The Prism: Efficient Signal Processing for the Internet of Things

Henry, M, Leach, F, Davy, M, Bushuev, OY, Tombs, M, Zhou, F & Karout, S

DOI 10.1109/MIE.2017.2760108
ISSN 1932-4529

Publisher: IEEE

The Prism – Efficient signal processing for IoT applications

Abstract

A new, networked approach to signal processing, efficient for both design and real-time computation, is proposed to match the requirements of the IoT. The Prism is a novel FIR filter which is fully recursive: the computational cost per sample is low and independent of the filter window length. The design cost is also low: any device capable of running a Prism-based signal processing scheme may also adapt it to match changing requirements. Various signal processing tasks can be implemented using networks of Prisms, thereby providing a useful toolset for a wide range of sensing and monitoring applications, including the IoT.

Introduction

The Internet of Things (IoT) [1] and Industrie 4.0 [2] propose substantial increases in the deployment of sensors - for collecting, processing and communicating measurement data in real time - in diverse working environments. The challenges in realising this ambition are considerable. Problems associated with complete Cyber-Physical Systems (CPS) have implications for the design and operation of networked sensors [3]: "As systems become more interconnected and diverse, architects are less able to anticipate and design interactions among components, leaving such issues to be dealt with at runtime. Soon systems will become too massive and complex for even the most skilled system integrators to install, configure, optimize, maintain, and merge."

The solution, according to IBM's vision of Autonomic Computing [3], is to develop systems with advanced capabilities, including self-configuration, self-optimisation, self-healing and self-protection. This places a unique burden on the sensors in any system:

- Sensors are required to diagnose themselves as far as possible [4-6] (although this task can be augmented by higher level modelling and fault detection);
- Sensors are required to provide data for higher level system diagnostics via their conventional measurement processes (this is of course the default understanding of system diagnostics);

• Sensors are also ‘windows onto the process’: typically the transducer can generate higher bandwidth than is used by the default measurement process. It has long been recognised that high frequency data might contain useful process diagnostic information. However, the default measurement procedure will typically filter out such detail, and the sensor’s external communication bandwidth is unlikely to support the transmission of raw data for external processing. Thus the processing of such extra-functional data would require access to internal computing resources, specified and directed by a higher level system component.

If communication bandwidth constraints require that “the signal will be processed entirely [at] the Point of Acquisition (PoA) [7]” (i.e. within the sensor itself), then the demands placed on sensor software are potentially very high indeed. It is certainly possible to develop sophisticated diagnostics and related functionality within commercial sensors [8]. For example over the last two decades, Coriolis mass flow metering has advanced significantly [9]: where once liquid/gas mixtures would induce operational failure, recent developments include a three-phase flow metering capability for the upstream oil and gas industry [10]. However, such improvements usually require a high level of specialist expertise, which may add significantly to the cost of delivering a new commercial sensor, and may only be justifiable for high volume and/or high value sensors.

In reality, as the demand for a wider range of IoT sensors increases, the available design expertise will arguably be diluted. For engineers developing new specialised transducers (e.g. micro-machined or chemical sensor array), the subsequent signal processing and software engineering generally may not be the primary focus, a problem identified more generally in the context of Industrie 4.0:

“… often software engineering is the last activity after mechanical and electrical design, facing a lack of information and limited development time because of delays in the other disciplines. On the other hand, bugs created in other disciplines need to be fixed by means of software [2].”

A solution to the challenge of IoT sensor signal processing is likely to mimic solutions to other complex problems already navigated in more advanced fields, often achieved by an adoption of networked, as opposed to monolithic, technologies. For example, high performance computer hardware is now created using modular architectures of repeated units (e.g. processors, GPUs or FPGA logic blocks) in networks:

this strategy generally outstrips industry's capability to develop ever more complex monolithic processor architectures. Recent breakthroughs in Artificial Intelligence are associated with Deep Neural Nets [11]. Of course the internet itself, and the IoT, are inherently networked technologies of enormous complexity and capability. Experience suggests that an appropriately organised network of relatively simple processing elements may often be able to outperform a single monolithic system, however complex.

While networking technologies such as FPGAs and neural nets have been deployed in sensor applications for many years [12-14], current sensor signal processing algorithms could fairly be described as monolithic: fixed algorithms are applied to data streams based on fixed or tuned parameters, with little scope for adaptation or flexible interaction with other system components. Sophisticated sensor self-diagnostics typically result in yet more specialised, design-specific code, rather than greater flexibility. Standards such as IEEE 1451 [15, 16] primarily provide networking frameworks into which monolithic code is placed: the network shares the results of the processing that has been carried out in isolation, but the network cannot readily adapt or distribute the signal processing tasks themselves.

Sources of signal processing inflexibility include the characteristics of basic building blocks such as Finite Impulse Response (FIR) filters. These are widely used in sensing and condition monitoring applications due to their numerical stability and linear phase characteristics. Their disadvantages include a high computational overhead, especially where a narrow transition is required between the pass band and the stop band, and the cost and complexity of filter design. FIR is usually understood to mean non-recursive i.e. the filter calculation is performed in full each time step. Sophisticated techniques have been developed (e.g. [17]) to structure FIR filters to include partially recursive calculations so that some results are carried over between time steps. These methods offer a reduction in computational load at the expense of a more complicated filter design process. Arguably, the limitations of FIR filters are becoming more significant. While Moore's law continues to provide faster processors and ADCs, corresponding improvements in measurement precision make slower progress. While the IoT requires high flexibility and low compute budgets, only rarely is FIR filter redesign (e.g. in response to changing process conditions and/or data processing requirements) achievable within field devices. So, the high design cost of most conventional filtering, coupled with the high computational cost of FIR filtering, contribute significantly to the monolithic

nature of a great proportion of sensor signal processing, and these costs form a major barrier to the implementation of adaptable IoT data processing.

This paper introduces the Prism, a signal processing module, as a suitable network technology for a wide range of sensor data processing tasks. Specifically, networks of Prisms can be assembled, whether at design time or autonomously in real time, to carry out standard signal processing tasks such as low pass, band pass and notch filtering, or tracking the frequency, phase and/or amplitude of one or more sinusoids within real-time signals. Both the design and real-time computational costs of Prism signal processing are low, thus offering benefits to both conventional sensor systems and especially to IoT devices.

Description of the Prism

The Prism (Figure 1) is a dual-layered signal processing object consisting of six networked integration blocks. Each block accepts an input time series (i.e. a sequence of values, with new values supplied at the sampling rate *fs*) and generates a corresponding output time series at the same rate. Each new input value is multiplied by the latest value of a modulating function with frequency *hm* Hz. The resulting product value is stored in the integration block along with all previous product values over the last 1/*m* seconds. The integral of these product values is calculated, and this result gives the corresponding output value for the block for the current time step. This entire calculation is performed in each block for every new input value.

