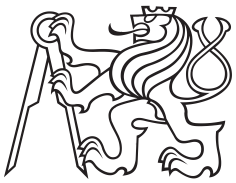
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Measurements

Analysis of the textile fibers unevenness in frequency domain

Ondřej Renza

Supervisor: Ing. Jakub Parák
Field of study: Sensors and Instrumentation
Subfield: Cybernetics and Robotics
May 2016

Acknowledgements

Děkuji.

Declaration

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 15. May 2016

Abstract

To be done.

Keywords: digital signal processing, textile defects, uneveness

Supervisor: Ing. Jakub Parák

Abstrakt

To be done.

Klíčová slova: zpracování digitálního signálu, textilní vady, nerovnoměrnost

Překlad názvu: Analýza periodických vad textilních vláken ve frekvenční oblasti

Contents

Figures

Tables

Chapter 1

Introduction

Textile manufacturing has always been important industry field. A major part of this industry is formed by a process called spinning, where twisting strands of fibers together form yarn. The modern spinners - textile machines, that execute the process of spinning - have been significantly improved and now they reach a high level of automation. This allows not only faster and cheaper production but also more focus on a quality of the produced textile yarn.

The quality of the yarn could devaluate the final product by creating defects, such as rapid changes in color or thickness etc., in the textile material. Even with modern technologies, it is still impossible to produce yarn without any defects. We can't prevent yarn defects by carefully selecting and preprocessing fiber because some defects can be created by spinning process itself. Out of many types of defects, this thesis is focused on the analysis of yarn unevenness (also called yarn irregularity). This is describing yarn with a diameter that is not even along its length, but it is changing its value periodically. We can measure this defect in the form of mass variation per unit length.

Designed system is not aiming to improve the quality of spun yarn, but to monitor quality (specifically unevenness) of produced yarn during the spinning process. Due to this monitoring, it is possible to stop spinning process if a defective yarn is detected. This allows the operator to resolve the issues that caused it, e.g. by replacing spinner fiber source with new one and reconnecting different yarn endings.

The project - described in this thesis - has been made in cooperation with company Rieter CZ s.r.o, who provided the device requirements, critical measurement data, and other important information. The goal of the project was to research and develop an algorithm for analysis and detection of yarn unevenness by using spectrograms and to design an embedded system capable of measuring a diameter of yarn while fulfilling the required time constraints and implement detection algorithm. The system is required to be controlled by ARM M4 microcontroller. Thus, the algorithm has to be implemented in a way that takes in consideration memory and computational limitation of such microcontrollers.

A very important specification was the requirement to analyze a quality of two textile fiber types: the yarn and the sliver. Where sliver is the input

textile fiber for the spinning process and the yarn is its final product. The both of which has significant physical differences. Mainly they differ in size because yarn diameter is usually in a range of micrometers and sliver diameter is in a range from millimeters to centimeters. The diameter of the sliver is measured on combing machine, which precedes the spinning process. This requires different measuring system and filtration processing, therefore, two systems and algorithms were designed for quality analysis of each fiber type. Their core content is equal but there are some major differences, which are described later in this thesis. The most significant difference is that processing of signal representing sliver diameter requires much more advanced techniques of digital signal processing. This is necessary due to a strong presence of periodical artefacts on sliver diameter caused by machine preprocessing. This type of diameter fluctuations has to be distinguished from the actual sliver unevenness, which is a task for complicated digital filtration in a frequency domain.

Chapter 2

Theoretical Introduction

Before the process of software and hardware development could begin, detailed theoretical research had to be done. Topics of the research include textile manufacturing, combing process, spinning process and textile fiber defects to understand better device requirements. Another step of research was focused on digital signal processing techniques that could be used on the project, mainly spectrogram estimation and it's calculation using Fast Fourier Transform, together with possible filtration algorithms. Another research topic was aimed to cover embedded systems and specifically micro-controller usage and it's real-time constraints.

2.1 Textile Engineering

The goal of textile manufacturing is to make fabric from textile fibers, which can be then used for clothes. This process can be separated into several stages:

- Preparatory Processes - prepares the textile fiber for spinning process by blending, carding and combing,
- Spinning - fibers are spun into yarns,
- Knitting or Weaving - yarns becomes fabric,
- Finishing - fabric is transformed into clothes etc.

In regards to the topic of this thesis only two - the spinning and combing processes - are described.

2.1.1 Textile Fibers

Fibers are the basis for all textiles. We distinguished the two main types: natural fibers and synthetic fibers. The natural fibers are:

- Cotton - from the cotton plant,
- Linen - from the flax plant,

- Wool - from sheep and
- Silk - from silkworms.

Examples of widely used synthetic fibers are:

- Viscose - from pine trees and petrochemicals,
- Acrylic, nylon, and polyester - from oil and coal.

The cotton is the most important natural fiber and analysis of the quality in this thesis is aimed specifically at cotton manufacturing. The fibers can have two main forms during the manufacturing process - the sliver and the yarn. The sliver is created by carding the fiber. In this process textile fibers are separated and then joined together into a loose strand of 1 cm to 4 cm in diameter. In the end of textile manufacturing process, a yarn is created. It is a textile fiber with significantly smaller diameter, in comparison to sliver, usually in a range of micrometers [9].

2.1.2 Combing Process

Combing is a preparation process during textile manufacturing. It is sub-part of a cleaning process which precedes the spinning process. Cotton contains a lot of impurities such as dirt, dust, foreign materials, neps and very short fibers. All of these should be eliminated by cleaning process. The combing process removes mainly short fibers and neps in sliver, which helps to produce stronger and cleaner yarn.

Combing is used in a production of medium-fine or fine yarns, where the quality of yarn is important. This quality improvement is at a cost of loss of raw material and high expenses for buying and operating the combing machines.

Comber (see Image 1) consists of three main parts:

- The Feed
- The Nipper
- The Comb

The process of removing the impurities is formed by attaching the input fiber in form of sliver to the feed roller

An input of this process is textile sliver (described in chapter xx).

2.1.3 Spinning Process

The term “spinning” in this context refers to the process that executes conversion of a large quantity of individual unordered fibers of relatively short length into a linear, ordered product of very great lengths by using spinning machines. There are three main methods of executing process of spinning:

- Ring Spinning,

- Rotor Spinning and
- Air-jet Spinning.

All of these systems yields yarn with different structures and properties. Each system has its advantages and limitations in terms of technical feasibility and economic viability.

2.2 Overview of Yarn Quality Sensors

An importance of yarn quality on final product became clear in the 1950s when first electronic yarn quality sensors were invented. Since then many principles were used in the detection of yarn defects - optical, mechanical or even chemical [3].

