Adaptive Signal Filtering Platform for a CPS/IoT Ecosystem

Haris Isakovic, Stefan Dangl, Zlatan Tucakovic, Radu Grosu
Technische Universität Wien, Vienna, Austria
name.surname@tuwien.ac.at

2021 22nd IEEE International Conference on Industrial Technology (ICIT) | 978-1-7281-5730-6/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICIT46573.2021.9453496

***Abstract*—The rapid increase in number of devices in Internet-of-Things generates astronomic amounts of data. Dealing with noisy and low quality data uses more effort than the data analysis itself. Dealing with noisy data at the source would significantly reduce the effort of pre-processing during analysis, as well as the storage and bandwidth overhead. In this paper we introduce an Adaptive Signal Processing Platform (ASPF) for CPS/IoT Ecosystems. It provides ability to dynamically detect noise variation in a signal and successfully filter these components out of the signal leaving only clean and useful data. The paper shows two approaches with different requirements on effort and scalability.**

I. Introduction

Industrial production systems are highly complex structures dealing with multiple levels of operation and heterogeneity. They possess data collection and diagnostic capabilities but they are often highly unsystematic and not designed for purposes of automated data analytic. Nevertheless, this data can be used to extract hidden technological or commercial insights in these production processes. By applying state-of-the-art tools and methods this data can be used to optimize the whole process, detect faulty products at run time, predict system failures and anticipate maintenance of individual components or the whole process. We define a CPS/IoT Ecosystem as a heterogeneous structure of hardware devices, and corresponding software components implemented in three scopes of operation: cloud, fog/edge, and sensor/actuator nodes [1]. It ensures data collection from the physical environment of the application, which is further aggregated and filtered in the fog and edge, and stored and analysed on a large scale in the cloud. Industrial facilities are generally noisy environments and sensors can be impeded to produce clean data. A denoising process is required to extract useful information before being analysed for higher levels of abstraction. Denoising is typically a pre-processing task during analysis and it uses various methods of approximation to reach most common subset of values regarded as basis for the data source. In this work we explore the ability to denoise data on runtime in the Fog. We propose a platform that will dynamical detect noise component in the signal and reconfigure the filter devices to remove them.

II. Background

A. CPS/IoT Ecosystem

A CPS/IoT Ecosystem is an heterogeneous system of systems that is combining two concepts, a Cyber-Physical System and the Internet of Things (IoT). It provides the ability to describe, perceive and interact with the physical environment in distributed way and on a large scale. As mentioned above we divide CPS/IoT Ecosystem in three scopes of operation, based on their unique abilities to manipulate and communicate data. **Sensors** and **Actuators** are devices in an immediate proximity to the physical environment, where they can either sense or manipulate the physical properties of a system. They have limited computational resources and communication capabilities. The **Fog** or **Edge** is represented by devices that can perform tasks such as real-time control or data aggregation and filtering and communicate this data vertically and horizontally with relatively low latency. They provide dedicated hardware interfaces for sensors and actuators that allow real-time control of these devices. The **Cloud** is providing computational and storage resources to store the data in large scale and allow complex analysis tasks that give us the ability to extract valuable emerging insights into these physical systems.

B. Hardware Accelerators in CPS

The development of Cyber-physical systems requires multiple disciplines working together in a unified development process. Physical systems need to be interfaced with computer systems and vice versa. At the same time computer systems need to be designed in the optimal way to ensure that properties like performance, dependability, power consumption and cost satisfy imposed standards and stakeholder requirements. As mentioned above, a computer system can be implemented using different combinations of hardware: a) commercial off-the-shelf hardware, or b) a dedicated hardware solution.

A commercial off-the-shelf hardware system provides all advantages of a generic platform with extensive set of system software and other resources (i.e., development tools) necessary to develop an application in a fast and efficient way. Such a system would be based on a standardized CPU and other components not especially designed for all types of functions present in a typical CPS (e.g., signal processing, complex mathematical operations). In addition all these abstractions implemented to improve user experience and improve development create additional overheads on the system.