The characteristic frequency *m*, together with the sampling rate *fs* and the harmonic number *h*, fully determine the Prism properties. Given the desired values of these parameters, the only 'design' effort required to instantiate the corresponding Prism is to calculate the linearly spaced sine and cosine values of the modulation functions – hence in principle any device capable of running a Prism can also design one.

The mathematics of the Prism permits recursive calculations within each integration block: the newest product value is computed and included in the updated integral value while the oldest product value is removed. Hence, only one multiply and accumulate operation is required in an integration block for each new input, so that the overall computational burden is low and fixed, irrespective of the number of samples in the data window of each integral.

For a sinusoidal input $s(t) = A\sin(2\pi ft + \phi_i)$ with amplitude A, frequency f and initial phase ϕ_i, each of the second stage integrals can be described mathematically as:

$$I^h_{[s|c][s|c]} = \quad m^2 \int_{-\frac{1}{m}}^{0} [\sin|\cos](2\pi hmt + q)\left(\int_{t-\frac{1}{m}}^{t} [\sin|\cos](2\pi hmt + q)\cdot s(t)dt\right)dt \tag{1}$$

The subscript notation [s | c] indicates the selection of one alternative between sine [s] and cosine[c] as the modulation function in the corresponding integral. The superscript h is the harmonic number, a positive integer giving the number of modulation function periods (each of duration 1/hm seconds) occurring within the integral period (duration 1/m s). In any digital implementation, it is a requirement that fs/m is an integer, so that the integration period corresponds to a whole number of samples. q is the (arbitrary) initial phase of the modulation function. Members of this integral family can be combined to form groups having simple analytic expressions. Specifically:

$$G^h_s = I^h_{ss} + I^h_{cc} = A\,\mathrm{sinc}^2(r)\frac{r^2}{r^2 - h^2}\sin(\phi(t) - 2\pi r) \tag{2}$$

$$G^h_c = I^h_{cs} - I^h_{sc} = A\,\mathrm{sinc}^2(r)\frac{hr}{r^2 - h^2}\cos(\phi(t) - 2\pi r) \tag{3}$$

where the instantaneous phase $\phi(t) = 2\pi ft + \phi_i$, $r = f/m$, the frequency ratio, and sinc is the 'sampling function', which arises frequently in signal processing and the theory of Fourier transforms. Note that $G_s{}^h$ and $G_c{}^h$ form an orthogonal (i.e. sine/cosine) pair, having the same linear phase delay $2\pi r$. Note also that here the subscripts s and c refer to the analytic forms (either sine or cosine) of the corresponding output functions $G_s{}^h$ and $G_c{}^h$. While the numerical evaluation of double integrals is generally expensive, these particular groups can be evaluated recursively as follows. Equations (2) and (3) remain true irrespective of the initial phase q of the modulation function in (1). Accordingly, it is possible to evaluate each integral in a 'sliding window' arrangement whereby only the oldest product value (i.e. data × modulation function) is removed and the newest product computed for inclusion in the updated integral value. Using this approach, it is possible to evaluate $G_s{}^h$ and $G_c{}^h$ efficiently. Numerical integration error may be significantly reduced by applying Romberg Integration (RI) [18], adapted to operate on time series data, at the expense of some additional computation and further constraints on the integral length in samples. Using RI, at each time step

a series of integral values are formed, based on different subsets of the data points, and from which a weighted sum is calculated to provide a best estimate. This technique is particularly effective for low noise input signals.

Figure 2 shows how the gains of G_s^h and G_c^h vary with frequency for h = 1, 3, 5 and 7. These gains are obtained mathematically simply by removing the sine and cosine terms from equations (2) and (3) respectively, and scaling relative to the original amplitude, A. Gains are negative for frequencies below hm Hz and positive thereafter. The lower plot shows the relative gain whereby each function is scaled by its maximum absolute value and plotted on a decibel scale. For every function, notches occur at all multiples of m (including DC). $G_s^1$and G_c^1 each have a broadly low pass characteristic. For higher values of h, G_s^h has a broadly bandpass characteristic centred around hm Hz; G_c^h behaves similarly, but has an additional high (negative) gain region below m Hz.

Having outlined the operation of the Prism, a comparison can be made with previously developed recursive filtering techniques. Conventional IIR filters ([19]) are widely used and have low computational cost, but typically have a non-linear phase response and are vulnerable to numerical instability. Partially recursive FIR filters can be created by identifying segments of the impulse response suited to polynomial approximation ([17, 20]). Similar techniques, often combined with median filters, can be applied for edge detection in image processing [21]. In the latter two cases partially recursive FIR calculations (i.e. with a modest reduction in computational cost) are obtained at the expense of a more complex filter design process.

Tables 1 and 2, together with Figure 3, show the advantages offered by Prism signal processing over conventional FIR filtering, based on the following example: consider a resonant transducer operating at 150 Hz, with a signal-to-noise ratio of (say) 80 dB, where its frequency, amplitude and phase are to be tracked. What computing cost (upper plot), and precision benefit (lower plot) arise from steadily increasing the sampling rate from 9.6 kHz up to 9.6 MHz? A natural first step is to filter the data to remove high frequency noise. As shown in Table 1 below, a data window covering exactly one period of the transducer signal (i.e. 6.67 ms) has a length of 64 samples at 9.6 kHz but 64,000 samples at 9.6 MHz.

Sample rate (kHz)	Filter order (samples)	Equiripple Filter Design Time (s)
9.6	64	0.021
96	640	0.508
960	6,400	54.08
9600	64,000	N/A

Table 1: MATLAB design time for Equiripple filter

Table 1 also shows the time required by MATLAB filter design function 'firpm' to create the desired Equiripple filter, given its order and sample rate. The selected passband is 150 Hz and the stopband is 300 Hz in each case. These and other results are obtained on a 2.5 GHz i7-2860 laptop with 16GB of RAM running 64 bit Windows 7. The design time rises rapidly with filter order, and for the 9.6 MHz sampling example, the system is unable to complete the design. While undoubtedly such a filter could be designed using other tools and/or with relaxed design criteria, Table 1 illustrates the significant design effort required for many conventional FIR (and other) filters. This high design costs forms a significant barrier to flexible signal processing, as anticipated by the IoT.

By contrast, the equivalent design requirement for a Prism is simply to calculate the linearly spaced sine and cosine values for the modulation function in Eq (1), given the values of fs, m (here m = 300 Hz) and h (here h = 1). This is a fast and simple task readily undertaken in any modest computing device, for example a self-configuring IoT node.

