TAMPERE UNIVERSITY OF TECHNOLOGY

Petteri Aimonen

Design of an embedded system for video performance measurements

Master of Science Thesis

Examiners: Prof. Timo D. Hämäläinen,
Dr. Erno Salminen
Examiners and topic approved by the
Faculty Council of the Faculty of
Computing and Electrical Engineering
on 5.2.2014

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Signaalinkäsittelyn ja tietoliikennetekniikan koulutusohjelma
PETTERI AIMONEN: Sulautetun mittalaitteen suunnittelu videomittauksia varten
Diplomityö, 64 sivua, ei liitesivuja
Helmikuu 2014
Pääaine: Digitaali- ja tietokonetekniikka
Tarkastajat: Prof. Timo D. Hämäläinen, TkT Erno Salminen
Avainsanat: videotoiston suorituskyky, kuvataajuus, käyttäjäkokemus, sulautettu järjestelmä

Tämä diplomityö kuvaa videomittauksiin käytettävän mittalaitteen suunnitteluprosessin. Kuvatun laitteen tarkoituksena on auttaa videotoiston optimoinnissa toistolaitteiden, kuten taulutietokoneiden, älypuhelimien ja televisioiden, suunnitteluprosessin aikana. Laite mittaa videon kuvataajuuden ja laskee mahdollisesti toistovaiheessa puuttumaan jääneiden kuvien lukumäärän käyttäen anturia, joka asetetaan toistolaitteen näyttöä vasten.

Ominaisuus, joka erottaa tämän mittalaitteen muista yleisistä tavoista mitata näytön kuvataajuus on se, että se toimii täysin mitattavan laitteen ulkopuolelta. Mitattavaan laitteeseen tai sen ohjelmistoon ei siis tarvita mitään muutoksia, ja kaikki laitteistotason ilmiöt, joita voi esiintyä esimerkiksi näytönohjaimessa, saadaan mitattua. Näin se mahdollistaa kattavamman analyysin toistolaitteen toiminnasta, ja mahdollistaa tulosten vertailun myös kilpailijoiden laitteisiin.

Mittalaitteen toteuttamiseksi suunniteltiin oma STM32F407-mikrokontrolleriin perustuva laitteistoalusta. Koska kohdemarkkinan tarpeet muuttuvat nopeasti uusien laitesukupolvien myötä, suunniteltiin mittalaite niin, että sitä voidaan helposti laajentaa esimerkiksi uusilla antureilla.

Mittausten tarvitsema signaalinkäsittely toteutettiin kokonaan ohjelmallisesti, jotta se olisi mukautettavissa eri tehtäviin. Tämä asettaa nopeusvaatimuksia laitteen ohjelmistolle, ja niiden täyttämiseksi ohjelmisto rakennettiin NuttX-reaaliaikakäyttöjärjestelmän päälle.

Projektin tuloksia arvioidaan sekä tavoitteiden saavuttamisen kannalta, että asiakkailta ja myyntiosastolta saadun palautteen perusteella. Laitteen kehittäminen ensimmäisistä suunnittelupalavereista valmiiseen ohjelmistoversioon saakka kesti 9 kuukautta, ja karakterisointimittauksissa todettiin 1 ms mittaustarkkuus 150 FPS kuvataajuuteen asti.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Signal Processing and Communications Technology
PETTERI AIMONEN: Design of an embedded system for video performance measurements
Master of Science Thesis, 64 pages, no appendix pages
February 2014
Major: Digital and Computer Systems
Examiners: Prof. Timo D. Hämäläinen, Dr. Erno Salminen
Keywords: video playback performance, frame rate, user experience, embedded system

This master's thesis describes the design process of a measurement instrument for measuring the video playback performance. The purpose of the instrument is to aid in optimizing the video playback quality of devices such as tablets, smartphones and televisions. In particular, the instrument measures the frame rate and number of missing frames using a sensor attached to the display.

The aspect that sets this instrument apart from common ways of measuring display frame rates is that it operates outside the video playback device. This means that no modifications are required to the device being tested, and all effects that may occur in e.g. display hardware are accurately measured. In this way it allows more complete analysis of the device behaviour, and is also suitable for testing competitor's devices.

To implement the measurement instrument, a custom hardware platform based around the STM32F407 microcontroller is designed. In order to adapt to the rapidly changing needs and focuses of the market sector, the hardware is designed with maximal extensibility to allow easy attachment of new kinds of sensors.

The signal processing required for the sensor signals is implemented entirely in software, in order to allow future measurement tasks to use different algorithms. This places requirements on real-time performance on the hardware and software, and the NuttX real-time operating system is chosen as a basis for the firmware development.

A software architecture composed of signal processing libraries, low-level device drivers and GUI user interface is described. The architecture is designed to allow easy reuse of components to implement several measurement tasks.

Finally, the success of the project is evaluated both on basis of meeting the set goals and based on feedback received from the pilot customers and sales team. The development of the instrument took 9 months from the initial planning meetings to the first official software version. Characterization measurements have shown the instrument to measure up to 150 FPS frame rates at 1 ms resolution.

PREFACE

The purpose of this thesis is to describe the high-level choices and process involved in the design of an embedded measurement instrument. Due to trade secrets, the signal processing algorithms used are not described in detail, while their development was a significant portion of the work in this project.

This thesis was done for OptoFidelity Oy, a test automation and machine vision company based in Tampere, Finland. Selling world-wide, OptoFidelity has sales relationships with many of the leading companies in the mobile device market, and is constantly looking for new opportunities to aid in their customer's research and development process. The instrument described in this thesis forms a part of their product offering for measuring video playback performance.

I would like to thank OptoFidelity for providing this interesting and challenging topic, and especially Kimmo Jokinen for his encouragement and highly useful feedback during the project. The sales team also helped by providing insight into the market demands, and by arranging demos and test uses with pilot customers. Finally, I would like to thank my family for support during the stressful year that this project and thesis took to complete.

Jyväskylä 6.2.2014

Petteri Aimonen

CONTENTS

TERMS AND SYMBOLS

ADC

Analog to Digital Converter, a device that allows conversion of analog voltage levels into numeric values.

API

Application Programming Interface, a documented interface that can be used by software applications to communicate with each other.

CNC

Computer Numerical Control, a class of machining tools that are used to cut and form metal according to a 3-dimensional model stored on a computer..

DAC

Digital to Analog Converter, a circuit that converts digital signals to analog voltages.

DMA

Direct Memory Access, a hardware unit that can autonomously perform data transfer tasks to and from memory, freeing the main processor to execute other tasks.

DPI

Dots Per Inch, a measurement unit for the pixel density on a display.

Dropped frame

A frame that is missing in video playback, usually due to insufficient performance of the playback device.

EEPROM

Electrically Erasable Programmable Read Only Memory, a memory circuit that allows permanent storage of data, but can be erased and rewritten when necessary.

FIR

Finite Impulse Response, a type of digital filter where a finite number of previous samples are combined using multiplication and addition.

FPGA

Field Programmable Gate Array, a microchip that can be programmed to implement logical circuits.

FPS

Frames Per Second, a measurement unit for the rate of individual images in video content.

FPU

Floating Point Unit, a processor submodule that accelerates computations with floating point numbers.

Frame time

The length of time for how long a specific video frame is visible on the display.

GPIO

General Purpose Input Output, a term for microcontroller pins that can be directly controlled by software to perform either digital input and output.

GUI

Graphical User Interface, a computer program's interface composed of graphical elements, such as buttons, images and text areas.

HAL

Hardware Abstraction Layer, a component in an operating system that supports several kinds of hardware and provides a common interface for applications to use.

I2C

Inter-Integrated Circuit, a type of digital interface which uses two wires and pull-up resistors for serial communications.

IC

Integrated Circuit, an electronic circuit that has been implemented on a single microchip.

LCD

Liquid Crystal Display, a display technology based on the polarization change of liquid crystals.

LDO

Low Drop-Out, a term for voltage regulators that need only a small (usually 0.1–0.5 V) voltage difference to operate.

LED

Light Emitting Diode, a semiconductor device that emits light when a current flows through it.

MCU

MicroController Unit, another term for microcontroller.

Microcontroller

An integrated circuit that combines a microprocessor with memory and other peripherals on the same microchip.

MOSFET

Metal Oxide Semiconductor Field Effect Transistor, a type of transistor, used to control the flow of electrical current.

OLED

Organic Light Emitting Diode, a display technology based on an array of small light emitting diodes that act as the pixels of the screen.

OS

Operating System, a software platform that provides services such as task scheduling, which make it easier to implement applications.

PC

Personal Computer, a general purpose desktop or laptop computer, usually based on Intel x86 architecture.

PCB

Printed Circuit Board, usually a fiber glass board with etched copper conductors for mounting components.

POSIX

Portable Operating System Interface, a family of IEEE standards specifying an API for operating systems, aiming for compatibility between various platforms.

RAM

Random Access Memory, a type of microprocessor memory that is used for temporary storage of data that is being processed.

RGB

Red Green Blue, the combination of three colors in a computer display which can produce most of the colors perceived by human eye.

RTC

Real Time Clock, a clock circuit that keeps track of the current time and date.

RTOS

Real Time Operating System, a class of operating systems that is especially designed to allow fast reaction to external events and predictable timing of operations.

SD

Secure Digital, a standardized form factor and interface for memory cards.

SMPS

Switching Mode Power Supply, a class of power supplies that can be designed to raise or lower a voltage efficiently by rapidly switching it on and off.

SoC

System on Chip, a technique where the processor core and other necessary hardware components are integrated on a single microchip.

