td>139</td>
</tr>
</tbody>
</table>

Table 5.1: Switched Capacitor Filter Circuit Parameters
filter should not be below 45.9 Hz, see (5.3). This design was optimized to work in grid-connected mode with 50 Hz or 60 Hz nominal utility frequency.

\[ f_o \geq \frac{2 \times f_c}{f_{clk}} = 42.44 \text{ Hz}, \]  

(5.3)

An anti-aliasing filter at the output of the switched capacitor filter block is a part of the acquisition device performing analog-to-digital conversion.

### 5.1.2 Frequency Detection

Several approaches are possible for fundamental frequency detection. A zero crossing detector, hysteresis comparator or digital single-phase PLL are some of possible implementations [108]. In the designed hardware, a hysteresis comparator was chosen to minimize the computational burden on the microcontroller performing acquisition and communication related tasks. Digital PLL would create additional computational burden [108]. A circuit diagram representing the frequency detection block is given in Fig. 5-3. The input low-pass filter consisting of \( R_1 \) and \( C_1 \) is designed with corner frequency of 160 Hz to limit the impact of fundamental component harmonics and noise on detection process. The filtered signal is AC coupled, scaled and referred to \( \frac{1}{2} V_{cc} \) (\( R_2 = R_3 \)). A comparator with positive feedback is used to implement hysteresis

![Figure 5-3: Hysteresis comparator circuit.](image-url)
comparator. Thresholds of the comparator circuit are defined below:

\[ V_l = V_{cc} \frac{R_3||R_4}{R_3||R_4 + R_2} \]  

\[ V_h = V_{cc} \frac{R_3}{R_2||R_4 + R_3} \]  

where \( R_2 = R_3 \), \( V_{cc} \) is the supply voltage and \( R_x||R_y \) denotes parallel combination of resistors \( R_x \) and \( R_y \). This means that voltage \( V_{con} \) changes state whenever voltage at node \( V_z \) exceeds the range between \( V_l \) and \( V_h \). Output signal \( V_{con} \) is a two-state signal and can be used as a trigger for timer interrupt of the microcontroller in diagram in Fig. 5-2. This provides a way to adjust the center frequency of the notch filter digitally by adjusting frequency of the control signal. Resistor \( R_4 \) in the designed hardware [106] is implemented as potentiometer to provide an additional degree of freedom to the user. The AC coupling network is designed to pass frequencies above 0.8 Hz, see Table 5.2 for details of hysteresis circuit. An actual circuit implementation has additional gain block between the output of the low-pass filter and the AC coupling capacitor \( C_{ac} \) that is not shown in Fig. 5-3.

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>1 kΩ</td>
</tr>
<tr>
<td>( C_1 )</td>
<td>1 μF</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>100 kΩ</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>100 kΩ</td>
</tr>
<tr>
<td>( R_4 )</td>
<td>( 1^{-5}-1 ) MΩ</td>
</tr>
<tr>
<td>( C_{ac} )</td>
<td>1 μF</td>
</tr>
<tr>
<td>( R_{ac1} )</td>
<td>100 kΩ</td>
</tr>
<tr>
<td>( R_{ac2} )</td>
<td>100 kΩ</td>
</tr>
</tbody>
</table>

Sample current data is presented in Fig. 5-4. It can be seen that the fundamental component is heavily filtered and that the sideband frequency components are distinct. It should be noted that in case of a heavily distorted signal, harmonics of fundamental component may still dominate the spectrum and the designed filter will not attenuate those frequency components.
5.2 Hardware Support for IEPE Transducer

The IEPE accelerometer is a de-facto standard for high-quality industrial vibration sensing. The AC component of the output voltage conveys information about the mechanical excitation. Despite the simplicity and robustness of the interface, other types of IEPE transducers are lacking. To address this need, an extension of the IEPE interface was proposed. This extension enables new measurements to be acquired over the IEPE interface. Due to the modulation technique it is possible to transmit DC measurements, despite the inherent AC-coupling of the IEPE interface. Details of the implementation and circuit performance are provided in Chapter 4.

So far, the proposed IEPE transducer results presented in this thesis were acquired in parallel to vibration measurements and frequency content was converted off-line to the measurand domain using Hilbert transform [26] and circuit parameters. This approach provides the instantaneous phase of the signal which can be mapped for instance to temperature (in case of sensing element reacting to temperature changes).

This chapter demonstrates hardware support for the stretched IEPE interface which enables real-time signal conversion. It uses an FPGA to perform real-time computations and provide a measured quantity to the user. Temperature measurement is used as an example application. Further sections present details of signal processing and implementation of the processing pipeline. The final result of conversion is displayed...
over the VGA interface to the user. The presented results were obtained with Nexys 4 DDR evaluation platform that is equipped with the XC7A100T-1CSG324C FPGA from Xilinx.

5.2.1 Processing Pipeline

The data processing pipeline requires several stages. The pipeline is realized in two clock-domains, the 100 MHz domain for signal processing and 65 MHz for VGA related logic (with 1024x768 resolution). Functional components of the pipeline are listed below:

- Sampling of the frequency modulated signal
- Digital band-pass filtering
- Calculation of the magnitude spectrum
- Detection of the fundamental frequency
- Frequency estimate to voltage conversion
- Voltage estimate to resistance of sensing element conversion
- Resistance to measurand conversion
- Display functions

The functionalities presented above are realized with a processing pipeline given in Fig. 5-5. Synchronization between modules is performed with AXI, however, custom modules do not necessarily strictly follow the AXI standard. Further sections provide details of the particular blocks, pipeline was implemented in the Xilinx Vivado Design Suite. List of modules in the 100 MHz pipeline:

- fir128
- xfft_0
5.2.2 Sampling and Filtering

The sampling and quantization of the input signal is performed with the xadc module. Physically sampling is performed at 961.54 kHz. Then the signal is decimated to provide the desired signal rate. The range of frequency modulation is between 8-15 kHz; sampling frequency was designed as 48 kHz. In order to minimize the interference
of undesirable content the sample signal is filtered with a band-pass FIR filter. The magnitude frequency response of the 128 coefficient FIR filter is presented in Fig. 5-6. The filter coefficients were designed with the Parks–McClellan algorithm.

![Figure 5-6: The magnitude frequency response of the band-pass filter.](image)

The FIR filter is implemented with a cyclic buffer of size equal to the number of filter coefficients. If the filter size is a power of two, then an unsigned index variable (index in the buffer) naturally overflows and does not need to be controlled. This enables efficient iteration over the buffer evaluating and accumulating sample-coefficient products. Similarly, the second index variable of the same size stores location of the most recent sample which is a starting point for calculation of a new filter output. Latency of this implementation equals to the number of filter coefficients, in this design it is equal to 128 clock cycles.

### 5.2.3 Time-frequency Conversion

The filtered signal is an input to the Xilinx IP subsystem performing FFT. The architecture of the FFT IP core is designed as fixed point pipelined streaming I/O. It uses the highest number of resources but provides highest throughput for continuous processing which is essential in this application. The Fast Fourier Transform v9.1 core was used in this project.
The frequency representation of data blocks is essentially the short-time discrete Fourier transform (STFT). There is no overlap and adjacent blocks are processed sequentially. Data blocks are extracted with a basic rectangular window. Frequency resolution is a key factor in this application, while magnitude of the frequency content is of no interest. Thus, a rectangular window, with smaller mainlobe width than many popular windowing functions, is a good candidate. The size of window was designed as 32768. Consequently, with a sampling frequency of 48 kHz the frequency resolution equals to $\Delta f = \frac{F_s}{2^15} = 1.46 \, Hz$. A sample view of the FFT module output is presented in Fig. 5-7, it is an output of 1024 long FFT. It should be noted that with the desired FFT size the output data in not available in one burst. The AXI bus with indication of last output element is necessary to synchronize the following module.

![Sample output of a 1024 long FFT.](image)

The following module in the pipeline extracts the magnitude spectrum. The square root of the final output is a vector of magnitude values and is used as an input to the peak extraction module. Peak is localized using the FFT properties for real input signals, and thus, only half of the spectrum is investigated. A simple approach iterating over all discrete frequencies between DC and Nyquist frequency is applied. All of the above modules are fully pipelined to maximize the throughput. The modules relevant to this section are reported in Table 5.3 with a corresponding latency.

The latency reported for the FFT module corresponds to a number of clock cycles between last input sample and last output sample. At 100 MHz it corresponds to approximately 650 ms.
Table 5.3: Module Latency

<table>
<thead>
<tr>
<th>Module</th>
<th>Latency (clk)</th>
</tr>
</thead>
<tbody>
<tr>
<td>fir128</td>
<td>128</td>
</tr>
<tr>
<td>xfft_0</td>
<td>65060(last in to last out)</td>
</tr>
<tr>
<td>square_and_sum</td>
<td>2</td>
</tr>
<tr>
<td>cordic_0</td>
<td>17</td>
</tr>
<tr>
<td>peak_search</td>
<td>1</td>
</tr>
</tbody>
</table>

5.2.4 Frequency to Resistance Conversion

The output of the `peak_search` module is a 16 bit unsigned integer that needs to be converted into frequency value in Hz. The frequency estimate \( F_{est} = \frac{\text{PeakIndex}}{N_{FFT}} F_s \) (\( F_s \) denotes sampling frequency and \( N_{FFT} \) denotes the FFT length) is not in general an integer; thus, the following operations require fixed point arithmetic. In the expression for \( F_{est} \) the FFT size \( N_{FFT} \) is a power of two number and bit shifting can replace the division operation. The fixed point data format was designed as Q16.16 format.

The frequency value is mapped to the value of the sensed voltage \( V_s \), see Fig. 4-6. Mapping is done according to (5.6)-(5.8). Please refer to Fig. 4-6 for circuit components details.

\[
\frac{1}{2F_{est}} = \frac{V_{pp} C_1}{I_{C_1}},
\]

where \( V_{pp} \) denotes the voltage ripple across feedback capacitor, which is fixed by the thresholds of the output comparator in Fig. 4-6b), and \( I_{C_1} \) denotes the feedback capacitor current.

\[
I_{C_1} = \left| \frac{0.5(V_{com} - G \times V_s)}{R_1} \right|
\]

(5.7)

Taking into account gain of the differential amplifier in Fig. 4-6b) \( F_{est} \) can be expressed in (5.8).

\[
F_{est} = \frac{V_{com} - G \times V_s}{4 \times V_{pp} R_1 C_1},
\]

(5.8)

where \( V_{com}, G = \frac{R_1}{R_{eq}} \), \( R_1 \) and \( C_1 \) are constants defined by circuit parameters and scaled to Q16.16 format. The \( V_s \) signal is the unknown input signal, i.e., output of the Wheatstone bridge. Inversion of this equation results in a multiplication by a
constant, subtraction and division operations. Gain of 0.5 ($\frac{R_f}{R_g} = 0.5$) can be included into values of $V_{ref}$ and $4 \times V_{pp} R_1 C_1$ to simplify the calculations.

By inspection of the circuit in Fig. 4-6d) it can be noticed that feedback capacitor current can be characterized by (5.9) depending on the state of switch $Q_1$. In the ideal case, currents in each phase of circuit operation are equal in magnitude, however, value of $R_{ds}$ will impact the symmetry.

\[
\begin{align*}
|i_{C_1}^{on}(t)| &= v_x(t) \frac{0.5R_1 - R_{ds}}{R_1(0.5R_1 + R_{ds})} \\
|i_{C_1}^{off}(t)| &= v_x(t) \frac{0.5R_1}{R_1},
\end{align*}
\]  

(5.9)

where $R_{ds}$ is the drain-to-source resistance when MOSFET is turned on. If $R_{ds}$ is neglected, $i_{C_1}^{on}(t) = -i_{C_1}^{off}(t)$. In order to account for the asymmetry, the average of current in both phases is considered $|i_{C_1}(t)| = 0.5(|i_{C_1}^{on}(t)| + |i_{C_1}^{off}(t)|)$. A value of $R_1$ in (5.8) becomes an effective resistance $R_x$ as written in (5.10). A revised formulation of (5.8), providing revised frequency estimate, is given in (5.11). An additional factor of two in (5.11) comes from the averaging of current expressions.

\[
R_x = \frac{1}{\frac{1}{R_1} + \frac{R_1 - R_{ds}}{R_1(0.5R_1 + R_{ds})}},
\]  

(5.10)

\[
F_{est} = \frac{V_{ref} - G \times V_s}{8 \times V_{pp} R_x C_1},
\]  

(5.11)

The voltage estimate allows to calculate the resistance estimate, i.e., sensing resistance, see Fig. 4-6a). Resistance is estimated with the following equation:

\[
R_{est} = \frac{V_{cc} R_x + V_s (3R_i + 2R_x)}{-V_s \left( \frac{R_i}{R_x} + 2 \right) + V_{cc}},
\]  

(5.12)

where $R_i$, $V_{cc}$, $R_x$ are constants defined by circuit parameters and scaled to Q16.16 format. The above calculations are performed in the `sensing_resistance` module. It performs necessary fixed point multiplications, scaling and division. Latency of the module equals to 37 clock cycles, as reported in Table 5.4. Latency is mostly driven
by the divider generator from Xilinx library. The divisor and dividend arguments are 32 bit numbers, output of the divider is a 64 bit number scaled to the original format. The divider is implemented with a non-blocking radix-2 architecture with fractional output.

Table 5.4: Module Latency, Frequency Conversion

<table>
<thead>
<tr>
<th>Module</th>
<th>Latency (clk)</th>
</tr>
</thead>
<tbody>
<tr>
<td>center_freq_est</td>
<td>2</td>
</tr>
<tr>
<td>center_freq_2_voltage</td>
<td>3</td>
</tr>
<tr>
<td>sensing_resistance</td>
<td>37</td>
</tr>
</tbody>
</table>

5.2.5 Resistance to Temperature Conversion

The characteristic of the sensing element is strongly nonlinear [94]. Equation (5.13) is used to calculate temperature estimate based on the value of $R_{est}$.

$$T_{est} = \frac{T_0 \beta}{T_0 ln\left(\frac{R_{est}}{R_0}\right) + \beta}$$ (5.13)

where $T_0$, $R_0$ and $\beta$ are constants defined by thermistor characteristic. Output of (5.13) is an unsigned number in the Kelvin scale.

It can be seen that evaluation of the natural logarithm is required to calculate the temperature estimate. Unfortunately, the Xilinx IP library provides only natural logarithm implementation for floating point arithmetic, this implementation is as a part of the Floating-Point Operator core. Thus, in order to complete the calculations a custom implementation of the natural logarithm for fixed point arithmetic is necessary. Due to the fixed point nature, the fast binary logarithm algorithm was used [39, 40].

The main idea behind the algorithm is based on the Al Kashi’s method. It uses sequential squaring and division operations. In the radix-2 representation, a division by two can be efficiently implemented with bit shifting. In order to obtain base two logarithm of $x$, i.e., $y = log_2(x)$, the following representation is used: $x = 2^y$. In the
radix-2 representation and for $1 \leq x < 2$ the following can be written:

$$x = 2^{2^{-1}(y_1+2^{-1}y_2+...)}$$  \hspace{1cm} (5.14)

Squaring both sides of (5.14) provides the following:

$$x^2 = 2^{y_1}2^{2^{-1}(y_2+2^{-1}y_3+...)}$$  \hspace{1cm} (5.15)

It can be recognized that $2^{2^{-1}(y_2+2^{-1}y_3+...)}$ is a number greater or equal to one and less than two. Thus, (5.15) is greater than two if $y_1$ is one, i.e., bit at position one has value one, see (5.16).

$$y_1 = \begin{cases} 
1, & \text{if } x^2 \geq 2 \\
0, & \text{otherwise} 
\end{cases}$$  \hspace{1cm} (5.16)

If $y_1$ is one, (5.15) is divided by 2; if not, there is no change to (5.15). In both cases, the reminder $2^{2^{-1}(y_2+2^{-1}y_3+...)}$ has the same form as (5.14), i.e., it can be squared and analyzed in the same fashion. This is the main idea behind the fast binary logarithm algorithm. It should be noted that the algorithm relies on $x$ values between one and two. Thus, the input number has to be scaled accordingly before the first algorithm iteration. If $N$ is the number of initial divisions by two and $M$ is the number of initial multiplications by two, the final mantissa $y'$ equals to:

$$y' = y + N - M$$  \hspace{1cm} (5.17)

The described algorithm calculates base 2 logarithm of $x$, however, the property in (5.18) can be used to convert it to a desired base. Conversion can be efficiently implemented by multiplication with a constant $\frac{1}{\log_2(b)}$.

$$\log_b(x) = \frac{\log_2(x)}{\log_2(b)}$$  \hspace{1cm} (5.18)

With the fixed point implementation of the natural logarithm, final evaluation of (5.13) is possible. Table 5.5 presents latency of the modules involved in the conversion
to temperature. The division in (5.13) is included in the convert_temperature module and contributes to its latency. This module uses the same architecture of the divider generator as the sensing_resistance module. The high latency of the base_2_log module is due to its iterative nature. A look-up table could be an alternative for logarithm evaluation, however, with 32 bit variables this would be considerable memory effort. Additionally, this algorithm can be scaled for operations with words of different sizes.

<table>
<thead>
<tr>
<th>Module</th>
<th>Latency (clk)</th>
</tr>
</thead>
<tbody>
<tr>
<td>base_2_log</td>
<td>170</td>
</tr>
<tr>
<td>natural_logarithm</td>
<td>3</td>
</tr>
<tr>
<td>convert_temperature</td>
<td>46</td>
</tr>
</tbody>
</table>

5.2.6 Temperature Conversion

The unsigned $T_{est}$ result is converted to the Celcius scale by subtracting 273.15. The result is a 64 bit signed number in (°C). The data width is a result of the divider block in the convert_temperature module which generates 64 bit result.

$$T_{est}^C = T_{est} - 273.15$$  (5.19)

5.2.7 Pipeline Performance

Latency values for each module were reported in the previous subsections. The overall data processing pipeline is implemented in fixed point arithmetic, mostly in the Q16.16 format. The values in Tables 5.6-5.7 present results for a sample of center frequencies. The results were obtained with both: fixed point representation and double-precision floating-point representation (floating-point arithmetic was implemented in Matlab).
Table 5.6: Fixed Point Performance, Temperature

<table>
<thead>
<tr>
<th>$F_{osc}$ (Hz)</th>
<th>$R_{est}/R_0$</th>
<th>$T_{est}$ (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FPGA Mat.</td>
<td>FPGA Mat.</td>
</tr>
<tr>
<td>11967.0117</td>
<td>1.18531 1.18531</td>
<td>294.396 294.395</td>
</tr>
<tr>
<td>11997.0117</td>
<td>1.1664 1.1664</td>
<td>294.644 294.768</td>
</tr>
<tr>
<td>12324.6875</td>
<td>0.97741 0.97743</td>
<td>298.947 298.945</td>
</tr>
<tr>
<td>12354.6875</td>
<td>0.9614 0.9615</td>
<td>299.341 299.340</td>
</tr>
</tbody>
</table>

Table 5.7: Fixed Point Performance, Sensed Voltage

<table>
<thead>
<tr>
<th>$F_{osc}$ (Hz)</th>
<th>$V_s$ (mV)</th>
<th>$\ln(R_{est}/R_0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FPGA Mat.</td>
<td>FPGA Mat.</td>
</tr>
<tr>
<td>11967.0117</td>
<td>94.665 94.680</td>
<td>0.1699 0.17005</td>
</tr>
<tr>
<td>11997.0117</td>
<td>85.678 85.675</td>
<td>0.1593 0.1540</td>
</tr>
<tr>
<td>12324.6875</td>
<td>-12.542 -12.538</td>
<td>-0.02287 -0.022875</td>
</tr>
<tr>
<td>12354.6875</td>
<td>-21.545 -21.531</td>
<td>-0.03932 -0.03925</td>
</tr>
</tbody>
</table>

5.2.8 Memory

A dual port BRAM memory is used to buffer the temperature data. Width of the memory word is designed as 8 bytes (final temperature result). Depth of the memory is designed as 128. This corresponds to temperature data for equivalent time window of 87 seconds, $N_{FFT}T_s \approx 87$ s.

Additional role of the dual port memory is to enable safe clock domain crossing as marked in Fig. 5-5. The signal processing pipeline operates at 100 MHz, while display pipeline operates at 65 MHz.

5.2.9 Display

The temperature measurement is displayed on a monitor over the VGA interface. The basic functionality displays temperature measurements corresponding to 64 measurement windows, i.e., approximately 43 seconds, see Fig. 5-8. Horizontal dimension of the display is discretized into 64 segments. During each monitor refresh cycle 64 segments are read from the BRAM starting at the most recent measurement.

Besides waveform display, graphical interface provides numerical value of current
frequency and temperature estimates $F_{\text{est}}$ and $T^C_{\text{est}}$ [109]. To enable this feature, a COE file with ASCII symbols is stored in the ROM memory. Symbols stored in the ROM are 8x16 pixels; thus, the ROM memory size is $16 \times 8 \times 128 = 16384$. Standard ASCII set was modified to include a degree symbol. The block diagram view of the text controller is given in Fig. 5-9.

An additional RAM memory is used to store the ASCII encoded text representation. Value of the encoded text is used to generate input address to the ROM memory storing value for a particular pixel, see Fig. 5-9. In every refresh cycle new values of $F_{\text{est}}$ and $T^C_{\text{est}}$ are written to the RAM memory. This provides a way to continuously update the displayed text.

![Figure 5-8: The graphical interface.](image)

The latency of the memory read cycle equals to two clock cycles. It is critical to synchronize the output of the RAM memory with the input to the ROM memory.

![Figure 5-9: Block diagram of the VGA text controller.](image)
Otherwise, at a given screen coordinate, the pipeline will read from a wrong address in the ROM memory. Without synchronization text distortion can occur as can be seen in Fig. 5-10. Delaying the hcount and vcount signals by two clock cycles (memory latency), see Fig. 5-9, provides correct screen coordinates to the glyph controller. In other words, output data of the RAM memory is provided to the glyph controller with correct hcount and vcount values, see Fig. 5-9.

Figure 5-10: The effect of lack of synchronization between RAM and ROM memories.

5.2.10 Device Utilization

This section provides device utilization summary, as well as power analysis results. The utilization report is shown in Fig. 5-11. It can be noticed that 52% of the available BRAM resources are used. High memory utilization arises to a large extent from the FFT implementation. A pipelined streaming architecture with 32768 long FFT utilizes most of the project BRAM. The result of power analysis is given in Fig. 5-12.