Sample rate (kHz)	Filter order (samples)	Equiripple filter time per sample (s)	Prism filter time per sample (s)
9.6	64	6.23e-8	2.46e-8
96	640	5.59e-7	2.48e-8
960	6,400	5.54e-6	2.52e-8
9600	64,000	(5.50e-5)	2.58e-8

Table 2: Processing time per sample for Equiripple and Prism filtering

Table 2 shows the processing time required to filter a single sample for the Equiripple and Prism methods, using single threaded C++ code. As expected, for the non-recursive FIR Equiripple filter, the compute time increases linearly with filter order: this trend has been extrapolated for the 9.6 MHz sampling case. By contrast the Prism filtering time is essentially constant.

Figure 3 extend these results to show the total real-time computational burden for each filter type, accounting for the increasing sampling rate. The Equiripple filter (black line) shows a quadratic increase in computing power with sampling rate: at 9.6 kHz, the filter requires approximately 0.05% of an i7 core, while at 9.6 MHz, the filter is projected to require approximately 500 i7 cores. This is as expected: the computation requirement per sample increases linearly with sample rate (Table 2), so the total computational load will increase quadratically. By contrast, the Prism (green line) has an effectively constant computation requirement for each new sample, irrespective of the window length in samples (Table 2), and therefore only a linear increase in total load with sample rate. At 9.6 kHz the computational effort to perform real-time Prism filtering is approximately 0.03% of an i7 core, and this increases to only 30% at 9.6 MHz.

One might reasonably ask: why sample at such high rates for a low bandwidth signal? Higher sampling rates may often deliver higher measurement precision, depending upon the particular structure of the input signal; they may also deliver faster measurement updates, leading to an improved dynamic response of the signal processing system or indeed new applications, as illustrated below.

For a sensing or data processing technology that currently uses FIR filtering, the Prism offers the prospect of an immediate performance improvement, equivalent to moving horizontally from the black line to the green line in Figure 3. For a given computational budget (y-axis), higher data throughput (x-axis) becomes possible, affording higher precision and/or a faster dynamic response. As Moore's law continues to deliver more powerful computing engines, Prism-based technologies can convert higher processing budgets into application performance improvements on a linear, rather than a quadratic, basis, thus maintaining or extending competitive advantage. In the IoT context, downsampled Prism data, from which high precision results may be recovered, could form the basis of a low bandwidth data communication protocol to facilitate network-based calculations.

Filtering Examples

With its range of properties – varying gain, regular notches, and orthogonal outputs – the Prism is like a signal processing Swiss Army knife, applicable to a wide range of tasks. Lowpass, bandpass and notch filters can be constructed as networks of Prisms, where simple rules can be used to determine the values of

m and *h* for each. Because the design cost is also low, the adaptation or reconfiguration of Prism signal processing schemes can be readily achievable in relatively low-power IoT sensor networks.

For example, Fig. 4 illustrates how bandpass filters can be constructed using pairs of Prisms in series. Fig. 4a shows the variation in gain against frequency for the Prism output G_s^h with $h = 1, .., 12$. For each *h* there is a positive and a negative peak, with the negative peak occurring in the range [*h*-1, *h*]× *m* Hz while the positive peak appears in the range [*h*, *h*+1]× *m* Hz. Fig. 4b shows how, by selecting the appropriate value of *m* in each case, a family of G_s^h functions can be created with either a positive or negative peak at some arbitrary frequency, here 100 Hz. For each value of *h* a pair of G_s^h functions is created: one with its positive peak positioned at 100 Hz and the other with its negative peak at 100 Hz. Fig. 4c shows the results of concatenating such Prism pairs. A sequence of bandpass filters are produced with, for increasing *h*, a narrowing passband around the chosen central frequency. Further pairs of Prisms may be networked in series to increase the attenuation of frequencies outside the desired passband. For example Fig 4d shows the performance of a bandpass filter consisting of 6 Prisms in series. Here the sampling rate *fs* is 48 kHz, and an arbitrary central frequency of 1234.5 Hz has been selected. Pairs of Prisms with *h* values of 500, 333 and 250 have been networked to create a filter with a passband (to -10 dB) of ± 0.5 Hz. The total filter length is 168,448 samples, but the computational burden is low, as only six Prism evaluations are needed for each sample, and only the G_s^h output is calculated for each Prism. Fig 4d also shows that, when applied numerically to white noise, the filter delivers its theoretical performance. The computational requirement is 1.67e-7s per sample, so that the real-time 48 kHz throughout requires only 0.8% of an i7 core. Perhaps more importantly, the design and instantiation of such a bandpass filter is straightforward, so that any moderately powerful IoT device could create one or more such filters, adjusting the central frequency and/or the pass bandwidth upon request or with changing environmental conditions. Note that while the computational burden for this filter is low, its sample length is high and hence the dynamic response is relatively slow. Improving the design efficiency of Prism-based bandpass filters (i.e. selecting the number of Prisms and their desired *h* and *m* values, assuming *fs* is fixed by the application), while ideally still retaining design simplicity, is a future research topic.

A new technique, dynamic notch filtering (DNF), is made possible by the absence of a conventional flat passband in the Prism frequency response. Fig. 5a shows two Prisms having identical values of m but with h = 1 and 2 respectively. In each case only the G_c^h output is generated. These outputs have matched phase at all frequencies, but different gains. Accordingly, a selected frequency component can be notched out by forming a weighted sum of the two G_c^h outputs, where the weighting is the ratio of the respective gains at the notch frequency. This dynamic notching can be done sample by sample at low cost, and where the notch frequency can be selected in real time, so that components with varying frequencies may be successfully separated and tracked. Different weightings, each notching a separate frequency, can be generated from the same Prism outputs, so that it is possible to split a multi-component signal into a corresponding set of single component signals to be tracked individually. In Fig. 5a, two signal components are split using DNF while Figs. 5b – 5d provide a simulation example. Fig 5b shows a two component input with frequencies at 91 Hz and 75 Hz, while Figs. 5c and 5d show the resulting component separation using m = 200 Hz. A three-component DNF example is described in [22], with a pressure sensing application in which multi-component ultrasonic pulses are decomposed into individual frequency components for diagnostic purposes.