Dedicated hardware allows more efficient implementation, with an obvious flexibility trade-off. Building custom hardware is a completely different process from the one used to write software programs. More complex and more time consuming, harder to debug and very difficult to change and adapt. Dedicated hardware has been used in applications with utmost

978-1-7281-5730-6/21/$31.00 ©2021 IEEE

importance on dependability, where the importance of reliable operation outweighs the cost. Introduction of FPGAs (Field Programming Gate Arrays) improved some of these properties and allowed engineers to implement flexible and effective hardware solutions. However, the design process and the ability to enroll applications still remains far behind software solutions on COTS hardware.

Hardware accelerated applications are systems where program is executed on a CPU in a classical way and parts of the program are implemented as hardware IP blocks and outsourced on an FPGA or other dedicated device. The two parts are seamlessly integrated with the software component acting as a control part. Introduction of hybrid SoC platforms (e.g., Xilinx Zynq, Altera Arria) creates even closer bond between the main software and individual accelerators.

C. Adaptive Signal Filtering

Filtering and noise reduction of signals is an extensive research topic in signal processing domain. However, the field of adaptive runtime filtering is still a relatively novel topic. Static runtime filters are applied in applications such as live audio processing, and dynamic post-processing filtering is used in audio post-production. In this paper, the terms dynamic and adaptive filtering refers to an adaption of the filter behavior according to a change in the input signal. E.g. the filtered frequency range could be adapted or the filter type could be switched when the input signal changes to achieve a better filtering result. The usage of dynamic filters has many advantages compared to static filters such as a better noise reduction but therefore the filtering system has to recognize the noise component of the input signal.

The problem with dynamic filtering is that it's quite difficult to analyse the input signal, adapt the filter curve at the same time, and still guarantee a short latency. In this paper, we take advantage of CPS/IoT Ecosystem and its different scopes of operation that handle these tasks individually. That leads to a reasonable reconfiguration time and a very short latency what is essential for systems with high dependability requirements. Although, due to the increasing power of computers in the last years it would also be possible to achieve good results with only one computation unit.

III. Adaptive Signal Filtering Platform (ASFP)

The proposed adaptive signal filtering platform is developed over all three scopes of operation defined in Section II-A. Further, it can be divided in the following stages: 1) Data Acquisition, 2) Noise Detection and Classification, 3) Runtime Adaptation, 4) Signal Filtering and 5) Data Storage. A sensor would acquire noisy data from the environment, this signal would be fed to the Noise Detection and Classification (NDC) component. NDC analyses signal and sends the signal features to the Runtime Adaptation (RA) component. Based on the information received from NDC the RA component will calculate the corresponding filter that will be delivered in the form of the image on the Signal Filtering (SF) component. The filtered signal would be then transferred into the cloud. Figure 1 provides an overview of the computing stages and the data-flow for the ASFP.

A. Noise Detection and Classification

Algorithm 1 Noise detection algorithm

Require: Signal $X_i, i \in \{1...n\}$
  if FFT Available for X **then**
    # Calculation of the mean value
    sum = 0
    cnt = 0
    for $i = 0; i < FFT.length; i++$ **do**
      if $FFT(X_i) > 0$ **then**
        sum += $FFT(X_i)$
        cnt = cnt + 1
      end if
    end for
    avrg = sum/cnt
    # Calculation of the lower cut-off frequency
    for $i = 0; i < FFT.length; i++$ **do**
      if $FFT(X_i) >=$ avrg **then**
        lowerCutOffFrequency = i/2
        break
      end if
    end for
    # Calculation of the upper cut-off frequency
    for $i = FFT.length - 1; i >= 0; i--$ **do**
      if $FFT(X_i) >=$ avrg **then**
        upperCutOffFrequency = i*2
        break
      end if
    end for
  end if