5.2.11 Sensor Resolution

Values of $T_{est}$ and $ln\left(\frac{R_{est}}{R_0}\right)$ for two frequencies are reported in Table 5.8. Both the hardware generated results and the software generated results (with double-precision floating point representation in Matlab) are reported. It can be noticed that the reported ratio of temperature change to frequency change equals to $\frac{\Delta T}{\Delta F} = 0.0135 \frac{K}{Hz}$ for the hardware implementation and $\frac{\Delta T}{\Delta F} = 0.0117 \frac{K}{Hz}$ for the software calculation. The value of the ratio $\frac{\Delta T}{\Delta F}$ will change depending on the investigated frequency range as can
be seen in the frequency-temperature characteristic in Fig. 5-13. Fig. 5-14 presents frequency-resistance characteristic. In the linear part of the frequency-temperature characteristic and with the FFT resolution of $\Delta F = 1.46 \, \text{Hz}$ the temperature resolution equals to $\Delta T \approx 0.017 \, \text{K}$. The range of frequency modulation for oscillator is given in Table 5.9, details of circuit implementation are provided in Appendix C.

<table>
<thead>
<tr>
<th>$F_{osc}$ (Hz)</th>
<th>$T_{est}$ (K)</th>
<th>$\ln(R_{est}/R_0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>11064</td>
<td>238.250</td>
<td>0.6511</td>
</tr>
<tr>
<td>11264</td>
<td>285.950</td>
<td>0.5430</td>
</tr>
</tbody>
</table>

Table 5.8: Sensitivity Data

Figure 5-11: View of the device utilization report.

Figure 5-12: View of the power analysis report.
Table 5.9: Modulation Range

<table>
<thead>
<tr>
<th>$F_{osc}$ (kHz)</th>
<th>Sensor Resistance (kΩ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.4</td>
<td>0</td>
</tr>
<tr>
<td>8.02</td>
<td>70</td>
</tr>
</tbody>
</table>

Figure 5-13: Temperature as a function of the oscillator frequency.

Figure 5-14: Resistance as a function of the oscillator frequency.

5.3 Acquisition Hardware

The condition monitoring approaches introduced in this work require high-quality data acquisition. The switched capacitor filter described in Section 5.1 enables meth-
ods based on current data. Vibration is another quantity used in the previously presented work. A custom acquisition platform was developed to enable high-resolution acquisition of wide-bandwidth vibration signals. All channels dedicated for vibration measurements are compatible with the IEPE standard and were tested with IEPE accelerometers from multiple vendors. Additionally, the IEPE microphones can be used with the developed hardware.

5.3.1 Analog Front-End

The analog front-end for vibration measurements is described in this section. The IEPE-compatible front-end consists of a controllable current source, AC coupling network, non-inverting adjustable amplifier, fully-differential amplifier performing filtering and single-ended-to-differential conversion and fully-differential ADC. A block diagram representing analog front-end is presented in Fig. 5-15.

![Analog front-end block diagram.](image)

5.3.1.1 Current Source and Coupling

Typical IEPE accelerometer has a 2-10 mA current rating, however, sensors with higher ratings are common [110]. Conventionally, the DC current source is generated with a current-regulating diode [81]. However, in order to comply with transducers from different vendors an adjustable DC current source is necessary. Several current source implementations are possible [92], including current mirrors and operational amplifier based configuration. Two implementations are presented in this section. The first one is based on the XTR111 precision voltage-to-current converter with built-in current mirror [111]. The relation between input voltage and output current is set by
the setting resistor $R_{set}$, see Fig. 5-16; thus, by adjusting input voltage it is possible to adjust the output current within the output voltage limitations. The input voltage is programmable with DAC. The XTR111 can be fed with voltage source between 7 to 44 V, thus, it complies with typical voltage for the IEPE standard, i.e., between 24 to 30 V. A diagram presenting device structure is presented in Fig. 5-16, refer to data sheet for details [111]. A magnitude spectrum of voltage drop across a high-precision 1.1 kΩ resistor due to the current generated by the circuit in Fig. 5-16 is given in Fig. 5-17. It reveals additive noise components at multiples of approximately 10 kHz. A detailed inspection of [111] reveals that the input-referred noise spectrum on page six shows potential for additive noise at multiples of approximately 10 kHz. Precision voltage-to-current converters are often optimized for the 4-20 mA current loop applications [80]. Such applications, either sensing or process control [79], are inherently low-bandwidth applications, thus, signal components above 10 kHz are typically acceptable. Unfortunately, acquisition of vibration signals with IEPE accelerometers requires a low-noise DC current source [28]. Current source implementation should not only be low-noise but high-bandwidth as well. High-bandwidth allows a current source to be an actual DC current source with load impedance presenting changes at frequencies in the range of tens of kHz.
Taking into account noise and bandwidth requirements an alternative discrete current source implementation was designed. A circuit diagram of an operational amplifier based current source implementation is given in Fig. 5-18. It consists of three operational amplifiers and allows to create current source in the sourcing configuration. Scaled voltage $V_{ref}$ and $V_n$ are summed and attenuated by an additional factor of two, however, gain of the non-inverting amplifier (amplifier 2) compensates for this attenuation. In order to achieve this compensation, all resistors $R_3$ should be high-precision and high-thermal stability resistors, ideally in a form of resistors array. For matched resistors $R_3$, voltage at node $V_m$ equals to the sum of voltage $V_n$ and scaled voltage $V_{ref}$. Thus, voltage across resistor $R_{ref}$ equals to $V_{ref} \frac{R_2}{R_2+R_1}$. Consequently, the relation between $V_{ref}$ and $I_{ref}$ is given in (5.20). Resistor $R_{ref}$ should be a high-precision and high-thermal stability resistor.

$$I_{ref} = \frac{V_{ref} R_2}{R_{ref} (R_2 + R_1)} \quad (5.20)$$

Amplifiers 2 and 3 were designed as LM6172. This operational amplifier presents a high gain-bandwidth product of 100 MHz. With low-gain circuits it presents wide bandwidth and allows to maintain DC current despite variable load impedance. Additionally, LM6172 has high output current driving capability of 50 mA and maximum
output voltage just 600 mV below the power rail. Power supply voltage and current rating of the IEPE standard are well within the ratings of LM6172. Dual package allows to optimize layout and minimize space by implementing both operational amplifiers 2 and 3 in one integrated circuit. A common-mode input voltage of LM6172 is just 1.5 V below supply voltage, thus, operational amplifier 3 can operate with high input voltage $V_n$. Resistor $R_{ref}$ helps with compensation of operational amplifier 3 in case of capacitive loading, however, higher values of $R_{set}$ limit the possible voltage range at node $V_n$. Design details are given in Table 5.10. Resistors matching impedance between amplifiers inputs are not marked in the circuit diagram.

Table 5.10: Current Source Circuit Parameters

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>2 kΩ</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$R_3$</td>
<td>5 kΩ</td>
</tr>
<tr>
<td>$R_{ref}$</td>
<td>249 Ω</td>
</tr>
<tr>
<td>$OP_1$</td>
<td>TLV316IDCKT</td>
</tr>
<tr>
<td>$OP_2$</td>
<td>LM6172</td>
</tr>
<tr>
<td>$OP_3$</td>
<td>LM6172</td>
</tr>
</tbody>
</table>
As an example of the benefits offered by the custom instrumentation, a comparison with the IEPE acquisition platform built according to the guidelines in [112] is shown in Figs. 5-19 and 5-20. It should be noted that the reference design recommends the XTR111 as IEPE current source. Comparisons presented in Figs. 5-19 and 5-20 reveal that current source implementation of the reference design introduces significant high-frequency noise into the measurements. The noise contribution is large enough to have a visible effect on the vibration and acoustic signals in the time domain representation. The IEPE microphone was used in front of speaker excited with a single tone. Additionally, IEPE accelerometer was mounted on the speaker excited with a single tone. This comparison clearly highlights need for a dedicated IEPE current source implementation. Many of the existing voltage-to-current converters are optimized for low-bandwidth applications and may not be appropriate for high-fidelity vibration measurements.

![Image](image.png)

**Figure 5-19:** Sample acoustic signal for current source performance comparison.

A diagram presenting current source and coupling network is presented in Fig. 5-21. The coupling capacitor $C_{ac}$ and resistors $R_{ac1/2}$ remove the high voltage DC component $V_{dc}$ and bias the scaled AC component $V_{ac}$ around voltage $V_{ref}$. This allows to process a single-ended signal with amplifiers powered with unipolar power supplies. Otherwise, a bipolar power supply would be necessary for the amplifiers in the analog front-end. Application of $C_{ac}$ not only removes the high voltage DC component

158
Figure 5-20: Sample vibration signal for current source performance comparison.

but also limits the bandwidth, i.e., there is a non-zero corner frequency associated with the coupling network $C_{ac}$ and $R_{ac1/2}$. The corner frequency $f_c$ associated with this network is 1.26 Hz as expressed in (5.21). For some applications, with low-frequency signals, a lower corner frequency may be desirable. The thermal stability offered by C0G (NP0) ceramics capacitors is desirable, however, capacitance and voltage requirements (needs to sustain up to 30 V) do not allow for such a design choice. The scaling $G$ introduced by resistors $R_{ac}$ is set according to (5.22), assuming operation in pass-band and negligible impedance presented by the capacitor. Design

Figure 5-21: AC coupling network.
details are given in Table 5.11.

\[ f_c = \frac{1}{2\pi (R_{ac1} + R_{ac2})C_{ac}} = 1.26 \text{ Hz} \]  
(5.21)

\[ G = \frac{R_{ac2}}{R_{ac1} + R_{ac2}} = 0.0476 \]  
(5.22)

Table 5.11: Coupling Network Circuit Parameters

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_{ac1} )</td>
<td>120 kΩ</td>
</tr>
<tr>
<td>( R_{ac2} )</td>
<td>6 kΩ</td>
</tr>
<tr>
<td>( C_{ac} )</td>
<td>1 μF</td>
</tr>
</tbody>
</table>

5.3.1.2 Single-Ended Amplifier

Scaled and biased voltage is amplified by a non-inverting amplifier. A circuit diagram is presented in Fig. 5-22 and circuit gain expression is given in (5.23). Developed hardware has two gain settings, gain of two and six. It is further explained in the following sections. Schottky diodes at the input line limit the impact of potential overvoltages. The absolute maximum input voltage at the input pin of operational amplifier is 0.5 V above the supply voltage [113], e.g., it corresponds to 3.5 V with 3 V power supply voltage. The forward voltage drop across diode depends on the severity of overvoltage and corresponding current through the diode, however, limiting resistor \( R_3 \) limits the current and corresponding forward voltage drop. Diode \( D_1 \) has 0.4 V forward voltage drop at 10 mA current. This would correspond to a 13.4 V overvoltage at the input node and 3.4 V at the diode connection node assuming 1 kΩ limiting resistor \( R_3 \) and \( V_{cc} \) of 3 V; \( V_{ss} \) is tied to a common ground. Thus, maximum 3.4 V input is within 3.5 V of amplifier absolute maximum input voltage. Table 5.12 summarizes the designed circuit.

\[ V_o = V_{ref} + V_i \left( 1 + \frac{R_2}{R_1} \right) \]  
(5.23)

Noise analysis for the circuit in Fig. 5-22 involves multiple noise sources. An
equivalent circuit diagram for the purpose of noise analysis is presented in Fig. 5-23 [114, 115]. Input current noise spectral densities, i.e., a square root of the current noise power spectral densities [116], $i_p$, $i_n$ and input voltage noise spectral density $v_p$ of the TLV316IDCKT amplifier are provided in [113]. Thermal Johnson noise of the feedback network is included in the analyzed model [117], sources $v_{n1-n3}$ represent voltage noise spectral densities corresponding to thermal agitation of electrons in $R_{1-3}$. Technically $i_n$, $i_p$, $v_1$, $v_{n2}$, $v_3$ and $v_p$ are square root of power spectral densities, i.e., have to be squared and integrated over given frequency band to provide a mean-square current or voltage noise in the band of interest. Based on the assumption of independence of random processes modeling noise sources, the total output voltage noise spectral density at the output of the non-inverting amplifier $v_{no}^{na}$ is expressed in (5.24). Contributions to the total output voltage noise spectral density $v_{no}^{na}$ from each source $v_{1-4}$, $v_p$ and $i_{p-n}$ are summarized in Table 5.13.
\[ v_{na}^n(\omega) = \sqrt{\sum_{x \in \{1,2,3,p\}} |T_{vx}(\omega)|^2 v_x^2 + \sum_{x \in \{p,n\}} |Z_{ix}(\omega)|^2 |T_{ix}(\omega)|^2 i_x^2} \]  

(5.24)

where \( T_{vx}(\omega) \) is a gain from the voltage noise source \( x \) to output, \( Z_{ix}(\omega) \) is a Thevenin equivalent impedance at the current noise source \( x \) node and \( T_{ix}(\omega) \) is a gain from the current noise source (converted to voltage) \( x \) to output.

Table 5.13: Non-Inverting Amplifier Noise Gains

<table>
<thead>
<tr>
<th>Gain Expression</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{v1}(\omega) )</td>
<td>( \frac{R_2}{R_1} )</td>
</tr>
<tr>
<td>( T_{v2}(\omega) )</td>
<td>1</td>
</tr>
<tr>
<td>( T_{v3}(\omega) )</td>
<td>( 1 + \frac{R_2}{R_1} )</td>
</tr>
<tr>
<td>( T_{ip}(\omega) )</td>
<td>( 1 + \frac{R_3}{R_1} )</td>
</tr>
</tbody>
</table>

The input referred amplifier voltage and current noise spectral densities \( v_p \) and \( i_{p/n} \) can be extended to model the low-frequency 1/f noise, also known as the flicker noise [114]. A corner frequency \( f_{cv} \) characterizing the voltage noise spectral density was determined graphically using the approach described in [114]. Based on data in [113],
$f_{cv} = 250 \ Hz$. Data sheet of TLV316IDCKT [113] does not provide noise spectral density plot for current noise; thus, corner frequency $f_{ci}$ cannot be determined and $\frac{1}{f}$ current noise component remains undefined. However, TLV316IDCKT is a CMOS amplifier with a very low input current noise specification of $1.3 \ \text{fA} \sqrt{\text{Hz}}$, so it is of lower importance in the noise analysis. The expanded input voltage noise power spectral density for the analyzed amplifier is described in (5.25) [114].

$$v_p^2(\omega) = v_p^2(1 + \frac{2\pi f_{cv}}{\omega}) \quad (5.25)$$

The output voltage noise spectral density according to (5.24) (including extension in (5.25)) is presented in Fig. 5-24, it is compared with the output voltage noise spectral density from analysis performed with the SPICE simulator. SPICE simulation includes amplifier and resistors noise sources. Both simulation and analytical model assume temperature of $25 \ ^\circ\text{C}$. It can be seen that the analytical model matches well the simulation results in a wide frequency range. Thus, it can be used for the analysis and optimization of noise-related circuit performance. The analytical model provides insight into the impact of particular noise source or circuit component on output voltage noise spectral density.

![Figure 5-24: Output voltage noise spectral density for the analytical model and SPICE simulation of the non-inverting amplifier.](image-url)
5.3.1.3 Single-Ended to Differential Converter and Filter

High-end ADCs are often implemented in a differential architecture. The input signal from IEPE vibration or acoustic transducer is inherently single-ended, thus, a conversion is necessary if differential ADC is to be used. An anti-aliasing filter is necessary before sampling of continuous-time signal. Conversion from the single-ended to differential signal and filtering can be combined in one stage. Additional requirement for the circuit is to provide a low-impedance path for the sampling circuit of the ADC [118]. A circuit diagram of the designed hardware is presented in Fig. 5-25.

![Circuit diagram of the fully-differential amplifier.](image)

Figure 5-25: Circuit diagram of the fully-differential amplifier.

The designed circuit implements a second-order low-pass filter [92]. The filter is implemented in a multiple-feedback architecture due to the low sensitivity to circuit parameters variation. A transfer function describing the input to output relationship in the Laplace domain is given in (5.26). SPICE simulation was performed with THS4551 fully-differential amplifier model and magnitude frequency response is compared with magnitude of the transfer function in (5.26). Comparison is presented in Fig. 5-26. A corner frequency of 36.87 kHz is marked in the plot, the designed circuit
presents a gain of two in the pass-band.

\[
\frac{V_{do}(s)}{V_{di}(s)} = \frac{\frac{R_2}{R_1}}{s^22C_1C_2R_2R_3 + s(R_3C_1 + R_2C_1 + \frac{R_2}{R_1}R_3C_1) + 1}
\]  

(5.26)

Figure 5-26: Frequency response of the fully-differential amplifier.

Additional components were added to the circuit as shown in Fig. 5-27. Capacitor \( C_3 \) was added to lower the source impedance in front of the sample-hold circuit of the ADC and provide charge buffer to minimize voltage distortion during the sampling action. Resistors \( R_4 \) and capacitor \( C_3 \) create additional pole in the frequency response of the filter, thus, response of the extended circuit is of third order. Resistors \( R_5 \) are compensation resistors at the output of the amplifier to improve circuit performance with capacitive loads. Capacitor \( C_4 \) was added to improve the loop phase-margin with minimal interaction with the active filter operation, as recommended in [119]. The final magnitude frequency response from SPICE simulation is presented in Fig. 5-28. The corner frequency of the extended filter circuit equals to 26.65 kHz and is labeled in Fig. 5-28. The gain in the pass-band is still two. The design details of the circuit in Fig. 5-27 are given in Table 5.14. In the designed circuit fully-differential amplifier is fed from 3 V power supply voltage, the same used to power the non-inverting amplifier at the input, see Fig. 5-15. Input of the non-inverting stage is protected from overvoltages as presented in Fig. 5-22. Thus, there is no need for
Figure 5-27: Circuit diagram of the extended fully-differential amplifier.

Table 5.14: Fully-Differential Amplifier Circuit Parameters

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>1 kΩ</td>
</tr>
<tr>
<td>$R_2$</td>
<td>2 kΩ</td>
</tr>
<tr>
<td>$R_3$</td>
<td>1 kΩ</td>
</tr>
<tr>
<td>$R_4$</td>
<td>10 Ω</td>
</tr>
<tr>
<td>$R_5$</td>
<td>15 Ω</td>
</tr>
<tr>
<td>$C_1$</td>
<td>2.2 nF</td>
</tr>
<tr>
<td>$C_2$</td>
<td>4.7 nF</td>
</tr>
<tr>
<td>$C_3$</td>
<td>22 nF</td>
</tr>
<tr>
<td>$C_4$</td>
<td>1 nF</td>
</tr>
</tbody>
</table>

Amplifier THS4551

Additional protection at the input pins of the differential stage.

Noise analysis for the circuit in Fig. 5-27 involves contribution from multiple noise sources. An equivalent circuit diagram for the purpose of noise analysis is presented in Fig. 5-29 [114, 115]. The input current noise spectral densities, i.e, a square root of the current noise power spectral densities [116], $i_p$, $i_n$ and input voltage noise spectral density $v_p$ of the THS4551 amplifier are provided in [119]. Thermal Johnson noise of the feedback network is included in the analyzed model [117], sources $v_{1-4}$ represent voltage noise spectral densities corresponding to thermal agitation of electrons in $R_{1-4}$. Noise from the resistive part of capacitors is neglected [120]. Technically $i_n$, $i_n$,
Figure 5-28: Frequency response of the fully-differential amplifier with additional pole.

\( v_1, v_2, v_3, v_4 \) and \( v_p \) are square root of power spectral densities, i.e., have to be squared and integrated over given frequency band to provide a mean-square current or voltage noise in the band of interest. In order to simplify the analysis, a Thevenin equivalent of the input network \( 2R_1 - C_2 \) is considered with two single-ended input sources [115]. This simplified circuit is presented in Fig. 5-30 with \( Z_1 = \frac{1}{2}(2R_1||\frac{1}{sC_2}) \). Based on the assumption of independence of random processes modeling noise sources, the total output voltage noise spectral density at the output pins of the fully-differential amplifier \( v^\text{da}_{\text{no}} \) is expressed in (5.27). Contributions to the total output voltage noise spectral density \( v^\text{da}_{\text{no}} \) from each source \( v_{1-4}, v_p \) and \( i_{n-p} \) are summarized in Table 5.15.

Figure 5-29: Circuit diagram for noise analysis of the fully-differential amplifier.
\[ v_{na}(\omega) = \sqrt{\sum_{x \in \{1,2,3,p\}} |T_{vx}(\omega)|^2 v_x^2 + \sum_{x \in \{p,n\}} |Z_{ix}(\omega)|^2 |T_{io}|^2 i_x^2}, \quad (5.27) \]

where \( T_{vx}(\omega) \) is a gain from the voltage noise source \( x \) to output and \( Z_{ix}(\omega) \) is a Thevenin equivalent impedance at the the current noise source \( x \) node, it is referred to output by gain \( T_{io} = \frac{1}{\beta} \), \( \beta \) is defined in (5.28).