Application Example

Prism signal processing is being used in a new Coriolis mass flow meter prototype, providing a substantial increase in the measurement update rate and opening up new application areas (Figure 6). The Coriolis measurement principle is that the vibration of an oscillating flowtube (transducer) is monitored using two sensors, each producing nominally sinusoidal data. The mass flow rate is a function of the phase difference between the two sensor signals. In current commercial devices, flow update rates are typically no faster than 10 – 100 Hz; a previous 'ultrafast' prototype generated updates at 1.5 kHz [23]. The new Prism-based prototype has been developed using a commercial flowtube coupled to a dual ARM core based transmitter (compute engine). This is capable of generating flow measurement updates at 48 kHz, and is being used to monitor laboratory engine fuel injection [24], a previously unimaginable application. In Figure 6, short (1.5 ms) pulses of diesel fuel are injected through the meter at an equivalent engine speed of 1800 rpm and at 500 bar pressure. The mechanically noisy test environment excites additional modes of vibration of the flowtube, contaminating the sensor signals with unwanted frequency components. Using Prism signal processing, the

frequency, amplitude and phase of each sensor signal is tracked at 48 kHz, as is the phase difference between the sensors. Without pre-filtering, the resulting measurement is too noisy to detect the fuel pulses (grey line). Including Prism-based notch filtering in the signal path removes the unwanted components and reveals each fuel pulse (black line).

The Prism applied to the IoT

The IoT presents significant challenges to the design and implementation of industrial sensors, as discussed in the introduction. Prism signal processing addresses a number of these challenges, offering various advantages over well-established FIR techniques: a recursive calculation enabling low computational burden, design simplicity, and a variety of filtering and tracking techniques. Prism technology can thus provide effective use of available computational power and simple signal processing design for new sensors, thus reducing the burden on available design expertise. It offers a network-style adaptability: any field device capable of running Prisms can also design them, so that new signal processing schemes can be created ad hoc as requested or required within the device or a sensor network. This also affords a more open approach to algorithmic development: the sensor can be viewed as a local processor of (relatively high bandwidth) transducer data, where the exact data analysis required can be decided in real-time and/or in the application context, rather than permanently fixed by the sensor designers in advance. Finally, the Prism offers an effective means of data compression via down-sampling to preserve communication bandwidth.

The Coriolis meter provides one example of the substantial increase in measurement update rate achievable using Prism signal processing, but the same techniques could benefit a wide range of other sensing and monitoring applications. It is particularly suited to signals with one or more discrete frequency components, where the frequency and/or amplitude properties may vary with time, such as the measuring of resonant sensors or the monitoring of electromechanical systems. Potential applications may be found in a wide range of IoT (and indeed conventional) fields such as industrial automation, transportation, healthcare, electrical power, condition and environmental (e.g. gas sensing [25]) monitoring, and the smart home [26].

Future Work

A patent has been filed by Oxford University, and the first Prism-based commercial products are planned to launch in 2018. While there is ample scope for using the current body of Prism signal processing techniques in a wide range of applications, topics for further research include more efficient bandpass filtering design, and the rapid and accurate tracking of dynamically changing sinusoidal signals.

References

[1] M. Wollschlaeger, T. Sauter, and J. Jasperneite, “The Future of Industrial Communication”, IEEE Industrial Electronics Magazine, pp. 17–27, March 2017. DOI : 10.1109/MIE.2017.2649104

[2] B. Vogel-Heuser, D. Hess, “Guest Editorial - Industry 4.0-Prerequisites and Visions”, IEEE Transactions on Automation Science and Engineering, Vol 14, No. 2, April 2016. DOI: 10.1109/TASE.2016.2523639.

[3] J. O. Kephart and D. M. Chess, “The vision of autonomic computing”, IEEE Computer, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[4] M.P. Henry and D.W. Clarke, “The Self-Validating Sensor: Rationale, Definitions and Examples”, Control Eng. Pract., vol. 1, pp 585–610, 1993.

[5] BSI, “BS7986:2005, Specification for data quality metrics for industrial measurement and control systems”, British Standards Institute, 2005.

[6] R. Taymanov, K. Sapozhnikova, I. Druzhinin, "Sensor Devices with Metrological Self-Check", Sensors & Transducers Journal, Vol. 10, pp. 30-45, February 2011.

[7] G. Monte, V. Huang, P. Liscovsky, D. Marasco, and A. Agnello,“Standard of things, first step: Understanding and normalizing sensor signals,” Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE, Vienna, 2013, pp. 118-123. DOI: 10.1109/IECON.2013.6699121

[8] M.P. Henry and G.G. Wood, "Sensor Validation: principles and standards", atp International 3, No. 2, pp 39-52 , 2005.

[9] T. Wang, R. Baker. “Coriolis flowmeters: a review of developments over the past 20 years, and an assessment of the state of the art and likely future directions”, Flow Measurement and Instrumentation vol. 40, pp99–123, 2014. DOI: http://dx.doi.org/10.1016/j.flowmeasinst.2014.08.015

[10] M Henry, M Tombs, F Zhou ,“Field experience of well testing using multiphase Coriolis metering”,. Flow Measurement and Instrumentation, vol. 52, pp 121–136, 2016. DOI: http://dx.doi.org/10.1016/j.flowmeasinst.2016.09.014

[11] G. Hinton et al., "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups", IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82-97, Nov. 2012. DOI: 10.1109/MSP.2012.2205597

[12] M.E. Zamora, M.P. Henry, "Digital control of a Coriolis mass flowmeter", IEEE Transactions on Industrial Electronics 2008; 55:2820–2831.

[13] T. Grimm, B. Janßen, O. Navarro and M. Hübner, "The value of FPGAs as reconfigurable hardware enabling Cyber-Physical Systems," IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), Luxembourg, 2015, pp. 1-8. DOI: 10.1109/ETFA.2015.7301496

[14] R.P. Liu, M.J. Fuent, M.P. Henry, M.D. Duta. "A neural network to correct mass flow errors caused by two-phase flow in a digital Coriolis mass flowmeter", Flow Meas. Instrum., vol 12, pp 53–63, 2001.

[15] E. Song and K. Lee, "Understanding IEEE 1451 - Networked Smart Transducer Interface Standard", IEEE Instrumentation & Measurement Magazine, April 2008, pp 11-17.

[16] A. Kumar, V. Srivastava, M. Singh and G. Hancke, "Current Status of the IEEE 1451 Standard Based Sensor Applications", IEEE Sensors Journal, pp. 2505-2513, 2014.

[17] R. Lehto, T. Saramäki, O. Vainio, “Synthesis of Narrowband Linear-Phase FIR Filters With a Piecewise-Polynomial Impulse Response”, IEEE Trans. Circuits and Systems, vol. 54, no. 10, p 2262-2276, 2007.

[18] J. Dutka, "Richardson Extrapolation and Romberg Integration", Historia Mathematica, Vol. 11, pp 3-21, 1984.

[19] Steven W. Smith, “Chapter 19: Recursive Filters”, The Scientist and Engineer’s Guide to DSP”.

[20] R. Lehto, T. Tauren and O. Vainio, “Recursive FIR Filter Structures on FPGA”, Microprocessors and Microsystems, V35, Issue 7, pp 595-602, Oct. 2011.