SPI

Serial Peripheral Interface, a type of digital interface which uses three wires for serial communications.

TFT

Thin Film Transistor, a technology used in display panels.

USART

Universal Synchronous Asynchronous Receiver/Transmitter, a digital circuit that can both receive and transmit data using a synchronous or an asynchronous serial protocol.

USB

Universal Serial Bus, an interface standard for connecting peripheral devices to computers.

1. INTRODUCTION

This master's thesis describes the design process of a measurement instrument for video playback measurements. The design work done by the author and described in this thesis includes the choice of hardware components, design of the electronic circuit and circuit board, design of custom enclosure, choice of software platform and implementation of the software running on the device. The purpose of the instrument, shown in Figure 1.1, is to aid in the development of video playback devices, such as tablets, smartphones and televisions.

Figure 1.1: *The device designed in this thesis work.*

The number of different models of video playback devices in development is nowadays higher than ever previously and new models are introduced every year. Popular examples include Apple's iPhone smartphones and iPad tablets, and the Android-based smartphones and tablets available from multiple manufacturers. An important part of the user experience on these platforms is the smooth playback of videos, game graphics and user interface animations. However, verifying the playback quality is a challenge.

If the video playback quality cannot be measured, it is likely that defects and inefficiencies will remain in the devices. These will eventually show up to the users, possibly only occasionally such as jitter that is visible in only certain kinds of motion. The user does not even need to consciously notice the problem for it to affect the *feel* of smoothness.

Most common methods are to add instrumentation to the programs, or to have a human perform subjective evaluation. See [1] and [2], respectively, for examples of these methods. Instrumenting the programs is often straight-forward, but doesn't allow comparisons to competitor products, verifying the product in the exact same configuration as the customer would use nor does it detect problems that occur in the display hardware. Subjective evaluation is expensive and does not provide exact, repeatable results, which makes it difficult to use during the development process.

Figure 1.2: *An example measurement setup where the video playback performance of a laptop computer is being measured.*

OptoFidelity Oy [3] is a company specialized in an external measurement method: the image on the screen is monitored using a high-speed camera or other sensors. This allows the verification of the complete playback path, including the display hardware. One of the product lines based on this principle is the *Frame Rate Meter* [4], which uses a graphical marker embedded in the video stream in order to measure frame times.

The device described in this thesis, called *Video Multimeter* [5], is a continuation of this product line. It improves upon the previous version by adding more capable sensors

and by allowing more extensibility for various measurement purposes, while retaining the basic mode of operation. Figure 1.2 shows an example of the measurement setup, where the marker is located at the upper right corner of the video and is measured using a fiber-optic sensor.

The rest of the thesis is divided into four parts. The first chapter will provide general background on video playback performance measurement, and describe in detail the challenges involved. The next two chapters will explain in detail the development of the measurement device, done as part of this thesis work. The final chapter will analyze the results of the work based on the achievement of goals and the customer feedback received for the pilot version.

2. STARTING POINT FOR DESIGN

Traditionally, measuring of video playback performance has focused on the image quality. This was reasonable with analog systems, where the primary problems are noise and distortion. Digital video systems have altogether different failure modes: it is far more common to encounter gaps and jitter than for the actual image quality to be degraded.

2.1 Background on video playback performance

The term *video playback performance* is used here for the aspects that affect the video reproduction quality and happen during the playback, as opposed to those that happen during the video production. Figure 2.1 shows the typical path from video production to video playback, and the part of the process that this thesis focuses on.

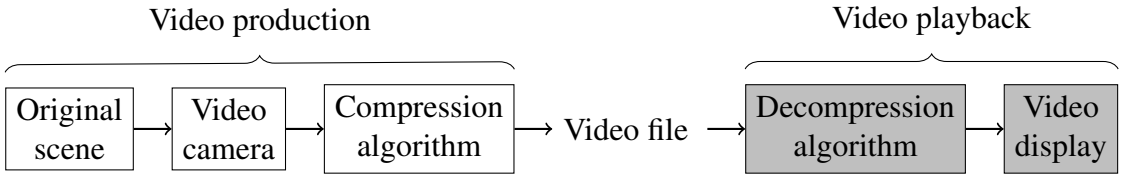


Figure 2.1: *A typical process of producing and displaying digital video. This thesis is concerned with measuring the performance of the video playback portion of the process.*

The overall user experience depends on a wide variety of factors, as illustrated in Figure 2.2, of which the playback performance is one important part. For detailed discussion and examples, see [6]- [7]. Leaving aside the audio portion and using a strict definition of ”video” as only the moving image, the video playback performance consists of two aspects:

Image quality is the visual aspects of each individual video frame as shown on the screen. It depends on things such as the display resolution, color reproduction and decompression accuracy.

Motion quality is the dynamic behavior of video frame changes. In the most basic case, it means how accurate the frame change time is. Further considerations are how fast the change occurs, whether the whole display changes in unison and whether all the video frames are actually displayed.

Of these, image quality can be roughly defined as the apparent clarity of the image, while motion quality can be defined as the apparent smoothness of motion. In the end, the perceived playback quality is always a complex combination of video quality, audio quality, content material and personal characteristics of the viewer [6].

Because video playback is a primary use for so many consumer electronic devices, it is clear that the quality of the playback affects the user's perceptions of the device. The smoothness of the image and the smoothness of the motion can have a great deal of effect on the market success of the tablet, computer or phone. A testimony to this is the decision of Apple to adopt the Retina displays [8], which provide high image quality, and the constant drive towards higher performance processors in tablets [9] to provide higher motion quality. Even the user interface and games are closely related to video playback, although some of the content is generated in real-time.

Therefore, it is beneficial for the manufacturer to optimize the video playback performance of their device. One way to attempt this is just to select the highest performance hardware on the market, and hope that it overcomes any deficiencies in the software. However, it would be more cost effective to improve the software, as the improvements there can be distributed to all devices for no extra cost.

Our goal is to find and eliminate performance bugs from the software. Some of them can be found out by simple testing by a human, but others are not immediately obvious when watching the video playback. Humans have a high tolerance for poor quality video when it is viewed alone, and the deficiencies only become obvious when compared against a better version. Even if such a golden reference is available, avoiding the subjective bias will be difficult [10].

To achieve objective results, we want to leave the content material and the personal characteristics of the viewer out of the consideration. This can be achieved by carefully designed double-blind studies, such as the *Double Stimulus Continuous Quality Scale* methodology [2, p.159], but the cost of such studies is prohibitive for the product development phase. It would be optimal to have an electronic device, which could accurately estimate the perceptions of the average viewer. Such a device could then be used during the whole product development project to improve the quality of the product, instead of just rating the product after it is already almost finished.

Electronic measurements that try to estimate the perceived video quality are usually termed *metrics*. The typical way to establish a metric is to define one or more measurable characteristics of the video, and then construct a model that maps the measurements into a single value. The accuracy of the metric can be studied using the double-blind studies. The goal is for the metric to correspond as well as possible to the average perceptions of the users.