**Table 5.15: Fully-Differential Amplifier Noise Gains**

<table>
<thead>
<tr>
<th>Gain</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{v1}(\omega) )</td>
<td>( \frac{R_2</td>
</tr>
<tr>
<td>( T_{v2}(\omega) )</td>
<td>( \frac{1}{\beta} )</td>
</tr>
<tr>
<td>( T_{v3}(\omega) )</td>
<td>( \frac{1}{\beta} \frac{1}{sC_1 + R_3 + R_2 \left( \frac{1}{2} \frac{2R_1}{sC_2} \right)} )</td>
</tr>
<tr>
<td>( T_{vp}(\omega) )</td>
<td>( \frac{1}{\beta} )</td>
</tr>
<tr>
<td>( Z_{ip}(\omega) )</td>
<td>( \frac{1}{\beta} \frac{1}{sC_1} \left( R_3 + R_2 \right) \left( \frac{1}{2} \frac{2R_1}{sC_2} \right) )</td>
</tr>
<tr>
<td>( Z_{in}(\omega) )</td>
<td>( \frac{1}{Z_{ip}} )</td>
</tr>
</tbody>
</table>

\[ \beta = \frac{\frac{1}{2} \left( \frac{2R_1}{sC_2} \right) + R_2 || (R_3 + \frac{1}{sC_1})}{\frac{1}{2} \left( \frac{2R_1}{sC_2} \right) + R_2 || (R_3 + \frac{1}{sC_1}) + \frac{R_2}{sC_1} || (R_3 + \frac{1}{sC_1}) R_3 + \frac{1}{sC_1}} \quad (5.28) \]

Symbol ‘\(||\)’ in Table 5.15 denotes parallel connection of impedances. The input referred amplifier voltage and current noise spectral densities \( v_p \) and \( i_{p/n} \) can be extended to model the low-frequency 1/f noise. Corner frequencies \( f_{cv} \) and \( f_{ci} \) charac-
terizing the noise spectral densities were determined graphically using the approach described in [114]. Based on data in [119], \( f_{cv} = 200 \, Hz \) and \( f_{ci} = 6 \, kHz \). The expanded input referred noise power spectral densities for analyzed amplifier are described in (5.29)-(5.30) [114].

\[
v_{np}^2(\omega) = v_p^2 \left(1 + \frac{2\pi f_{cv}}{\omega}\right) \tag{5.29}
\]

\[
v_{nm/p}^2(\omega) = i_{p/n}^2 \left(1 + \frac{2\pi f_{ci}}{\omega}\right) \tag{5.30}
\]

The expanded output voltage noise spectral density \( v_{no}^{'d_o} \) at the output of the fully-differential amplifier is referred to output of the \( 2R_4 - C_3 \) network and noise contribution of \( R_4 \) resistors is added as summarized in (5.31).

\[
v_{no}^{'d_o}(\omega) = \sqrt{\left[v_{no}^{'d_o}(\omega)\right]^2 \frac{1}{2R_4j\omega C_3 + 1}^2 + v_4^2}, \tag{5.31}
\]

where \( v_4 = \sqrt{4k_BT R_4} \), \( k_B \) is the Boltzman constant and \( T \) is temperature in the Kelvin scale. The output voltage noise spectral density \( v_{no}^{'d_o} \) according to (5.31) is presented in Fig. 5-31, it is compared with the output voltage noise spectral density from analysis performed with the SPICE simulator. SPICE simulation includes amplifier and resistors noise sources. Both simulation and model assume temperature of 25 °C. It can be seen that the analytical model matches well the simulation results in a wide frequency range and can serve as a design and optimization guide.

### 5.3.1.4 Analog Inputs

The analog circuitry was designed and optimized to perform high-quality measurements of vibration and acoustic signals. However, developed hardware provides an alternative input path for generic voltage-input measurements. There are four channels that can be independently configured to read input from either IEPE transducer or generic voltage input. Selection is performed with a mechanical switch on the top side of upper PCB, see Fig. 5-32. Analog input designation on the silkscreen denotes the voltage input.
Figure 5-31: Output voltage noise spectral density for the analytical model and SPICE simulation of the fully-differential amplifier.

Figure 5-32: Input type selection switches.

Input signal at the analog input is scaled and protection from potential overvoltages is provided by Schottky diodes, see Fig. 5-33. In the developed hardware $R_1 = 120 \, k\Omega$, $R_2 = 0 \, \Omega$ and $R_3 = 80.6 \, k\Omega$. In this configuration, $5 \, V_p$ input signal is scaled down to $2.009 \, V_p$. This is within the common mode input range of the operational amplifiers in voltage follower configuration, i.e., TLV316IDCKT, which is $\pm 2.7 \, V$ with $\pm 2.5 \, V$ power supply. Reference voltage of the ADC is $2.048 \, V$, thus, maximum range of input signal at the analog inputs is limited to $\pm 5 \, V$. Output of the voltage followers is filtered by a differential low-pass filter with cut-off frequency of $23.45 \, kHz$. Voltage follower configured amplifiers provide low source impedance for the sampling circuit. Additionally, loading of the input sensing circuit is limited to the current via scaling network $R_1-R_2-R_3$. Scaling network is built with $0.1 \%$ tolerant resistors. Capacitor $C_1$ provides charge buffer for the sample-and-hold circuit of
the ADC. In the configuration described above $R_2 = 0 \, \Omega$ and circuit is configured for a single-ended input signal, $V_{i-}$ should connected to a common ground by the user. This is very typical configuration to measure signals at the output of voltage sensors, current sensors or pressure transducers. However, replacing $R_1$ and $R_2$ to create a symmetrical network allows to perform measurements for differential input signals. Clamping diodes are connected to the same $\pm 2.5 \, \text{V}$ rails. The absolute maximum input voltage at the input pin of operational amplifier is $0.5 \, \text{V}$ above the positive supply voltage and $0.5 \, \text{V}$ below the negative supply voltage. The forward voltage drop across diode depends on the severity of overvoltage and corresponding current through the diode, however, resistors $R_1$ and $R_2$ limit the current and corresponding forward voltage drop. Diodes $D_1$ and $D_2$ have $0.4 \, \text{V}$ forward voltage drop at $10 \, \text{mA}$ current. Resistor $R_2 = 0 \, \Omega$, so it doesn’t limit current through diode $D_2$. However, in this configuration, for single-ended measurements, $V_{i-}$ is connected to a common ground. Table 5.16 summarizes the designed circuit.

![Analog front-end for the voltage inputs.](image)

**Figure 5-33: Analog front-end for the voltage inputs.**

### 5.3.1.5 Overall Transmission Path

This section presents experimental results for a test input signal. The input signal is presented in Fig. 5-34, a $3 \, \text{V}_p$ $100 \, \text{Hz}$ sinusoidal input signal. The input signal
Table 5.16: Analog Input Circuit Parameters

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>120 kΩ</td>
</tr>
<tr>
<td>$R_2$</td>
<td>0 Ω</td>
</tr>
<tr>
<td>$R_3$</td>
<td>80.6 kΩ</td>
</tr>
<tr>
<td>$R_4$</td>
<td>4.99 Ω</td>
</tr>
<tr>
<td>$C_1$</td>
<td>0.68 µF</td>
</tr>
<tr>
<td>$D_1$</td>
<td>BAT54S-E3-08</td>
</tr>
<tr>
<td>$D_2$</td>
<td>BAT54S-E3-08</td>
</tr>
<tr>
<td>Amplifier</td>
<td>TLV316IDCKT</td>
</tr>
</tbody>
</table>

is applied as indicated in Fig. 5-21. The effects of AC coupling, scaling and biasing with $V_{ref}$ are presented in Fig. 5-35. An offset of $V_{ref} = 1.5$ V can be seen and 3 V amplitude is scaled to 0.429 V, it should be noted that the results presented below were obtained with $G=0.143$, compare with gain defined in (5.22). Scaled and biased signal is amplified by the non-inverting amplifier according to (5.23). Voltage at the amplifier output is presented in Fig. 5-36. It should be noted that gain of two was selected when performing the measurement. Thus, the AC component has amplitude of 0.858 V.

The output of the non-inverting amplifier presents a single-ended signal biased
around $V_{ref}$ and is converted to a differential signal by the fully-differential amplifier circuit. Balanced output of the fully-differential amplifier is presented in Fig. 5-37. A differential signal, being a difference of signals in Fig. 5-37, is presented in Fig. 5-38, a gain factor of two with respect to the signal in Fig. 5-36 is evident. High precision resistors are indispensable to achieve good matching of both feedback paths in the fully-differential amplifier circuit. Symmetry in both paths is desired from the noise.
Figure 5-37: Signals at the outputs of the fully-differential amplifier.

Figure 5-38: Differential signal at the output of the fully-differential amplifier.

perspective and common-mode voltage rejection. The output noise level is strongly affected by the input noise at the common-mode input pin in case of asymmetric feedback paths [121]. Additionally, common-mode voltage rejection depends on the resistor matching error, e.g., a 0.1% error can result in 60 dB of CMRR [122].

The reference voltage of the ADC is 2.048 V. Thus, maximum value of input signal scaled and biased according to previously presented rules should not exceed this value
at the input pins of the ADC. A chain of voltage levels is discussed for 10 V_p input signal. It could correspond to a mechanical input of 100 g for accelerometer with sensitivity of 100 mV/g [123], assuming this peak acceleration is within the safe operating range. The input signal is scaled according to (5.22), providing signal of 0.476 V_p, and biased around $V_{ref}$ of 1.5 V. This scaled and biased signal is amplified by the non-inverting amplifier according to (5.23). As explained, input to the non-inverting amplifier varies between 1.024 V to 1.976 V. For TLV316IDCKT amplifier supplied with 3 V, a common-mode input voltage range corresponds to (-0.2)-(+3.2) V. Thus, with maximum input signal of 10 V_p, common-mode input voltage constraints of the non-inverting amplifier are not violated. The output signal of the non-inverting amplifier has 0.952 V amplitude and the same offset of 1.5 V, a gain factor of two is assumed in the non-inverting stage. Finally, the fully-differential circuit converts signal to a differential form and provides gain of two in the pass-band below 26.65 kHz, i.e., differential output signal has amplitude of 1.904 V. SPICE simulation of the fully-differential amplifier circuit revealed that with 10 V_p excitation the common-mode input voltage at amplifier pins varies between 1.19 V to 1.82 V. A common-mode input voltage range of THS4551 at 25 °C and supplied with 3 V corresponds to (-0.2)-(+1.9) V [119]. Thus, with maximum input signal of 10 V_p, common-mode input voltage constraints of the fully-differential amplifier are not violated. The differential signal at the input of the ADC has amplitude of 1.904 V and can be correctly digitized with the designed reference voltage of 2.048 V. Such selection of gains was designed taking into account maximum swing of the AC component at the input of the IEPE interface. An interface with 24 V power supply can provide at most 12 V of symmetrical swing around offset of 12 V. Implemented current source is fed with 24 V, however, due to limitations of operational amplifiers it cannot provide full voltage at the output, i.e., at the connection point of IEPE transducer. Thus, the peak value of AC component in the above consideration was limited to 10 V. For input signals with smaller peak values and low dynamic range, the risk of clipping and distortion is low and gain of the non-inverting amplifier may be increased from two to six. Gain is selected with a jumper on the top side of upper PCB, see Fig. 5-39. The selection marked in Fig.
5-39 corresponds to the lower gain setting.

Figure 5-39: Gain selection for the IEPE channel.

Noise analysis performed for the non-inverting amplifier and fully-differential amplifier should be combined to provide the total output voltage noise spectral density or equivalent root mean-square voltage at sampling point of the ADC. The voltage noise spectral density at the output of the non-inverting stage $v_{no}^{na}$, see (5.24)-(5.25), is an input to the fully-differential amplifier. Thus, the total output voltage noise spectral density for the fully-differential amplifier $v_{no}^{da}$, see formula (5.31), can be expanded with input from the non-inverting stage. The final noise spectral density is denoted as $v_{no}^{da'}$ and expressed in (5.32).

$$v_{no}^{da'}(\omega) = \sqrt{(v_{no}^{da})^2 + (v_{no}^{na})^2 |T_1(\omega)|^2 \frac{1}{2R_4j\omega C_3 + 1}^2}, \quad (5.32)$$

where $T_1(\omega) = \frac{R_2}{R_1} \frac{1}{s^2C_1C_2R_2R_3 + s(R_3C_1 + R_2C_1 + \frac{R_2R_3}{R_1}C_1) + 1}$ is a transmission path from $v_{no}^{na}$ to output of the fully-differential amplifier. Circuit parameters in (5.32) and in $T_1$ refer to the fully-differential amplifier circuit. A comparison of the analytical model and SPICE simulation results for the output voltage noise analysis is shown in Fig. 5-40. It can be seen that the analytical model matches well the SPICE simulation results in high-frequency range, low-frequency voltage noise spectral density obtained with the analytical model is higher than the SPICE simulation results. It should be highlighted that noise contribution from the $V_{ref}$ potential was neglected in the SPICE simulation and analytical model by assuming perfectly matched feedback paths in the circuit of fully-differential amplifier [122].

176
In order to evaluate SNR of the overall circuit, the output voltage noise spectral density \( v_{\text{no}}^{\text{da}} \) expression in (5.32) is squared and integrated in the band of interest. This provides a mean square voltage noise, RMS value of the output voltage noise is obtained according to (5.33).

\[
v_{\text{rms}}^{\text{da}} = \sqrt{\int_{F_{\text{min}}}^{F_{\text{max}}}[v_{\text{no}}^{\text{da}}(2\pi f)]^2 df},
\]

(5.33)

where \( F_{\text{min}} = 1 \) Hz and \( F_{\text{max}} = 64 \) kHz. Numerical integration results in RMS value of the output voltage noise \( v_{\text{rms}}^{\text{da}} = 24.3 \) μV. Assuming Gaussian distribution of the random process modeling output voltage noise and zero mean, standard deviation \( \sigma \) equals to RMS value and the peak value of the output voltage noise can be calculated according to 3σ rule [124]. Thus, 99.7% of the instantaneous noise values are assumed to be within \( \pm 3\sigma = \pm 72.9 \) μV. With differential output signal \( V_{\text{do}} \), see Fig. 5-27, equal to the reference voltage of the ADC, i.e., \( V_{\text{do}} \) equals to 2.048 Vp, the SNR of the circuit is calculated according to (5.34).

\[
SNR = 20\log\left(\frac{V_{\text{do}}}{3 \times v_{\text{rms}}^{\text{da}}}\right) = 88.97 \text{ dB}
\]

(5.34)
The SINAD figure of the fully-differential amplifier circuit is calculated according to (5.35).

\[ SINAD = 10 \log \left( \frac{P_s + P_{THD} + P_N}{P_{THD} + P_N} \right) \]  

(5.35)

where \( P_s \) is signal power, \( P_{THD} \) is power of the signal harmonics and \( P_N \) is power of the noise component. The SINAD of the ADC, described in the further sections, is evaluated with input signal having peak value of 94.4% of full-scale voltage [125], i.e., 1.93 \( V_p \). Thus, this signal level is considered in the SINAD calculations. \( P_{THD} \) is calculated following amplifier’s data sheet [119]. Data sheet specifies only second harmonic \( H_2 \) and third harmonic \( H_3 \) properties, thus, calculations are limited to those components. According to [119], for differential output voltages of 4 \( V_{pp} \) and signal frequency equal to 100 kHz \( H_2 \) level and \( H_3 \) level are approximately -127 dBc -125 dBc, respectively. Thus, \( P_{THD} \) power of the signal harmonics is expressed in (5.36).

\[ P_{THD} = \left(10^{-127/20} \times 1.93\right)^2 + \left(10^{-125/20} \times 1.93\right)^2 \]  

(5.36)

The noise power \( P_N \) is evaluated as a square of the RMS value of the output voltage noise \( v_{rms}^{da} \). The final value of SINAD according to (5.35) is 95 dB.

### 5.3.2 Analog-to-Digital Converter

A low-noise differential signal is provided at the output of the designed analog front-end. The ultimate component in the path is the ADC which performs signal digitization. The noise shaping property of a single-bit \( \Delta \Sigma \) modulator is beneficial for high precision measurements [26]. Digital low-pass filter and decimation filter at the output of the modulator allow to achieve sub-MHz data rate with actual sampling in the MHz range. This simplifies an anti-aliasing filter design and increases oversampling rate for the signal of interest [126]. Modern \( \Delta \Sigma \) ADCs can achieve effective resolutions above 20 bits with data rates above 100 kHz. Low-noise \( \Delta \Sigma \) converters can achieve effective number of bits above 25 with data rate close to 40 kHz [127]. Those features make modern \( \Delta \Sigma \) converters great choice for critical applications requiring
Converters built around $\Delta\Sigma$ modulators are not the best choice for multiplexed inputs, however, high-fidelity acquisition often requires simultaneously-sampled data. Thus, a $\Delta\Sigma$ ADC with simultaneously-sampled channels was chosen for the designed hardware.

Several factors were taken into account in the design process. The effective number of bits, data rate, simultaneous sampling of multiple channels, digital interface and scalability were key factors driving the design decisions. A minimum number of three simultaneously-sampled channels allows to perform vibration measurements in three axis or in three different locations. Additional input allows to perform acoustic signal measurements, thus, four-channel ADC was chosen. A 10- and 12-bit resolution is an industry standard, requirement for the designed hardware was to perform signal digitization with at least 16 bits at data rate above 60 kHz. Additionally, extending number of channels by daisy chaining several devices is of great interest and extends future applications. Taking into account the aforementioned arguments, ADS131A04 ADC from Texas Instruments was chosen for the design. It is available in two and four-channel versions [125]. Two versions of hardware were developed. The major platform has four channels [22], however, hardware described in [106] has two ADS131A04 daisy chained together to provide effectively eight simultaneously-sampled channels. Selected ADC allows to achieve data rates up to 128 kHz with programmable resolution of 16 or 24 bits. This is a valuable feature allowing to trade data size for resolution. Digital communication with ADS131A04 uses SPI bus between converter and microcontroller. Chosen converter can operate in several modes of digital interface like asynchronous mode, synchronous slave and master modes. In a four-channel platform converter operates in asynchronous mode, driving external interrupt to the microcontroller with a programmed data rate. From the electrical configuration point of view, ADS131A04 operates with bipolar power supply $\pm 2.5$ V and disabled negative charge pump.

There are two input sources for the channels of ADS131A04. One from the IEPE front-end and one from generic analog input. In the first case, the fully-differential
amplifier is fed from 3 V unipolar power supply. According to [119], maximum output voltage of fully-differential amplifier corresponds to 2.8 V. In the second case, operational amplifiers providing input to the ADC, see Fig. 5-33, are fed from ±2.5 V bipolar power supply. According to [113], maximum output voltage of operational amplifier corresponds to ±2.375 V. Thus, in both configurations maximum input voltage is within the absolute maximum input range of ADS131A04, which corresponds to ±2.8 V with a bipolar ±2.5V power supply [125].

The input capacitance of the specified converter is 3.5 pF. In order to facilitate the charging process of the sampling capacitor of the converter, an external capacitor \( C_{ext} \) is added to lower the source impedance. The external capacitor buffers charge and minimizes voltage distortion during charge sharing process between sampling and external capacitors [118]. In both input configurations, i.e., IEPE front-end and analog inputs, amplifiers lower the effective source impedance in front of the external capacitor.

The external capacitor is sized based on the size of sampling capacitor and the number of bits. Based on the circuit in Fig. 5-41 and charge balance equation, voltage after closing the sampling switch S is described in (5.37).

\[
V_f = \frac{C_s V_s + C_{ext} V_i}{C_s + C_{ext}},
\]

where \( C_{ext} \) is the external capacitor, \( C_s \) is the sampling capacitor, \( V_f \) is the voltage

![Figure 5-41: Sampling capacitor buffer.](image-url)
after closing switch S, \( V_i \) and \( V_s \) are initial voltages across \( C_{ext} \) and \( C_s \), respectively. Typically, charge sharing process should not distort the voltage \( V_f \) by more than half of the least significant bit from the initial voltage \( V_i \), see (5.38). Assuming zero initial voltage across \( C_s \) and N number of bits, the requirement for size of \( C_{ext} \) can be written in (5.39).

\[
V_i \left(1 - \frac{1}{2^N}\right) = \frac{C_{ext}V_i}{C_s + C_{ext}} \tag{5.38}
\]

\[
C_{ext} \geq (2^{N+1} - 1)C_s \tag{5.39}
\]

Thus, for 16 bits of resolution external capacitor \( C_{ext} \) should be 131071 times bigger than \( C_s \). Table 5.17 defines external capacitance for both input sources, i.e., IEPE front-end and analog inputs. External Capacitor \( C_{ext} \) for analog inputs satisfies (5.39),

<table>
<thead>
<tr>
<th>Input</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEPE Front-End</td>
<td>22 nF</td>
</tr>
<tr>
<td>Analog Inputs</td>
<td>680 nF</td>
</tr>
</tbody>
</table>

i.e., 0.68 \( \mu \)F > 0.46 \( \mu \)F. In case of IEPE front-end, condition (5.39) is slightly relaxed to achieve higher bandwidth of the signal being acquired. Even though sizing of the \( C_{ext} \) capacitor is a function of the sampling circuitry [118], not filter bandwidth, it contributes to the effective bandwidth of the anti-aliasing filter. In case of IEPE front-end, it provides third pole to the frequency response of the anti-aliasing filter, see Fig. 5-28.

Another critical requirement for designing the input of the ADC is the effective source resistance \( R'_s = 2R_s \) [118], where \( R_s \) is defined in Fig. 5-41. In other words, \( C_{ext} \) is sized to provide charge buffer, minimize voltage deviations as described in (5.38) and replenish missing charge within the conversion cycle \( T_{ADC} \). Dynamics of the process is defined by \( C_{ext} \) and effective source resistance \( R'_s \). Time \( T_1 \) necessary for the circuit to restore voltage across \( C_{ext} \) within half of the least significant bit.
satisfies (5.40). Thus, \( T_1 = \ln(2^{N+1}) R_s C_{ext} \).

\[
V_i (1 - e^{-\frac{T_1}{R_s C_{ext}}}) = V_i (1 - \frac{1}{2^{2^N}})
\tag{5.40}
\]

For a 16 bit converter with conversion cycle \( T_{ADC} \), \( T_{ADC} \geq 11.78 R_s C_{ext} \). It can be summarized that for time constant \( \tau = R_s C_{ext} \) the following condition must be met:

\[
T_{ADC} \geq \ln(2^{N+1}) \tau.
\tag{5.41}
\]

It should be highlighted that (5.41) is critical for ADCs with multiplexed channels. If (5.41) is not satisfied, residual charge from previously converted channel, i.e., channel \( i - 1 \), will gradually accumulate over time in \( C_{ext}^i \) on channel \( i \). Such phenomenon is known as a channel cross-talk. In the designed hardware all channels have separate sample-hold circuits; thus, (5.41) is not the major design constraint. However, effective output small-signal resistance for IEPE front-end and analog inputs is provided in Table 5.18. It was estimated with the SPICE simulator.