Sensors of yarn quality are in the textile industry often called *yarn clearer*. This term was created for first such devices because goal of yarn clearers wasn't only to discover possible yarn defects but also to immediately remove them. Today, yarn clearers analyze defects that are much more complex (given their properties such as periodicity etc.) and difficult to be immediately removed. Such defects are invisible to a naked eye on single yarn and they appear only when turned to fabric.

The following overview is concerned with yarn quality sensors that have similar purpose and functions as the developed device. Therefore, only online yarn clearers that check quality of spun yarn on every spinning point in real-time are listed. All of the following devices are also aimed to be used on the yarn spun by a rotor spinners (see 2.1.3).

Uster Quantum 3

Uster Quantum 3 is modern yarn clearer, which provides detection of many types of defects and includes several interesting features:

- full yarn body display,
- foreign matter sensor with multicolored light sources,
- polypropylene detection,
- detection of moiré,
- unevenness calculation CV%,
- IPI classification,
- calculation of hairiness,
- spectrograms calculation.

Uster Quantum 3 has been developed by company Uster Technologies, which is developing yarn clearers for 30 years. Their devices often feature an application of new technologies and they offer high-quality products.

This yarn clearer is also one of the two commercially sold products capable of calculating spectrograms and evaluate yarn unevenness CV%.

2.2.1 State of Art

2.3 Digital Signal Processing

The largest part of work on this project is oriented on digital signal processing (DSP). Usage of modern advanced algorithms from this field allowed designing projected device in the first place. This section covers the most important algorithms that are used in this project.

Digital signal processing is an area of science and engineering that has developed rapidly over the past 40 years as a result of significant advances in digital computer technology. Today, many of the signal processing tasks that were conventionally performed by analog means are now realized by less expensive digital hardware [2].

To perform the processing digitally, there is the need for the conversion between an analog signal and digital signal. This is done by an interface called analog-to-digital (A/D) converter, which yields a digital signal as it's output that is appropriate as an input to the digital processor [2, 1].

2.3.1 Discrete Fourier Transform

To perform frequency analysis on a discrete-time signal $x[n]$, we convert the time-domain sequence to an equivalent frequency-domain representation. This conversion is obtained by Discrete Fourier Transform that can be algebraically formulated as (according to [2, 1]).

Given N consecutive samples $x[n], 0 \leq n \leq N-1$ of a periodic or aperiodic sequence, the N-point Discrete Fourier Transform(DFT) $X[k], 0 \leq k \leq N-1$ is defined by

$$X[k] = \sum_{k=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn}. \tag{2.1}$$

Given N DFT coefficients $X[k], 0 \leq k \leq N-1$, we can recover the N sample values of sequence $x[n], 0 \leq n \leq N-1$ using Inverse Discrete Fourier Transform (IDFT) given by

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}kn}. \tag{2.2}$$

If $x[n]$ has infinite duration, the frequency samples $X[2\pi k/N], k = 0, 1, ..., N-1$ correspond to a periodic sequence $x_p[n]$ of period N, which is an aliased version of $x[n]$. When the sequence $x[n]$ has finite duration of length $L \leq N$, then $x_p[n]$ is simply a periodic repetition of $x[n]$.

The DFT defined in (2.1) can also be rewritten as

$$X[k] = \sum_{k=0}^{N-1} x[n] W_N^{kn},\ k = 0, 1, ..., N-1, \tag{2.3}$$

where

$$W_N^{kn} = e^{-j\frac{2\pi}{N}kn} = \cos(\frac{2\pi kn}{N}) - j\sin(\frac{2\pi kn}{N}),\ 0 \leq k, n \leq N-1 \tag{2.4}$$

The parameters W_N^{kn} are called the twiddle factors [6].

Understanding properties of DFT is critical for application of the transformation to practical problems. List of the main DFT properties contains:

- Linearity,
- Periodicity,
- Complex Conjugate,
- Circular Convolution,
- DFT and the z-transform.

For detailed description of DFT properties see [2, 6].

The operation of selecting a finite number of samples called windowing is equivalent to multiplying the actual sequence $x[n]$ defined in a range $-\infty < n < \infty$, by a finite-length sequence $w[n]$ called window. Using simplest rectangular windowing (truncation) on a signal can cause an effect called *leakage*, which transfers power from frequency bands that contain a large amount of signal power into bands that contain only a little. This may create "false" peaks, peaks at wrong frequencies or changes the amplitude of existing peaks.

Another effect of time-windowing is *smearing*. Which causes a spread of spectrum accordingly to the width of the mainlobe of the window spectrum. This result in loss of resolution [1] .

Therefore, a "good" window should have low-level sidelobes and a narrow mainlobe to minimize both of these effects. There are four most known windows used for time-windowing:

- Rectangular,
- Triangular (or Bartlett),
- Hann,
- Hamming.

Their differences (as shown in image XXX) relays in a different width of mainlobe and peak sidelobe level.

2.3.2 Fast Fourier Transform

Difficulty in using the DFT for practical applications is its high computational requirements. Direct computation of the N-point DFT requires computational cost of N^2. However class of efficient DFT algorithms called *Fast Fourier Transform (FFT)* has computational cost proportional to $Nlog_2N$ [6, 1].

Decimation-in-time FFT algorithms are based in splitting the N-point DFT summation into two summations, that one sum over the even-indexed points of $x[n]$ and another sum over the odd-indexed points of $x[n]$. Therefore, we obtain

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n]W_N^{kn},\ k = 0, 1, ..., N-1 \\ &= \sum_{m=0}^{N/2-1} x[2m]W_N^{k(2m)} + W_N^k \sum_{m=0}^{N/2-1} x[2m+1]W_N^{k(2m)} \end{aligned} \tag{2.5}$$

Dividing sequence $x[n]$ we get two shorter sequences:

$$a[n] = x[2n], \qquad n = 0, 1, ..., N/2-1 \tag{2.6}$$

$$b[n] = x[2n+1], \qquad n = 0, 1, ..., N/2-1 \tag{2.7}$$

Shorter sequences are obtained by *decimating* [1] the sequence $x[n]$, thus, this FFT algorithm is called decimation-in-time. Substituting definitions 2.6 and 2.7 into 2.5 yields

$$A[k] = \sum_{m=0}^{N/2-1} a[m]W_{N/2}^{km}, \qquad k = 0, 1, ..., N/2-1 \tag{2.8}$$

$$B[k] = \sum_{m=0}^{N/2-1} a[m]W_{N/2}^{km}, \qquad k = 0, 1, ..., N/2-1 \tag{2.9}$$

where $A[k]$ and $B[k]$ are $N/2$-point DFTs [1, 2].