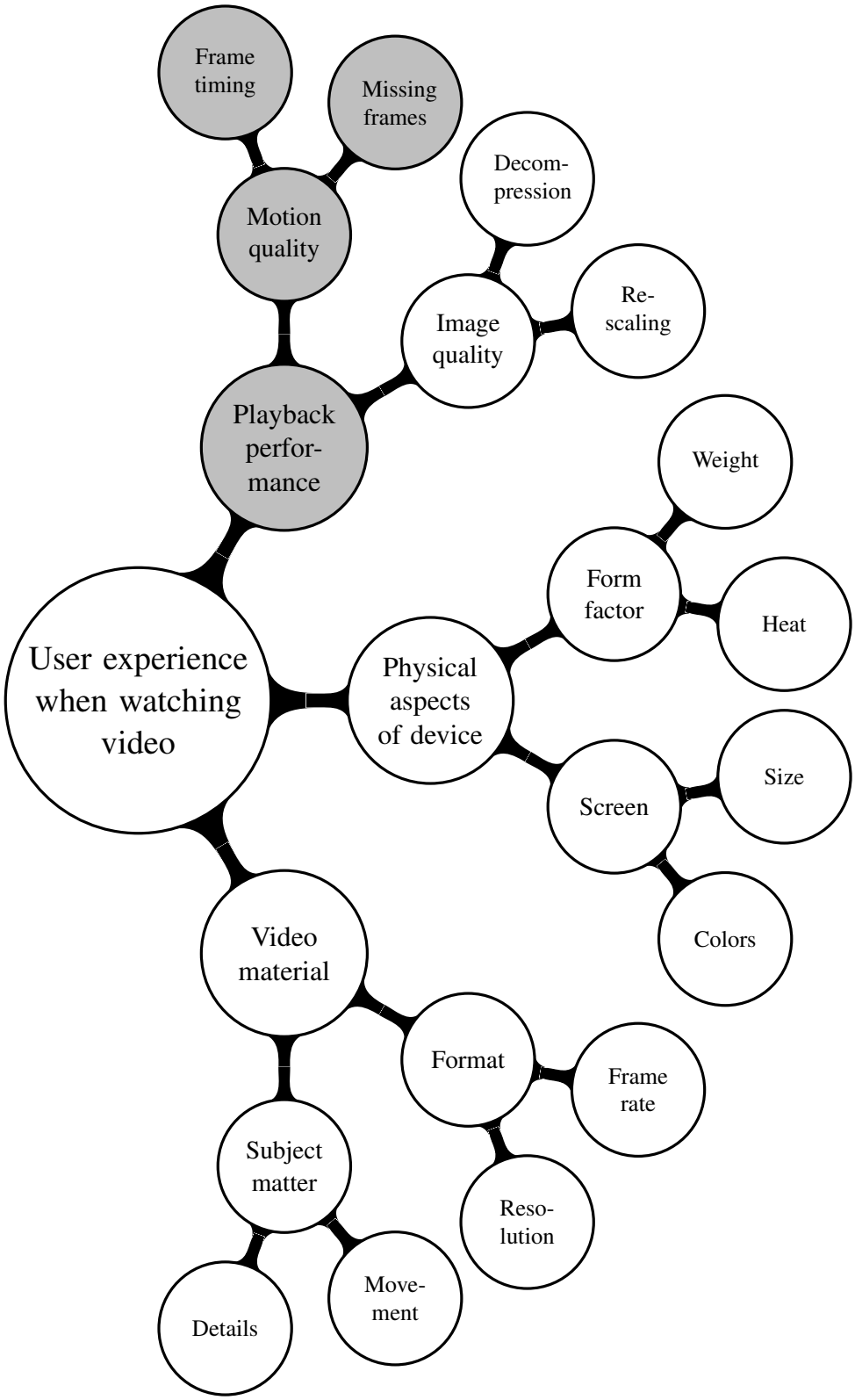


Figure 2.2: *Illustration of the wide variety of aspects that affect the user experience when watching a video played back on an electronic device. The parts that are of central interest for this thesis are shaded in gray.*

There are multiple widely used metrics for video quality, ranging from the PSNR (Peak Signal-to-Noise Ratio) to more advanced metrics that more closely approximate the human perception [10]. However, most of them focus on the image quality, ignoring the motion quality. There are several reasons for this. Historically, the signal path has consisted mostly of analog systems that could degrade the image quality, but could not change the timing of the material as there was no buffering. Similarly, with the analog systems, changing the dynamic characteristics such as the frame rate of the material was not feasible and it was not important to measure something which you cannot improve.

However, the introduction of digital video has radically changed the practical aspects of video quality. Because most digital video systems buffer one or many frames before playback, the timing of the frames can be easily and also accidentally altered. Similarly, higher quality equipment can use higher frame rates than traditional analog systems have. Consequently, it is now equally important to measure motion quality as it is to measure the image quality.

There exists ongoing research into the development of metrics for video motion quality. Notable examples include the *Motion-based Video Integrity Evaluation* index [7] and *V-Factor* [10]. However, there is no general consensus on the reliability of these metrics, and neither has achieved widespread use.

The use of metrics to estimate the perceptions of the user can also be misleading. The traditional *Peak signal-to-noise ratio* (PSNR) metric for the image quality has been widely used to optimize the compression in video codecs, mostly because of its mathematical simplicity. However, further research has shown that this actually leads to excessively blurry images, and codecs that rate better in this metric can actually look worse in reality [11]. Therefore if a metric is to be used, it must be carefully selected to avoid any undesired bias.

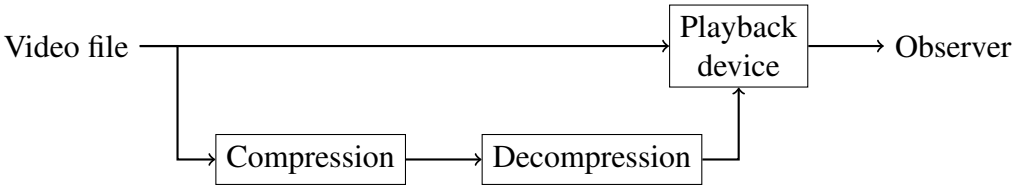


Figure 2.3: *A single playback device can be used to perform subjective comparisons of video quality by alternating between two sources, for example between original and compressed material. However, measuring the quality of the playback device itself is not possible with this setup.*

Both the double-blind studies and most of the metrics rely on the availability of a golden reference. The test subjects and the metrics compare *degraded material* against the source material. The principle is shown in Figure 2.3. This type of study is well suited for video compression, transmission and storage testing, where the degraded material and source material can be played on the same screen. However, when we are interested in the

degradation that occurs in the playback device itself, how do we play back the reference? Or in the case of a metric, how do we capture the degraded material?

There does not exist a general solution to this problem. It might be feasible to use a very high-quality video display as the reference, and set the test up so that the two devices are indistinguishable to retain the double-blindness. For comparative studies, comparing the playback from several screens is sufficient, but this also requires careful arrangements to avoid subjective bias if the devices can be distinguished. For the use of metrics, a high-quality video camera can be used to record the display, and through careful calibration and positioning the recording can be compared to the source material.

However, there is also a different way to approach the problem. We can define how the ideal video playback device should function, even though it cannot be implemented in reality. Objective measurements can then be used to determine how close the device is to the ideal behavior, and what aspects should be improved. Because the reference device doesn't actually exist this precludes the use of human subjects for the testing, and we need accurate measurement devices to replace them instead.

2.1.1 Ideal video playback

Video playback can be considered to be ideal when it fulfills the following criteria:

1. The color values for every pixel in every frame are exactly the same as in the source material.

2. Each frame is displayed for the same length of time, which is equal to 1/frame rate.

3. Frame changes are immediate and all pixels on the screen change the color instantly and at the same moment.

Note that the quality of the source material is ignored in the consideration of video playback performance. Even a poor quality video can be played back ideally, and the quality of the material depends solely on the quality of the recording device and any compression that has been applied. It is also assumed that there is an agreed upon way to decode the source material, i.e. that the color formats or compression is not ambiguous or implementation-dependent in any way.

Even with these restrictions, the ideal video playback cannot be achieved in practice due to several inherent physical constraints. There are also many possible implementation problems that can cause the performance to degrade. If we cannot achieve the ideal performance, how close to it should we strive to be?

2.1.2 Limits of human perception

In some market segments, it is enough to optimize the video playback performance so that it satisfies the average user. In the high-fidelity market, the manufacturer's goal is to push the quality beyond what their most demanding users can perceive. However, in all situations it is unnecessary to optimize beyond the limits of the human perception.

The human eye functions quite differently from a camera [12]. Instead of evenly spaced frames, each light receptor transmits its own sequence of neuroimpulses to the brain. When a new object becomes visible, brains are aware of its position and velocity much sooner than the actual color or details of the object. A camera in similar situation would have all the details of the object available at the same time, when the first frame is captured.

It is relatively straight-forward to determine the maximum visual acuity and color accuracy for the human eye [13]. Based on these, it can be calculated how many pixels and how many color values can be seen when a display is viewed from a certain distance. Modern high-DPI (Dots Per Inch) displays, such as Apple's Retina, approach or exceed these limits, leading to a viewing experience where the individual pixels can no longer be distinguished. When combined with good color quality, it is theoretically possible to reproduce still scenes exactly as they would appear in real life, except for depth perception.

We can also define a limit for the perception of the motion. Even though the eye operates in such a way that it is impossible to define a strict frame rate for it, it does have a limit after which an increase of the frame rate of a display cannot be detected. Such studies have been conducted, and the typical result is in the range of 30–120 FPS [14]. The large variance is to be expected both due to individual differences and due to differences in the video content.

Even if a user cannot distinguish the difference in the frame rate, any *variation* in the frame times may still be visible. Some studies suggest that the impact of dropped frames on perceived video quality decreases quickly at frame rates over 25 FPS [15], which would mean that a jitter (difference of shortest and longest frame time) of up to 40 ms (1 s / 25) would be allowable. However, there is a lack of studies that would have actually tested higher frame rates, so the impact is difficult to estimate. Personal experience shows that variations of around 10 ms are visible, especially in slowly panning aerial imagery. Even though determining an exact limit of perceptibility would require further studies, it can be assumed to be somewhere in the tens of milliseconds.

These values give an idea of how accurately we should aim to measure the display behavior, and how close to the perfect result the displays should aim to be.

2.1.3 Challenges in video playback

Any video playback device has physical restrictions that prevent it from reaching the ideal video playback. Many of these are cost issues, so a compromise can be reached which provides adequate performance compared to the price of the device.

The most common physical limitations are:

1. Color gamut of the display, limiting the reproduction of color values.
2. Refresh rate of the display, limiting the frame rate.
3. Latency of the pixel color changes, limiting the frame change speed.

However, there are also several problems that are primarily caused by the software implementation of the video playback device. Often these can be overcome with careful design, without increasing the price of the hardware.

Most common software limitations are:

1. Inaccuracies in decoding of the video contents or color space conversions.
2. Unpredictable timing in multiprocessing operating systems.
3. Gaps in playback caused by unexpected delays in fetching the source material from disk or network.
4. Mismatch between display frame rate and content frame rate.

Of these, limitation 1. causes problems in image quality, while limitations 2-4. cause problems in motion quality. Particularly 2. and 3. are difficult to test, as they can vary depending on background programs and network load, requiring long test runs in different kinds of environments.

Least obvious of the problems is number 4, which is caused by the refresh rate of the display hardware not being evenly divisible by the frame rate of the content. If the refresh rate can not be changed, the best compromise is to duplicate single frames at even intervals. However, if not explicitly designed and verified, it is much more common for the frame duplication to vary randomly based on minute timing variations of the system. This can lead to some frames being displayed for e.g. 3 refresh cycles, while other frames would be shown only for 1 cycle.