[21] L.J. Morales-Mendoza, et.al., “A New Recursive Scheme of the Unbiased FIR Filter to Image Processing”, Procedia Engineering 35 (2012) 202 – 209.

[22] M. Henry, O. Bushuev, O. Ibryaeva. “Prism Signal Processing for Sensor Condition Monitoring”, 26th IEEE International Symposium on Industrial Electronics, Edinburgh, UK, 2017.

[23] C. Clark, M. Zamora, R. Cheesewright, M. Henry. “The dynamic performance of a new ultra-fast response Coriolis flow meter”, Flow Measurement and Instrumentation, vol. 17, no. 6, p391–8, 2006. DOI: https://doi.org/10.1016/j.flowmeasinst.2006.07.002

[24] F Leach, S Karout, F Zhou, M Tombs, M Davy, M Henry. “Fast Coriolis mass flow metering for monitoring diesel fuel injection”, Flow Measurement and Instrumentation. Available online Sept 20 2017. http://dx.doi.org/10.1016/j.flowmeasinst.2017.09.009

[25] R. A. Potyrailo. “Multivariable Sensors for Ubiquitous Monitoring of Gases in the Era of Internet of Things and Industrial Internet”, Chem. Rev. 2016, 116, 11877−11923. DOI: 10.1021/acs.chemrev.6b00187

[26] A Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash. “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications”, IEEE Commucation Surveys and Tutorials, Vol. 17, No. 4, 2015. DOI: 10.1109/COMST.2015.2444095

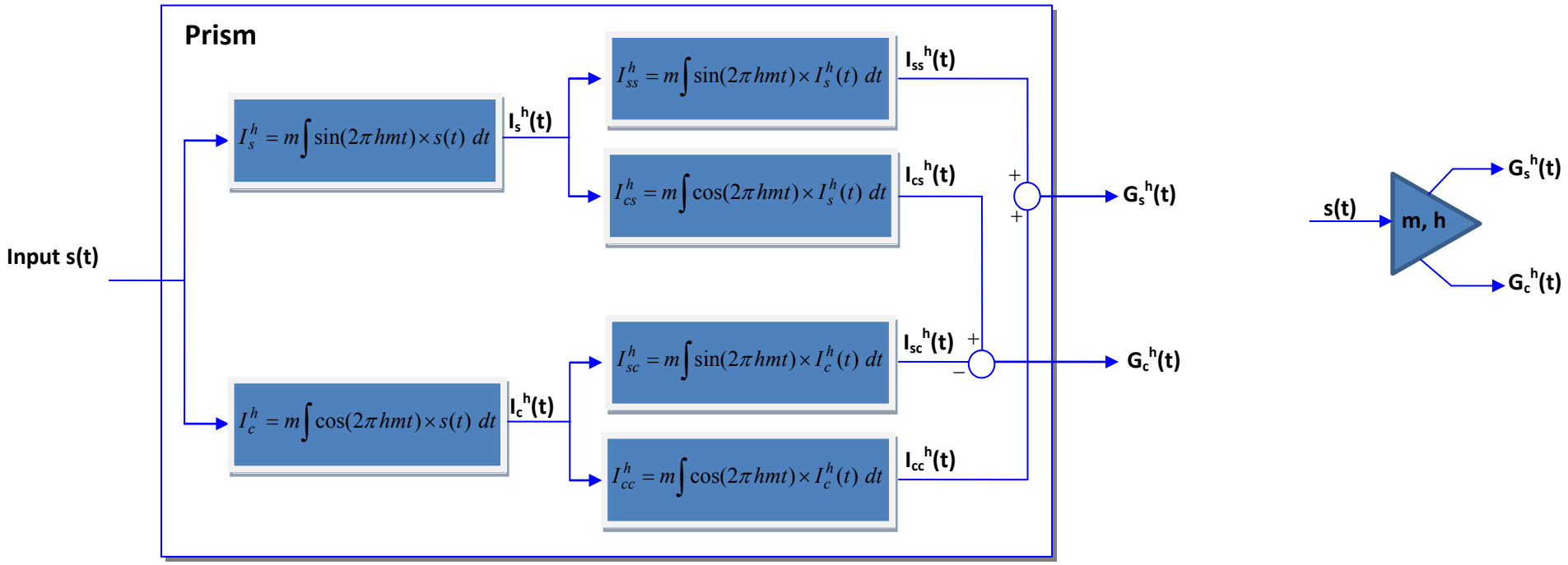


Figure 1: Structure of the Prism signal processing object. Right: Prism symbol.