Thus, we can calculate N-point DFT $X[k]$ from the $N/2$-point DFTs $A[k]$ and $B[k]$ (2.8, 2.9) using the following merging formulas

$$X[k] = A[k] + W_N^k B[k], \qquad k = 0, 1, ..., N/2-1 \tag{2.10}$$

$$X[k + \frac{N}{2}] = A[k] - W_N^k B[k], \qquad k = 0, 1, ..., N/2-1 \tag{2.11}$$

These formulas (2.10, 2.11) can be applied to any FFT of even length [1].

This procedure is shown in Figure XXX ([1]). The displayed structure in the figure is called the butterfly network. Each butterfly consists of just a single complex multiplication by the twiddle factor W_N^k, one addition and one subtraction.

[1] Decimation of a signal with sampling rate f_s by a integer factor D results in the lower sampling rate $f_s' = f_s/D$.

An example for $N = 8$ is shown in Figure XXX. Each $N/2$-point DFT can be computed by two smaller $N/4$-point DFTs. By repeating the same process, we will obtain a set of two-point DFTs, which is illustrated in Figure XXX [6].

FFT algorithm *decimation-in-frequency* is similar to the decimation-in-time, with important differences, that the decomposition and symmetry relationships are reversed. The bit reversal occurs at the output instead of the input and the order of the output samples $X[k]$ will be rearranged [6].

The FFT algorithms shown in the previous paragraphs can be modified to calculate the inverse FFT (IFFT).

2.3.3 Power spectral density

Energy Spectral Density $S(\omega)$ defined as 2.12

$$S(\omega) = |Y(\omega)|^2 \tag{2.12}$$

Which can be obtained from Parseval's theorem (2.13)

$$\sum_{t=-\infty}^{\infty} |y(t)|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} S(\omega) d\omega \tag{2.13}$$

This equality shows that $S(\omega)$ represents the distribution of sequence energy as a function of frequency. For this reason, $S(\omega)$ is called the *energy spectral density*. Most of the signals in practical applications are such that their variation in the future cannot be known. It is only possible to make probabilistic statement about the variation. Such sequences are called *random signals*.

A random signal usually has finite average power, therefore, we can use average power spectral density for its characterization. Which is, for simplicity, better known under name *power spectral density* (PSD). As shown in [4] is defined as

$$\phi(\omega) = \sum_{t=-\infty}^{\infty} r(k) e^{-i\omega k} \tag{2.14}$$

where $r(k)$ is auto covariance function $r(k) = E[y_t y_{t+k}]$. The PSD is very useful in the analysis of random signals since it provides us with information about the distribution of the average power over the frequency. There are several different methods for estimating the PSD.

Methods that require direct use of a finite signal for purposes of auto-correlation calculation are called *non-parametric* methods. On the other hand, methods that rely on a model for signal generation are call *parametric* methods [6]. Widely used non-parametric methods are

- Bartlett method,
- Blackman-Turkey method,
- Welch method or

- Danielle method.

They differ mainly in the resolution and variance level of the result. For a detailed description of methods see [2].

2.4 Digital Processing Techniques

2.4.1 Spectrograms

2.5 Embedded systems and microcontrollers

There are many definitions of embedded systems. The one picked for this thesis subjectively seems well accurate in the description of the given term. It goes as follows.

"An embedded system is a specialized computer system that is usually integrated as part of a larger system. An embedded system consist of a combination of hardware and software components to form a computational engine that will perform a specific function. Unlike desktop systems which are designed to perform a general function, embedded systems are constrained in their application [5]."

Embedded systems very often perform in reactive and real-time environments, which has to fulfill the real-time constrains that are described in 2.5.3. Not satisfying these constraints can cause significant system consequences. If the consequences consist of a degradation of performance, but not a failure, the system is referred to as a soft real-time system (e.g. highway car counter). On the contrary, if the consequences are a system failure, the system is referred to as a hard real-time system (e.g. a braking system in a vehicle).

Typical embedded system receives information about the surrounding environment via sensors and responds with actuators. General block diagram of such system is in Figure XXX.

2.5.1 Embedded systems

2.5.2 DSP Hardware Options

DSP algorithms can be implemented on different types of digital hardware. The following are the widely used options for DSP systems:

- Special-purpose chips such as application-specific integrated circuits (ASICs).
- Field-programmable gate arrays (FPGAs).
- General-purpose micro-processors or micro-controllers (μP/μC).
- General-purpose digital signal processors.

- DSP processors with application-specific hardware accelerators [6].

Each hardware (HW) platform has different advantages and constraints for different applications, thus, there is no *best* HW platform that could be used for every practical project. Instead, each option should be carefully considered from point of flexibility, required design time, power consumption, performance and cost.

Characteristic of mentioned hardware options are summarized in Table XXX ([6]).

2.5.3 Real-time constraints

Generally, a real-time system is one that must process information and produce a response within a specified time, else risk severe consequences, including failure. A real-time DSP system demands that the signal processing time t_p, must be less than sampling period T, that is

$$t_p + t_o < T, \tag{2.15}$$

where t_o is overhead time of input-output (I/O) processing. Thus, this limitation gives constraint to the highest frequency signal that can be processed by DSP systems in sample-by-sample processing, given as

$$f_M \leq \frac{fs}{2} < \frac{1}{2(t_p + t_o)} [6]. \tag{2.16}$$

Using different techniques of processing can reduce the I/O overhead time and increase the performance of the DSP hardware platforms. For example by applying a block-by-block processing, where the I/O operations are handled by DMA controllers, which place data samples in memory buffers [6].

Today, with performance improvement of hardware platforms, it is even possible to calculate FFT (see 2.3.2) of 64-points in a matter of tenths of milliseconds with low-cost ARM microprocessor (e.g. specifically in $0.16ms$ for microcontrollers of STM32F1xx series according to [7]).

2.5.4 ARM Cortex-M3

First manufactured microprocessor of Cortex series was ARM Cortex-M3. It was developed in 2006 and its target group of application was 32-bit microcontrollers. Its main advantage is an excellent efficiency which yields high performance and a low energy consumption without the need for a very high system clock frequency. It is based on architecture *ARMv7*.

An ARMv7 architecture was developed as a modern type of a general architecture that could be used for low-level microprocessors as well as for high-performance application processors. Design of the architecture is split into three main profiles:

- Profile A - aimed at application processors for performance-intensive systems capable of using embedded operational systems (such as Symbian, Android or Linux Embedded).