The applied Noise Detection Algorithm is defined on the basis of Fast Fourier Transformation (FFT) and it is implemented on a microcontroller (MCU) platform specially designed for signal processing. The MCU uses an FFT based algorithm described in the Algorithm 1 to analyse the input signal that is coming from the sensor node. Note that only amplitudes of FFT are used. The process is performed in following steps:

1) The input signal is analysed by the FFT algorithm.
2) The mean value of all spectral lines of the FFT result is calculated. For that only spectral values greater than 0 are taken into account.
3) The lower cut-off frequency of the band pass is calculated. Therefor the frequency of the lowest spectral value of the FFT greater then the previous calculated main value is taken. The iteration starts at the lowest frequency (0 Hz) and goes up to the highest measured frequency (22 016 Hz). If the value of a spectral line is greater than or equal to the previously formed mean value, it is assumed that this frequency already belongs to the actual signal. This frequency is chosen and used

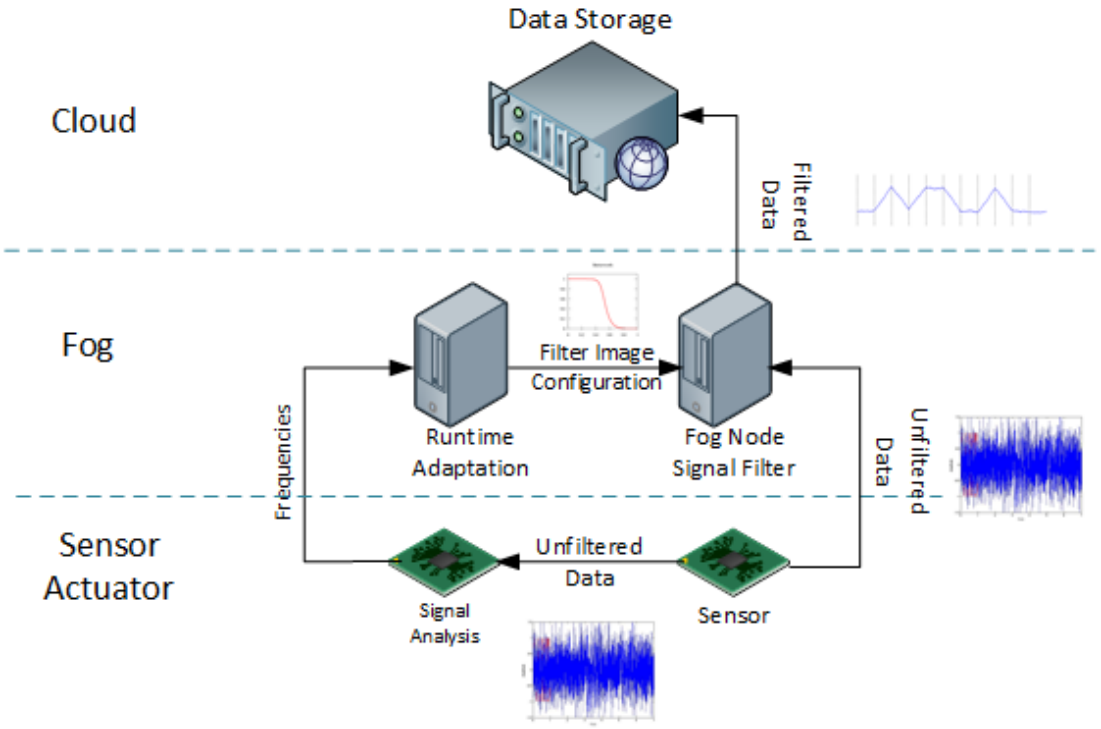

Fig. 1. Adaptive Signal Filtering Platform (ASFP)

for further calculations. It is guaranteed that such a value is found.

4) Similar to the previous step, the higher cut-off frequency of the band pass is calculated. Here, the iteration starts at the highest frequency and is gradually reduced. The filtering should influence the actual signal as little as possible. Accordingly, the obtained value of the lower cut-off frequency is halved and that of the upper one is doubled.