Figure 2.4 shows an example of this behavior by comparing the behaviour of two playback devices. On both devices, the display refresh rate is approximately 60 Hz and the video frame rate is 30 FPS. Even though the nominal frequencies are divisible by each other, small variations can cause them to drift relative to each other. Device 1 handles the

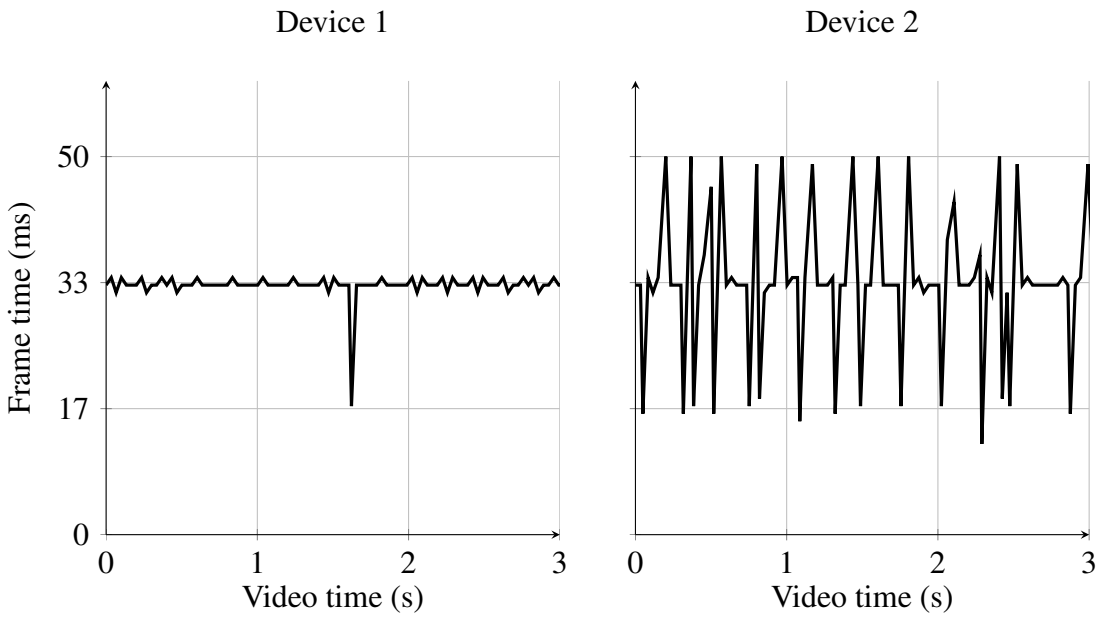


Figure 2.4: *Frame timings measured from two video playback devices.*

situation well, only requiring an occasional shorter frame to compensate, while the device 2 unnecessary also lengthens the frame times and causes excessive variation.

The jitter in frame times is seen as unevenness in motion, but the visibility of the problem depends heavily on the contents that is being played. In the example case, device 1 has a jitter of 17 milliseconds and device 2 has a jitter of 33 milliseconds. The jitter also occurs much less often on device 1, which will further reduce the visibility of it.

Overall, the visibility of the software limitations listed above depends on the user. Some users will tolerate even high inaccuracies in the playback, and in some kinds of content the problems may not even be visible. However, it is reasonable to expect that the differences will be noticed by reviews and when customers are comparing devices side-by-side in a store.

2.1.4 Need for timing measurement

Comprehensive testing of video playback performance will require the combination of several methods. Some of the aspects can be verified using traditional software testing methods, such as running a known file through the decoder and verifying the output. Other aspects depend more heavily on the interaction between software and hardware, and require therefore measurements from the real device.

One useful characteristic to measure is the timing of various events, most importantly the frame changes. This timing information can provide a good estimate of the motion quality of the playback. It is also relatively straightforward to measure, if the played back material is modified to include suitable markers.

The central point of this thesis is the development of a method and device for such timing measurements. It is also important that the solution can be integrated in a larger measurement system, which can then measure the complete scope of video playback performance.

2.1.5 Existing solutions

There exist several ways in which the timing information of video playback can be captured:

1. By modifying the playback software to capture timestamps when frames are shown.
2. Using generic electronic instruments, such as an oscilloscope connected to the display hardware.
3. Using a specialized instrument, which monitors the display using a sensor or a camera.

Method 1 is the most inexpensive, and it has found widespread use. However, it doesn't usually capture the delays introduced in the graphics card or display hardware. Furthermore, as it requires modification of the software, it cannot be used for competitor analysis nor for testing the final production units.

Method 2 can be very flexible in the kinds of measurements that can be performed. An example of this method is an application note by Rohde & Schwarz on lip-sync measurement using an oscilloscope [16]. However, the signal analysis is rarely automated. This means that long test runs will require laborious manual verification of the waveforms, or the development of custom signal analysis software for the purpose. In the latter case, the complexity of the system would approach that of the method 3.

Method 3 is ideally the least laborious and the most widely applicable option. If the algorithms included in the instrument perform well, the method can be applied with little modification to any kind of video playback device. However, the downside is that this method requires the purchase of a specialized instrument, the expense of which may not always be justified for a single kind of measurement.

2.2 Towards a measurement instrument

The measurement instrument developed in this thesis uses a light sensor, which is attached to the screen of the video playback device. A microprocessor will then analyze the waveforms captured using the light sensor and collect the requested data derived from the frame timings. Figure 2.5 shows an overview of the measurement setup.

In addition to measuring the frame changes, it is useful to be able to output a synchronization pulse to an external camera. This capability allows building a larger measurement system, where this instrument is used to detect the timing of the video frames, and another instrument is used to verify the content of the images. An example of such a system is the OptoFidelity AV100 [17].

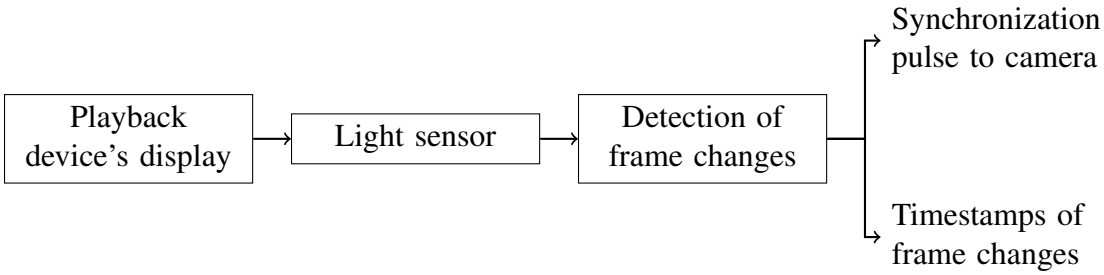


Figure 2.5: *The basic measurement setup for camera synchronization and frame rate measurements.*

The need to measure the frame timings and to synchronize a camera is central to many video playback performance measurement systems developed by OptoFidelity. Therefore, the kind of device described in this thesis was first developed very early in the company history. First version was under the name of *Synchronization Module* and then refined as *Frame Rate Meter*, shown in Figure 2.6. The device described in this thesis is a continuation of this product line, named *Video Multimeter* to reflect the wider measurement possibilities.

Figure 2.6: *The OptoFidelity Frame Rate Meter is a predecessor to the device described in this thesis.*

Throughout the various versions, the basic measurement setup has remained essentially the same. However, the use of more refined hardware and algorithms allow more accurate data to be extracted from the measurements. Equally important goal of the new version is to address some limitations that have restricted the applicability of the previous versions.

2.2.1 Issues of the previous version

The Frame Rate Meter is a device that can detect a black-and-white marker on the display using a fiber-optic light sensor. The actual detection of the light levels is implemented as an analog filter circuit, the output of which is then monitored by an 8-bit microcontroller. The device can perform the two basic functions: output a synchronization pulse, and capture timestamps when a change is detected. The analog signal path has proven to have predictable operation and low latency of just a few microseconds, both of which are important for this kind of a device.

However, it does also have drawbacks: adapting the device to different purposes is difficult, because the only way to change the signal path is through hardware modifications. It also translates to a large number of discrete parts in the hardware, which increases production costs. Some of the parts used are no longer manufactured, leading to reduced availability.

The sensor used is a single photodiode, connected to the fiber-optic cable. This kind of a sensor can sense the brightness of the light at one point of the screen. It is designed to be used with a test video, which is modified to contain a black-and-white blinking square in one corner. Detecting the difference of black and white light levels is reliable, but it has one severe limitation: if a frame is missing in playback, the square will be the same color for two frame times. For example, if a frame with black marker is dropped, the marker will remain white for two frames in a row. Consequently, one dropped frame in playback leads to two dropped frames in capture. Similarly, two dropped frames can go undetected.

This problem has previously been circumvented using a hardware modification, which links two units together. In this way it has been possible to monitor two separate synchronization markers on the screen. By having these markers use different frequencies, dropped frames can be handled more accurately. Essentially this forms a 4-state marker, by using 2 bits. However, the system consisting of two separate devices and fibers is difficult to set up and is rarely used. Also, even with the modification, the device does not collect any information about the dropped frames.

Another problem that has been noticed in practical use is the limited capture rate of cameras. When emitting the synchronization pulse to the camera, it is usually desirable to minimize the latency between the frame change and the pulse. However, if two frames occur very close to each other, the external camera cannot capture them fast enough and will ignore the second trigger pulse. This shows a need for flexibility when integrating into other parts of a larger measurement system: to adapt to the speed of the camera, a

delay has to be inserted between two synchronization pulses if they occur too closely. Previously this has also required a hardware modification, which is rarely practical.

Finally, the old version does not attempt to handle the backlight flicker of the display. In practice this means that the video playback device has to be set to full brightness, an option that may not be available on all devices.