<table>
<thead>
<tr>
<th>Input</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEPE Front-End</td>
<td>26 Ω</td>
</tr>
<tr>
<td>Analog Inputs</td>
<td>20 Ω</td>
</tr>
</tbody>
</table>

In practice figures of merit like SNR and SINAD of the analog front-end should be properly matched on the interface between amplifier and ADC [126, 124]. The SNR of the analog front-end is given in (5.34). There is no analytical bound on the SNR of the circuit driving input of the ADC, however, practically the SNR should be comparable or higher than the converter’s SNR in order to avoid performance limitations. Another figure of merit that should be considered is SINAD. It is recommended that SINAD of amplifier circuit is not lower than the converter’s SINAD [118]. Table 5.19 summarizes the SNR and SINAD values, values for the converter were read from [125] for operating points close to the circuit operating point. It can be seen that both figures of merit, i.e., SNR and SINAD, are very close in value. Parameters for the ADC were considered
for temperature of 25 °C.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Amplifier</th>
<th>ADC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNR (dB)</td>
<td>88.84</td>
<td>107</td>
</tr>
<tr>
<td>SINAD (dB)</td>
<td>95.1</td>
<td>101</td>
</tr>
</tbody>
</table>

**5.3.3 Wireless Platform**

Analog circuitry described in this chapter is critical to perform high-resolution and high-bandwidth measurements, however, digital interface is necessary to create a useful embedded platform in field applications. Modern advances in connectivity and embedded systems allow to create a platform with remote memory, computational power and with wireless communication link. Wireless networking technologies like Wi-Fi or Bluetooth allow for reliable and secure communication. Wi-Fi networks offer bigger range of communication and higher data exchange rates; thus, Wi-Fi was used as a communication link in the developed hardware platform. Additionally, battery operation is not a primary mode of operation; otherwise, modern Bluetooth technologies, i.e., Bluetooth 4.0 – 5.0: Bluetooth Low Energy, would be considered.

There are several modes of operation for IoT devices utilizing Wi-Fi technology. Two modes of operation that are considered in this thesis include AP and STA modes. AP is a base station in a wireless LAN, potentially providing internet access. while STA is any device capable to use Wi-Fi technology. However, in this thesis STA refers to a wireless client connected to a particular AP, e.g., embedded computer, laptop or phone. Developed hardware has ability to work in both modes. Operation in AP mode is convenient from the user perspective because it eliminates the need for separate AP, e.g., wireless router with built-in AP capability. It is especially convenient in field applications and temporary installations when operator wants to use the device to acquire data for particular machine. In such case, there is no need to carry wireless router or connect to existing infrastructure, if such exists. STA mode of operation requires external AP to connect to, however, this modes of operation allows multiple
devices to connect to the same wireless LAN and central computer with monitor software can access multiple nodes in the system. This allows to effectively increase the number of channels available for measurements.

Apart from communication tasks like running the TCP/IP protocol, embedded system has to perform low-level communication with ADC that is responsible for real-time data acquisition. There are additional tasks related to file management and user interaction. Thus, a separate network co-processor and main microcontroller were specified in the device architecture. The network co-processor is responsible for running the TCP/IP protocol and operating the physical layer of the Wi-Fi standard. Based on the requirements stated above a CC3100 network co-processor from Texas Instruments Simplelink family was chosen [128]. It allows to operate the device in AP and STA modes. Three security protocols are supported, namely: WEP, WPA and WPA2. All application layer protocols have to be run on the main microcontroller, e.g., file transfer protocol. Communication with the main microcontroller is possible over SPI bus or UART. In the developed hardware SPI communication is used.

Main microcontroller has to provide real-time performance and wide range of peripherals to enable all of the functionalities. Communication with network co-processor and real-time data handling were already mentioned. Additionally, external SDRAM memory is used in the device to extend the high-speed memory to buffer acquired data. Finally, data is stored in the µSD card in the FAT file system. The developed hardware has several auxiliary functions that require input and output lines provided by the microcontroller. Lastly, it is desirable for the microcontroller to have support for running real-time operating systems for efficient and scalable firmware implementation. Taking into account the aforementioned requirements, Texas Instruments ARM® Cortex™-M-4 TM4C129ENCPDT microcontroller from Tiva family was chosen [129]. TM4C129ENCPDT provides a wide range of peripherals like parallel memory port for interfacing with the external SDRAM, built-in media access controller and physical layer for Ethernet, multiple SPI buses, floating-point unit and a wide range of flexible general-purpose ports. Additionally, popular real-time operating systems like free-RTOS or TI-RTOS can be run on this microcon-
controller. The built-in hibernation module and real-time clock allow to optimize power consumption for demanding applications and enable potential battery operation. Additionally, real-time clock enables file management with respect to time and date of the particular data set, a valuable feature for condition monitoring applications.

A high-level diagram presenting developed hardware is shown in Fig. 5-42. Mechanically device consists of two PCBs. The bottom PCB is equipped with microcontroller, network co-processor, SDRAM memory, as well as radio-frequency circuitry, while the top PCB is equipped with analog front-end and ADC. Views of two boards stacked together via headers are presented in Fig. 5-43.

![High-level diagram of the embedded platform.](image)

The design with two separate boards serves two purposes. First one, flexible layout of power planes and shielding provided by solid ground plane between boards. Second one, a form factor, device is supposed to link benefits of embedded system, which by definition should be compact and ideally portable, with high-end analog front-end for vibration and acoustic signals acquisition. The final form factor is 10 cm in diameter and 5.1 cm in height, dimensions include designed case which is presented in Figs. 5-44,5-45. Due to the dimensions close to the dimensions of ice hockey puck [130], developed device is often referred to with abbreviated name "puck". Aluminum case is connected to circuit ground and provides electrical shielding and mechanical
Figure 5-43: Views of both PCBs stacked together.

protection. Case is connected to circuit ground by the standoff above header $H3$ and battery connector. Connection can be removed if resistor $R53$ is removed. Metal screws should be used to mount PCB, otherwise case may not be connected to circuit ground. Case has build-in magnets that allow installations on ferromagnetic surfaces, e.g., motor frame or milling machine body. Four SMA connectors on the cover allow to connect IEPE transducers.

Figure 5-44: Custom machined aluminium case.
Four U.FL input connectors of the IEPE front-end are presented in Fig. 5-46, these are connected to the SMA connectors on the case cover. The LEDs marked in red indicate presence of sensor on each channel; when sensor is not detected LED is bright. Fuses $F_1$-$F_4$ are protecting output lines. Input connector allowing connection of analog inputs is shown in Fig. 5-47. Polarity of inputs and rating are marked on the PCB silkscreen. Additionally, this connector provides ground and 3.3 V to the user. Analog inputs are accessible via left opening seen in the case in Fig. 5-45.

![Figure 5-46: IEPE front-end U.FL connectors.](image)

Figure 5-46: IEPE front-end U.FL connectors.
A 2x5.5 mm power-jack connector is labeled in green in Fig. 5-43. Center pin has positive polarity and range of input voltage corresponds to 3-10 V. Input is protected with diode against reversed polarity of the input voltage. An alternative power source can be connected to the 2-pin header marked in red in Fig. 5-43. This header is compatible with a common 2-pin plug on lithium-ion battery packs. Battery should not be connected together with the power-jack source. Although diode protecting the power-jack input will prevent discharging battery if power-jack source voltage is lower than battery voltage, battery is not protected in case power-jack source voltage is higher than battery voltage. Both inputs are fused for safety, however, only the power-jack input is protected from reversed polarity. Power source should be able to provide more than 500 mA at start-up due to the inrush current of filtering capacitors. Inspection hole seen in Fig. 5-44 allows to see the LED indicating presence of power source. The LED can be seen on Fig. 5-45 showing device without case cover. The upper board with analog front-end has additional LED, designated as LED1, to indicate that power is provided from the bottom board. Apart from the main power source, device has a 1025 coin battery holder on the bottom side of the bottom PCB. This battery powers the hibernation module and real-time clock when main power
source is disconnected. Device has three headers that need jumpers in particular positions to operate. Header designated as VCC_3P3 needs to be closed in order to provide power to the device, see header in Fig. 5-48. Header designated as HIB allows to configure some of the hibernation functionalities. If header is closed as marked in Fig. 5-48, i.e., in position two, then hibernation functionality is disabled and on-board power supply is enabled whenever main power source is provided. Use this option if not operating in battery mode. When header HIB is closed in position one, hibernation pin of the main microcontroller controls the enable and disable lines of the on-board power supply. Third header designated as WakeSrc connects input from the WAKE push-button to either tamper input pin or wake pin of the hibernation module. This selection is not important if hibernation functionality is not used.

The designed device is equipped with five push-button, with four being available when both boards are assembled together. Push-buttons RST and WAKE control reset signal and wake signal of the main microcontroller, respectively. Push-buttons nRST1 and nHIB1 control reset signal and hibernation signal of the network co-processor, respectively. Fifth push-button S1 does not serve any particular reason and is left for future use or as a debugging tool. The main microcontroller is programmed with JTAG debug probe over dedicated header, see Fig. 5-49. It is compatible with the Blackhawk USB100v2-ARM JTAG Emulator with 10 pin header. A dot on the silkscreen around JTAG header designates polarity of the debug probe header. Network co-processor’s firmware can be updated using UART lines exposed on header

Figure 5-48: Power configuration headers.
CC3100UART, see Fig. 5-49. Transmitting line, receiving line, ground and power pins are designated on silkscreen, see [131] for details of co-processor programming.

![Figure 5-49: Programming headers.](image)

An external 2.4 GHz antenna can be seen in Fig. 5-44. A U.FL connector for external antenna is located on the bottom PCB and can be seen in Fig. 5-50, marked in red. If the device is operated without metal case a built-in PCB antenna can be used, marked in green in Fig. 5-50. In order to use the external antenna, place resistor R108 in orientation marked in red in Fig. 5-50. For internal antenna, place resistor R108 in the opposite direction. Besides the Wi-Fi communication, developed hardware has USB port that allows for basic device configuration. It is an isolated port that uses FT232RL USB-to-UART converter. Isolation provides safety to the user, while USB-to-UART adapter puts less burden on the microcontroller side. This serial communication channel allows to set time and date of the real-time clock, change the IP address or verify MAC address of the device. The USB port is shown in Fig. 5-51, LEDs next to the USB port indicate communication in both directions.

![Figure 5-50: Wi-Fi antenna selection.](image)
As mentioned previously, device performs file management functions that allow to store files in a remote non-volatile memory, a µSD card. The µSD card that was used during the device development and verification is recommended as a default card choice. It is optimized for memory access time, model: SanDisk Extreme PRO shown in Fig. 5-52. Other card models may be used but performance has to be verified. A slot for the µSD card is located under the USB port. Device is equipped with the external SDRAM memory connected with the main microcontroller via parallel port. It is a volatile memory used to buffer high-sample rate data before writing to the non-volatile memory.

The developed device generates all necessary voltage levels on-board. The main microcontroller, network co-processor and SDRAM memory operate from 3.3 V power supply. The upper board is supplied with 3.3 V through one of the mechanical headers (header H2). Voltages supplying analog front-end are generated locally on the upper board from the 3.3 V rail provided via header. The upper board generates 3 V to supply the fully-differential amplifier and non-inverting amplifier, ±2.5 V to supply
the ADC and analog inputs buffers. The current source implementation on the upper board requires 24 V power source. The required 24 V rail is generated on the bottom board with isolated DC-DC converter fed from 3.3 V rail. The isolated DC-DC power supply U4 is located in one of the openings of the case to enable better cooling. Header H3 provides 24 V to the upper board, where it is filtered and connected to the current source circuitry. A block diagram representing voltage distribution among the loads is presented in Fig. 5-53, there is one ground potential for both boards.

Figure 5-53: Block diagram of the on-board voltage levels.

Connector H9 on the upper board provides auxiliary functions to the device. Fig. 5-54 shows connector H9 and Table 5.20 summarizes auxiliary functionalities staring from the left side of connector H9. Despite the functionality mentioned in Table 5.20, fifth pin from the right side of connector H9 can be used as a digital input signal triggering measurement. The input pin is configured with pull-up resistor to 3.3 V. The input signal should be held steady in the low state for 10 ms to indicate measurement request. Analog output on connector H9 is provided from the on-board DAC.

Two high-current outputs are provided on connector H10, see Fig. 5-55. Two solid-state relays provide normally open contacts, each output is protected with 1 A fuse. No clamping diode is provided for this circuitry. Input voltage should not exceed 48 V_{ac/dc}. Thess high-current outputs may be used in applications where
device has to drive high-power circuitry, e.g., relay coils. Detailed circuit schematics for the wireless platform are provided in Appendix C.

### 5.3.4 Wired Platform

The wireless embedded device described in this chapter is a major hardware platform used in the work presented in this thesis, however, a version with Ethernet connectivity was developed as well. It was designed to allow installations where wireless network is not desirable or prohibited. Additionally, Ethernet connection allows to achieve higher data rates and can provide power over the Ethernet cable. The designed hardware serves as a platform for two devices. One described in [106], served as a new communication and acquisition engine for a power monitor platform in professor Steven Leeb’s research group, see Fig. 5-56, it was developed together with Dr. Kahyun Lee. The second device is described in this section. The main microcontroller in the wireless embedded platform, TM4C129ENCPDT, has built-in media access
controller and physical layer for Ethernet. It conforms with 10BASE-T/100BASE-TX IEEE-802.3 specifications. Thus, the same microcontroller was used in device with Ethernet support. Views of the device are presented in Fig. 5-57.

![Solid-state relays output connector.](image)

Figure 5-55: Solid-state relays output connector.

Layout of the analog and digital circuitry is arranged on two PCBs, similarly to the wireless device, see Fig. 5-42. Two boards are connected mechanically with three headers. The bottom board contains analog circuitry with two IEPE inputs and six analog inputs, only six channels are available in total. IEPE transducers should be connected to BNC connectors BNC1_X1 and BNC1_X2. Analog inputs are exposed with connector H10. Channels one and two on connector H10 are shared with BNC connectors BNC1_X1 and BNC1_X2, only one signal source should be

![Developed hardware in a power monitor.](image)

Figure 5-56: Developed hardware in a power monitor.
used at a time. Current source circuitry can be disconnected from the BNC1\_X1 and BNC1\_X2 connectors when headers H5 are open, see Appendix C for details. The upper board contains micronontroller, SDRAM and remaining digital circuitry. Power can be provided to the device in two ways. First one, via USB port Pwr. This USB port serves only power purpose, there is no communication link provided. Standard USB voltage level is sufficient to power the device. In general, input voltage on this port should not exceed 10 V or fall below 3 V. Second way to power the device is over the Ethernet RJ45 port, the PoE standard is supported. The PoE-related circuitry supports the IEEE 802.3at standard as a 13-W type 1 power device [132]. Both power inputs are protected with fuse. Only one power source can be used at a time. In order to enable the device, jumpers J2, J3, J6 and J8 have to be closed. Using PoE as a power source is convenient from the user perspective, only one Ethernet cable is required to power and communicate with the device. Connector J1 provides circuit ground, 5 V, 3.3 V and 24 V to the user. Besides the Ethernet connectivity, device is equipped with a USB port and FT232RL USB-to-UART converter. This allows to use serial communication link between device and computer. The USB port used for communication is labeled as SERIAL. The SDRAM memory is not located directly on the upper board but has to be connected with header H\_EPI1. The view of the device with connected SDRAM extension board is presented in Fig. 5-58.
card slot is located underneath the SERIAL USB port.

![Device with the external SDRAM adapter board.](image)

**Figure 5-58**: Device with the external SDRAM adapter board.

Connector I2C provides I2C bus to the user for auxiliary purposes. Either 3.3 V or 5 V can be provided to the I2C connector. Populate resistor R25 with 0 Ω to use 3.3 V or populate resistor R26 to use 5 V, only one resistor should be populated at a time. The data and clock lines of the I2C bus are pulled-up to voltage selected by the same resistors. Connector Enc enables to connect signals from a quadrature encoder. This connector is compatible with popular E3 type encoder from US Digital, see [133] for details. It should be noted that mechanical orientation of connector Enc should be flipped with respect to what is shown in Fig. 5-58. Encoder signals, i.e., two channels and index signal, are connected to peripheral of the microcontroller via isolator. Connector Enc provides 5 V and ground potentials to power the encoder circuitry. The main microcontroller can be programmed with the Blackhawk USB100v2-ARM JTAG Emulator debug probe using 10 pin JTAG1 header, the same one as for the wireless device. Detailed circuit schematics are provided in Appendix C.
Chapter 6

Software and Firmware Support

The hardware platforms described in Chapter 5 are built around embedded system merging high-end analog circuitry for acquisition of vibration signals with IoT connectivity. Low-noise circuitry and high-bandwidth amplifiers enable quality data acquisition, however, software and firmware tools are necessary to enable field applications. User interface and network-related functions must be provided jointly by the software and firmware implementations. This chapter is organised as follows. Section 6.1 discusses firmware for wireless platform and Python software run on the user computer. Section 6.2 discusses RTOS-based firmware for wired platform, as well as, related software. Software code listings for both platforms are provided in Appendix B.

6.1 Firmware and Software for the Wireless Platform

The wireless hardware was described in Section 5.3. Block diagram presenting high-level architecture of the device is presented in Fig. 5-42. In order to enable applications of the developed hardware, a firmware for the embedded device and client software need to be implemented. Details of implementation and functionalities are discussed in the following subsections.
6.1.1 Embedded Device Firmware

Several functionalities like communication with the ADC, application layer tasks, file management, wireless network related tasks have to be enabled by firmware of the embedded system. Based on the device architecture, network co-processor offloads Wi-Fi and Internet protocols from the main microcontroller [128]. It handles whole TCP/IP stack allowing the main microcontroller to perform other tasks. Communication between both devices uses SPI bus with 12 MHz clock. Industry-standard BSD socket application programming interface is provided for the main microcontroller to perform socket programming [134]. The network co-processor drives external interrupt line to provide asynchronous indication of network events that have to be handled by the main microcontroller. Besides the SPI bus and interrupt line, the main microcontroller controls hibernation line of the network co-processor. This allows to minimize the power consumption when Wi-Fi connectivity is not required or to disable Wi-Fi related resources for measurement quality.

As indicated in Section 5.3, device can operate in either AP or STA modes. However, independently of mode of operation, device operates as a server, while remote user is a client from the communication point of view. Firmware can be implemented taking advantage of the RTOS and its scheduler. Many RTOS implementations use pre-emptive schedulers, this allows to coordinate different firmware threads based on their priorities. RTOS-based firmware is a good compromise between tight real-time requirements and OS functionalities like scheduling or synchronization allowing to coordinate multiple threads responsible for different parts of the application. Many market solutions like FreeRTOS or TI-RTOS are optimized for real-time applications [135, 136], however, finite time is necessary to run RTOS-related functions. Thus, some bandwidth of the application is lost. An example of this limitation is presented in the following section with code profiling results. The goal of the developed device is to perform state of the art vibration signal acquisition. Thus, the firmware implementation described in this section does not use any RTOS, it is so called bare-metal implementation.
In a bare-metal implementation all parts of the application have to be scheduled and synchronized by the application code itself. A typical bare-metal application code has a main application loop used for this purpose. RTOS-based implementations often use blocking application programming interface. If a given resource is not available or event did not occur, thread can be blocked until certain condition happens because RTOS scheduler is in operation. In a bare-metal application, blocking application programming interface may hold entire application in one point in the code. A pseudo-code of the main application loop is presented in Fig. 6-1. The state

```c
while(1)
{
    /* Application loop, non-OS implementation.*/
    if(TcpServerFlag == TcpServerListening)
    {
        TcpAccept();
    }
    else if(TcpServerFlag == TcpServerWaitingCmd)
    {
        TcpServerReceive();
        RunRequest();
    }
    else if(TcpServerFlag == TcpServerRunningCmd)
    {
        FinishRequest();
        ...
        ...
    }
    /* The serial command driver.*/
    cmd_pool();
    /* The network co-processor driver.*/
    _SilNonOsMainLoopTask()

    /* End of application loop.*/
}
```

Figure 6-1: Pseudo-code of the non-RTOS application loop.

variable TcpServerFlag is used as an indication of the application state. Application allows a single user to be connected to the device at a time, thus, stepping over the application without dynamic instances of particular functionality is acceptable. The application programming interface used in this firmware implementations uses mostly non-blocking implementations. This prevents non-RTOS application from
getting stuck, while application loop repeats querying for up-coming requests.

During the device initialization a listening TCP socket is created and bound to port 50000 by BsdTcpServer() routine, not shwon in the main application loop. This socket is configured as a non-blocking socket. Thus, periodic calls to application programming interface in TcpAccept(), see Fig. 6-1, trying to accept incoming connection request do not block the entire application. In particular, sl_Accept() function is responsible for accepting incoming connection requests in case of the SimpleLink CC3100 host driver. If connection request is detected, a new TCP socket is created and serves the entire data exchange process, the original socket is not modified and bound to port 50000. After successful connection, incoming requests on the new socket are handled by the TcpServerReceive() routine, see Fig. 6-1. First data byte in the incoming TCP segment, commonly referred to as TCP packet, encodes the particular user request. In the final firmware revision there are ten possible requests with identifiers listed in Table 6.1. Together with request identifier, first TCP segment sends necessary arguments for particular request, e.g., acquisition time or sampling frequency. Details of the argument list are provided in Appendix A. TCP server, i.e. embedded device, responds to client confirming that request was detected and processes the request with RunRequest() routine, see Fig. 6-1. After performing client request, device informs client and terminates the connection. The new socket is closed and server continues listening on the initial socket bound to port 50000.

<table>
<thead>
<tr>
<th>Function</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measure</td>
<td>101</td>
</tr>
<tr>
<td>Download file</td>
<td>102</td>
</tr>
<tr>
<td>Stop flag</td>
<td>103</td>
</tr>
<tr>
<td>Check files in memory</td>
<td>104</td>
</tr>
<tr>
<td>Delete files</td>
<td>105</td>
</tr>
<tr>
<td>Error flag</td>
<td>106</td>
</tr>
<tr>
<td>Measure and download</td>
<td>107</td>
</tr>
<tr>
<td>Check number of sensors</td>
<td>108</td>
</tr>
<tr>
<td>Configure automatic measurements</td>
<td>109</td>
</tr>
<tr>
<td>Check automatic configuration</td>
<td>110</td>
</tr>
</tbody>
</table>
Client request to perform measurement can be handled in various ways, depending on the particular hardware configuration and mode of operation. The designed device had three revisions, mostly due to gradual increase in specifications like number of IEPE channels or noise performance of the analog circuitry. However, first two revisions were operated only in the STA mode, i.e., with an external AP creating wireless LAN used for communication between embedded devices. The reason for that was noise induced by the operation of CC3100 network co-processor. The network co-processor operation would disturb voltage rail and affect performance of the analog circuitry. In order to mitigate this, Wi-Fi related circuitry was disabled for the time of acquisition. However, if device was operated in the AP mode it would result in loss of wireless LAN and severely affect the user experience. Constant reconnections to the wireless network would be necessary. As mentioned previously, upon finishing request device responds to client and terminates the connection. It is highly desirable to implement shut down of the network co-processor in a way that is virtually invisible to the user. Thus, after receiving measurement request device disables network co-processor, while client side opens a new server socket listening on port 50001. After finishing measurement, device enables the network co-processor and requests connection on port 50001. In this case, the embedded device and user’s device switch roles, i.e., first becomes a client and second server. This allows new client to inform server that measurement is over and terminate connection without disrupting user experience. A diagram summarizing work around the problem of coupled noise is presented in Fig. 6-2.