Algorithm 1 shows the fundamental basics of the *Noise detection Algorithm*. It should be noted that the used FFT function offers a resolution of 43 Hz.

B. Runtime Adaptation

Runtime Adaptation stage is implemented on a scale of the Fog or Edge devices. It provides enough computation resources to execute mathematical modeling tools used to calculate filter properties and to generate configuration code for the filter device. The cut-off frequencies received from the Noise Detection and Classification node are used to calculate filter coefficients that are further used to generate the code and the configuration for the filter device. This step could be performed in the cloud as well. However, if we were to use this approach in a control loop the difference in a response time between cloud and edge could be quite significant [2]. Thus, performing this task in the Fog is better suited for the use case.

C. Signal Filtering and Cloud Storage

In the final two steps signals are filtered and then stored in the cloud. The filtering is performed on the separate node in the fog designed specifically for this purpose. In this work we focused on two specific types of devices: FPGA and Digital Signal Processor (DSP) Node. Runtime Adaptation Node would calculate required filter parameters and compile them into existing code templates for each device. Each device is than directly flashed from the Runtime Adaptation Node.

Signal Filter Node is connected via communication gateway via cloud, where the filtered signal is stored after filtering. Alternatively, the data could be fed to another Fog Node, a Programmable Logic Controller (PLC) or another embedded platform performing a real-time control task. The stored data can be further used for an offline analysis.

IV. Implementation

In the previous section we learned all necessary stages of the proposed ASFP. In this section we describe implementation specific details.

For the Noise Detection and Classification Node we used a *Teensy 3.6* micro-controller board [3]. It performs the frequency spectrum measurements of the input signal using FFT. The input signal is generated by function generator with ability to randomise noise, and interfaced with the Teensy board over a *Audio Adapter Board for Teensy 3.0 - 3.6* [4]. The audio adapter is used to increase the resolution (16 bit, 44.1 kHz) and improve the analysis capabilities [5]. Noise detection

is performed using an algorithm described in Section III-A. The latest eight cut-off frequencies calculated by the Noise Detection Node are buffered in a local storage. The lowest and the highest cut-off frequencies in the buffer are transmitted to the Runtime Adaptation node on a request.

The Runtime Adaptation Node is implemented on a Windows 10 machine. A tool-chain for mathematical modeling, and code generation consists of: a) GNU Octave [6] used to calculate filter parameters, b) Python [7] scripting and task automation, c) Intel Quartus Prime Software Suite [8] FPGA management and configuration, d) Sigma Studio [9] for DSP programming and configuration.

We implemented Signal Filtering Node on two different hardware platforms, namely an FPGA and DPS. The FPGA approach uses the Altera DE2-70 FPGA Board [10]. To optimize the power consumption the FPGA operates at 12 MHz, and to minimize the quantization error, the on-board audio chip (WM8731) [11] works with a resolution of 24 bit, 96 kHz (high-quality audio). This filter implementation strictly follows the state space representation of the corresponding filter and uses arithmetic floating point modules for the calculation. The filter implementation is fixed while the filter coefficients are configured on runtime via UART interface. The coefficient transfer is performed using two buffers to ensure coefficient synchronization. The buffers are implemented in DRAM to optimize energy consumption.

The second approach uses the DSP platform to implement Signal Filtering Node. For this purpose we used a Wondom DSP board [12]. The onboard ADCs and DACs were configured to operate with the same resolution of 24 bit and 96 kHz as in the case of FPGA. The filter skeleton was implemented in the Sigma Studio software. For the configuration, compilation and flashing process Sigma Studio is also used, because it allows writing automation scripts. The Sigma Studio script reads the cut-off frequencies sent by the MCU, calculates a new configuration for the DSP platform, and applies this new configuration.

Both implementations enable the simultaneous sampling and generation of two signals. This allows to process stereo signals.