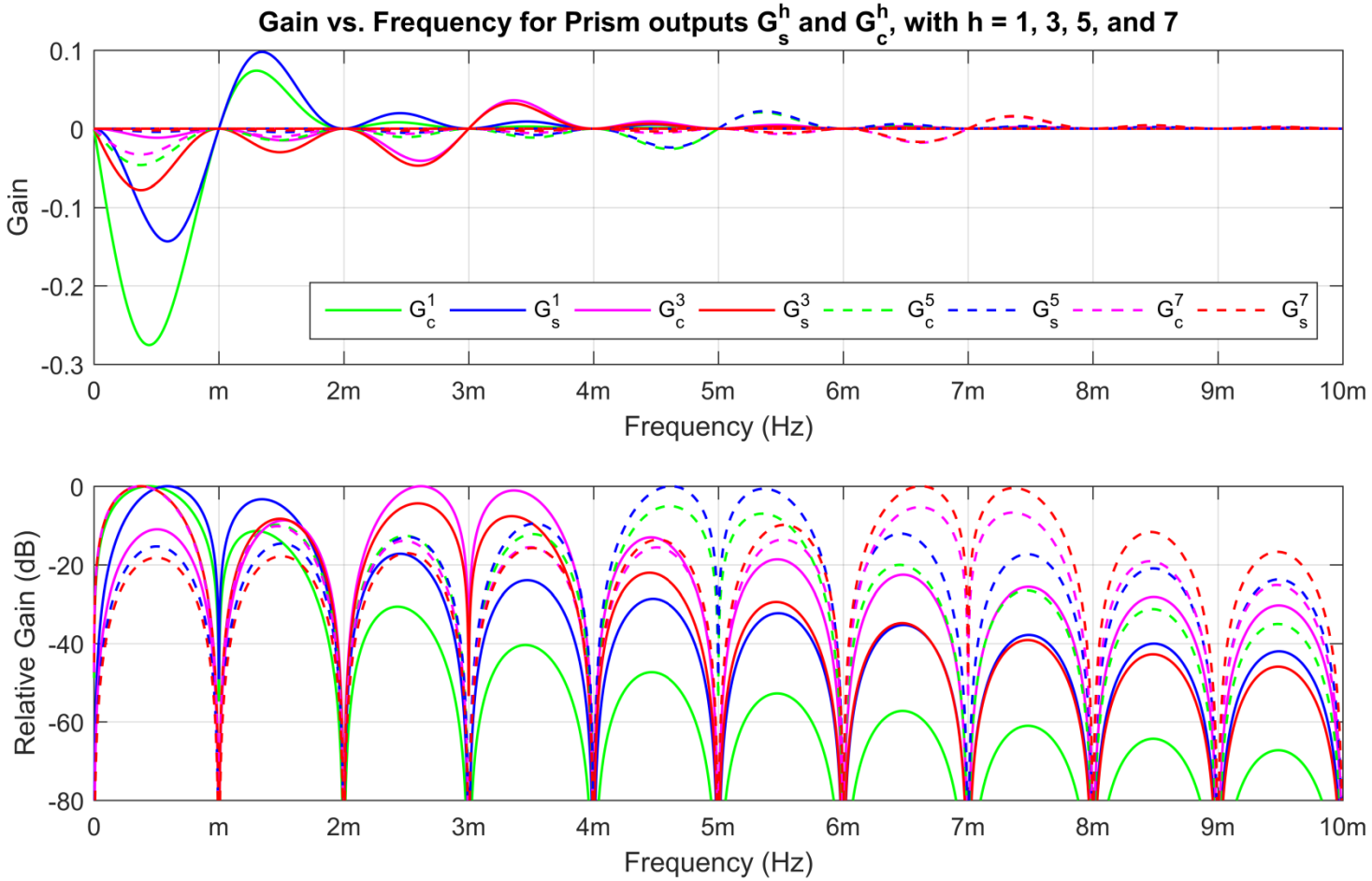


Figure 2: Gains of Prism outputs G_s^h and G_c^h against frequency, for h = 1, 3, 5 and 7.

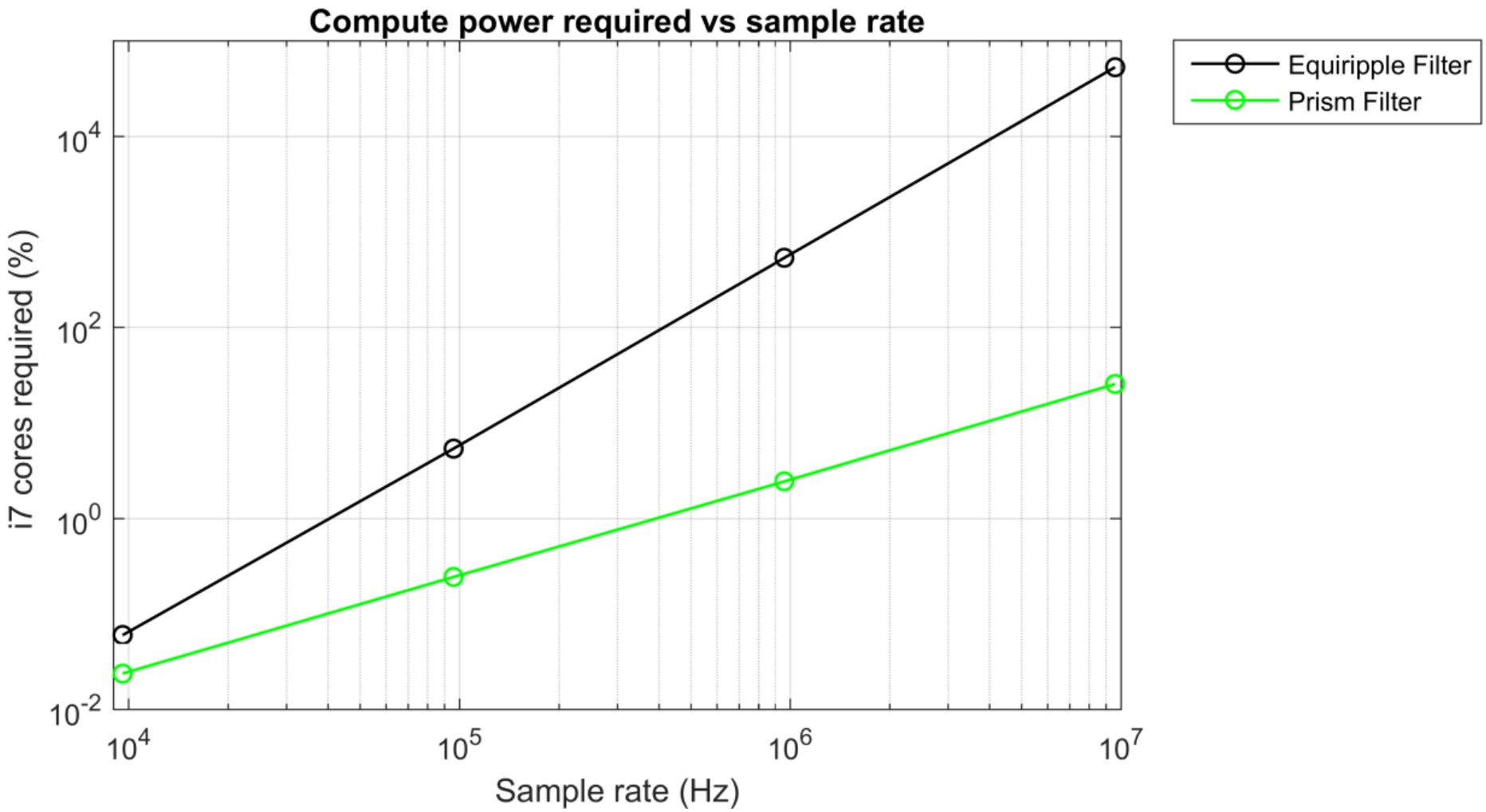
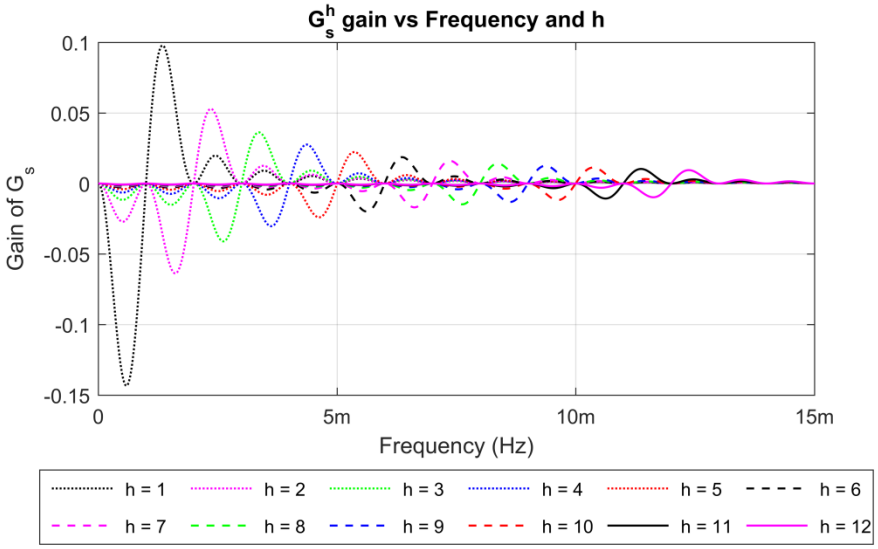
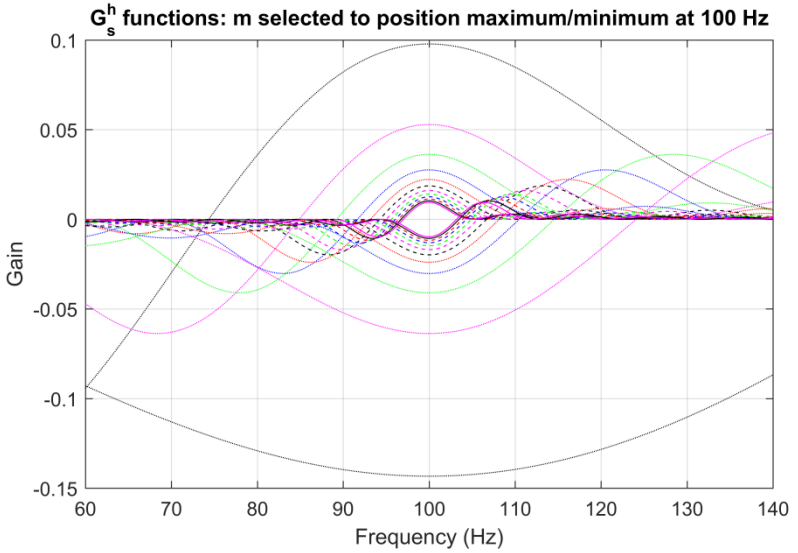