If the aforementioned swap of roles is omitted, device would suffer from noise induced by the operation of Wi-Fi transceiver. As described in Chapter 5, wireless platform consists of two separate PCBs shielded by power planes and ground plane. Additionally, 2.4 GHz interference is out of the frequency band of interest. However, those precautions did not eliminate the interference of the Wi-Fi transceiver. In particular, every AP in IEEE 802.11 based wireless LAN periodically transmits network management frames, so called beacon frames. Beacon frames are used to advertise the network’s SSID and parameters, as well as, are used by the connected devices for
synchronization purposes. As already stated, transmission frequency of the communication channel is not an issue, however, beacon frames periodically arriving to the Wi-Fi transceiver cause peak power draws disturbing supply voltage rail. Fig. 6-3 shows block diagram representing voltage distribution among the loads on early hardware revisions. An example of the disturbance influence is shown in Fig. 6-4, the 3.3 V rail was probed on the bottom digital PCB while Wi-Fi transceiver was enabled. Standard beacon interval of 100 ms can be seen in the figure. First two hardware revisions used a single 3.3 V rail which was filtered and regulated by linear regulators to voltage used by the analog circuitry, see Fig. 6-3. When compared with block diagram of the final design, shown in Fig. 5-53, it is clear that single 3.3 V rail was used in the initial hardware revisions. Peak power draw presented by the transceiver caused voltage disturbance with a wide-frequency content and finite-bandwidth voltage regulation did not completely eliminate the impact on voltage rails feeding the analog circuitry. As an illustration, data from IEPE channels without any excitation
applied is presented in Fig. 6-5, data was collected with early hardware revision. It is clear that disturbance induced by the transceiver operation affects quality of data acquisition. Fig. 6-6 shows the same measurement repeated for the final revision of hardware, where separate 3.3 V rail feeds analog circuitry and Wi-Fi transceiver is not disabled. Complete elimination of 100 ms disturbance is obvious. Thus, final hardware revision can reliably work in the AP mode without disabling the network co-processor during measurements.

Figure 6-4: Disturbance on the 3.3 V rail.

Two of the interface functions involving data acquisition require to use the role
swap described above, namely options 101 and 107 in Table 6.1. Client requests that do not require data acquisition are less complicated from the communication point of view, see Fig. 6-2. For example, downloading or deleting data from the remote memory, i.e. μSD card, do not require to disable the network co-processor. μSD card is formatted in FAT filesystem and open-source FatFs firmware module is used to manage data stored in the memory [137]. The FatFs is a generic FAT/exFAT file system module for small embedded systems, it is independent of the physical layer of microcontroller [137]. Thus, it can be ported to multiple hardware platforms. Long file names were enabled in the default FatFs implementation to allow up to 16
characters in the file name field. In case of download request, files read from the µSD card have to be transferred to the user. Two application layer protocols have been implemented and tested. First one, is a custom implementation of the TFTP protocol. Second one, is a custom protocol based on the TCP transportation layer protocol that simply sends partitioned file in a minimum number of TCP segments. Advantage of the TFTP solution is small footprint and multiple ready software tools to implement TFTP client. In the tested configuration, the embedded device runs a TFTP server listening on port 69, only read actions are allowed by the server. Application layer protocol like TFTP is based on the UDP transportation layer protocol. Such a connection-less protocol lacks error-correction or re-transmission mechanisms that provide reliability. On the application layer, the TFTP server monitors indices of data blocks that are acknowledged by the TFTP client. However, this application layer safety mechanism slows down the file transfer rate in case on many lost UDP datagrams. Unfortunately, the wireless LAN that devices operate in tends to have significant rate of lost network packets that need to be detected by the application layer safety check. In practice, performance of file download functionality was often too slow. Thus, the second file transfer solution is used in the final firmware version. In the second implementation, the already opened socket is used to send requested data using TCP transportation layer protocol. For each file transfer device determines the number of TCP segments to be sent by reading the file size in the memory and dividing it by the size of the TCP segment. In the implemented firmware the TCP segment size is maximized to minimize the number of sent segments; segment size is set to 1400 bytes. TCP segments are assembled together by the client and saved in the text file. Once all of the file segments are sent, client can send a 'NEXT_PLS' message to request more files. All file names are specified in the very first request message send by the client.

In case of missing IEPE transducer, the on-board current source saturates and provides maximum voltage at the connection point. In case of connected IEPE transducer, voltage at the connection point is proportional to transducer’s impedance and current value. The DC voltage at the connection point can be used to indicate pres-
ence of the IEPE transducer. Due to the lack of free four digital lines on the top PCB that could indicate presence of each transducer separately, a voltage-controlled oscillator is used to encode transducers count to frequency, see Appendix C for circuit details. Such solution does not allow to distinguish which transducers are connected but provides a control and diagnostic tool to the user. Such tool can monitor how many transducers are connected, this is very helpful to test the transducer-cable and cable-device connections. The output of the oscillator is used as an input signal to the edge-triggered configured timer. Firmware interprets discrete frequency levels as an indication of transducers count and provides it to the user, see functionality 108 in Table 6.1.

Another important functionality is the automatic measurement feature. This functionality allows to trigger measurement based on an external logic signal, e.g., auxiliary relay terminals, that indicates the event of interest. It is a very valuable functionality for field scenarios where user cannot use manual requests. The functionality with identifier 109 in Table 6.1 can be used to configure parameters of the automatic measurement. Desired channels, sampling frequency and acquisition time can be specified by the user. User can query the device to verify the configuration, see identifier 110 in Table 6.1. A 3.3 V logic signal should be applied to pin five of connector H9 on the top PCB, see Fig. 6-7. The input signal is pulled high and measurement is triggered by the falling edge. The input signal has to be kept low for at least 10 ms to trigger the measurement. Device was tested up to 3500 automated measurements, 10 seconds file were recorded.

![Figure 6-7: Trigger signal input pin.](image)
6.1.2 Client Software

On the user side client software is necessary to communicate with the embedded device. For the final hardware release, software operates only as a TCP client. However, in the older hardware releases software had to implement the role swap client-to-server as described previously. Software providing the necessary functionality was developed in Python as a command-line program.

A pseudo-code describing client software is presented in Fig. 6-8, final hardware release is assumed. After establishing connection over TCP socket, software, i.e., TCP client, sends the request message to the TCP server. Message is constructed based on the output of a parser and user-provided parameters. The TCP server responds with data returned by `RecvWithEotFlag()` function. The data returned by the `RecvWithEotFlag()` function provides a safety mechanism to verify that the TCP server correctly identified the user request. First byte in the returned data corresponds to the action request received by the TCP server. Based on the action request, particular action is performed. The `ActionDict` dictionary in the pseudo-code encodes the action identifiers provided in Table 6.1. For instance, action identifier 107 corresponds to measure and download action which has a dictionary key `MEASURE_DOWNLOAD`. In case of measure and download action, function `FileTransfer()` returns the downloaded data and `ConvertData` method of class instance `Conv` is used to convert and visualize the data. The `FileTransfer()` function implements the custom application layer file transfer protocol that was described previously. The `Conv` class instance (an instance of `DataConversion` class) contains all methods necessary to convert data from raw format to floating-point representation and visualize it using tools from the `matplotlib` library. Software terminates the connection after successful indication from the embedded device in the form of an ASCII encoded flag, it is provided by `CloseWithEotFlag()` function.

The `argparse` Python module enables the command-line interface framework and parses the user input to determine the request configuration [138]. Low-level networking interface and socket programming are provided by the Python `socket` module [139].
The embedded device allows one user connection at a time, thus, there is no need for multithreaded programming that is supported by the Python `threading` module. The developed software operates sequentially and is meant to perform user request and terminate. There is no idle loop waiting for input. Sockets are configured as non-blocking or blocking for specified amount of time. This allows the user to terminate the command-line program in case of failure. In case of autonomous operation, there is no user input to terminate the program, thus all sockets should be configured as blocking with specified timeout. A detailed manual for the software is included in Appendix A, it contains example use cases and error messages. The TCP client software source code is provided in Appendix B in Section B.1. Additionally, an example Matlab script to analyze the acquired data is provided in Appendix B in Section B.3.

6.2 Firmware and Software for the Wired Platform

The wired embedded system hardware was described in Section 5.3. In order to enable applications of the developed hardware, a firmware for the embedded device and client
software need to be implemented. Details of implementation and functionalities are discussed in the following subsections.

### 6.2.1 Embedded Device Firmware

Firmware implementation for the device with Ethernet connectivity was developed using the TI-RTOS real-time operating system [136]. Contrary to the firmware of wireless platform, this firmware benefits from synchronization and scheduling tools offered by RTOS. This provides a lot of benefits like scalability, ease of code maintenance and threads synchronization. However, some of the application bandwidth is used by RTOS. An example of this limitation is presented further in this section. Two firmware versions are described, first one streaming data in-real time and second one sending acquired data once measurement is finalized.

Firmware described below differs significantly from the firmware for wireless platform. The difference is not only application of RTOS but also basic functionalities. While wireless platform firmware used the local memory to store permanently files, this firmware streams data over the TCP socket and files are not stored remotely. A block diagram representing application code is shown in Fig. 6-9. The application involves 5 threads: two tasks and three hardware interrupts and one semaphore to provide synchronization. The `tcpHandler` task creates a blocking TCP socket listening on port 50000. The TCP socket blocks the task until connection request

![Figure 6-9: Block diagram of the RTOS-based firmware.](image-url)

...
from client arrives. Once connection request is successfully accepted, a new socket is opened and application creates dynamically new task \textit{tcpWorker} responsible for streaming data to the TCP client over the new socket. Additionally, \textit{tcpHandler} disables timer interrupt responsible for blinking the inspection LED during the idle state. The newly-opened socket receives requested measurement time within the \textit{tcpWorker} task. Once requested time is recovered, \textit{tcpWorker} task sets global state variable indicating start of data acquisition. Sampling frequency is fixed in this firmware implementation, i.e., defined when code is built. The change in the state variable informs hardware interrupt routine servicing asynchronous interrupt from a data ready line of the ADC to start reading the incoming data. Once all data is read over the SPI bus, the SPI interrupt routine is triggered and data is read from the receive FIFO buffer and copied to the network buffer. The network buffer is defined as 1424 byte array (assuming 16 bit resolution, i.e., 2 bytes per word, and six channels with two additional status words) holding data to be sent. Once network buffer is filled, a semaphore is posted to indicate that data is ready to be sent. The \textit{tcpWorker} task pends on the same semaphore. Thus, after posting the semaphore, the \textit{tcpWorker} task sends data in the network buffer to the TCP client. When all data is acquired according to the requested measurement time, \textit{tcpWorker} closes connection with the TCP client, enables timer interrupt and terminates. The \textit{tcpHandler} task continues listening on port 50000 for a new connection request.

The second version of the firmware uses TFTP server to transfer acquired data. Incoming data is buffered in the external SDRAM memory connected to the microcontroller via parallel port. Another difference with respect to the first firmware version is that the \textit{tcpWorker} task is pending on the semaphore until measurement is completed. Additionally, there is a second semaphore used to synchronize the TFTP server task. After measurement completion, the \textit{tcpWorker} task closes the TCP socket and posts the second semaphore which blocks the TFTP server task. The TFTP server listening on port 69 transfers file to the TFTP client. The TFTP server implementation uses a non-standard block size of 1400 bytes, instead of 512 bytes, in order to improve the file transmission rate. The TFTP server implementation was
customized based on [140]. The TFTP server implementation verifies requested file name after client connection, expected file name is 'Data'.

It has been mentioned before that operation of RTOS can affect firmware performance. Table 6.2 compares maximum sampling rates for the bare-metal firmware and two versions of the RTOS-based firmware, i.e., version streaming data and version with the TFTP server. The reported values were obtained for a six channel measurement with 16 bit resolution. It can be seen that the bare-metal firmware can achieve higher data rates. A snapshot of code execution profile in Fig. 6-10 shows performance of RTOS-based firmware, abscissa is scaled in µs. The routine name INTSSI2 refers to the SPI ISR in the block diagram in Fig. 6-9. Code execution profile shows performance of the application with 21 kHz sampling rate, time spent in the DRDY and SPI routines is marked in the figure. RTOS tools require some computation time for RTOS to operate. A code execution profile in Fig. 6-11 shows potential impact of RTOS operation on real-time performance of the application. Comparing Figs. 6-10 and 6-11, it can be seen that firmware did not manage to read and buffer

<table>
<thead>
<tr>
<th>Firmware</th>
<th>Max Sampling Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bare-metal</td>
<td>64 kHz</td>
</tr>
<tr>
<td>RTOS</td>
<td>10 kHz</td>
</tr>
<tr>
<td>RTOS with TFTP</td>
<td>21 kHz</td>
</tr>
</tbody>
</table>

Figure 6-10: Code execution profile.
the incoming data in real-time. This is one of the major reasons for differences in firmware performance reported in Table 6.2.

6.2.2 Client Software

Client side software is implemented in Python, similarly to the software for wireless platform. Two software versions are described according to the requirements of two firmware versions described above. In both cases client software opens a TCP socket and attempts connection to port 50000, the socket is configured as blocking. Measurement request is sent to the embedded device, it consists of the request opcode and requested measurement time. Upon successfully receiving request, the embedded device sends confirmation to the client. The first software version enters a loop periodically receiving segments of data from the device and streams it to the file, by default located in ‘C:/Users/’ directory. After the last segment transmission, data file and connection are closed and program returns. The second software version enters a loop waiting for notification from the embedded device that measurement was finished. Then, the TFTP client requests measurement data on port 69. A Python TFTP client implementation was used from [141], data block size was modified to 1400 bytes. Low-level networking interface and socket programming are provided by the Python socket module [139].

Figure 6-11: Code execution profile, RTOS impact.
Chapter 7

Conclusion

Diagnostic metrics introduced in this thesis provide reliable condition monitoring tools tailored for periodically operated devices. Monitoring of cyclostationary components with the MES in vibration signal was applied as a candidate method and proved to be a solid diagnostic tool. It presents a superior performance for signals where periodic modulations interfere with essentially random content. Such a scenario is common for a wide range of industrial actuators with diaphragm pump and CNC milling machine being good examples. The MES indicator relies on the detection of hidden periodicity in a vibration signal, hence can be applied for all electromechanical, pneumatic or hydraulic actuators driving periodic loads. Additionally, energy of the vibration signal at the mechanical resonant frequency was successfully deployed for condition monitoring. Another approach presented in this thesis is based on the analysis of the frequency content in the current signal. Adaptive discrete-time notch filter allows for optimal analog-to-digital conversion of the signal of interest. Two frequency locations were identified as candidates for detection of the valve clogging and diaphragm degradation phenomena.

A new IEPE transducer extends the measurement ability of the IEPE interface to slowly-varying and DC signals. The IEPE transducer uses a voltage-controlled oscillator to encode signal amplitude to frequency by means of frequency modulation. The performance of the transducer was validated with temperature and strain sensing elements. An IEPE microphone was also considered in this thesis. A small-signal
analysis provides a gain expression that closely matches experimental results. Presented comparisons with commonly accepted microphone amplifier topology validate the design. The developed instrumentation stretches the applications of the IEPE interface to a wide range of measurands.

An embedded acquisition platform presented in this thesis combines high-end analog circuitry, optimized for the IEPE transducers, with IoT connectivity. It results in a compact and portable platform that is optimized for field applications, including temporary as well as permanent installations. The hardware platform was optimized for low-noise and compact form factor. Design of a system combining high-end analog circuitry with embedded system presents a difficult mixed-signal system design problem. Impact of wireless communication links provides additional design constraints.
Bibliography


219


[102] PUI AUDIO. *POM-3535P-3-R Microphone*, September 2013. Rev. A.


[107] Linear Technology. *LTC1060 Universal Dual Filter Building Block*. Rev. B.


[110] Norsonic. *IEPE Outdoor Microphone Nor1218 for semi-permanent installations*.


[125] Texas Instruments. *ADS131xA0x 2- or 4-Channel, 24-Bit, 128-kSPS, Simultaneous-Sampling, Delta-Sigma ADC*, 2020. Rev. E.


Appendix A

Puck Interface

This appendix provides a user manual for software used to communicate with the wireless platform described in Chapter 5.

A.1 Interface Manual

This manual is valid for firmware and software 1.6.0 revisions.

1. Connection with PUCK is established over Wi-Fi. Stand-alone Python script is used to handle data measurement requests and data exchange.

- On Windows machine open PowerShell or command line with administrator privileges (downloaded files are saved on the PC side). Call Python ScriptName.py from appropriate directory.

- There are several parameters to run ScriptName.py.
  - ‘x’: defines action type ‘x’. It is a necessary parameter! There are 8 action types:
    * a: stands for acquire data request
    * m: stands for remote data inspection request
    * g: stands for file request
    * d: stands for file delete request
* ag: stands for file acquire and download request. Files aren’t saved remotely in this operation!

* cs: stands for checking number of connected IEPE transducers

* st: stands for configuration of automatic measurements

* ct: stands for checking configuration of automatic measurements

- `-f 'FileName': provides a file name for requested data. It allows up to 16 characters. The limit on the delete action and download action is 99 files in one operation, i.e., 99 files can be specified to be downloaded or deleted. Accepted delimiters: space, new line character or NULL termination.

- `-t X: specifies time to acquire (X is an integer number). If none specified, 10 seconds will be used.

- `-fs X: specifies sampling frequency. X: 0-7812Hz, 1-15624Hz, 2-20833Hz, 3-31249Hz, 4-62498Hz

- `-ch xxxx: specifies channels to acquire. Example: `-ch 1101 specifies channels one, two and four. By default all channels are acquired.

- `-c X: turns on or off the current source. “-c 1” turns on the current source. Remember about IEPE transducer start-up time.

- `-ip xxx.xxx.x.x: specifies the device IP address. If not specified 192.168.2.7 will be used.

- `-path: path to save files when downloaded from the remote memory. Default: C:/Users/ (REMEMBER ABOUT WRITE PRIVILEGE)

- Type Python ScriptName.py -h to get the help menu, see Fig. A-1.

- If SD card is missing the error message in Fig. A-2 will appear.

- If real time clock was not configured (via serial port), the error message in Fig. A-2 will appear.

- Example usage commands:
Acquire 10 seconds of data (t not specified): Python ScriptName.py -a a -fs 4 -f 'TCS3' -c 1 -ch 1100 -ip 192.168.2.7

Check automatic measurement configuration: Python ScriptName .py -a ct -ip 192.168.2.7

Set automatic measurement configuration: Python ScriptName .py -a st -t 3 -ch 11 -fs 3 -ip 192.168.2.7

Check memory content: Python ScriptName .py -a m -ip 192.168.2.7

Download two files: Python ScriptName .py -a g -f 'TestIEPE5 TestIEPE6' -ip 192.168.2.7 (new line character can be used to separate file names)

Delete two files: Python ScriptName .py -a d -f 'TestIEPE5 TestIEPE6'
2. A serial interface allows to configure the device.

- The following options are available:
  - Set local time: `set_time HH:MM:SS`.
  - Set local date: `set_date DD/MM/YYYY`.
  - Check local time and date: `get_time_date`.
  - Hibernate (work in progress). Currently hib will turn off the device for 30 seconds.
  - Set IP address: `set_IP num1.num2.num3.num4`, address is changed permanently (even after device restart).
  - Check hardware MAC address and IP address: `get_IP_MAC`.
- Any software for serial port communication (Putty, CoolTerm) can be used. If handling of the delete symbol on the PC software side (Putty, CoolTerm) is enabled, puck interface will recognize the delete symbol and correctly interpret the command.
- Type `help` in the serial port interface to get the description of interface options, see Fig. A-3.

3. Example notifications:

- Deleting/downloading multiple files, space or new line character are acceptable as delimiters between file names, see Fig. A-4.
- Deleting/downloading over 99 files, error message, see Fig. A-5.

4. The µSD card that was used during device development and verification is recommended as a default card choice. It is optimized for memory access times.
help : Display list of commands.
hib : Place system into hibernate mode.
set_IP : Set IP address with style "num1.num2.num3.num4".
set_date : Set Date "DD/MM/YYYY".
set_time : Set Time 24HR style "HH:MM:SS".
get_time_date : Display system date and time.
get_IP_MAC : Display the MAC and IP addresses.
cls : Clear the terminal screen

Figure A-3: Help menu in the serial interface.

Figure A-4: Deleting/downloading multiple files.

Model: SanDisk Extreme PRO, see Fig. A-6. Other card may be used but performance has to be verified.

5. Notes:

- Current version of interface uses only TCP transportation layer protocol. There should be no need to create additional firewall rules. If any problems occurs try to set up wireless LAN between devices as a private network and disable firewall to see if this enables operation.

- There is no DHCP server on the device side. Thus, client should set a static IP on a correct subnet.

A.2 Revision History

Revision history of the bottom PCB is described in Table A.1.
Example connection with the CC31XXEMUBOOST module from TI is presented in Fig. A-7. It can be used to update firmware in the network co-processor of the wireless platform, refer to [131] for details.
### Table A.1: Revision History

<table>
<thead>
<tr>
<th>Bottom Board Revision</th>
<th>Details:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>No IEPE channels, only MEMS sensors</td>
</tr>
<tr>
<td>1.1</td>
<td>Single IEPE channel, compatible with daughter board extension</td>
</tr>
<tr>
<td>1.2</td>
<td>All IEPE channels available through daughter board, improved current source implementation</td>
</tr>
<tr>
<td>1.3</td>
<td>All IEPE channels available through daughter board, improved current source implementation and resolved AP mode noise problem</td>
</tr>
</tbody>
</table>

**Figure A-7:** The CC31XXEMUBOOST-CC3100 programming connection.
Appendix B

Source Code

This appendix provides code listings for software dedicated to work with hardware platforms described in Chapter 5.