V. Results and Discussion

In this section we will discuss the results in terms of filtering capabilities, power consumption, latency and scalability. The focus of the paper is to establish feasibility of CPS/IoT Ecosystem to perform adaptive signal filtering on runtime. This would ensure optimisation of data flow, bandwidth optimisation in case of applications with limited communication capabilities e.g., applications using narrow-band IoT networks. Further, noise detection and classification can be used to detect faults or deterioration in Quality of Service (QoS). Data analysis is often sidetracked by noisy and redundant data. This effort can be reduced significantly by using data pre-processing in production stage.

A. Filtering Behavior

Figure 2 and 3 show how the filters response on a random noisy input signals. At figure 2, a 100 Hz sine wave is applied, and in Figure 3, the frequency of the input signal was set to 1 kHz. The test system reacts automatically to this change and reconfigures the filters. Figure 3 shows the filter behavior after the reconfiguration. It can be seen, that there's still a good noise reduction, but no reduction of the magnitude of the output signal, which was a requirement. The upper part of the figures compares the input signal (marked as blue or darker) with the output signal (marked as red or lighter) when no filtering is applied. The lower part compares the input signal with the filtered output signal. The left side corresponds to the DSP implementation and the right side to the FPGA implementation. The ASFP successfully re-configures the filter in both cases and manages to reduce noise with a significant margin.

B. Power Consumption

For the theoretical calculation of the power consumption, data sheet information and the PowerPlay - Early Power Estimator [13] by Altera were used. The following table compares the calculated power consumption of the FPGA implementation with those of the DSP implementation under standard conditions (regarding temperature and pressure):

Platform	Power
Altera DE2-70 FPGA Board	235,23 mW
Teensy DSP Board	362,10 mW

We observe that the FPGA implementation has a power consumption that is 35.04% lower than the DSP's.

C. Response time and latency

The reconfiguration time of the two implementations is currently around 100 ms. That means that it usually takes around 100 ms for the filter instrument to respond to a frequency change of the input signal with an adaption of the filter curve. This value is strongly dependent on parameters such as transfer rate and processing speed and can be subject to optimization in the future work.

The latency here is defined by the sample and hold time of the ADC and DAC and the response time of the filtering systems. The response time of the FPGA-Implementation is around two 96 kHz periods. This leads to a latency of around 20 μs.

D. Scalability

The effort to create the FPGA implementation was much higher than the effort to create the DSP implementation. Initially, the FPGA implementation requires more hours to implement than the DSP. So FPGA is more suitable for larger projects and DSP for faster prototyping. However, the FPGA approach has significant advantage in scalability. While it requires significantly less effort, the DSP platform is limited with the predefined function blocks and less flexible toolchain. If we consider dynamic reconfiguration as an use case, the