2.2.2 Strengths of the previous version

One advantage of the analog signal processing is the guaranteed low latency and jitter, which are just a few microseconds. The camera synchronization requires predictable timing of the synchronization pulses, and is then adjusted to give the best possible captured image quality. To have reliable measurement results, the timing has to be equal for all captured frames.

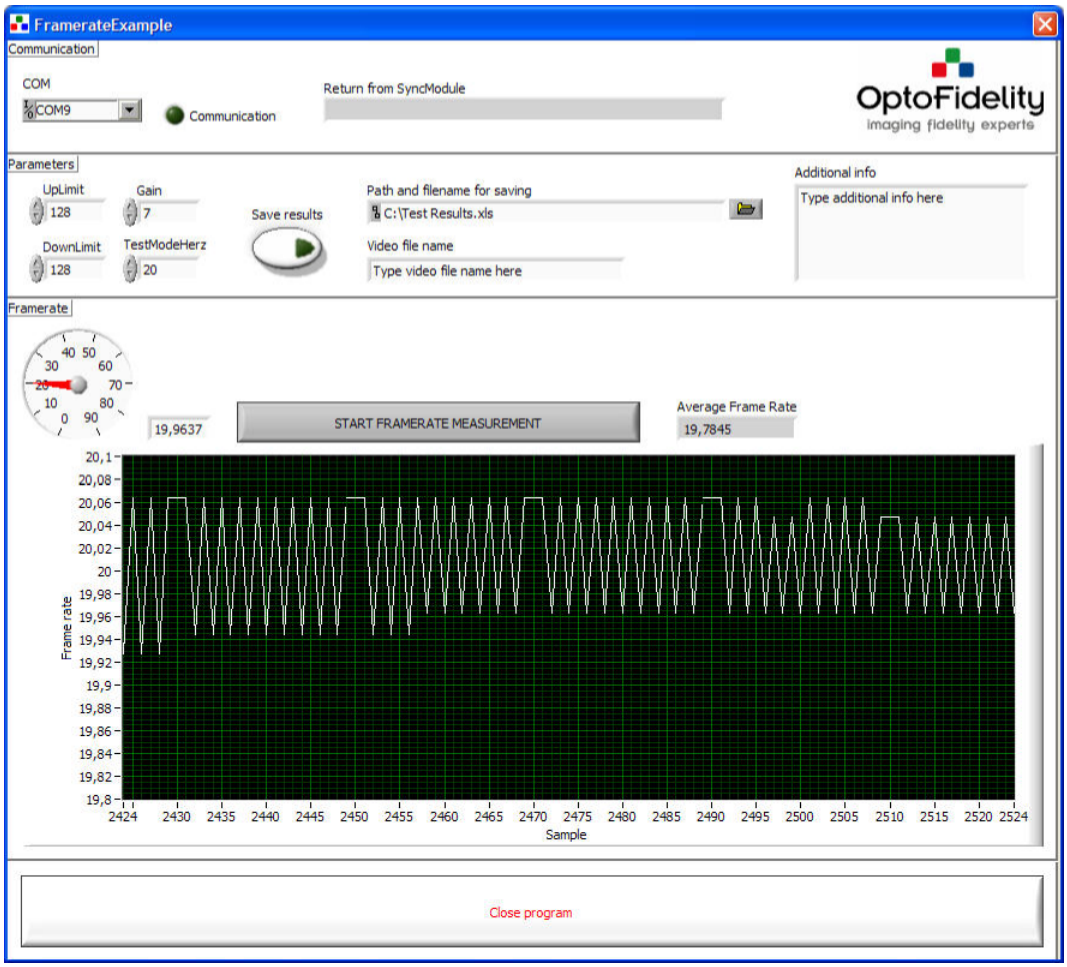


Figure 2.7: *The graphical user interface previously developed for the Frame Rate Meter.*

A PC program, shown in Figure 2.7, has been developed to act as a GUI for the device. It can show the frame rate in real time, and measure the variations in it. The previous

version has already been integrated to various other measurement systems, which would benefit from retaining the same USB interface and functionality.

The previous version has also been extensively used in the field, and found to perform reliably. When acting as a synchronization device, it is only a small part of the whole measurement system. Therefore it is especially important that synchronization is accurate and consistent, as errors in results are difficult to pinpoint to a single part of the system. The simplicity and predictability of the analog signal processing path has helped in achieving this goal.

2.2.3 New market needs

Despite fulfilling its immediate goal, the Frame Rate Meter has not been a large success on the market. Feedback from the sales indicates that the narrow application area and limitations in measurement are the prime cause for this. Many customers consider the expense of a separate instrument too large, if it can only perform a basic frame rate measurement.

Early in the planning of this thesis project, several new possible market areas were identified. Examples of these are: more thorough frame rate measurement, with dropped frames detection; camera latency measurement; audio-video synchronization measurement. All of these have something in common with the basic task, namely the measurement of timing information from a display device. However, they all require a different method of analyzing the signal, and also extra hardware for some of the tasks.

Overall, it was identified that the new instrument would have to be more flexible and extensible than the previous version. This would allow the sale of a single hardware unit, which could later be expanded using different software options.

2.2.4 Similar products and methods

There are not many products on the market that perform frame rate measurement from the playback device's display. All of these rely on markers embedded in the video feed.

VDelay [18], shown in Figure 2.8, is a program for latency and frame rate measurements in video calls, developed by Columbia University. It uses a small display in front of the transmitting camera to show a bar code, which is detected at the receiving end using software. By synchronizing the clocks of the devices, the latency of the transfer can be computed. By detecting when the barcode changes, the frame rate of the video is known. The main challenges in the system are related to the refresh rate of the LCD display and the camera shutter speed, which affect the blurriness and thus detection of the barcode.

An off-the-shelf high-speed video camera can be used to study the frame-rate in single-use scenarios. This involves recording the display at a high frame rate and careful studying of the resulting video to identify frame change points. The method is limited by the length

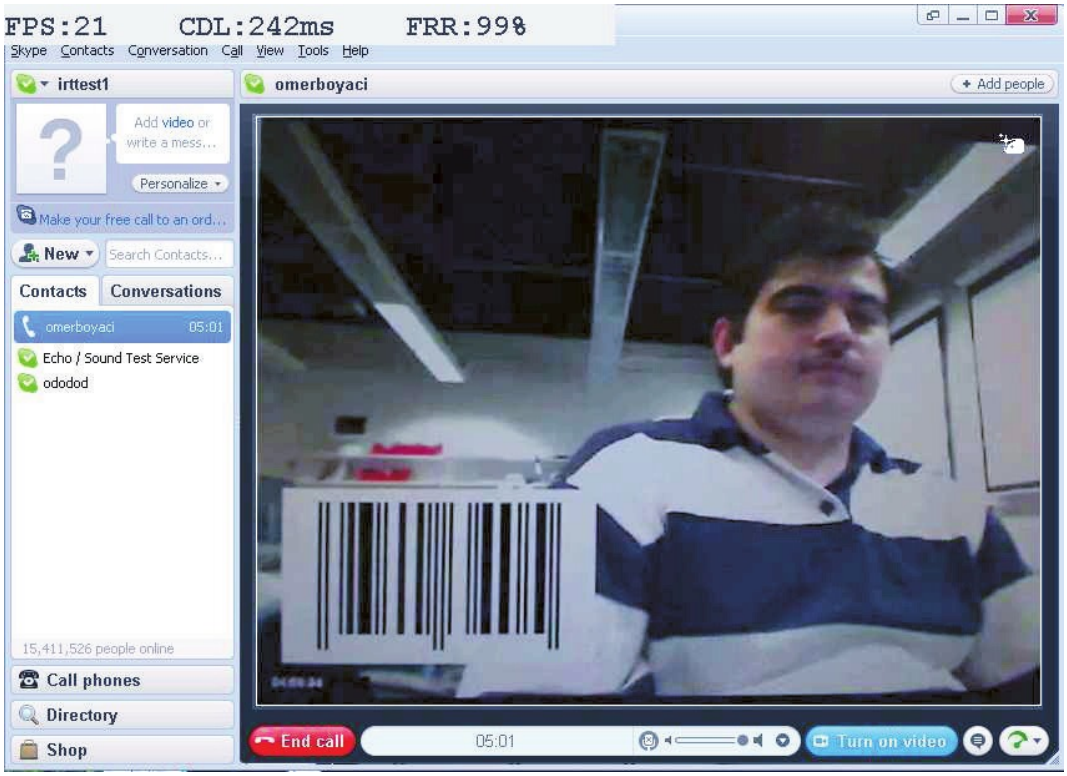


Figure 2.8: *Screen shot of the vDelay application. Image from [18].*

of video that the high-speed camera can record, which is often just a few seconds, and by the manual work required to process the resulting video. These limitations make studying longer periods of playback infeasible.

Spirent Communications (previously known as Metrico Wireless) has a test system called Chromatic [19] for measuring frame rate and other video performance aspects from a device display. It uses a high-speed video camera to capture spinning circular markers that have been embedded in the video. An example measurement setup is shown in Figure 2.9. The results are processed on a normal PC laptop. The same camera is also used to perform further analysis of the video content, such as detecting transmission errors. There is no public information available on the speed of the camera used nor the measurement accuracy that can be achieved using this setup, but it is reasonable to assume that it is limited by the camera speed and available processing power.

Pixel Instruments has a specialized instrument for lip-sync measurements. The Lip-Tracker [20] automatically detects the speaker from the image and uses computer vision algorithms to match the lip movements to the audio signal. In this way the measurement can be performed without a reference source and without any added markers.