B.1 Wireless Platform Client Software

#-*- coding: utf-8 -*-
#
# Created on Tue Aug 28 22:03:50 2018
# @author: Lukasz
# Modified on February 2021
# @author: Lukasz Huchel and Thommas Krause
# revision: 1.6.0

# Features:
#  - this mode is compatible with STA version of the
#    embedded device
#  - reads binary files from remote storage
#  - deletes binary files from remote storage
#  - if '.txt' extension is provided code will automatically reject
#      it
#  - frequency selection: supported
#  - current source selection: supported
#  - channel selection: supported
#  - multiple downloads: supported
# TODO: Decide course of action once file mismatch occurs
# Send date/time to RTC
# Autonomous functions for beaglebone use
# ""
import os
import socket
import datetime
import time
import argparse
import csv
import re
from ConvertFile import DataConversion

fmt = '%Y_%m_%d %H:%M:%S' # ex. 20110104172008 -> Jan. 04, 2011 5:20:08 pm
### Set the sampling frequency and data frame:
freq=0
# freq=7812
# freq=20833
# freq=31249
# freq=62498
data_frame=695##14
### Set the server IP address and TCP port:
# TCP_IP = '18.62.14.82' # MIT registered
# TCP_IP = '192.168.2.7'
# TCP_IP = '192.168.2.4'##no-OS
# TCP_IP = '192.168.1.209'##OS
TCP_PORT = 50000
TCP_PORT2 = 50001# Port to listen on (non-privileged ports are > 1023)
BUFFER_SIZE = 1460
### MESSAGES:
MESSAGE1 = bytes([101])##Start measurement message
MESSAGE2 = bytes([102])##Start downloading message
MESSAGE3 = bytes([103]) ## Stop message
MESSAGE4 = bytes([104]) ## Get names of files saved in the remote memory message
MESSAGE5 = bytes([105]) ## Delete a file in the remote memory
MESSAGE6 = bytes([106]) ## Error message
MESSAGE7 = bytes([107]) ## Start measurement and download after message
MESSAGE8 = bytes([108]) ## Check number of IEPE sensors present
MESSAGE9 = bytes([109]) ## Configure automatic measurements, triggered externally
MESSAGE10 = bytes([110]) ## Check automatic measurements configuration, triggered externally
PARSER_ACQUIRE = 'a' # acquire
PARSER_DOWNLOAD = 'g' # get file
PARSER_ACQUIRE_DOWNLOAD = 'ag' # acquire and get file
PARSER_CHECK_MEMORY = 'm' # memory
PARSER_DELETE_MEMORY = 'd' # delete file
PARSER_CHECK_SENSORS = 'cs' # check IEPE sensors
PARSER_CONF_TRIGGER = 'st' # configure automatic measurements
PARSER_CHECK_CONF_TRIGGER = 'ct' # configure automatic measurements
FreqDict = { '7812':0, '15624':1, '20833':2, '31249':3, '62498':4} # Dictionary with available sampling frequencies
FreqKeys = list(FreqDict.keys())
FreqValues = list(FreqDict.values())
FILE_LIMIT = 99 # Limit of files that can be downloaded/deleted at once

### METHODS:
''
VerifyPath: - Verifies that the path provided by the user exists.
It is used to save the downloaded file in the memory.

This function will append "\" if omitted (last element is expected to be either \ or /).

```python
def VerifyPath(Path):
    if Path[-1] != "\\" or Path[-1] != "/":
        Path = Path + "\\"
    assert os.path.exists(Path) == True, 'Specified local path does not exist'
    return Path
```

-Verifies that the path provided by the user exists.

It is used to save the downloaded file in the memory.

This function will append "\" if omitted (last element is expected to be either \ or /).

```python
def PrintNames(Names):
    ind1 = 0
    ind2 = 0
    for i in range(0, len(Names) - 1):
        if Names[i] == 0x0a or Names[i] == 0:
            ind2 = i
            print(Names[ind1:ind2].decode('ASCII'))
            ind1 = i + 1
    return
```

-Name is the file name or list of file names, it will get rid of possible space at the beginning.

',' ','\n' and NULL are acceptable as delimeters

-Action is the action identifier.
- Returns : [ Names , Number of Files to Transfer ,

109

Number of Packets to send during init ]
With 99 files of full length , more than 1460

110

bytes is needed .
It is important for delete and download actions .

111
112

TODO :

113

Not allow space in file name for aquire
Increase allowed number of downloads

114
115

’’’

116

def VerifyFileName ( Name , Action ) :

117

if ( Action == P A R S E R _ A C Q U I R E _ D O W N L O A D and Name == None ) :

118

now = datetime . datetime . now ()

119

Name = now . strftime ( " % d_ % m_ % Y_ % H_ % M_ % S " )

120

# Name += ’. txt ’

121

print ( ’ File name : ’ + Name ) # print file name that will be
used
return Name , 1 , 1

122
123

# # PARSER TAKES CARE OF CASES LIKE BELOW :

124

# if (( Action == P A R S E R _ A C Q U I R E _ D O W N L O A D or Action ==
P A R S E R _ D E L E TE _ M E M O R Y ) and Name == None ) :

125

#

print ( ’ Warning : No file name specified ! ’)

126

#

return None , 0 # Returning 0 as second argument will

terminate the program
127
128

elif Name != None :
for ch in Name :

129

# print ( element , end = ’ ’)

130

if ch != ’ ’: # # It will eliminate possible space at the
beginning of the file name or file name list

131

ind = Name . find ( ch )

132

break

133

if ( Name [ ind : len ( Name ) ]. find ( ’ ’) > -1 or Name [ ind : len ( Name ) ].
find ( ’\ n ’) > -1 or Name [ ind : len ( Name ) ]. find ( ’ \0 ’) > -1 or Name [ ind :
len ( Name ) ]. find ( ’ \10 ’) > -1) :

134

# global num_total

135

WordList = re . split ( ’ |\0|\10|\ n ’ , Name [ ind : len ( Name ) ])

136

# WordList = Name [ ind : len ( Name ) ]. split ( ’ ’)

239


# print(WordList)
for element in WordList:
    if element.find('.txt') != -1:
        WordList[WordList.index(element)]=WordList[WordList.index(element)][:element.find('.txt')-1]
Name = ' '.join(WordList)
if len(WordList) > FILE_LIMIT:
    print('Warning: 99 file download/delete limit')
    return 'Error', 0, 0
return Name, len(WordList), ((len(Name)>1460)+1)
else:
    # print(Name[ind:len(Name)], Name[ind:len(Name)].find('.txt'))
    if (Name[ind:len(Name)].find('.txt'))>-1:
        # print(1)
        return Name[ind:len(Name)-4], 1, 1
    else:
        # print(2)
        return Name[ind:len(Name)], 1, 1
    else:
        return None, 1, 1

def ListToString(s):
    # empty string
    str1 = ""
    # loop the string
    for element in s:
        str1 += str(element)
    # return string
    return str1

def UserInput(Option,Time,Name,IEPE,Fs,Channels,PacketCount):
    global freq
    if Fs != 0 and Fs != None:# Get sampling frequency or assign default frequency
        freq=int(FreqKeys[FreqValues.index(Fs)])
else:
    freq=7812#Default sampling frequency
    # print(freq)
    if Option==PARSER_ACQUIRE:
        NumofWords=sum([int(element) for element in Channels])+1#
        Channels plus status word
        if NumofWords>5:
            print('Warning: Wrong channel specification format. Four
channels will be used')
        NumofWords=5
        data_frame=(NumofWords)*139#4 channels plus status
        ->5*139=695, this many 16 bit words
        buff=Time*freq//((data_frame//((NumofWords)))##Number of data
buffers needed to store desired amount of data
        print('Actual buffer size:',buff)##Print number of ethernet
frames to transfer
        mes=bytearray(MESSAGE1)
        mes_temp=buff.to_bytes(4, byteorder='big', signed=True)
        mes+=mes_temp##Start command is concatenated with the
desired number of buffer frames (assuming buffer will be encoded
in 4 bytes!!)
        mes+=bytearray("IEPE:", 'utf-8')
        if IEPE != None:
            mes+=bytes(str(IEPE), 'utf-8')
        else:
            mes+=bytes(str(0), 'utf-8')
        mes+=bytearray("FREQ:", 'utf-8')##Send sampling frequenecy
        mes+=bytes(str(Fs), 'utf-8')
        mes+=bytearray("CH:", 'utf-8')##Send desired channels
        # Channels=[str(abs(int(c)-1)) for c in Channels]##Revert the
channel labeling
        if len(Channels)!=4:
            # print("\u0332".join("Warning: "))
            print('Warning: Wrong channel specification format. Four
channels will be acquired')
            Channels=[1,1,1,1]
mes += bytes(ListToString(Channels), 'utf-8')
mes += bytearray("NAME:", 'utf-8')  # Send file name, if custom
# if Name != None:
if Name != None and len(Name) > 16:
    # print(len(Name))
    print('Warning: File name above 16 characters. Default file naming will be applied')
else:
    # print(Name)
    mes_temp = bytearray(Name, 'utf-8')
    mes += mes_temp  # Start command is concatenated with the file name
    mes += bytes([0])  # NULL termination
    return mes
elif Option == PARSER_DOWNLOAD:
    mes = bytearray(MESSAGE2)
    mes += bytearray("PACKET:", 'utf-8')  # Inform server is second packet will be sent
    mes += bytes([PacketCount])
    mes_temp = bytearray(Name, 'utf-8')
    mes += mes_temp  # Start command is concatenated with the file name
    mes += bytes([0])  # NULL termination
    return mes
elif Option == PARSER_CHECK_MEMORY:
    mes = bytearray(MESSAGE4)
    return mes
elif Option == PARSER_DELETE_MEMORY:
    mes = bytearray(MESSAGE5)
    mes += bytearray("PACKET:", 'utf-8')  # Inform server is second packet will be sent
    mes += bytes([PacketCount])
    mes_temp = bytearray(Name, 'utf-8')
mes+=mes_temp##Start command is concatenated with the file name
mes+=bytes([0])##NULL termination
return mes
elif Option==PARSER_CHECK_SENSORS:
    mes=bytearray(MESSAGES)
    return mes
elif Option==PARSER_ACQUIRE_DOWNLOAD:
    NumofWords=sum([int(element) for element in Channels])+1#
    Channels plus status word
    if NumofWords>5:
        print('Warning: Wrong channel specification format. Four channels will be acquired')
        NumofWords=5
        data_frame=(NumofWords)*139#4 channels plus status ->5*139=695
        buff=Time*freq//(data_frame//(NumofWords))##Number of data buffers needed to store desired amount of data
        print('Actual buffer size:',buff)#Print number of ethernet frames to transfer
        mes=bytearray(MESSAGE7)
        mes_temp=buff.to_bytes(4, byteorder='big', signed=True)
        mes+=mes_temp##Start command is concatenated with the desired number of buffer frames (assuming buffer will be encoded in 4 bytes!!)
        mes+=bytearray("IEPE:", 'utf-8')
        if IEPE != None:
            mes+=bytes(str(IEPE), 'utf-8')
        else:
            mes+=bytes(str(0), 'utf-8')
        mes+=bytearray("FREQ:", 'utf-8')#Send sampling frequency
        mes+=bytes(str(Fs), 'utf-8')
        mes+=bytearray("CH:", 'utf-8')#Send desired channels
        # Channels=[str(abs(int(c)-1)) for c in Channels]#Revert the channel labeling
        if len(Channels)!=4:
# print("\\u0332".join("Warning: "))
print('Warning: Wrong channel specification format. Four channels will be acquired')
Channels=[1,1,1,1]
mes+=bytes(ListToString(Channels), 'utf-8')
mes+=bytearray("NAME:", 'utf-8')
if Name == None:
    Name='default.txt' #This isn’t used, remote end isn’t saving on SD card anyway
    # print(Name)
    mes_temp=bytearray(Name, 'utf-8')
mes+=mes_temp## Start command is concatenated with the file name
    mes+=bytes([0])## NULL termination
    return mes
elif Option==PARSER_CONF_TRIGGER:
    NumofWords=sum([int(element) for element in Channels]) +1## Channels plus status word
    if NumofWords>5:
        print('Warning: Wrong channel specification format. Four channels will be acquired')
        NumofWords=5
        data_frame=(NumofWords)*139#4 channels plus status ->5*139=695
        buff=Time*freq/(data_frame/(NumofWords))## Number of data buffers needed to store desired amount of data
        print('Actual buffer size:',buff)# Print number of ethernet frames to transfer
        mes=bytearray(MESSAGE9)
        mes_temp=buff.to_bytes(4, byteorder='big', signed=True)
        mes+=mes_temp## Start command is concatenated with the desired number of buffer frames (assuming buffer will be encoded in 4 bytes!!)
        mes+=bytearray("IEPE:", 'utf-8')
        if IEPE != None:
            mes+=bytes(str(IEPE), 'utf-8')
else:
    mes+=bytes(str(0), 'utf-8')
mes+=bytearray("FREQ:", 'utf-8')# Send sampling frequency
mes+=bytes(str(Fs), 'utf-8')
mes+=bytearray("CH:", 'utf-8')# Send desired channels
# Channels=[str(abs(int(c)-1)) for c in Channels]# Revert the channel labeling
    if len(Channels)!==4:
        # print("\u0332".join("Warning: "))
        print('Warning: Wrong channel specification format. Four channels will be acquired')
    Channels=[1,1,1,1]
mes+=bytes(ListToString(Channels), 'utf-8')
mes+=bytearray("NAME:", 'utf-8')
    if Name == None:
        Name='default.txt' # This isn’t used, remote end isn’t saving on SD card anyway
        # print(Name)
    mes_temp=bytearray(Name,'utf-8')
    mes+=mes_temp## Start command is concatenated with the file name
mes+=bytes([0])## NULL termination
return mes
elif Option==PARSER_CHECK_CONF_TRIGGER:
    mes=bytearray(MESSAGE10)# Just send the right action flag, no arguments
    mes+=bytes([0])## NULL termination
    return mes

def CloseWithEotFlag(s,flag):
    while True:
        try:
            data = s.recv(BUFFER_SIZE)
        except BlockingIOError:## Blocking socket would keep waiting
            pass
        else:
if data.find(bytes('EndOfData','utf-8'))>0: # Check if EOT flag is present
    if data[0]==103:
        if flag == 1:
            print('Measurement successfully finished, data saved on Sd card:
\',datetime.datetime.now().strftime(fmt)+')
s.close()## Should provide FIN flag from client side
    elif flag == 0:
        print('Measurement successfully finished:
\',datetime.datetime.now().strftime(fmt)+')
        return data
    else:
        print('Connection unexpectedly closed..')
s.close()## Should provide FIN flag from client side
        return data
else:
    print('Connection unexpectedly closed...')
s.close()## Should provide FIN flag from client side
    return 0

ReceivWithEotFlag: This function looks for end-of-transmission (EoT)
flag and returns when received.

TODO:

def ReceivWithEotFlag(s):
    while True:
        try:
            data = s.recv(BUFFER_SIZE)
            # print(type(data))
        except BlockingIOError:## Blocking socket would keep waiting
            # print('t')
pass
    else:
        if data.find(bytes('EndOfData','utf-8'))>0: #Check if EOT flag is present
            # print('t2')
            return data[0:(len(data)-len('EndOfData'))]
        # return data[0:(len(data))]
    else:
        pass

'RecvWithEotFlagPrint: This function looks for end-of-transmission (EoT) flag and returns when received. It prints payload received before the packet with end-of-transmission (EoT) flag. Preamble is optional string to display at the beginning.

TODO:

def RecvWithEotFlagPrint(s,preamble):
    if bool(preamble)==True:
        print(preamble)
    while True:
        try:
            data = s.recv(BUFFER_SIZE)
            # ind1=0
            # ind2=0
        except BlockingIOError:##Blocking socket would keep waiting
            pass
        else:
            if data.find(bytes('EndOfData','utf-8'))>0: #Check if EOT flag is present
                return data[0:(len(data)-len('EndOfData'))]
# return data[0:(len(data))]  

else:
    if(len(data)>0):
        print('Payload size: ',len(data))
        print(data.decode('ASCII'))

def StartConnection(s,TCP_IP, TCP_PORT):
    try:
        s.connect((TCP_IP, TCP_PORT))
    except:
        print('Connecting to '+TCP_IP+' failed\n')
        exit('')

FileTransfer:  -TCP_IP is the IP address  
                -s is the socket identifier  
                -Name is a name of a file to create  
                -FileCounter counts how many files have been downloaded  
                -Action Identifier allows to pass 'Name' in case of the MEASURE_DOWNLOAD action,  
                use 1 for MEASURE_DOWNLOAD action  
                -Returns 1 on success and 0 on error.

    def FileTransfer(TCP_IP,s,Name,FileCounter,ActionIdentifier,Path):
        save_path=Path
        BUFFER_SIZE = 2000#1460#20000#1400
        mes=bytes('NEXT_PLS','utf-8')
        mes+=bytes([0])## NULL termination
        assert s.send(mes) >= 0 , print('Failed to send')##
        while True:
            try:
                data = s.recv(BUFFER_SIZE)
            except BlockingIOError:##Blocking socket would keep waiting  
                # print('t')
                pass

248
else:
    if data.find(bytes('SIZE2TX:', 'utf-8')) >= 0:
        # Find indicies of Name, Count, Channel, Freq, and Number of Bytes
        ChInd = data.find(bytes('CH:', 'utf-8'))
        FreqInd = data.find(bytes('FREQ:', 'utf-8'))
        NumOfBytesStr = data[0:ChInd]  # Bytes literal form

        # print(NumOfBytesStr)
        NumOfBytes = int(NumOfBytesStr)

        # Extract the information using indicies
        if ActionIdentifier != 1:
            NameInd = data.find(bytes('FILENAME:', 'utf-8'))
            CountInd = data.find(bytes('FILECOUNTER:', 'utf-8'))
        
        FileName = data[NameInd: len('FILENAME:') + len(data)]
        FileName = FileName.decode('utf-8')
        CountStr = data[CountInd + len('FILECOUNTER:') + 2]  # Bytes literal form
        Count = int(CountStr)
    else:
        FileName = Name
        Count = 1  # Only one download allowed for MEASURE_DOWNLOAD action

        # Print values to console
        # print(FileName)
        print('Number of kB to be transmitted: %.3f' % float(NumOfBytes / 1000))

        if Count != FileCounter:
            # Puck and script have become mismatched
            print('Warning: file counter mismatch with remote end')

        ChannelsList = data[ChInd:ChInd + 7]
        ChannelsListInt = [int(s) for s in ChannelsList]  # Revert the channel labels

    249
TempStr1 = 'Channels selected: '
TempStr2 = 'Frequency: '
TempInd = 0
for Ch in range(len(ChannelsListInt)):
    if ChannelsListInt[Ch] == 1:
        TempStr1 += str(Ch + 1) + ' '
        TempInd += 1
print(TempStr1) ## Print selected channels in the terminal
TempStr2 += FreqKeys[FreqValues.index(data[FreqInd + 5])] + 'Hz '
print(TempStr2) ## Print selected frequency
break
elif data.find(bytes('No file found','utf-8')) >= 0:
    print(data.decode("utf-8"))
    time.sleep(1)
    s.close()
    return 0
else:
    # print(data)##Just for debugging
    print('Unexpected event....')
    time.sleep(1)
    s.close()
    return 0
mes = bytes('START','utf-8')
mes += bytes([0]) ## NULL termination
assert s.send(mes) >= 0 , print('Failed to send file request')##
print('Downloading started')
NamePath = save_path + FileName
file = open(NamePath + '.txt', 'w')
file.write('Channels count:%d\n' % TempInd)
file.write('%s\n' % TempStr1)
file.write('%s\n' % TempStr2)
# filewriter = csv.writer(file)
```python
# file = open(NamePath, 'w')
# file = open(NamePath, 'w', encoding='utf-8')
# filewriter.writerow(["Byte:", "Data:"])  
# file.write("Byte: Data:\n")
file.write("Data:\n")
i = 0
##Loop waiting for frames:
while True:
    try:
        data = s.recv(BUFFER_SIZE)
    except BlockingIOError:  
        # print('t')
        pass
    else:
        if not data:  
            file.close()  
            s.close()  
            print('Exepcted event...')
            return 0
        elif i < (NumOfBytes - 1 * BUFFER_SIZE):
            # print(len(data))  
            # Just for debugging
            for value in data:
                # filewriter.writerow([i, value])  
                # filewriter.writerow([value])  
                # file.write('%c' % value)
                # file.write('%d,%d
' % (i, value))
                file.write('%d
' % (value))
                i = i + 1
        elif data.find(bytes('EndOfData', 'utf-8')) >= 0:
            # We don’t want to keep looking all the time for EndOfData, just at the end.
            file.close()  
            # print(i, NumOfBytes)
            print('Downloading finished')
            return 1
```
```python
def DeleteFile(TCP_IP,s):
    # print('Downloading finished')
    
def Parser():
    Description='This code allows to connect, acquire data, check
    currently stored files and download desired data from the Hockey
    Puck.\nIt uses default TCP port 50000.'
    parser = argparse.ArgumentParser(description=Description,
        epilog="End",add_help=False)
    requiredNamed = parser.add_argument_group('required arguments')
    requiredNamed.add_argument('-a','--a', action='store', dest='action', required=True,
        help='Chose action:
- acquire data: a,
- check files in the remote memory: m,
- download file from the remote memory: g,
- delete file from the remote memory, file names separated by space or new line character: d,
- acquire data and download immediately (not saved remotely): ag,
- check number of connected IEPE sensors: cs,
- configure automatic measurements: st,
- check automatic measurements configuration: ct, use st prior to using ct')
    parser.add_argument('-c','--c', dest='IEPE', action='store', type=int,
```

help='Turn on current sources for IEPE sensors with 1, default is 0')

parser.add_argument('-f', '--file', action='store', dest='FileName',
    help='Name of file that will be created or file to be downloaded \n    (up to 16 characters). If no name is provided, default naming scheme will be applied. \n    Multiple file names can be provided for "d" and "g" actions. Acceptable delimiters: \n    ",", new line character or NULL character. File list should not contain more than 99 names')

parser.add_argument('-f', '--file', action='store', dest='FileName',
    help='Name of file that will be created or file to be downloaded \n    (up to 16 characters). If no name is provided, default naming scheme will be applied. \n    Multiple file names can be provided for "d" and "g" actions. Acceptable delimiters: \n    ",", new line character or NULL character. File list should not contain more than 99 names')

parser.add_argument('-t', '--time', dest='Time', action='store',
    default=10, type=int,
    help='Acquisition time in seconds')

parser.add_argument('-ch', '--channels', dest='Channels', action='store',
    default=[1,1,1,1], type=list,
    help='Desired channel has to be marked with one in the list, example of input: 1001')

parser.add_argument('-fs', '--fs', dest='Fs', action='store',
    default=0, type=int,
    help='Sampling frequency (7812Hz is default). Option:
    -0: 7812Hz,
    -1: 15624Hz,
    -2: 20833Hz,
    -3: 31249Hz,
    -4: 62498Hz')

parser.add_argument('-ip', '--ip', action='store', dest='TCP_IP',
    default='192.168.2.7',
    help='Server IP address, default address: 192.168.2.7')

parser.add_argument('-path', '--path', action='store', dest='Path',
    default='C:/Users/',
    help='Path to save files when downloaded from the remote memory. Default: C:/Users/ (REMEMBER ABOUT WRITE PRIVILEGE)')