- Profile R - aimed at high-performance processors for real-time applications.
- Profile M - aimed at wide-range processors for deeply embedded applications [8].

Instruction Set

There are two instruction sets supported by ARM cores in general: the ARM instructions which are 32-bit and the Thumb instructions which are 16-bit. During the execution, the processor can switch either to the ARM state or to the Thumb state accordingly to the instruction set it is currently using.

The ARMv7 architecture is using new instruction set called a Thumb-2 which consists of the 32-bit Thumb instructions and also 16-bit Thumb instructions. This yields big improvement from the perspective of the ease of use, performance and code size. It allows execution of a complex operation in state Thumb which increases the effectivity [8].

Chapter 3

Practical Implementation

Prior to the design of the system architecture, a list of requirements for the device had to be created. This was done in discussion with company Rieter CZ s.r.o. who provided resources for development. The most important requirement is that the device shall be able to analyze the quality of a yarn and sliver. This separates the project in two parts, where core software algorithms are shared but other parts significantly different.

3.1 Yarn Quality Analysis

The yarn quality analysis requires only software algorithm and no additional hardware. This is because measurement of a yarn diameter is done by a row camera sensor that is connected to processing board which was already developed in Rieter. Processing board has CAN interface that sends values of diameters to the microcontroller.

Therefore, in this part of the project, the only input we consider are values of diameter send over CAN. This determines the project as software system with main requirements and features defined as follows:

- The system shall receive yarn diameter values and process them in a way that yields spectrogram values as output.
- The system shall communicate via CAN bus.
- The system shall be able to run on ARM microcontrollers STM32F1xx series.

3.1.1 Description of Algorithm for Uneveness Analysis

3.1.2 Flowchart Diagram

3.2 Sliver Quality Analysis

The sliver quality analysis requires development of software and also additional hardware. In this case - opposite to yarn quality system - it is necessary to design new processing board due to the fact that sliver and yarn have

significant size differences (see 2.1.1 and different sensors need to be used). Also, it was decided that device developed in this project will be only prototype version, thus, it has additional functionality that can be removed in a final product. The main requirements and features for the device prototype were defined as follows:

- The device shall be able to measure the diameter of a sliver.
- The device shall be able to filter, and process measured data in a way that allows calculating CV% value and Variance-length Curve.
- The device shall be able to measure the diameter with one or more specified sensors simultaneously.
- The device shall be able to send measured data to the service computer in real-time.
- A PC application shall be implemented for receiving and storing data received from the device.
- The device shall be able to send measured data to the service computer in real-time.
- The device shall run on ARM microcontroller STM32F446.

3.2.1 Hardware

The purpose of the hardware section relays in providing the correct supply to the control unit and the sensors. Another feature of the hardware platform is processing the signals from sensors and converting it to the range which is suitable for ADC[1] unit. Additionally, the hardware platform allows communicating with other distributed systems or PC by providing necessary interfaces.

The hardware block diagram of the system is shown in Figure 3.1. Description of each block follows in next paragraphs.

Control Unit

The main part of the hardware is ARM microcontroller STM32F446 from STMicroelectronics. Its choice was given by a requirement from Rieter CZ s.r.o with a description that it was chosen for its low cost, sufficient performance and company's good long-term experience with manufacturer's support.

Microcontroller STM32F446 is based on ARM Cortex-M4 technology with ARMv7 architecture 2.5.4. With DSP instructions, it is capable of handling DSP algorithms such as FFT calculation in very short time (less than $0.5ms$ for 1024-point FFT). Maximal operation frequency is $180MHz$ at which gives $225DMIPS$ (Dhrystone benchmark). The most important features and peripherals of this microcontroller are:

[1] Analog to digital converter

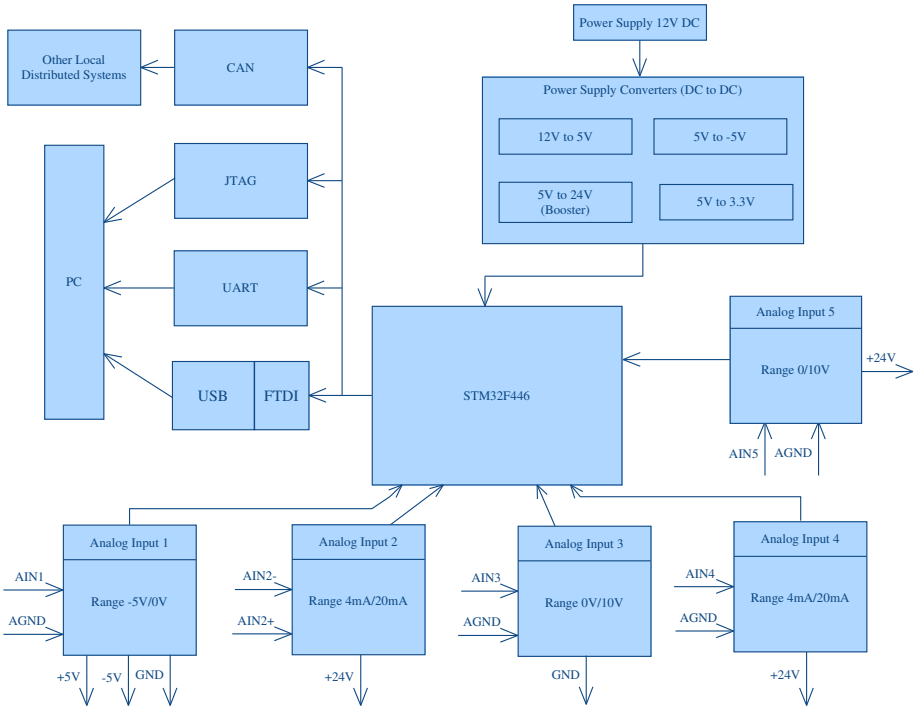


Figure 3.1: Block Diagram of the System

- ARM 32-bit Cortex-M4, 180 MHz,
- 512 kB of Flash memory,
- 128 kB of SRAM,
- SWD and JTAG debug interface,
- 17 Timers and 3 ADC,
- Communication interfaces such as USART, CAN, SPI, SAI, SDIO, I2C, LIN etc.

Power Supply Converters

Hardware platform of this system has to provide correct supply voltages to microcontroller and sensors. Supplying the sensors is especially complicated since they require three different voltages:

- $5V$,
- $-5V$,
- $24V$.

Voltage supply of the microcontroller is $3.3V$.

The power supply of the hardware board was required to be $12V$. Therefore, four DC-DC converters has to be designed, specifically:

- $12V$ to $5V$,
- $5V$ to $-5V$,
- $5V$ to $24V$,
- $5V$ to $3.3V$.