Compared to the similar products, the Frame Rate Meter stands apart by its use of a simpler marker scheme. Instead of using a 1-dimensional marker, such as a barcode, or a 2-dimensional marker like a spinning disc, it uses a single blinking rectangle. This allows the instrument to capture the marker using a simple photodiode sensor, which can operate

Figure 2.9: *Running measurement with Spirent Chromatic test system. Image from [19].*

at a significantly faster sample rate than a camera. This yields higher accuracy measurements, but on the other hand limits the amount of information that can be conveyed by the marker.

2.3 The idea: a Video Multimeter

Early in the project, the idea began to form about a highly extensible hardware device, which could interface various sensors and perform real-time signal processing on them. Although the name *Video Multimeter* was coined only later, it encompasses well the purpose of this device: to be a convenient instrument to measure many aspects of video playback, providing both quick feedback during development and accurate results for quality assurance.

To achieve this goal, the hardware has to be designed to be both powerful and extensible. Furthermore, the user interface needs a good usability, in order to make a versatile tool. All this had to be done in the scope of a relatively small research and development project, in order to keep the development costs reasonable. Fortunately, with modern digital parts and ready-made software components, the task is much more feasible than even just a decade ago.

2.3.1 Basic principle for new version

Advances in microprocessor technology have increased the performance steadily over the years. Modern microcontrollers and small microprocessor systems are already fast enough to allow the complete implementation of the signal processing path in software. By capturing all the sensor readings into digital form as early as possible, the signal path can be defined in software to suit the needs of each measurement task. It also reduces the part count, as many microcontrollers now include a fast analog-to-digital converter and a large selection of other peripherals on the same chip.

In addition to performing well in the measurement tasks that have been decided ahead of time, the device needs to be adaptable for new purposes. Therefore sufficient extension capabilities must be designed into the hardware, and also the software needs to allow efficient reuse and modification.

2.3.2 New features

The black-and-white marker of the Frame Rate Meter is the simplest possible way to detect frame changes, but over the years it has turned out to be too limiting. Instead, a multi-state marker was required to be able to gather more information about the video playback. Multiple options were considered, such as using a row of binary markers, but eventually it was decided to focus primarily on color information to provide the additional states.

By using the three color channels (red, green and blue) in a binary fashion, 8 combinations can be created. This is enough to detect multiple dropped frames in the video playback, while the marker can still be read through a single optical fiber. This is useful as it takes up minimal space on the display, which may need to remain visible for other measurement devices.

The multi-state marker allows for the most important new feature: ability to detect dropped frames. This allows accurate frame rate and dropped frame measurements, which were difficult with the previous version if any frame dropping occurred.

However, just doing frame rate measurement is a too narrow market segment to cover the development costs. By leveraging the flexibility of software, the device can be used for a variety of tasks: latency measurements, lip sync measurements, camera synchronization and backlight analysis to name a few. Some of these were implemented for the first release, in the scope of this project, while others were left for future development.

3. HARDWARE DESIGN

To realize the goals set in previous chapter, a suitable hardware platform is needed. The choices made in this chapter have wide implications for the rest of the design. For example, the choice of processor affects the kind of software it can run efficiently. Higher performance processor can allow fast software development in a high-level language, but it will also have drawbacks in price, power usage and complexity.

Most important concerns in the hardware design are the manufacturability, cost and flexibility of the device.

For manufacturability, a premade platform would be optimal. There exists a variety of embedded development platforms, some of which are also suitable for use in products. Premade platforms have also a price advantage, as higher production volumes reduce the manufacturing costs.

On the other hand, maximum flexibility and suitability to the task is achieved with a custom platform. In this way, every component choice can be tuned to achieve the optimal compromise between price and capabilities of the device. The price is raised by the design costs and need for manufacturing resources.

3.1 Design options

The hardware design is necessarily a compromise between price and features. There exists a very wide range of possible designs, but only a few can be investigated in depth. To select the optimal design, a hierarchical approach is used: a high-level concept is decided first, after which the lower level options are explored. This way the amount of options at each level is limited and manageable.

Figure 3.1 illustrates the hierarchical decision process. The choices will be further explained in the following sections.

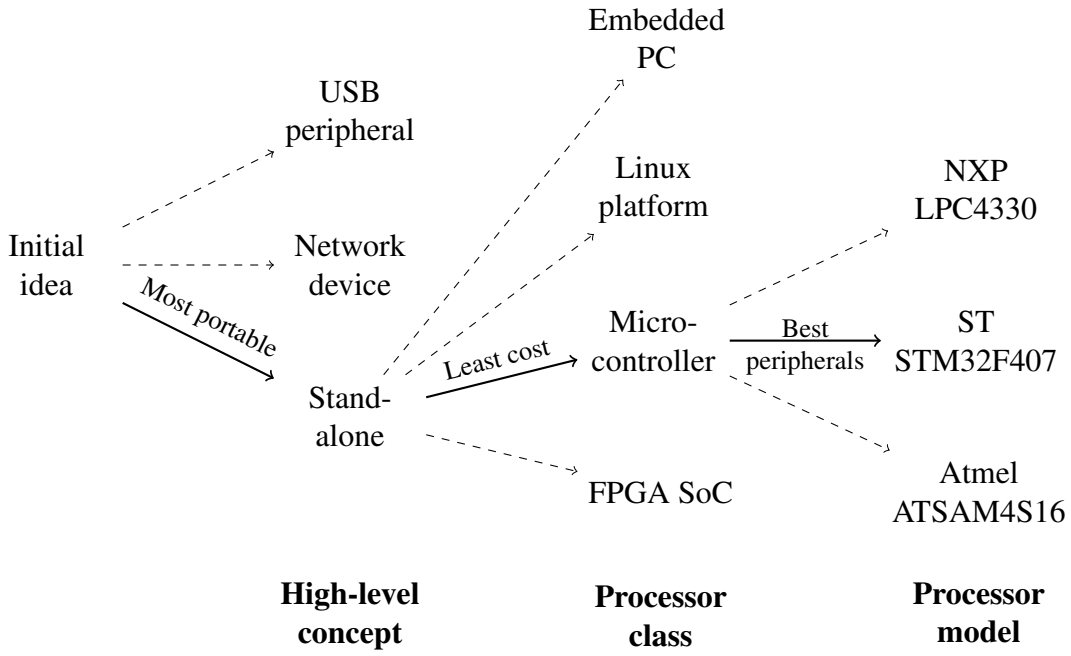


Figure 3.1: *Decision tree of the design possibilities explored.*

3.1.1 High level concept

At the beginning of the design process, three top-level design options were identified:

1. USB-connected device, which is operated from a control program running on a PC. This would be similar to the concept for previous versions.

2. Network connected device, which would be operated using a web browser. The user interface logic would run on the device itself, and the web browser would provide the display and control devices.

3. Completely stand-alone device, with its own touchscreen display and GUI running on it.

Of these, 1. has the lowest requirements for the hardware, and would be consequently the most economical to implement. At the other end of the spectrum, 3. requires display device and powerful enough platform to provide an user interface. Option 2. avoids the need for display, but still requires a powerful processor and also network connectivity.

From the sales feedback, there was a slight preference for the stand-alone option 3. A stand-alone device would be the easiest to demonstrate at sales events, and also is easiest to use in many of the use cases. Portability of the device would be another advantage, as would the independence of PC operating system. By including USB connectivity, the stand-alone device could also be controlled from a PC if necessary.

3.1.2 Processor class

To proceed with the design, the needed processing power must be evaluated and a suitable processor class has to be selected. The exact model of the processor is not important at this stage, but the general performance class is selected. The design choice of a stand-alone device with touchscreen display already precludes the lowest-end microcontrollers. This leaves the following options:

1. Embedded Windows PC, for example Intel Atom [21] or AMD Geode [22] based.
2. Embedded Linux platform, such as BeagleBone [23] or Pandaboard [24].
3. High-end microcontroller, such as ARM Cortex-M4 [25] based ones.
4. Custom system on a chip (SoC), implemented on an FPGA.

Figure 3.2 shows example products from each of these categories. In practice, options 1. and 2. would be implemented using a premanufactured hardware platform, because the hardware complexity makes custom design prohibitively expensive. Options 3. and 4. have simpler hardware design and could use either a premanufactured platform or a completely custom one.

It is important to notice that some of these options will not be sufficient by themselves. The options 1. and 2. would not be able to fulfill the real-time needs of the system, due to overhead in high-level operating systems. Therefore they would require an auxiliary microcontroller to handle the real-time processing.

Option 1. has a high cost, in the order of 1000-2000 EUR compared to the 100-200 EUR of the other options. It also lacks portability. Because it would still require a separate processor for all real-time tasks, it offers little advantage over just using a laptop and a USB-connected device. Consequently, it is easy to rule out.

Option 4. would have the highest flexibility, as the complete processor system would be on a re-programmable logic device, in an approach called *soft-core processor*. The processor could for example be augmented with custom instructions for specific tasks. However, the processing requirements would require a very high-performance FPGA. Also, as design costs have to be kept low, it is unlikely that the processor would be heavily customized in this project. Therefore using a soft-core processor would unnecessarily raise the price, as traditional processors in the same performance class are generally cheaper than an FPGA required to run one implemented in re-programmable logic. Therefore custom SoC on FPGA is not the best option for this device.