# parser.add_argument('--version', action='version', version='%(prog)s 1.0')

parser.add_argument('-h', '--help', action='help', default=argparse.SUPPRESS,
```python
help='Help...')

results = parser.parse_args()

if (results.action == PARSER_ACQUIRE and results.Time <= 0):
    parser.error('The acquisition time value is incorrect')
if (results.action == PARSER_DOWNLOAD and results.FileName == None):
    parser.error('The name of the file to download has to be specified')
if (results.action == PARSER_DELETE_MEMORY and results.FileName == None):
    parser.error('The name of the file to delete has to be specified')
return results

def main():
    par=Parser()
    par.FileName, Files2Transfer, PacketsNumber=VerifyFileName(par.FileName, par.action)
    par.Path=VerifyPath(par.Path)
    # print(par.FileName)
    # print(par.Path)
    if Files2Transfer == 0:
        return
    global freq
    freq=par.Fs # Read sampling frequency from user input
    TCP_IP=par.TCP_IP
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(4.0)
    StartConnection(s, TCP_IP, TCP_PORT)
    s.setblocking(0)
    print('Connected with: '+TCP_IP)
    mes=UserInput(par.action,par.Time, par.FileName, par.IEPE, par.Fs, par.Channels, PacketsNumber)
```
input('Press enter to start...')  # Press to continue
assert s.send(mes) >= 0, print('Failed to send')  #
AssertionError if socket has failed to send
data = recvWithEotFlag(s)
SdInd = data.find(bytes('Sd', 'utf-8'))
EndInd = len(data)  # EndInd = data.find(bytes('EndOfData', 'utf-8'))
DataStr = data.decode('ASCII')
# print(DataStr)  # Debugging

if data[0] == ActionDict['MEASURE']:  # Check if start command was correctly received by server
    print('Application status:')
    # print('-' + DataStr[SdInd:TftpInd-1] + '
' + '-' + DataStr[TftpInd:EndInd])
    print('-' + DataStr[SdInd:SdInd + 15])
    print('Measurement started:
', datetime.datetime.now().strftime(fmt))
    # s.close()  # Should provide FIN flag from client side
    # s2 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # s2.settimeout(6*par.Time)
    # # print('"', s2.gettimeout())  # Verify the timeout of blocking socket
    # s2.bind(('0.0.0.0', TCP_PORT2))
    # s2.listen()
    # conn, addr = s2.accept()
    closeWithEotFlag(s, 1)
    s.close()  # Should provide FIN flag from client side
    # s2.close()
    print('Closing')
elif data[0] == ActionDict['MEASURE_DOWNLOAD']:  # Check if start command was correctly received by server
    print('Application status:')
    # print('-' + DataStr[SdInd:TftpInd-1] + '
' + '-' + DataStr[TftpInd:EndInd])
    print('-' + DataStr[SdInd:SdInd + 15])
print('Measurement started:
', datetime.datetime.now().strftime(fmt))

# s.close()## Should provide FIN flag from client side
# s2 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# s2.settimeout(5*par.Time)
# print("","s2.gettimeout()")## Verify the timeout of
blocking socket
# s2.bind(('0.0.0.0', TCP_PORT2))
# s2.listen()
# conn,addr = s2.accept()
CloseWithEotFlag(s,0)

FileCounter=1## For 'MEASURE_DOWNLOAD' action there is only
one file allowed
Flag=FileTransfer(TCP_IP,s,par FileName,FileCounter,1,par.
Path)## Use 1 as ActionQualifier
if Flag == 1:
    Convert=input('Do you want to process the data? (y/n)\n'
)
    if Convert == 'y':
        Conv=DataConversion(Name =par FileName, Path =par.
Path, freq=freq)
        Conv(ConvertData())
        # s2.close()
        s.close()## Should provide FIN flag from client side
        print('Closing')
    elif data[0]== ActionDict[‘CHECK_MEMORY’]:
        print(‘Application status:’)
        # print(‘-’+DataStr[SdInd:TftpInd-1]+’\n’+ ‘-’ + DataStr[
TftpInd:EndInd])
        print(‘-’+DataStr[SdInd:SdInd+15])
        preamble='Files stored in the remote memory:
    FileName=RecvWithEotFlagPrint(s,preamble)
    FileNameAscii = FileName.decode('ASCII')
    # PrintNames(FileNameAscii)
    print(FileNameAscii)
    print('Closing')
# Should provide FIN flag from client side

```python
def FileTransfer(TCP_IP, s, par.FileName, Counter, 0, par.Path):
    if Counter <= FileCounter:
        Flag = FileTransfer(TCP_IP, s, par.FileName, Counter, 0, par.Path)
        if Flag == 0:
            print('Warning: file transfer error')
        print('Closing')
        s.close()  # Should provide FIN flag from client side
        return
```

```python
elif Files2Transfer > 1:
    print('Downloading file %d of %d' % (Counter, Files2Transfer))
    for Counter in range(FileCounter, Files2Transfer + 1):
        print('Downloading file %d of %d' % (Counter, Files2Transfer))
        Flag = FileTransfer(TCP_IP, s, par.FileName, Counter, 0, par.Path)
        if Flag == 0:
            print('Warning: file transfer error')
        print('Closing')
        s.close()  # Should provide FIN flag from client side
        return
```

```python
elif Files2Transfer == 1:
    print('Downloading file %d of %d' % (FileCounter, Files2Transfer))
    Flag = FileTransfer(TCP_IP, s, par.FileName, FileCounter, 0, par.Path)
    if Flag == 1:
        Convert = input('Do you want to process the data? (y/n)')
        if Convert == 'y':
            Conv = DataConversion(Name=par.FileName, Path=par.Path, freq=freq)
            Conv.ConvertData()
            print('Closing')
    s.close()  # Should provide FIN flag from client side
```

```python
elif data[0] == ActionDict['DELETE']:
```
print('Application status:
 # print('-'+DataStr[SdInd:TftpInd-1]+'
'+ '-' + DataStr[TftpInd:EndInd])
print('-'+DataStr[SdInd:SdInd+15])
DeletedFiles=RecvWithEotFlag(s)
DeletedFilesAscii = DeletedFiles.decode('ASCII')
print(DeletedFilesAscii)
pin('Closing')
s.close()## Shpuld provide FIN flag from client side
elif data[0]==ActionDict['CONF_TRIGGER']:
    print('Configuring automatic measurements')
    print('Application status:
 # print('-'+DataStr[SdInd:TftpInd-1]+'
'+ '-' + DataStr[TftpInd:EndInd])
print('-'+DataStr[SdInd:SdInd+15])
print('Closing')
s.close()## Shpuld provide FIN flag from client side
elif data[0]==ActionDict['CHECK_CONF_TRIGGER']:
    print('Verifying automatic measurements configuration')
    print('Application status:
 # print('-'+DataStr[SdInd:TftpInd-1]+'
'+ '-' + DataStr[TftpInd:EndInd])
print('-'+DataStr[SdInd:SdInd+15])
    Configuration=RecvWithEotFlag(s)
    ChInd=Configuration.find(bytes('CH:','utf-8'))
    FreqInd=Configuration.find(bytes('FREQ:','utf-8'))
    BuffInd=Configuration.find(bytes('TIME:','utf-8'))
    TempStr1=' Channels selected:
    TempInd=0
    ChannelsList=Configuration[ChInd+3:ChInd+7]
    ChannelsListInt=[int(s) for s in ChannelsList]#Revert the channel labels
    for Ch in range(len(ChannelsListInt)):
        if ChannelsListInt[Ch] == 1:
            TempStr1=TempStr1+str(Ch+1)+',
    TempInd+=1
Buffer=int(Configuration[BuffInd+5:BuffInd+15])  # 10 bits are designated to carry this

# print(Buffer)  # Debugging

Time=round(Buffer*139/int(FreqKeys[FreqValues.index(Configuration[FreqInd+5]))])

print('Frequeney:'+FreqKeys[FreqValues.index(Configuration[FreqInd+5])+ 'Hz']

print(TempStr1)

print('Time:'+str(Time))

print('Closing')
s.close()  # Should provide FIN flag from client side

elif data[0]==ActionDict['CHECK_IEPE']:
    print('Application status: ')
    print('-'+DataStr[SdInd:SdInd+15])
    FileName=RecvWithEotFlag(s)
    FileNameAscii = FileName.decode('ASCII')
    print(FileNameAscii)
    print('Closing')
s.close()  # Should provide FIN flag from client side

elif data[0]==ActionDict['ERROR']:  # Error flag
    print('Error ...
Application status: ')
    # print('-'+DataStr[SdInd:TftpInd-1]+'
'+ '-' + DataStr[TftpInd:EndInd])
    print('-'+DataStr[SdInd:SdInd+15])
s.close()  # Should provide FIN flag from client side

elif data[0]==ActionDict['STOP']:  # Stop flag
    if EndInd==1:
        print('Connection closed
    else:
        print('-'+DataStr[1:len(DataStr)])
        print('Connection closed
    s.close()  # Should provide FIN flag from client side

###
B.2 Wired Platform Client Software

B.2.1 Software Working with Data Streaming Firmware

```python
# -*- coding : utf-8 -*-
# ""
# Created on Tue Aug 28 22:03:50 2018
# @author : Lukasz
# Features:
# - This code implements connection over tcp socket. It serves as a
#   client side.
# - Channels: 6

#!/usr/bin/env python

import socket
import datetime
import os.path
import csv

#IMPORTANT TO CHOOSE analyze_measurement file that corresponds to MCU
# configuration
#, formatting of most significant byte of ADC result:
#-data_arrays_update_08_2018 : MS byte converted on the PC
#-analyze_data_arrays_update_08_2018_MSB_at_MCU : MS byte converted
#   at MCU side
#-16 BIT RESOLUTION!!

fmt = '%Y_%m_%d %H:%M:%S' # ex. 20110104172008 -> Jan. 04, 2011
now_str = datetime.datetime.now().strftime(fmt)
```

260
```python
##
index=0
stop=0

# freq=31249#Select right frequency
# freq=20833#Select right frequency
# freq=7812*2#Select right frequency
freq=10416#Select right frequency
# freq=7812#Select right frequency
data_frame=704##Actual size minus start frame
###Set the server IP address and TCP port:
TCP_IP = '192.168.1.4'
# TCP_IP = '192.168.1.209'
TCP_PORT = 50000
BUFFER_SIZE = 200

###MESSAGES:
MESSAGE2 = bytes([101])##Start message
MESSAGE3 = bytes([103])##Stop message
###
str=b'New Connection'
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(10.0)
s.connect((TCP_IP, TCP_PORT))

##Send new connection string and print the echoed reply
s.send(str)
data = s.recv(BUFFER_SIZE)
d = data.decode('ASCII')
print(d+'. Connected with:'+TCP_IP)

###Define file path:
#save_path = 'C:/Users/Lukasz/Dropbox (MIT)/MIT/Prof. Leeb/Ti/
CCS7_rtos/
save_path = 'C:/Users/
time_date = now_str
```
# Input file name from keyboard
name_of_file = input('Type file name:\n')
Name = os.path.join(save_path, name_of_file+'\.txt')
file = open(Name, 'w', newline='')
filewriter = csv.writer(file)
# Write date and time for reference
filewriter.writerow([time_date])
filewriter.writerow([‘Byte:’, ‘Data:’])

meas_time=int(input(‘Estimated measurement time in [s]:’),10)## Convert str to int, base 10
num_of_words=8## Number of words (16 bit) send per one sampling
interval. 8 for 6ch (2 status, 6 channels).
buff=meas_time*freq//(data_frame//(num_of_words))## Number of data
buffers needed to store desired amount of data
print(‘Buffer size:’,buff)## Print number of ethernet frames to
transfer

## Below start command is concatenated with the desired number of
buffer frames (assuming buffer will be encoded in 4 bytes!!)
mes=bytearray(MESSAGE2)
mes_temp=buff.to_bytes(4, byteorder=’big’, signed=True)
mes+=mes_temp

input(‘Press enter to start\n’)
s.send(mes)## Send start command plus desired measurement time
data = s.recv(BUFFER_SIZE)## Read received data
i = int.from_bytes(data, byteorder=’big’)## Convert bytes to int

if data[0]==101:## Check if start command was correctly received by
server
    print(‘Transmission started:\n’,datetime.datetime.now().strftime
    (fmt))
BUFFER_SIZE = 20000

### Loop waiting for frames:
while 1:
    data = s.recv(BUFFER_SIZE)
    if not data: ##If empty frame is received break and close the
c     file. close()  
    #     print('Sth went wrong')
    #     s.send(MESSAGE3)#Send stop command
    break
    i=0
    for value in data:
        filewriter.writerow([i, value])
        i=i+1
    #     print(index)
    index=index+1
### End of loop waiting for frames

s.close()##Shpuld frovide FIN flag from client side

print('Connection closed:
',datetime.datetime.now().strftime(fmt),'
\n')
print('Processing data...')

B.2.2 Software Working with TFTP Server Firmware

# -*- coding: utf-8 -*-
# ""
# Created on Tue Aug 28 22:03:50 2018
# @author: Lukasz
# Features:
# - This code implements connection over tcp socket. It serves as a
# client side.
# - TFT server: yes
# - Channels: 6
import tftpy2
import os
import socket
import datetime
import time

""
TCP PART
""
fmt = '%Y_%m_%d %H:%M:%S' # ex. 20110104172008 -> Jan. 04, 2011 5:20:08pm
now_str = datetime.datetime.now().strftime(fmt)
###
index=0
stop=0
freq=62498 # Select right frequency
# freq=31249 # Select right frequency
# freq=20833 # Select right frequency
# freq=int(7812*1.33) # Select right frequency
# freq=7812 # Select right frequency
# freq=10416 # Select right frequency
# freq=7812*2 # Select right frequency
# freq=int(2*7812*1.33) # Select right frequency
# data_frame=695##888
data_frame=704 # 8 channels (6+2 status) * 88, 16bit resolution
### Set the server IP address and TCP port:
TCP_IP = '192.168.1.4' # no-OS
TCP_PORT = 50000
BUFFER_SIZE = 200

###MESSAGES:
MESSAGE2 = bytes([101]) # Start message
MESSAGE3 = bytes([103]) # Stop message
###
str = b'New Connection'
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(10.0)
s.connect((TCP_IP, TCP_PORT))

# Send new connection string and print the echoed reply
s.send(str)
data = s.recv(BUFFER_SIZE)
d = data.decode('ASCII')
print(d + '. Connected with:' + TCP_IP)

meas_time = int(input('Estimated measurement time in [s]:'), 10)## Convert str to int, base 10
num_of_words = 8; ## Number of words (16-bit) send per one sampling interval. 8 for 6ch (2 status, 6 channels).
buff = meas_time * freq // (data_frame // (num_of_words))## Number of data buffers needed to store desired amount of data
print('Buffer size:', buff)## Print number of ethernet frames to transfer

## Below start command is concatenated with the desired number of buffer frames (assuming buffer will be encoded in 4 bytes!!)
mes = bytearray(MESSAGE2)
mes_temp = buff.to_bytes(4, byteorder='big', signed=True)
mes += mes_temp

input('Press enter to start
')
s.send(mes)## Send start command plus desired measurement time
data = s.recv(BUFFER_SIZE)## Read received data
i = int.from_bytes(data, byteorder='big')## Convert bytes to int

if data[0] == 101:## Check if start command was correctly received by server
    print('Transmission started:
', datetime.datetime.now().strftime(fmt))
### Loop waiting for frames:

```python
while 1:
    data = s.recv(BUFFER_SIZE)

    if not data:  # If empty frame is received break and close the connection from client side
        break;

s.close()  # Should provide FIN flag from client side
#filewriter.writerow(["End"])  
```

```python
print('Connection closed:
', datetime.datetime.now().strftime(fmt),'
')

""
TFTP PART, modified tftp protocol (non 512 byte blocks send over UDP)
""
```

```python
# input('Press enter to download file\n')
time.sleep(4)
# input('Press enter to download file\n')
print('Downloading started\n')
save_path = 'C:/Users/'
file='tftp_test'
#Name = os.path.join(save_path,file + ".txt")
Name = os.path.join(save_path,file)
client = tftpy2.TftpClient(TCP_IP, 69)
client.download('Data',Name)
```

```python
print('Downloading finished\n')
```

B.3 Matlab Conversion Software

B.3.1 Wireless Platform
function [RetStruct] = ConvertDataFunFlexibleCh16bit(File,Gain,Vref)

% ConvertDataFun Provide Converted floating point measurement
% Converts data assuming 16 bit resolution and 4 channels:
% - File is a file location, string
% - Gain is the gain of the analog front-end
% - Vref is the reference source of the ADC in [V]
% - Data is a matrix which rows are status word
% and channels 1 to 4
resolution = 2*Vref/Gain/2^16; % In 16 bit resolution mode
name = File;

data_frame = 695; % Number of data chunk, without check word
start_fr_len = 8; % The check word len
start_fr = [1,1,0,0,0,0,1,1,1,1,1,0,0,0,0]; % Check word
start_fr = start_fr';
ByteRes = 2; % Number of bytes per word

% Get the data and convert to signed representation
fid = fopen(name);
if fid == -1
disp('Error');
end
numLines = 3; % First lines with file specification

% The file specification looks as follows:
% Channels count: 2
% Channels selected: 2 3
% Frequency: 15624 Hz
%}
FileInfo = cell(numLines,1);
IndInfo = zeros(numLines,1);
for i = 1:numLines
    FileInfo(i) = {fgetl(fid)};
    IndInfo(i) = strfind(FileInfo{i},": ");
end
ChannelsCount = str2double(FileInfo{1}(IndInfo(1)+1:end));
num_of_words = ChannelsCount + 1; % Status word plus channels.
data_frame = (ChannelsCount + 1) * 139;  % 4 channels plus status -> 5 * 139 = 695
ChannelsSelectionStr = FileInfo{2}(IndInfo(2) + 1: end);
ChannelsSelection = zeros(ChannelsCount, 1);
for i = 1:1:ChannelsCount
    ChannelsSelection(i) = str2double(ChannelsSelectionStr(i + (i - 1)));
end
fmt = ['Selected channels:' repmat(' %1d', 1, numel(ChannelsSelection)) ' \n'];
fprintf(fmt, ChannelsSelection)
f = str2double(FileInfo{3}(IndInfo(3) + 1: end - 2));  % Sampling frequency
datacell = textscan(fid, '%d', 'HeaderLines', 1);  % Assumes specific format of received data stored in txt file
M = datacell{1}(1: end - 1);
fclose(fid);
dim = max(size(M));
num_of_frames = floor(dim / (data_frame + start_fr_len));
num_of_FIFO_per_frame = floor(data_frame / num_of_words);
short = zeros(1, floor(num_of_frames * data_frame / ByteRes));  % Allocate memory
% Convert 8 bit data transferred via ethernet to 16 bit values (variable short), there is a start frame (check word) at the beginning of each data frame
j = 1;
l = 1;
T = 0;
for i = 1:1:dim - 15
    if (M(i:i + 15) == start_fr)  % Look for the beginning of new data frame
        if (M(i + 16) ~= 0 & M(i + 17) ~= 34)
            disp(M(i + 16))
            disp(M(i + 17))
        end
        r(j) = fix(i / ((data_frame + start_fr_len) * 2));
        mod(i, ((data_frame + start_fr_len) * 2));
        for k = 1:1:1: data_frame
             if((r(j) * (data_frame * 2) + (r(j) + 1) * start_fr_len * 2 + k * 2) <= dim
                  ...
short(l)=M(r(j))*(data_frame*2)+(r(j)+1)*start_fr_len*2+k*2-1)+M(r(j))*(data_frame*2)+(r(j)+1)*start_fr_len*2+k*2) *2^-8;

if(mod(k, num_of_words) == 1)
    if((short(l) ~= 8704) && (short(l) ~= 8720))%% Debugging
disp(l)%% Debugging
disp(r(j)*(data_frame*2)+(r(j)+1)*start_fr_len*2+k*2-1))%% Debugging
    end%% Debugging
end
l=l+1;
end
end
j=j+1;
end
dim_short=max(size(short));

Status=zeros(1,floor((dim_short-num_of_words)/num_of_words));
Ch=zeros(ChannelsCount,floor((dim_short-num_of_words)/num_of_words) +1);
Chs=zeros(ChannelsCount,floor((dim_short-num_of_words)/num_of_words) +1);

%%
% Vector short contains data as received from ADS131. If we have 16
% bit
% resolution, one MCU FIFO word is one result without further
% conversion.
k=0;
for i=1:num_of_words:(dim_short-num_of_words)
k=k+1;
    %-For 24 bit resolution--
%     status(k)=short(i+1);
    % adc1(k)=short(i+3)+(bitand(short(i+2),255)) *2^-16;
    % adc2(k)=short(i+4) *2^-8+bitshift(bitand(short(i+5),65280)
\[
\text{adc3}(k) = \text{short}(i + 6) + (\text{bitand}(\text{short}(i + 5), 255)) \times 2^{16};
\]
\[
\text{adc4}(k) = \text{short}(i + 7) \times 2^{8} + \text{bitshift}(\text{bitand}(\text{short}(i + 8), 65280), -8);
\]
\[
\text{---For 16 bit resolution---}
\]
\[
\text{Status}(k) = \text{short}(i);
\]
\[
\text{for } j = 1:1: \text{ChannelsCount}
\]
\[
\text{Ch}(j, k) = \text{short}(i + j);
\]
\[
\text{end}
\]
\[
\text{for } i = 1:1: \text{ChannelsCount}
\]
\[
\text{Chs}(i, :) = \text{int32}((\text{uint16}(\text{Ch}(i, :)), \text{'int16'})); \text{Get 2's complementary signed 16 bit int and cast 32 bit type}
\]
\[
\text{end}
\]
\[
\text{Chf} = \text{single}(\text{Chs}) \times \text{resolution}; \text{Convert to "V"}
\]
\[
\text{RetStruct.X} = [\text{Status}; \text{Chf}];
\]
\[
\text{RetStruct.Ch} = \text{ChannelsSelection};
\]
\[
\text{RetStruct.Fs} = f;
\]
\[
\text{end}
\]