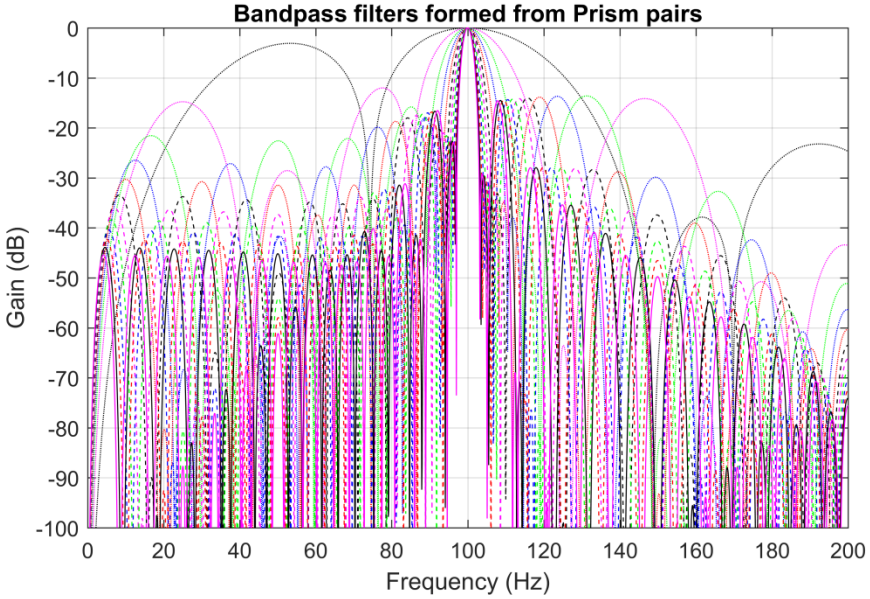
FIGURE 3 – The impact of the increased sampling rate on the real-time computational requirement.



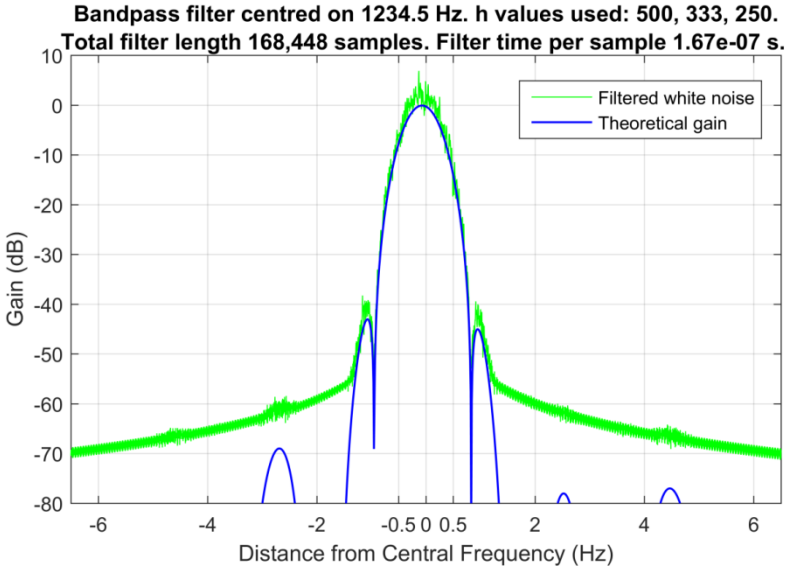
(a)



(b)



(c)



(d)

Figure 4. Prism-based bandpass filtering.