The final two options 2. and 3. are both very good alternatives. A Linux-based platform would allow using commonly available software development tools and libraries.

a) Advantech ARK-3403 [26], an Intel Atom based embedded Windows PC.

b) BeagleBone [23], an ARM Cortex-A8 based embedded Linux platform.

c) STM32F4 Discovery [27], a development board for STM32F4xx series of ARM Cortex-M4 microcontrollers.

d) Altera DE0-Nano [28], a development board for Altera Cyclone IV FPGA.

Figure 3.2: *A visual comparison of the processor class options. The photos are from the cited sources and are shown in approximate scale relative to each other.*

It would also have enough processing power to implement many signal processing algorithms, but with reduced real-time responsiveness. Even though achieving the required timing resolution under a non-real-time operating system is difficult, it would only require a separate microcontroller or FPGA for the real-time tasks. Such a separation could also make the software more structured, even though the required communication between processors would make it more complex.

Nevertheless, the processing requirements for the GUI are not exceedingly large. Option 3. would easily handle the GUI, while also executing the real-time tasks at the same time. Software development would be somewhat complicated, because there are not as many high-quality libraries available and the memory limitations must be considered at all times. On the other hand, other parts of the software are simplified because there is no need to communicate between separate processors. Also the hardware is simplified and power usage is reduced when everything is connected to a single processor, and the closer connection to hardware allows more determinism in some measurement tasks.

The decisive factor is that option 3. is the simplest, lowest-cost design that can fulfill the requirements.

3.1.3 Processor model

The most common processor cores used in modern 32-bit microcontrollers are ARM Cortex-M series [25], AVR32 [29] and MIPS [30]. Of these the ARM Cortex-M series has the widest array of associated manufacturers. Consequently it also has the largest amount of available libraries and good tool support. For example Texas Instruments TMS320 Delfino digital signal processors have very high performance, but their proprietary C28 processor core is not supported by many embedded operating systems. Because the ARM Cortex-M processors have high performance, low cost and best software support, they are the most reasonable choice for this design.

ARM Cortex-M series includes M0, M3 and M4 models. Of these, M4 has the highest performance and also includes a floating point unit in the M4F variant. Because the price difference is negligible in small production volumes, choice is simple. Highest performance processor will simplify the software development and allow larger flexibility in applications.

Manufacturers of high performance Cortex-M4 based microcontrollers include NXP Semiconductors, ST Microelectronics and Atmel. Details of the available options are presented in Table 3.1.

Table 3.1: *Comparison of high-end Cortex-M4 microcontrollers from three manufacturers, as available for purchase in September 2012. From each product series, the most suitable model was selected for the table.*

Manufacturer	**NXP**	**ST**	**Atmel**
Model	LPC4330FBD144 [31]	STM32F407ZG [32]	ATSAM4S16CA [33]
Clock frequency	204 MHz	168 MHz	120 MHz
Flash memory size	External	1 MiB	1 MiB
RAM size	264 kiB	192 kiB	128 kiB
FPU	Yes	Yes	No
ADC	10 bit, 8 channels, 400 kS/s	12 bit, 24 channels, 3 $\times$ 2.4 MS/s	12 bit, 16 channels, 1 MS/s
Other features	TFT controller, Cortex-M0 co-processor		
Price for 1 unit	8.11 €	12.75 €	9.94 € [34]

The most important application requirement is high general purpose computing performance, including high amount of RAM. For connecting different kinds of analog sensors, good analog capabilities are useful. Finally, a relatively large amount of flash memory, atleast 1 MiB, is needed to give room for software extensions.

All of the microcontrollers in the table have high enough performance to run basic signal processing and user interface tasks. However, the LPC4330 is clearly inferior in

its analog capabilities, and also requires external flash memory which makes the circuit more complex.

Of the remaining two, the STM32F407 is slightly more powerful and has more memory. The lack of a floating point unit also limits the performance of the ATSAM4S. Combined with the superior analog capabilities of the STM32F407, this makes it the best option.

3.2 Implementation of the main electronics

With the processor chosen, the rest of the electronics can be designed around it. These are quite straightforward design choices, as the only need is to provide the processor the necessary peripherals for interfacing the external world. Figure 3.3 shows the components and interconnections on the main board of the device.

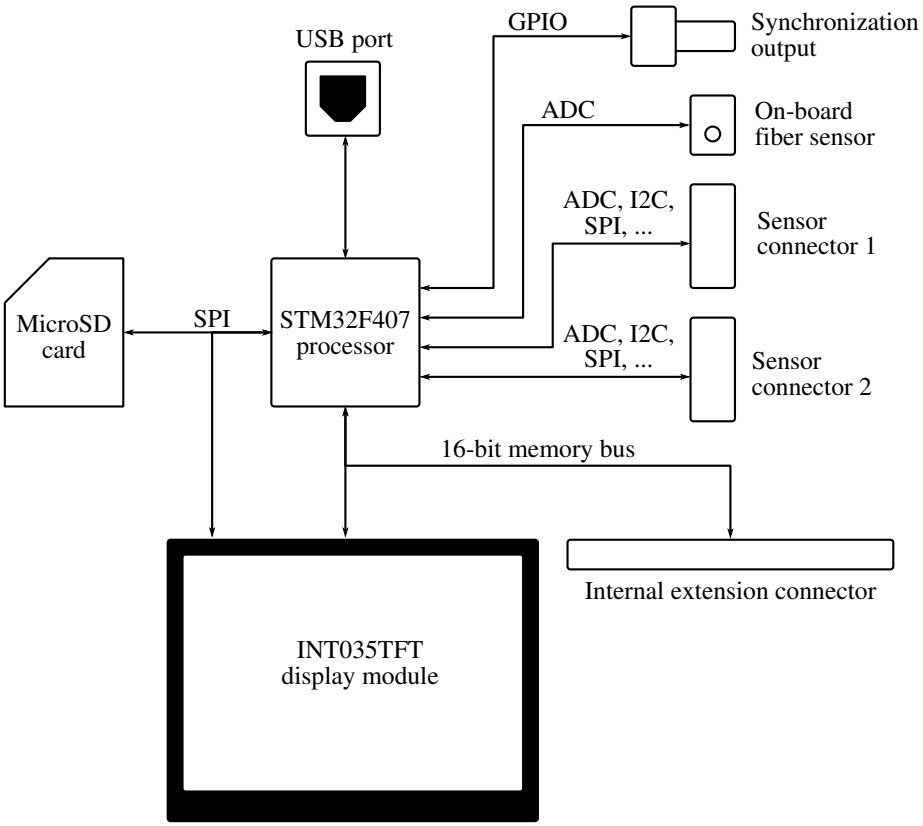


Figure 3.3: *Overview of the hardware components of the system, and the interconnections between them.*

The following sections describe the subsystems in detail. All parts of the main electronics are mounted on the same circuit board, which was also designed as part of this work. The circuit board is shown in Figure 3.4.

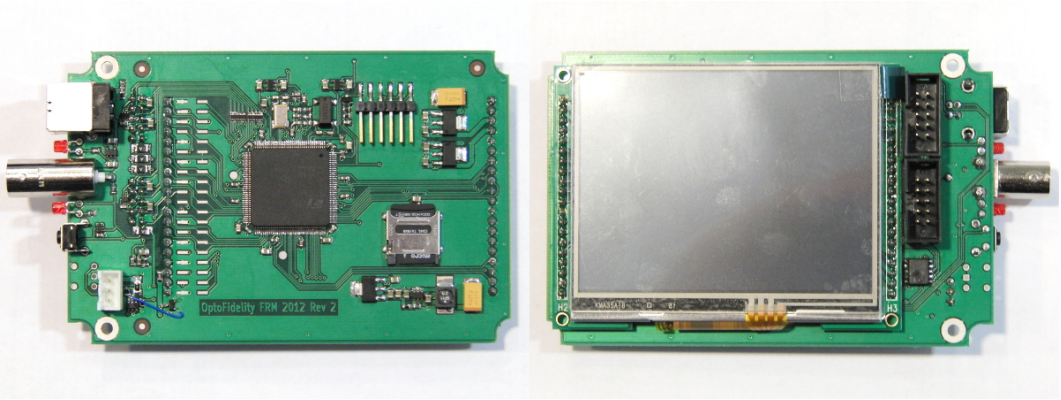


Figure 3.4: *A hand-soldered prototype version of the main circuit board of the device.*

3.2.1 Touchscreen

The implementation of a touchscreen based user interface requires several parts: the display panel itself, a backlight, a TFT controller, touch screen film, touch screen controller, and finally the processor controlling the user interface. To reduce the design complexity, an integrated display module is desired. These usually contain all of the parts except for the main processor in a single module.

A significant challenge with the integrated modules is to ensure their continued availability. Many modules are manufactured in small volumes and low profits, and consequently may be discontinued at any time. Furthermore, there is no standard form factor for TFT modules, so changing the module type would involve a redesign of the main PCB as well as changes to the enclosure.

Displaytech is one prominent manufacturer of integrated TFT and touchscreen modules. Their products are available from multiple distributors, providing good availability. The INT035TFT-TS module [35] was chosen for this project. It is a 320×240 screen with a resistive touchscreen film. While it is not comparable to modern smartphone displays in image quality, the availability in small quantities and simple hardware design make it a good choice for a design where production quantities are going to be small and low hardware development costs are important.

The INT035TFT-TS module integrates a SSD1963 TFT driver IC (Integrated Circuit). The driver IC has memory buffer for a single display frame, and automatically handles refreshing the display. The main microprocessor can write and read the screen contents using a 16-bit data bus. In this design, the TFT display is connected to the STM32F407 microcontroller's external memory bus to provide fast access.