For the first conversion from $12V$ to $5V$ was used an integrated step-down voltage regulator LM2594. To obtain $-5V$ a charge pump inverter TPS60400 was used. The conversion from $5V$ to $3.3V$ is done by simple linear regulator MCP1825. The only step-up was necessary to get $24V$ from $5V$, for this purpose was chosen step-up voltage regulator LM1577.

Concrete circuits schematics of these converters are added as an attachment to the CD annexed with this thesis.

Sensors and Input Modules

3.3 Algorithm for Sliver Quality Analysis

Algorithm overview diagram is shown in Fig. 3.2. This is the simplified representation of software used for sliver quality analysis in the microcontroller. The input of the system is a sample - measured by ADC - which represents the diameter of the sliver. Required output shall be a variance-length curve which is composed of CV(%) values calculated for several wavelengths[2]. Each software components are individually described in following paragraphs.

The main principle of designed quality analysis lies in determining the coefficient of variation (CV(%)) value. This value can be obtain from signal that corresponds to appropriate wavelength (cut length). Different wavelengths can be yielded by filtrating the input signal with lowpass filter that passes only frequencies that correspond to given wavelength. Specifically if it is chosen that signal of wavelengths between λ_{min} and λ_{max} is required, then signal can contain the frequencies only up to f_{max}. Where relation between λ_{min} and f_{max} is based on knowledge of the *sliver speed* v_s[3] determined as follows in equation 3.1:

$$f_{max} = \frac{v_s}{\lambda_{min}}. \tag{3.1}$$

From this, we can see that the lower boundary λ_{min} is more important and as a matter of notation it is used to specify the whole wavelength range. This means that when speaking of CV(%) for the wavelength of λ it is actually meant λ_{min}.

[2]For explanation of the terms used in this paragraph see chapter XXX

[3]Speed of winding up sliver during the process of combing.

When sufficient amount of CV(%) values for different wavelength is obtained, then they are joined in *variance-length curve* (see chapter XXX). This curve can be easily represented graphically and as such it allows users to see the quality of a sliver for different wavelengths just by looking at the shape of the curve. Taking new samples may cause a change in appropriate CV(%) values, therefore, the variance-length curve is periodically updated. It is important to mention that part of the curve with the lowest wavelength can be updated several times before even one update of higher wavelengths can be calculated. This is due to the fact that higher wavelengths require significantly more input samples.

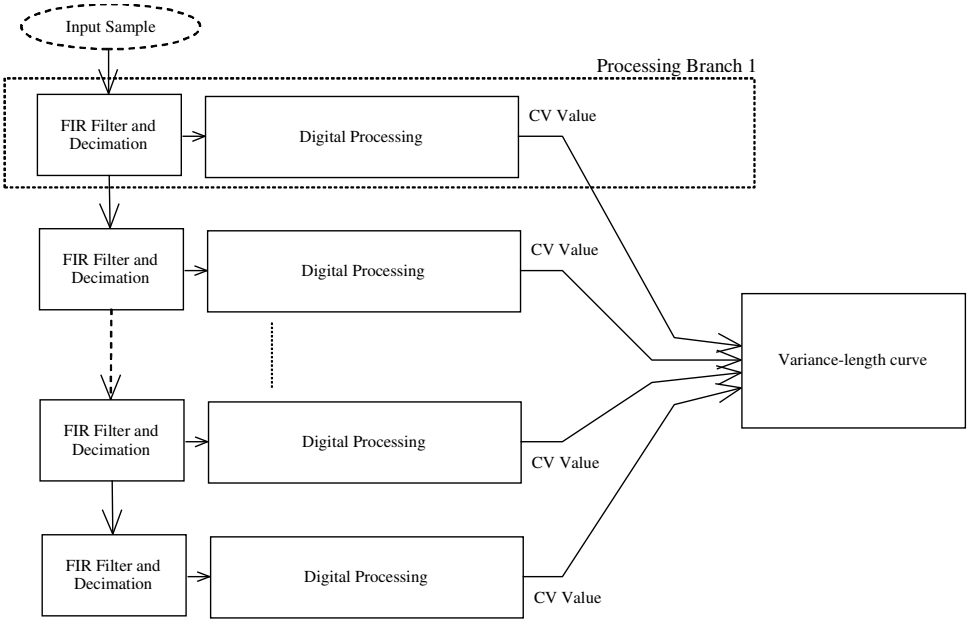


Figure 3.2: Software Architecture Overview

3.3.1 Solution for Embedded Systems Constraints

As described in chapter 2.5.3 using embedded systems brings several important constraints. The most significant issues this system has to solve were DSP performance and memory restriction of used microcontroller. The memory issue turned to be critical, because if we had control unit capable of storing hundreds of thousands of 16bit samples, we could easily store enough of input samples that allow calculation of all chosen wavelengths and then sequentially use low pass filter to obtain corresponding CV(%) values.

This could be done with a computer and was used in first version algorithm programmed in MATLAB. However, porting software to the microcontroller requires usage of decimation (see chapter XXX) which allows us to the lower number of samples used for each stage by throwing out every M-th sample (M is called the decimation factor). Of course, to prevent an *aliasing*[4] low pass

[4]Effect of violating the Nyquist-Shannon theorem.

filter has to be used to filter out frequencies higher than *Nyquist frequency*[5]. But, as mentioned in previous paragraphs our application requires the use of low pass filter which will always filter out frequencies higher than Nyquist frequency. Thus, no other filter is required prior decimation.

Another issue is limitation of microcontroller is that DSP function for calculation of radix-4 FFT: *cr4_fft_1024_stm32* requires maximally 1024 samples as input. This disallows usage of FFT on a much larger amount of samples and processing its frequency spectrum in a way that would also provide us information on the quality of sliver, since (with enough samples) all required frequencies - corresponding to wavelengths - would be visible. This issue is also resolved by the decimation of the signal and using multi-rate analysis.

Using this solution means that only buffers for 1024 samples are needed. Specifically one for each stage of calculation.

3.3.2 Wavelengths Specification

One of the requirements for the software was determined that the device has to be capable of evaluating the quality of a sliver for wavelengths in the range from $0.25cm$ to $512cm$ because others wavelengths are not significant enough to the final product. This range is then divided among twelve values, each wavelength representing one stage of calculation. This division will yield each wavelength to be the double of the previous and that allows us to set the decimation factor (see 3.3.1) to $M = 2$ for decimators in all stages. The table XXX shows these wavelengths and their corresponding sampling frequencies f_s calculated according to equation 3.1 with sliver speed $v_s = 30cm/s$.