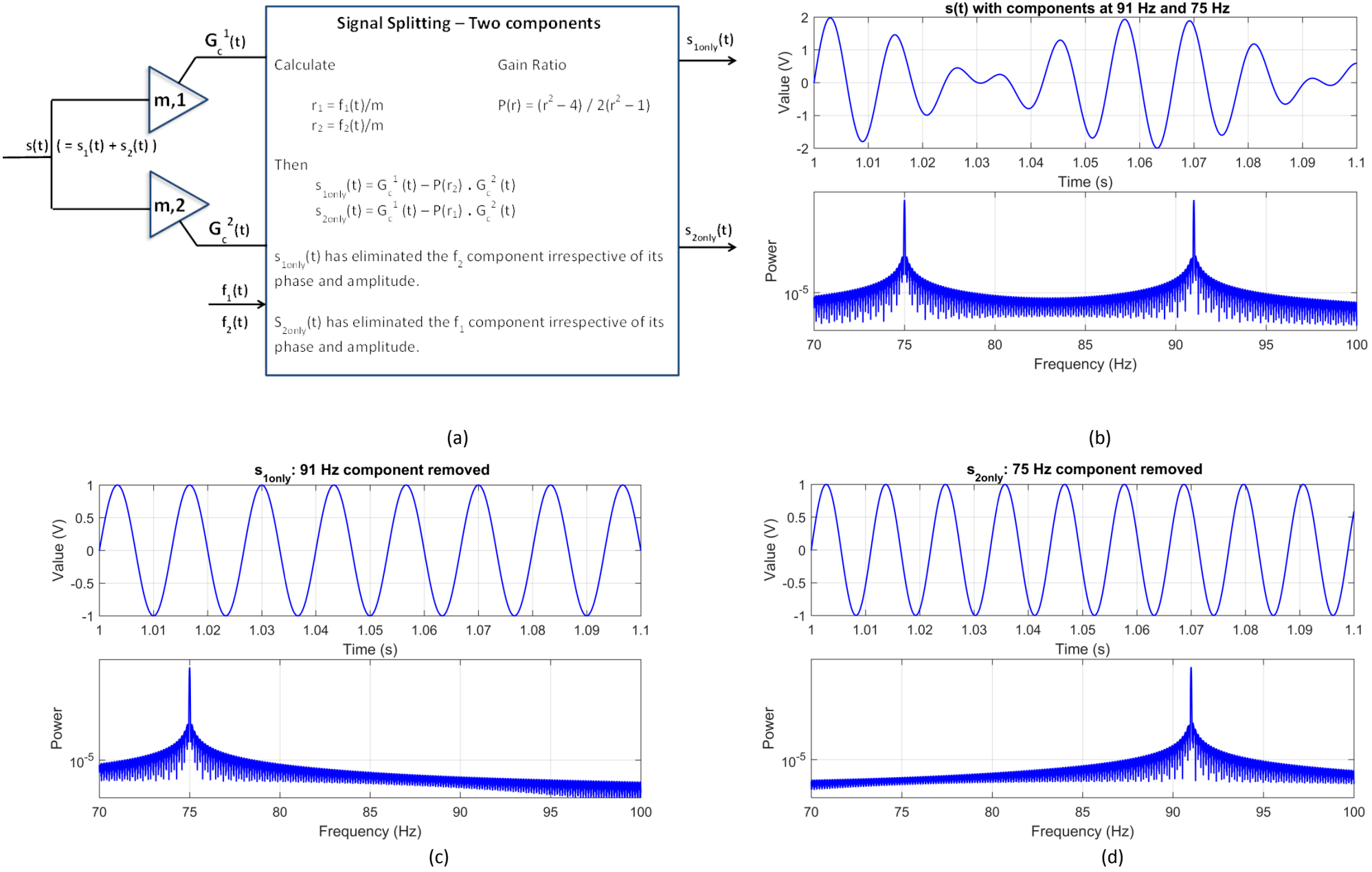


Figure 5. Dynamic notch filtering.

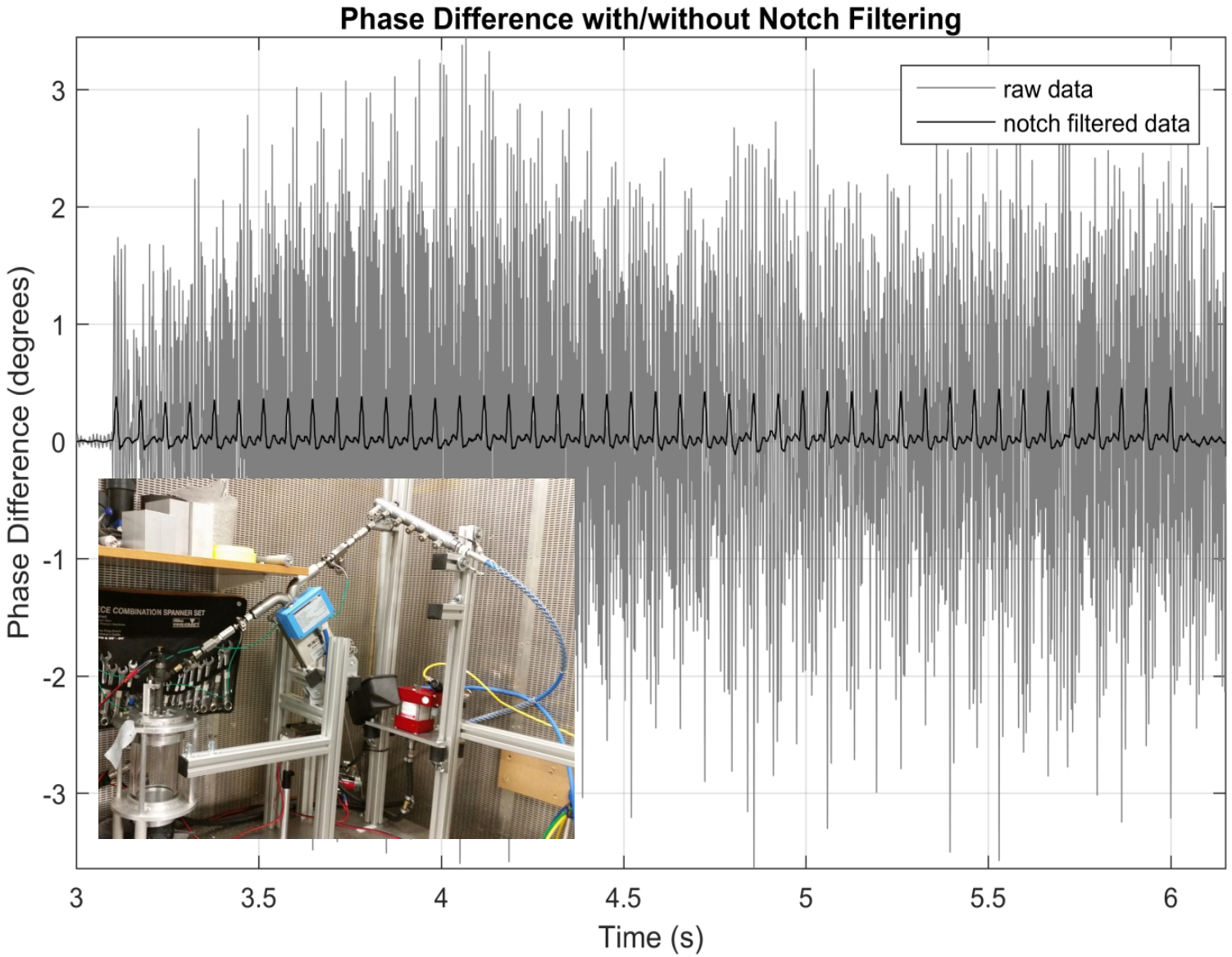


Figure 6: Prism-based Coriolis metering of diesel fuel injection pulses.