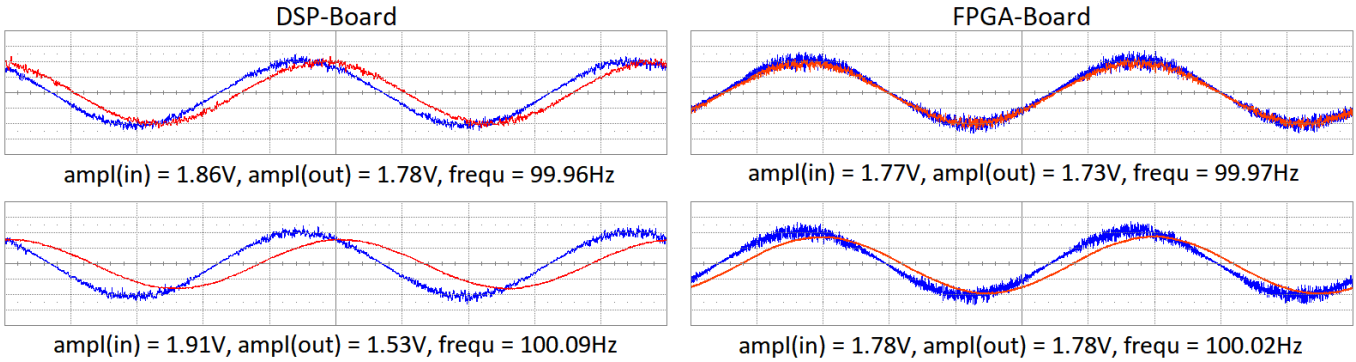

Fig. 2. frequency: 100Hz, Calculated cut-off frequencies: 0 Hz and 288 Hz

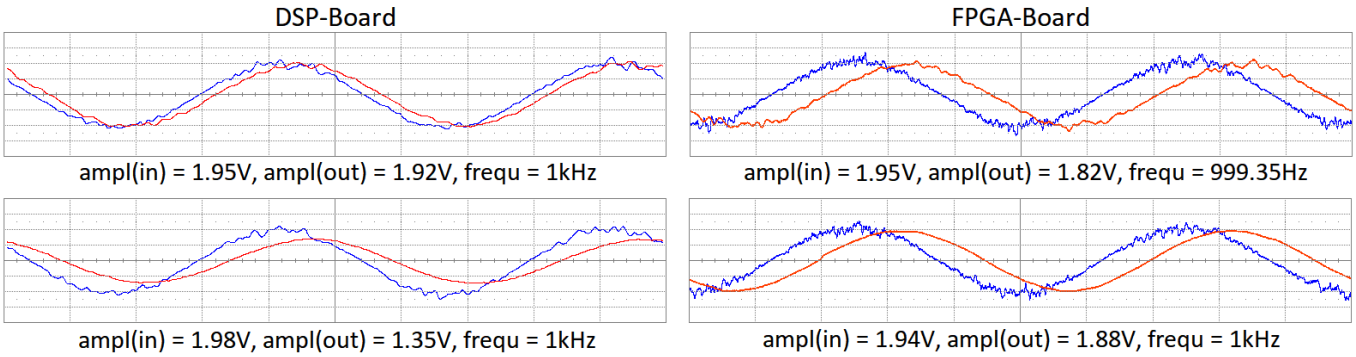

Fig. 3. frequency: 1kHz, Calculated cut-off frequencies: 576 Hz and 2304 Hz

FPGA approach and its toolchain are certainly better suited for this purpose.

VI. Related Work

In this paper we propose an approach for adaptive signal filtering using FPGA and DSP computation platforms. The authors of the paper [14] showed that other devices such as graphics hardware (GPUs) could be used for real-time FIR filtering of audio signals. In that paper, the filtering is also done in the frequency domain, and not in the time domain. Their main use case is the audio rendering of complex scenes, and their work is evaluated on NVIDIA GTX 285 card providing more than 200 channels with individual filters. In this paper, only 2 channels with the same filter configuration are provided, but can easily be expanded. The latency of the NVIDIA GTX 285 based test system is about 1 ms to 6 ms, which is slower than the approach proposed in this paper. It takes about 20 μs for the signal reading, filtering and direct outputting. The IoT communication overhead is not taken into account here. Paper [14] also features a dynamic filter update rate in range from 40 Hz to 100 Hz, and in their test system, 64 channels are update at once, which leads to a reconfiguration time of 10 ms to 25 ms. That is faster than in our approach. However, it is not clear if they considered the analysis of the input signal which takes most of the reconfiguration time, as shown in this paper.

A novel and interesting approach to reduce noise in signals would also be the use of deep learning AIs (Artificial Intelligence). The authors of [15] introduced a Deep Neural Network (DNN) - based framework for speech enhancement. Despite the advantages of this approach, the realization is very laborious, i.e. an extensive training set that encompasses many possible combinations of speech and noise types has to be designed. The authors used 104 different types of noise such as restaurant and street noises and trained the system with more than 100 hours of data.