This sliver speed was chosen as it is default settings of combing machines and was used during testing. If different sliver speed is set on combing machine, its value is sent over a CAN communication during initialization and the basic sampling frequency is adjusted in a way that decimation factor always stays set on $M = 2$. This allows using the same filter coefficients independently on value of sliver speed.

Knowing this information allows us to determine a *cut off frequency*[6] f_c of FIR filters that are used for limiting the bandwidth and an anti-aliasing. This is done according to the formula $f_c = f_s/2$ for each wavelength as shown in table 3.2.

3.3.3 Processing Branch

CV values for each of these wavelengths (as shown in table 3.2) are calculated in individual processing branch. They are in sequential order and input for each branch is a partial result of the previous one.

With every sampling period, one sample is taken by ADC peripheral and its values are stored in memory by DMA[7] and an interrupt is invoked. Then,

[5] Half of the sampling frequency
[6] XXX
[7] Direct Memory Access peripheral

Index of Stage	$\lambda_{min}[cm]$	$\lambda_{max}[cm]$	$\lambda_s[cm]$	$f_s[Hz]$	$f_c[Hz]$
1	0,25	0,5	0,125	240	120
2	0,5	1	0,25	120	60
3	1	2	0,5	60	30
4	2	4	1	30	15
5	4	8	2	15	7,5
6	8	16	4	7,5	3,75
7	16	32	8	3,75	1,875
8	32	64	16	1,875	0,9375
9	64	128	32	0,9375	0,46875
10	128	256	64	0,46875	0,234375
11	256	512	128	0,234375	0,1171875
12	512	1024	256	0,1171875	0,05859375

Table 3.1: Wavelengths and corresponding frequencies

in the interrupt handler a volatile flag determining that new sample was taken is set and main software state machine periodically checks its state. If the flag is set, the state is changed to first processing branch and sample processing begins.

Individual processing branch can be represented in the component diagram as shown in 3.3.

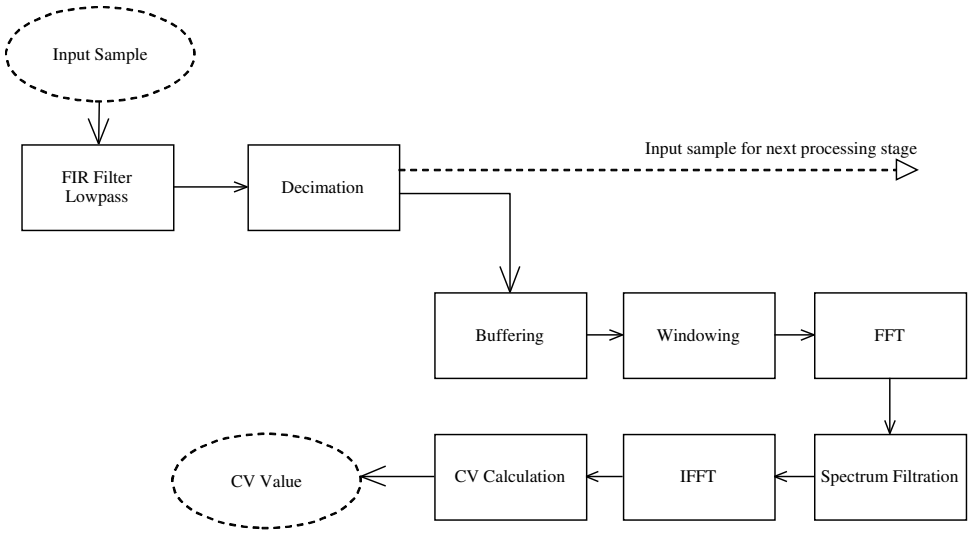


Figure 3.3: Component Diagram of Individual Processing Branch

Following paragraphs contain detail description for each component of processing branch.

Input Sample

Input sample is taken by ADC peripheral of the microcontroller. Particular input pin of ADC is set during initialization phase to the input module that is connected to the mechanical sensor of diameter (see 3.2.1). This sensor was chosen as for the first manufactured version of the device, because of its sufficiently accurate results and significantly lower price than other types of possible sensors. The ADC itself is set in DMA mode, with enabled interrupt after one sample is stored by DMA and to 12b size of the sample. The sampling frequency is set to be the double of the frequency required for the first wavelength $\lambda = 0.25cm$. This frequency is dependent on sliver speed which is sent over a CAN communication during initialization. Adjusting the sampling frequency is the only solution that allows using the same filter coefficients independently on a value of the sliver speed. It is done by changing settings of the ADC peripheral.

FIR Filter and Decimation

Each branch has to contain lowpass FIR filter because it is necessary to prevent aliasing and to remove frequencies that correspond to the smaller wavelengths. As mentioned in 3.3.1 each branch also has to implement decimation. Both of these operations can be combined in a component called *FIR decimator*. This is an implementation of FIR filter that calculates only each $M - th$ output, where M is a decimation factor. Thus, FIR decimator reduces required computational resources. The last version of DSP library for ARM microcontrollers does contain functions for the execution of FIR decimation. Unfortunately, these are not appropriate to use in this algorithm, since we need to the sample-by-sample filtration. This means that input for our implementation of FIR decimator has to be only one sample and output as well only one (or none) sample. Implementation is shown in the flowchart diagram 3.4.

Input sample for the FIR decimator is added to the Input Buffer. This buffer is unique for each stage. First, algorithm checks whether Input Buffer already has at least 33 samples, which is a number of taps in used FIR filter (also number of filter coefficients). If there are 33 samples, first filter output can be calculated. If there are fewer samples (it never can be more) than 33, buffer index is iterated by one and function returns -1 which indicates that output sample is not ready.

Next, the decimation flag is checked. In case, it is false no output is required during this run of FIR decimator. But it is still necessary to shift the Input Buffer to the left so that the actual input sample is correctly used in next filter calculation. In opposite case, when it is true, this run should calculate and return the output value. This is the decimation part of the algorithm. The flag is negated after each valid output value is produced and this yields downsampling with factor $M = 2$.

Output itself is calculated as a sum over the result of multiplication of Input Buffer and coefficients. This value is returned from the function that

implements this filtering and is passed to other components in processing branch and to the next stage FIR decimator.