**B.3.2 Wired Platform with Data Streaming Firmware**

```matlab
\%%
clc;
clear all;
close all;

\% f = 31300; \% Select right frequency
\% f = 20833; \% Select right frequency
\% f = 7812.5 \times 1.33; \% Select right frequency
\% f = 7812.5; \% Select right frequency
\% f = 10416; \% Select right frequency
\% f = 7812.5 \times 2; \% Select right frequency
\text{Vref} = 2.4991; \% Reference voltage on board
\text{start_fr} = [1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0];
\text{start_fr} = \text{start_fr'};
```
%% Read from file

% name = 'C:\Users\100Z_50Y_1000RPM_bad';
fid = fopen('C:\Users\new.txt');  %<- Set the right name!
if fid == -1
    disp('Error');
end

datacell = textscan(fid, '%d', 'HeaderLines', 2);  % Assumes specific format of received data stored in txt file
M = datacell{2};
fclose(fid);
dim = max(size(M));
data_frame = 704;
start_frame = 8;
% FIFO = 8;
num_of_words = 8;  % Number of words (16 bit) send per one sampling interval. 8 for 6ch (2 status, 6 channels).
num_of_frames = floor(dim / (data_frame + start_frame));
num_of_FIFO_per_frame = floor(data_frame / num_of_words);

%% Convert 8 bit data transferred via ethernet to 16 bit values (variable short), there is a start frame at the beginning of each data frame
j = 1;
l = 1;
T = 0;
for i = 1:dim - 15
    if (M(i:i+15) == start_frame)  % Look for the beginning of new data frame
        r(j) = fix(i / (data_frame + start_frame) * 2));
        mod(i, (data_frame + start_frame) * 2));
        for k = 1:1:15
            if ((r(j) * (data_frame * 2) + (r(j) + 1) * start_frame * 2 + k * 2) <= dim)
                short(l) = M(r(j) * (data_frame * 2) + (r(j) + 1) * start_frame * 2 + k * 2 - 1) + M(r(j) * (data_frame * 2) + (r(j) + 1) * start_frame * 2 + k * 2) * 2^-8;
                l = l + 1;
            end
        end
        j = j + 1;
    end
end
dim_short = max(size(short));

status = zeros(1, floor((dim_short - num_of_words)/num_of_words));
ad1 = zeros(1, floor((dim_short - num_of_words)/num_of_words));
ad2 = zeros(1, floor((dim_short - num_of_words)/num_of_words));
ad3 = zeros(1, floor((dim_short - num_of_words)/num_of_words));
ad4 = zeros(1, floor((dim_short - num_of_words)/num_of_words));
status2 = zeros(1, floor((dim_short - num_of_words)/num_of_words));
ad5 = zeros(1, floor((dim_short - num_of_words)/num_of_words));
ad6 = zeros(1, floor((dim_short - num_of_words)/num_of_words));

k = 0;
for i = 1:num_of_words:(dim_short - num_of_words)
    k = k + 1;
    extr = bitand(short(i), 255);
    status(k) = short(i+1);
    adc1(k) = short(i+3) + (bitand(short(i+2), 255)) * 2^16;
    adc2(k) = short(i+4) * 2^8 + bitshift(bitand(short(i+5), 65280), 272);
adc3(k) = short(i +6) + (bitand(short(i +5), 255)) * 2^16;

adc4(k) = short(i +7) * 2^8 + bitshift(bitand(short(i +8), 65280), -8);

status(k) = short(i);

adc1(k) = short(i +1);
adc2(k) = short(i +2);
adc3(k) = short(i +3);
adc4(k) = short(i +4);
status2(k) = short(i +5);
adc5(k) = short(i +6);
adc6(k) = short(i +7);

end

%%%%
% Accordign to the ADS131’s datasheet convert results into positive and
% negative values
Cell = { status, status2, adc1, adc2, adc3, adc4, adc5, adc6};
for j = 3:1:max(size(Cell))
    for i = 1:1:k
        if(Cell{1,j}(i) >= 32768) % Negative converions
            Cell{1,j}(i) = (Cell{1,j}(i) - 65536);
            Cell{1,j}(i) = double(Cell{1,j}(i)) / 32768;
        else
            Cell{1,j}(i) = double(Cell{1,j}(i)) / 32768*1; % Positive range has one bit lower range
        end
    end
end

%%
time_vector = 0:1/f:(k -1)/f;
%% Duration of measurement set
time = k/f
figure;
subplot(1,2,1)
plot(Cell{1,1},'-sr','MarkerSize',2)
ylabel('Status Word')% for i=2:1:k-350
subplot(1,2,2)
plot(Cell{1,2},'-sr','MarkerSize',2)
ylabel('Status Word')% for i=2:1:k-350

figure;
subplot(3,2,1)
plot(time_vector,Cell{1,3}*Vref,'-sr','MarkerSize',0.5)
if max(Cell{1,3})~=0
    ylim([-1.5*Vref*abs(max(Cell{1,3})) 1.5*Vref*abs(max(Cell{1,3}))])
end
xlabel('time, [s]')% for i=2:1:k-350
ylabel('Normalized measurement')
subplot(3,2,2)
% scrollplot(plot(Cell{1,3},'-sr','MarkerSize',2));
plot(time_vector,Cell{1,4}*Vref,'-sr','MarkerSize',0.5);
if max(Cell{1,4})~=0
    ylim([-1.5*Vref*abs(max(Cell{1,4})) 1.5*Vref*abs(max(Cell{1,4}))])
end
xlabel('time, [s]')% for i=2:1:k-350
ylabel('Normalized measurement')
subplot(3,2,3)
plot(time_vector,Cell{1,5}*Vref,'-sr','MarkerSize',0.5)
if max(Cell{1,5})~=0
    ylim([-1.5*Vref*abs(max(Cell{1,5})) 1.5*Vref*abs(max(Cell{1,5}))])
end
xlabel('time, [s]')% for i=2:1:k-350
ylabel('Normalized measurement')
subplot(3,2,4)
plot(time_vector,Cell{1,6}*Vref,'-sr','MarkerSize',0.5)
if max(Cell{1,6})~=0
    ylim([-1.5*Vref*abs(max(Cell{1,6})) 1.5*Vref*abs(max(Cell{1,6}))])
end
xlabel('time, [s]')% for i=2:1:k-350
subplot(3,2,5)
% B.3.3 Wired Platform with TFTP Server Firmware

%%
c1c;
clear all;
close all;

% f = 31300; % Select right frequency
% f = 20833; % Select right frequency
% f = 7812.5 * 1.33; % Select right frequency
% f = 7812.5; % Select right frequency
% f = 10416; % Select right frequency
% f = 7812.5 * 2; % Select right frequency
Vref = 2.4991; % Reference voltage on-board
start_fr = [1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0];
start_fr = start_fr';

%% Read from file, binary file
name = 'C:\Users\tftp_test';
fid = fopen(name);
if fid == -1
disp('Error');
end
M = fread(fid);
fclose(fid);
dim = max(size(M));
data_frame = 704; \% 695 for 4 channels with 16 bit, 704 for 16 bit 6 channels
start_frame = 8;
num_of_words = 8; \% Number of words (16 bit) send per one sampling interval. 8 for 6ch (2 status, 6 channels).
num_of_frames = floor(dim/(data_frame+start_frame));
num_of_FIFO_per_frame = floor(data_frame/num_of_words);
\%
\% Convert 8 bit data transferred via ethernet to 16 bit values (variable short), there is a start frame at the beginning of each data frame
j = 1;
l = 1;
T = 0;
for i = 1:1:dim-15
  if (M(i:i+15) == start_fr) \% Look for the beginning of new data frame
    \% (i-1)/1406
    r(j) = fix(i/((data_frame+start_frame)*2));
    mod(i,((data_frame+start_frame)*2));
    for k = 1:1:data_frame
    if ((r(j) *(data_frame*2) + (r(j) + 1) * start_frame*2 + k*2) <= dim)
      short(l) = M(r(j) *(data_frame*2) + (r(j) + 1) * start_frame*2 + k*2-1) + M(r(j) *(data_frame*2) + (r(j)+1) * start_frame*2 + k*2) * 2^8;
      l = l + 1;
      end
    end
    j = j + 1;
  end
end
dim_short = max(size(short));

status = zeros(1, floor((dim_short-num_of_words)/num_of_words));
adc1 = zeros(1, floor((dim_short-num_of_words)/num_of_words));
adc2 = zeros(1, floor((dim_short-num_of_words)/num_of_words));
adc3 = zeros(1, floor((dim_short-num_of_words)/num_of_words));
adc4 = zeros(1, floor((dim_short - num_of_words)/num_of_words));
status2 = zeros(1, floor((dim_short - num_of_words)/num_of_words));
adc5 = zeros(1, floor((dim_short - num_of_words)/num_of_words));
adc6 = zeros(1, floor((dim_short - num_of_words)/num_of_words));

% status = zeros(1, floor((num_of_frames * num_of_FIFO_per_frame / 2) - 1));
% adc1 = zeros(1, floor((num_of_frames * num_of_FIFO_per_frame / 2) - 1));
% adc2 = zeros(1, floor((num_of_frames * num_of_FIFO_per_frame / 2) - 1));
% adc3 = zeros(1, floor((num_of_frames * num_of_FIFO_per_frame / 2) - 1));
% adc4 = zeros(1, floor((num_of_frames * num_of_FIFO_per_frame / 2) - 1));
% status2 = zeros(1, floor((num_of_frames * num_of_FIFO_per_frame / 2) - 1));
% adc5 = zeros(1, floor((num_of_frames * num_of_FIFO_per_frame / 2) - 1));
% adc6 = zeros(1, floor((num_of_frames * num_of_FIFO_per_frame / 2) - 1));
%
% Vector short contains data as received from ADS131. Since we have 16 bit resolution, one MCU FIFO word is one result without further conversion.

k = 0;
for i = 1: num_of_words : (dim_short - num_of_words)
    k = k + 1;
    extr = bitand(short(i), 255);

    status(k) = short(i + 1);
    adc1(k) = short(i + 3) + (bitand(short(i + 2), 255)) * 2^16;
    adc2(k) = short(i + 4) * 2^8 + bitshift(bitand(short(i + 5), 65280)
                                                      ,-8);
    adc3(k) = short(i + 6) + (bitand(short(i + 5), 255)) * 2^16;
    adc4(k) = short(i + 7) * 2^8 + bitshift(bitand(short(i + 8), 65280)
                                                      ,-8);
    status(k) = short(i);    adc1(k) = short(i + 1);    adc2(k) = short(i + 2);    adc3(k) = short(i + 3);    adc4(k) = short(i + 4);
status2(k)=short(i+5);
adc5(k)=short(i+6);
adc6(k)=short(i+7);
end

%%
%% Accordign to the ADS131’s datasheet convert results into positive
% and
%% negative values
Cell={status,status2,adc1,adc2,adc3,adc4,adc5,adc6};
for j=3:1:max(size(Cell))
   for i=1:1:k
      if(Cell{1,j}(i)>=32768) % Negative converions
         Cell{1,j}(i)=(Cell{1,j}(i)-65536);
         Cell{1,j}(i)=double(Cell{1,j}(i))/32768;
      else
         Cell{1,j}(i)=double(Cell{1,j}(i))/32768*1; % Positive range has one bit lower range
      end
   end
end

%%
time_vector=0:1/f:(k-1)/f;
%% Duration of measurement set
time=k/f
figure;
subplot(1,2,1)
plot(Cell{1,1},'-sr','MarkerSize',2)
ylabel('Status Word') % for i=2:1:k-350
subplot(1,2,2)
plot(Cell{1,2},'-sr','MarkerSize',2)
ylabel('Status Word') % for i=2:1:k-350
figure;
plot(time_vector,Cell{1,3}*Vref,'-sr','MarkerSize',0.5)
if max(Cell{1,3}) ~= 0
    ylim([-1.5*Vref*abs(max(Cell{1,3})) 1.5*Vref*abs(max(Cell{1,3})))
end
xlabel('time, [s]')
% ylabel('Normalized measurement')
subplot(3,2,2)
% scrollplot(plot(Cell{1,3},'sr','MarkerSize',2));
plot(time_vector,Cell{1,4}*Vref,'-sr','MarkerSize',0.5);
if max(Cell{1,4}) ~= 0
    ylim([-1.5*Vref*abs(max(Cell{1,4}))) 1.5*Vref*abs(max(Cell{1,4})))
end
xlabel('time, [s]')
% ylabel('Normalized measurement')
subplot(3,2,3)
plot(time_vector,Cell{1,5}*Vref,'-sr','MarkerSize',0.5);
if max(Cell{1,5}) ~= 0
    ylim([-1.5*Vref*abs(max(Cell{1,5}))) 1.5*Vref*abs(max(Cell{1,5})))
end
xlabel('time, [s]')
% ylabel('Normalized measurement')
subplot(3,2,4)
plot(time_vector,Cell{1,6}*Vref,'-sr','MarkerSize',0.5);
if max(Cell{1,6}) ~= 0
    ylim([-1.5*Vref*abs(max(Cell{1,6}))) 1.5*Vref*abs(max(Cell{1,6})))
end
xlabel('time, [s]')
% ylabel('Normalized measurement')
subplot(3,2,5)
plot(time_vector,Cell{1,7}*Vref,'-sr','MarkerSize',0.5);
if max(Cell{1,7}) ~= 0
    ylim([-1.5*Vref*abs(max(Cell{1,7}))) 1.5*Vref*abs(max(Cell{1,7})))
end
xlabel('time, [s]')
% ylabel('Normalized measurement')
subplot(3,2,6)
plot(time_vector,Cell{1,8}*Vref,'-sr','MarkerSize',0.5);
if max(Cell{1,8}) ~= 0

279
ylim([-1.5*Vref*abs(max(Cell{1,8})) 1.5*Vref*abs(max(Cell{1,8}))])
end
xlabel('time, [s]')
% for i=2:1:k-350
Appendix C

Schematics and Technical Documents

This appendix provides circuit schematics and technical documentation related to the hardware presented in this thesis.

C.1 Wireless Platform Schematics

C.1.1 Top PCB
Channels 1-4 go with ADS131A04

ADC_BUS is used for communication with ADC
Past this divider there is a gain of 2 (at least) from the amplifier and 2 from differential amplifier.
To match +/-10V input range into the 2.048V of the reference voltage we set divider to 6/126.

\[ \text{Gain} = \frac{2 \times 2}{6/126} \]

At least 2.048V of the reference voltage is required for proper operation.
Used to configure ADC
M0 float is such slave
M0 to Vdd is asynchronous

Used to configure ADC
M1 float is 16bit res
M1 to GND is 24bit

XTAL2 floating, master
clock mode

DOUT goes to MISO or Rx
pin
DIN goes to MOSI or Tx pin

Charge pump enabled, VNCPE
bit enables it.
Inout voltage matching: +/-5V to +/-2.048V. Clamping diode protection.
Buffer to decouple sensor impedance
Low pass filter. Fc=23.4kHz
S&H circuit accumulator

Title:
Number: PIC2901
Revision: COC29
Size: PIC2902

Date: 4/9/2021
File: C:\Users\AnalogInputs.SchDoc
Drawn By:
Analog PS is preferred, however, analog 3.0V would be problematic with digital 3.3V logic. Vo is filtered.
Connect to pin 91 of the MCU to use TSCCP0!

High-pass filter, 8kHz

Low-pass filter, 17kHz

Low-pass filter, 17kHz

Gain stage

Hysteresis comp.
Add C to ground, look only at the DC component!
For prototype either 3 V can be used for all analog VCC (then use jumper resistor) or two separate 2.5 V and 3 V planes are available.

+2.5 V and -2.5 V LDO

Linear regulator is bypassed by the 0R resistor. In future regulator circuit may be used.

10 mA load is assumed to be drawn by ADS131 digital part and oscillator.
C.1.2 Bottom PCB
Title
Number
Revision
Size
Date: 4/9/2021
Sheet  of
File: C:\Users\..\TIDM_TM4C129POE_Connector.SchDoc

Debug Header

USB Connector

Switches
Temperature & Humidity Sensors

To share SPI bus there is external CS line to the IC

SSI 3 - CLK  
SSI 2 - CS  
SSI 1 - MOSI  
SSI 0 - MISO

<table>
<thead>
<tr>
<th>SSI 1</th>
<th>2</th>
<th>3</th>
<th>DNP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSI 0</td>
<td></td>
<td></td>
<td>DNP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DNP</th>
<th>89</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>100</th>
<th>101</th>
<th>102</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>103</th>
<th>104</th>
<th>105</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CS</th>
<th>SCL/SPC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SDA/SDI/SDO</td>
</tr>
<tr>
<td>SDO/SA0</td>
<td></td>
</tr>
</tbody>
</table>

U8

R84 10k
R85 10k
R83 0R

I2C 0 - I2C CLK  
I2C 1 - I2C Data

U6

ADDR
SCL
SDA

R86 10F
R87 DNP

CS8 1F
C60 0.01F
C59 1F

VCC_3P3

PIR8301 PIR8302
PIR8401 PIR8402
PIR8501 PIR8502
PIR8601 PIR8602
PIR8701 PIR8702
PIR8801 PIR8802
PIR8901 PIR8902
PIR9001 PIR9002
PIR9101 PIR9102

PIU601 PIRU602
PIU603 PIRU604
PIU605 PIRU606
PIU607 PIRU608
PIU609 PIRU6010
PIU6011 PIRU6012

PIU701 PIRU702
PIU703 PIRU704
PIU705 PIRU706

PIU801 PIRU802
PIU803 PIRU804
PIU805 PIRU806
PIU807 PIRU808
PIU809 PIRU8010
PIU8011 PIRU8012

PIU901 PIRU902
PIU903 PIRU904
PIU905 PIRU906
PIU907 PIRU908
PIU909 PIRU9010
PIU9011 PIRU9012

To share SPI bus there is external CS line to the IC.
Enable SD card. High side switch.
CR2025 battery for Hibernation Module

Ext. crystal for Hibernation Module

WAKE button as WAKE pin or Tamper event

Title

Date: 4/9/2021

Sheet of

File: C:\Users\..\HibernationModule.SchDoc

Drawn By:
C.2  Wired Platform Schematics

C.2.1  Top PCB
Alternative PS instead of battery, back-up option

Provision to use WAKE pin
C.2.2 Bottom PCB
DRDY signal from ADC is included in Tiva_GPIO4, DRDY_IN can be used for other purposes.

Channels 1-4 go with ADS131A04, 5-6 ADS131A02.

SSI0 bus is used for communication with ADC.
Minimum loading of 10% for reliable operation, load resistor
Title

Size | Number | Revision
--- | --- | ---
A | | 

Date: 4/9/2021

File: C:\Users\ADS131A04.SchDoc

Sheet of

Drawn By:
Option to use 5V as AVDD
C.3 IEPE Transducer and Microphone

Table C.1 presents the details of circuit implementation of the proposed IEPE transducer. The differential amplifier in Fig. 4-6b) is realized with an instrumentation amplifier in series with an inverting amplifier and overall gain of 0.5. Component designations follow Fig. 4-6. Table C.2 presents the details of circuit implementation of the IEPE microphone described in this thesis.

Table C.1: IEPE Transducer Circuit Parameters

<table>
<thead>
<tr>
<th>Component</th>
<th>Value:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst. amplifier</td>
<td>AD8226BRMZ-R7</td>
</tr>
<tr>
<td>Inv. amplifier</td>
<td>LTC6255CDC</td>
</tr>
<tr>
<td>Voltage regulator</td>
<td>LT3009IDC-3.3</td>
</tr>
<tr>
<td>Osc. amplifier</td>
<td>LTC6255CDC</td>
</tr>
<tr>
<td>Osc. comparator</td>
<td>TLV3201AIDBVR</td>
</tr>
<tr>
<td>$Q_2$</td>
<td>BSS138W</td>
</tr>
<tr>
<td>$R_b$</td>
<td>5.1 kΩ</td>
</tr>
<tr>
<td>$C_b$</td>
<td>1 μF</td>
</tr>
<tr>
<td>$R_1$</td>
<td>43 kΩ</td>
</tr>
<tr>
<td>$R_2$</td>
<td>21.5 kΩ</td>
</tr>
<tr>
<td>$R_3$</td>
<td>43 kΩ</td>
</tr>
<tr>
<td>$R_4$</td>
<td>43 kΩ</td>
</tr>
<tr>
<td>$R_5$</td>
<td>100 kΩ</td>
</tr>
<tr>
<td>$R_6$</td>
<td>100 kΩ</td>
</tr>
<tr>
<td>$R_7$</td>
<td>43 kΩ</td>
</tr>
<tr>
<td>$C_1$</td>
<td>470 pF</td>
</tr>
<tr>
<td>$R_i$</td>
<td>0.499 kΩ</td>
</tr>
<tr>
<td>$R_x$</td>
<td>1 kΩ</td>
</tr>
<tr>
<td>$R_d$</td>
<td>1 kΩ</td>
</tr>
</tbody>
</table>

Table C.2: IEPE Microphone Circuit Parameters

<table>
<thead>
<tr>
<th>Component</th>
<th>Value:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capsule</td>
<td>POM-3535P-3-R</td>
</tr>
<tr>
<td>$R_1$</td>
<td>6.19 kΩ</td>
</tr>
<tr>
<td>$R_{in}$</td>
<td>0.511 kΩ</td>
</tr>
<tr>
<td>$R_e$</td>
<td>0.422 kΩ</td>
</tr>
</tbody>
</table>
C.4 Case Technical Drawings

This section includes technical drawings for case, i.e., base and cover, and parts list of mechanical components.

Parts list:

- Magnets: neodymium magnet with straight unthreaded hole, 3360K862
- Board standoffs: 11.13 mm long, 2-56, 2056-256-AL-7
- U.FL-to-SMA adapter: CSI-SGFE-100-UFFR
- SMA-to-BNC adapter: 29-3855
- PCB mounting screws: 6.35 mm long, 2-56, PMS 256 0025 PH
- Standoffs mounting screws: 6.35 mm long, 2-56, 91263A119
- Cover mounting screws: 22.23 mm long, 2-56, 92210A086
4xØ1.78 10.00 2-56
3xØ1.78 2.70 2-56

4xØ2.38
82.0°

Ø100.00
4xØ1.78
Ø88.00
4xØ5.21

Base Drawing

Lukasz Huchel 2/10/2021

BaseDrawing

A4

1 / 2

SHEET 1 OF 1
Two unannotated holes on the bottom side can be used for battery holder. These were used in the previous two-channel version of the case.