Both works show efficient and novel methods to achieve adaptive signal filtering, however their focus is on different scope of operation with little to none consideration on resource utilization, or communication capabilities. The focus of the work in this paper is signal filtering in domain of IoT and industrial IoT, with the constraint of limited resources, computational and communication capabilities.

VII. Conclusion

Number of IoT devices is rising almost with an exponential rate. This new wave of devices is creating astronomic amounts of data. Noise is an unavoidable effect of the data acquisition process. It originates in different forms and from different sources and it significantly reduces analysis capabilities of data and increases effort [16][17]. Reducing noise is a significant challenge in IoT due to non-standardized nature of

IoT devices. In this paper we proposed the Adaptive Signal Filtering Platform (ASFP) with two possible implementations and showed how it can be applied in domain of IoT. We showed that these approaches could be applied in a dependable applications and also low effort hobby projects. In the future, we will extend the number of filter approaches and investigate feasibility of these implementations. It will also reduce response times of the reconfiguration process. In addition to aforementioned approaches, it would be interesting to compare a Tensor Processing Unit approach with the existing two.

ACKNOWLEDGMENT

This work has been conducted within projects that has received funding from the Austrian Government through the Federal Ministry Of Education, Science And Research (BMWFW) in the funding program Hochschulraum-Strukturmittel 2016 (HRSM). This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871319.

REFERENCES

[1] H. Isakovic, D. Ratasich, C. Hirsch, M. Platzer, B. Wally, T. Rausch, D. Nickovic, W. Krenn, S. Dustdar, and R. Grosu, "CPS/IoT Ecosystem: A platform for research and education," p. 8.

[2] S. Maheshwari, D. Raychaudhuri, I. Seskar, and F. Bronzino, "Scalability and Performance Evaluation of Edge Cloud Systems for Latency Constrained Applications," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, Oct. 2018, pp. 286–299.

[3] "Teensy USB Development Board," https://www.pjrc.com/teensy/.

[4] "PJRC Store," https://www.pjrc.com/store/teensy3_audio.html.

[5] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. Urbana, IL: University of Illinois Press, 1949.

[6] "GNU Octave," https://www.gnu.org/software/octave/index.

[7] "Python.org," https://www.python.org/.

[8] "FPGA Design Software - Intel® Quartus® Prime," https://www.intel.com/content/www/de/de/software/programmable/quartus-prime/overview.html.

[9] "SigmaStudio® — Analog Devices," https://www.analog.com/en/design-center/evaluation-hardware-and-software/software/ss_sigst_02.html.

[10] T. Technologies, "Terasic - All FPGA Boards - Cyclone II - Altera DE2-70 Board," https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=53&No=226&PartNo=4.

[11] "WM8731 — Cirrus Logic," https://www.cirrus.com/products/wm8731/.

[12] Boomaudio, "Wondom DSP Board," https://www.boomaudio.de/wondom-dsp-board.

[13] "Early Power Estimators and Power Analyzer," https://www.intel.com/content/www/us/en/programmable/support/support-resources/operation-and-testing/power/pow-powerplay.html.

[14] F. Wefers and J. Berg, "High-performance real-time fir-filtering using fast convolution on graphics hardware," 01 2010.

[15] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 23, no. 1, p. 7–19, Jan. 2015. [Online]. Available: https://doi.org/10.1109/TASLP.2014.2364452

[16] A. Waldherr, D. Maier, P. Miltner, and E. Günther, "Big data, big noise: The challenge of finding issue networks on the web," *Social Science Computer Review*, vol. 35, no. 4, pp. 427–443, 2017. [Online]. Available: https://doi.org/10.1177/0894439316643050

[17] S. Gupta and A. Gupta, "Dealing with noise problem in machine learning data-sets: A systematic review," *Procedia Computer Science*, vol. 161, pp. 466 – 474, 2019, the Fifth Information Systems International Conference, 23-24 July 2019, Surabaya, Indonesia. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050919318575