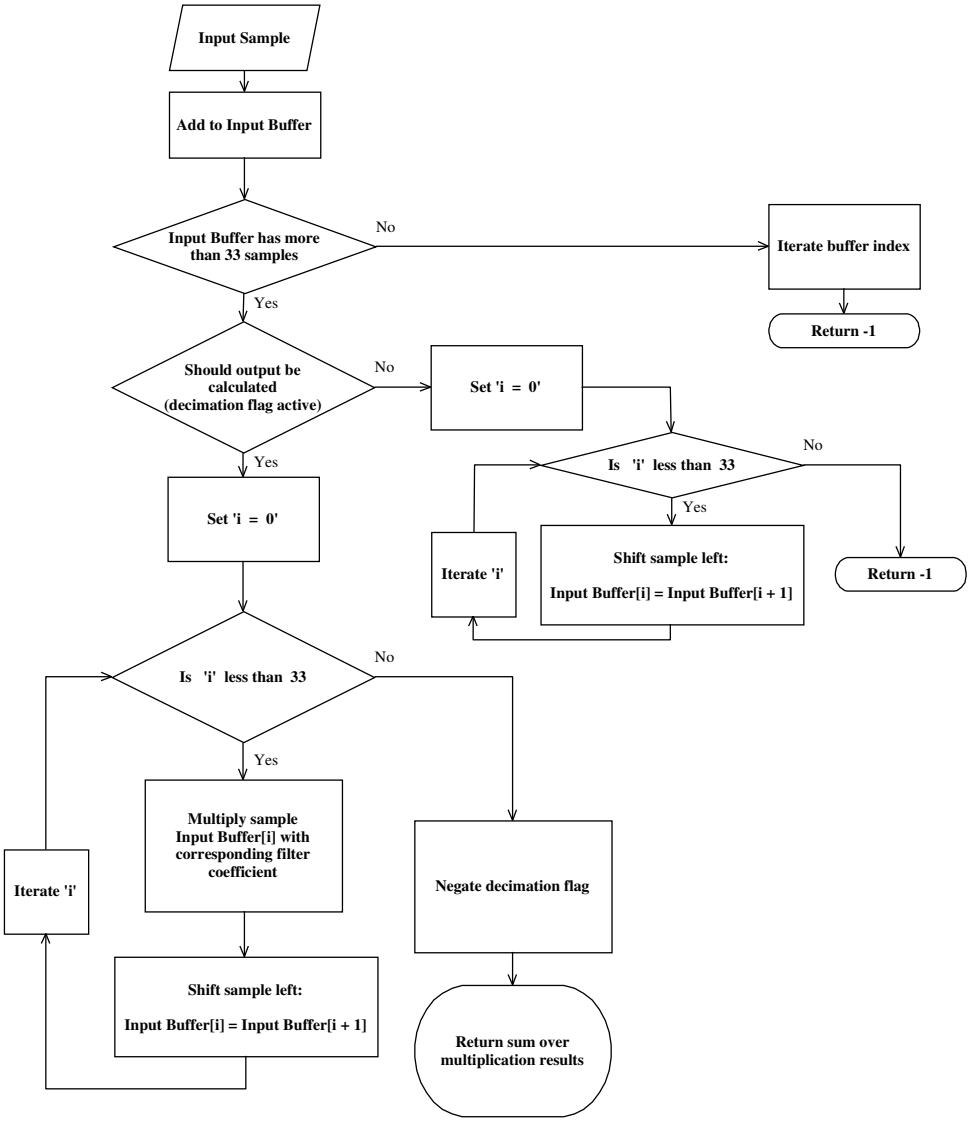


Figure 3.4: Flowchart diagram of FIR decimator component

FIR filter used in the FIR decimation was chosen as lowpass filter with order 32 and *Bartlett-Hanning* window. Using this order and window yields required frequency properties as can be seen in its frequency response shown in figure 3.5. Also its order is still small enough to allow sufficiently fast calculation. The cut-off frequencies are shown in table 3.2. The cut-off frequency is the same in normalized frequency[8] for all the branches. Therefore, the same filter coefficients can be used for all the processing branches.

[8]Range 0 to 1 and 1 corresponds to the $f_s/2$

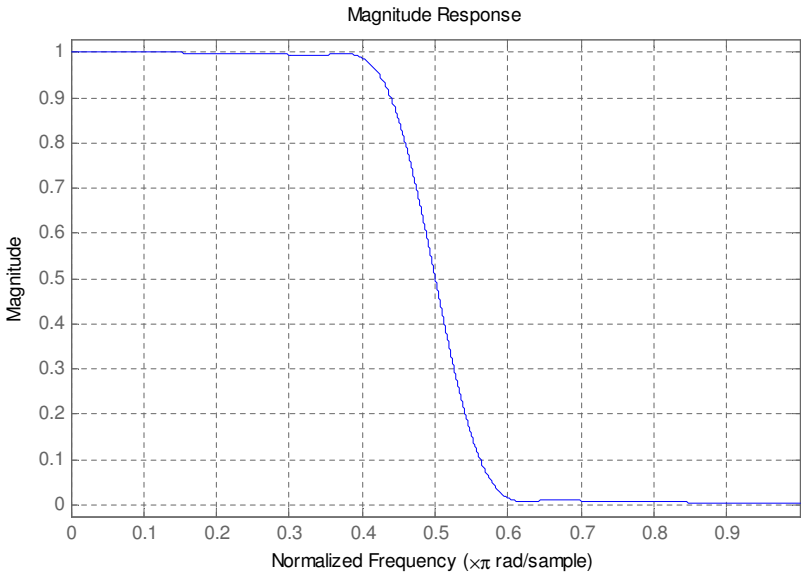


Figure 3.5: Magnitude response of designed FIR filter

Buffering

Output samples from FIR decimator are buffered in vectors of length 1024. When the buffer is filled to full an appropriate flag is set and calculation can proceed to the next component. This creates a requirement for memory space. We know that it is required to has 12 processing stages and each stage has to have one buffer of 1024 sample and one sample is stored in 2B integer word. Another buffer for 1024 samples has to be ready for an output of FFT. From this conclude that $Memory\ required = 12 \cdot 2 \cdot (1024 \cdot 2) = 49152B \cong 49.2kB$. This is not an issue for used microcontroller STM32F446 which has 128 kB of SRAM (see 3.2.1).

Windowing

Before calculation of FFT can be executed an operation called windowing (see chapter 2.3.1) should be performed. This will prevent leakage and smearing, which are effects that are distorting the frequency spectrum obtained by FFT.

FFT

The frequency spectrum is obtained by algorithm called FFT (see chapter 2.3.2). This is a fast implementation of Fourier transform and it is the most widely used technique used for obtaining frequency spectrum of signals. DSP libraries for ARM systems do contain functions for complex and real FFT. For our case we can use real FFT function *arm_rfft_fast_f32* with length of FFT 1024. This calculation can be performed on used microcontroller in tenths of milliseconds. [7].

The FFT is defined over complex data but in many applications is the applied input real. Real FFT algorithms have speed advantage over complex algorithms of the same length because they properly use the symmetry properties of the FFT.

Spectrum Filtration

With frequency spectrum of the signal we can filter out the frequencies that are proclaimed as artefacts. Such errors in signal are caused by used mechanical sensor. This sensor measures the diameter of a sliver but during the measuring it is affected by mechanical vibration caused by turning gears inside the combing machine. Each of this gear creates vibration with frequency that is depending on a number of teeth and shaft speed.

It would be possible to determine the frequencies that are distorting the signal from a detailed technical materials about combing machine and its gears. But unfortunately non such materials were available. Therefore, different approach to determine the error frequencies had to be used.

For this purpose, a special measurement was executed (see chapter XXX). For this measurement it was possible to run the combing machine without any sliver, thus, only the mechanical vibrations were measured by sensor. Signal from this case was digitally post-processed in MATLAB and from obtained frequency spectrum specific artefact frequencies were distinguished. They were clearly visible even to the naked eye as the given frequencies had much higher amplitude than their surrounding. To obtain concrete frequency bins a peak detection was implemented and used. Result is shown in frequency spectrum of a signal measured without sliver with sampling frequency $f_s = 240Hz$ (see figure 3.6).

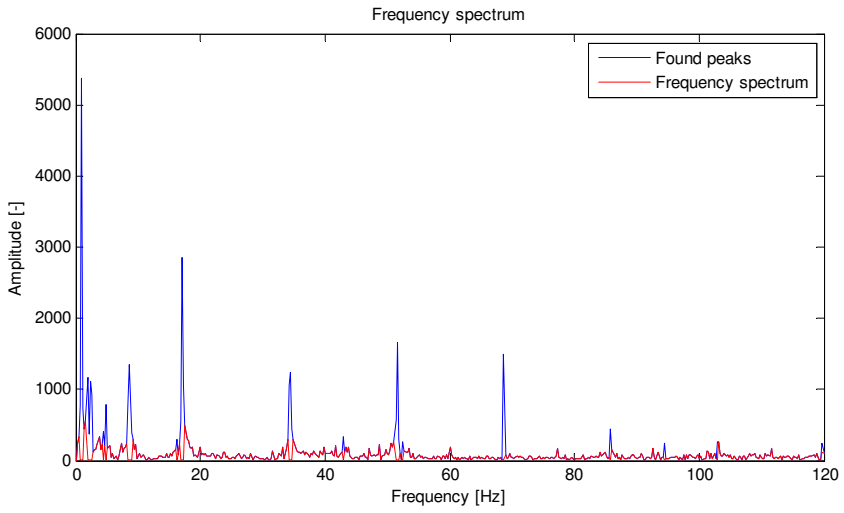


Figure 3.6: Frequency spectrum of signal measured without sliver

The most significant found frequencies are listed in table **??**
This peak detection algorithm is based on

Frequency [Hz]	0.94	1.88	2.34	4.45	4.92	8.672	17.11	34.45	42.89

Table 3.2: List of distorting frequencies

Although we knew which frequencies should occur in spectrum, it was unsure During the test data measurement it was possible to start measuring without any materials,

They are shown in table XXX.

3.3.4 Flowchart Diagram

3.4 Description of Designed System

3.4.1 Block Diagram

3.5 Implementation of Software

3.5.1 Filtration

3.5.2 Detection of Defects

3.5.3 Automatic Evaulation

3.5.4 Example of Designed Application

3.6 Implementation of Hardware

3.6.1 Electronic Circuits Design

3.6.2 Description of possible sensors

3.6.3 Designed Prototype

Chapter 4

Conclusions

Proof. 8 Bla

1. Blo

□

Appendix A

Bibliography

[1] V. K. I. Dimitris G. Manolakis, *Applied Digital Signal Processing*, Cambridge University Press, 2011.

[2] D. G. M. John G. Proakis, *Digital Signal Processing*, Pearson Prentice Hall, 2007.

[3] P. Kousalík, *Inteligentní snímač pro on-line vyhodnocování kvality příze*, dissertation, Technická univerzita v Liberci, 2011.

[4] R. M. Petre Stoica, *Specral Analysis of Signals*, Prentice Hall, 2005.

[5] M. K. Robert Oshana, *Software Engineering for Embedded Systems*, Elsevier, 2013.

[6] W. T. Sen M. Kuo, Bob H. Lee, *Real-Time Digital Signal Processing*, John Wiley and Sons, Ltd, 2013.

[7] STMicroelectronics, *STM32F10x DSP library*, 2010.

[8] J. Yiu, *The definitive guide to the ARM cortex-M3*, Newnes, 2010.

[9] A. Ziabicki, *Fundamentals of Fiber Formation*, John Wiley and Sons, Ltd., 1976.

název práce
Analýza periodických vad textilních vláken ve frekvenční oblasti
Analysis of the textile fibers unevenness in frequency domain

katedra obhajoby
katedra měření

obor
<neurčen>
Nesouhlasí obor práce s oborem studijní plánu studenta!

studijní program
Magisterský

vedoucí
Parák Jakub Ing.
katedra teorie obvodů (13131)
parakjak@fel.cvut.cz

oponent

student
Renza Ondřej (studuje) - zadáno
renzaond@fel.cvut.cz
Senzory a přístrojová technika
studijní plán: Kybernetika a robotika - Senzory a přístrojová technika (MPKYR2)
katedra obhajoby podle studijního plánu: katedra měření

literatura
[1] Joseph Yiu: The Definitive Guide to the ARM Cortex-M3, Oxford, 2007.
[2] Kousalík, P.: Inteligentní čidlo pro on-line vyhodnocování kvality příze. Liberec, 2011. dizertační práce, TUL.
[3] Uhlíř, J., Sovka, P.: Číslicové zpracování signálů. Ediční středisko ČVUT, Praha, 2002. Monografie CVUT FEL.
[4] Katalogové listy jednotlivých komponent a součástek.

pokyny
1. Seznámit se s problematikou vad textilních vláken a spřadních strojů
2. Navrhnout systém pro detekci vad textelních vláken
3. Navrhnout implementeci filtračního detekčního a detekčního algoritmu ve frekveční oblasti
4. Implementovat softwarovou knihovnu s navrženým algoritmem pro mikrokontrolér
5. Experimentálně otestovat navržený systém s implementovaným algoritmem

zadavatel
spolupráce s Rieter CZ s.r.o.

---

vytištěno: 28.11.2015 14:04:00
vytiskl: Parák Jakub Ing.