The touchscreen controller in the display module is MAX11802, which uses a SPI serial bus to connect to the main processor. It also has a separate interrupt line, which can be configured to provide information about whether the display is currently being

touched or not. The touchscreen controller manages the analog-to-digital conversions automatically, and provides numeric coordinates through the SPI bus.

Overall, the placement of the display determines a large part of the form factor of the device. The main electronics are placed on the other side of the main PCB, behind the display, in order to keep the size of the device small. This way the connections for the display could be routed directly from the processor to the display pins, while the external connectors are placed to the right of the display.

3.2.2 Power management

To make the device portable, a lithium-ion battery was chosen as the power supply. The battery will be charged through the USB connection. For safety purposes, a ready-made li-ion charger IC and a prepackaged battery with a protection circuit were selected for the design.

The operation on USB power places a constraint on the power usage of the device and the attached sensors. The maximum permitted current by the USB specification is 500 mA. In order to be able to also charge the battery simultaneously, the total power usage of the device should remain around 200-300 mA. The processor, at 100 mA, and the display module, at 150 mA, are the most power intensive devices on the main board. Fortunately, the total usage of them still leaves 250 mA leeway for external sensors and battery charging, which seems adequate.

The device is to be powered on and off by means of a slide switch. The switch could in the simplest case just cut the power supply to the device, but this would cause problems in practice. Because the device may be in the middle of a write operation to the memory card, it has to have control over the power-down. This is implemented using a MOSFET (Metal Oxide Semiconductor Field Effect Transistor), which allows the processor to control its own power supply independent of the power switch. This is used to force the power to stay on while important operations are in progress.

The components on the main board require a variety of supply voltages; Figure 3.5 shows an overview of this. Even though the number of separate supplies is large, most of them are implemented using one-chip linear regulators. Consequently the board area required by the power supply parts is relatively small.

The main supply voltage for the processor and display is +3.3 volts. This is easily produced from the nominal 3.7 V voltage of the li-ion battery using a low-dropout linear regulator (LDO). The regulator chosen operates at 150 mV dropout, which allows the supply voltage to remain stable until the battery voltage decreases to 3.5 V. At this point most of the energy in the battery has already been depleted.

The integrated color sensor and some of the planned external sensors require a +5 V supply voltage. This is generated from the battery voltage using a combination of a boost-type SMPS (Switching Mode Power Supply) and a linear regulator. The SMPS is necessary in order to be able to raise the voltage, but the switching action generates considerable noise in the output voltage. In order to make the output voltage stable enough for the sensors, the SMPS is configured to supply a +5.5 V output, which is then regulated down to +5 V using a LDO.

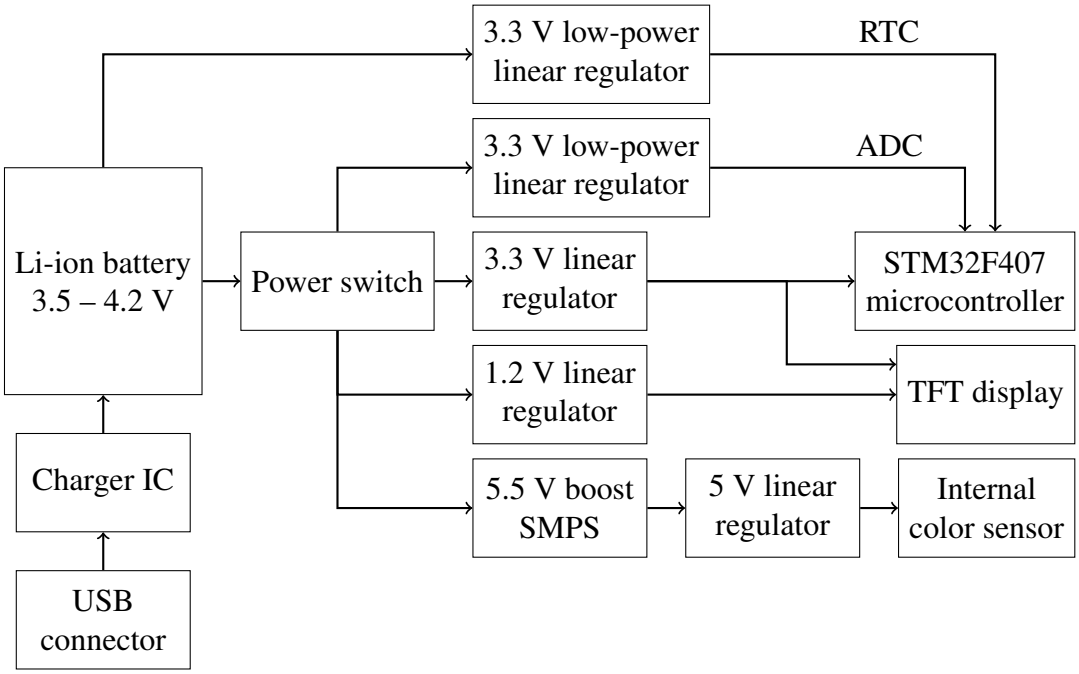


Figure 3.5: *Diagram of the power supply scheme of the device.*

The display module also needs an additional +1.2 V supply for the display controller, which is also generated using a linear regulator. At a voltage difference this large, a SMPS could provide better efficiency in dropping the voltage. However, the power usage on the +1.2 V supply is low enough that it does not matter in this device.

Finally, there are also two separate +3.3 V supplies that are needed by the processor: a backup supply for the RTC (Real Time Clock), and a separate supply for the ADC (Analog-to-Digital Converter). These have a low current drain, and are generated using small linear regulators. The RTC supply is regulated directly from the battery voltage, before the power switch. This way the device can keep correct time also when switched off. The separate ADC supply is implemented only to reduce the noise in the measurements, and it is regulated similarly to the main +3.3 V supply.

3.2.3 Data storage

An internal non-volatile memory is required to save configuration settings and the measurement results. This memory can be accessed through the USB port, so it does not have to be removable. In order to provide enough storage space for the measurements, some of which can be many hours long and consist of millions of data points, at least a 128 MB memory is needed.

The most reasonable memory type is flash memory, which comes in various forms. Flash memories are available with both parallel and serial bus interfaces. While a parallel bus is faster, it is unnecessarily complex to layout on the PCB due to the large number of connections. Unfortunately, there are only few options of serial flash memory having capacity over about 8 MB.

Instead, a more common solution is to use microSD cards as a storage medium. They provide large capacities in a small form factor, and are available from many manufacturers. Due to the standardized interface, this also reduces the risk of the manufacture of an important part being discontinued later.

One potential problem of microSD cards is the unpredictable write latency. The card contains a built-in controller, which handles the erasure and allocation of memory areas for storage. If the controller has to relocate a large amount of data in order to free up a location, the write operation may stall for several hundred milliseconds. This is a problem if real-time measurement data is being saved to the card. However, in this design, the microcontroller has sufficiently high amount of RAM memory in order to buffer the data if a stall does occur.

3.2.4 Built-in sensor

The amount of built-in sensors to include on the main board was necessarily a compromise. Including a sensor on the main board simplifies the hardware and reduces the cost, as it can be assembled in the same step with the main board. However, including unnecessary sensors would increase both the cost and the risk of some sensor part becoming deprecated and unavailable in the future, requiring design changes.

The color sensor is central for the basic measurement situations, so its inclusion was justified. By having the color sensor built-in, the two external sensor connectors are available for other purposes.

3.3 Extension capabilities

To provide for the needs of any future measurement tasks, the device has to be extensible. Naturally, the choice of microcontroller and platform will limit the capabilities of the system, and any electrical device will be obsolete eventually as technology progresses. However, by bringing out as many as possible of the interfaces present on the microcontroller, we can make sure that anything that could have been added in the design phase, can also be added as an extension in the future.

Most important extension capability is the connectors for external sensors. These should accommodate many kinds of sensors with minimal interfacing. Therefore the connector has to provide both digital and analog interfaces, and suitable power supply for the sensors. Some ideas for sensors that have come up during the project are: sound input, sound output, second color sensor, RGB LED for light output.

In order to allow for higher speed peripherals, there is also an internal extension port. This provides a 16-bit extension bus, which will allow fast data transfer between the extension board and the main processor. This bus could allow for example connection of a camera, an external display or high-speed data storage.

3.3.1 Sensor connectors

The chosen microcontroller already contains a large range of useful peripherals for interfacing sensors: it has SPI, I2C and USART buses, ADC and DAC converters, timer inputs and outputs, and naturally also general-purpose IO pins. Any of these can be useful for some kind of a sensor or extension, but dedicating a pin for each function would lead to a large and cumbersome connector.

The microcontroller itself contains a partial solution to this: many of the functions share pins, and have software-configurable multiplexers that select which function is connected to each pin. Not all of the desired functions are available on any set of pins, but we can further combine pins electrically using resistors. In any case, a series resistor of about $50 - 100\ \Omega$ is appropriate for an external sensor bus, in order to act as series termination. The main benefit of the resistor is to reduce the electromagnetic emissions from the system.

However, the series resistor can serve a dual purpose: by connecting each connector pin to two microcontroller pins through separate resistors, we can double the amount of available functions on each pin. By configuring either of the two microcontroller pins as high-impedance input, the functions of the other pin can be used independently.

Figure 3.6 shows a schematic of the sensor pin arrangement. The external sensor connects to the 10-pin connector J1. The resistors R1 and R2 are the series resistors that connect the two microcontroller pins to the sensor pin IO 1. The other three IO pins have same kind of connection, which is left out of the diagram for clarity.

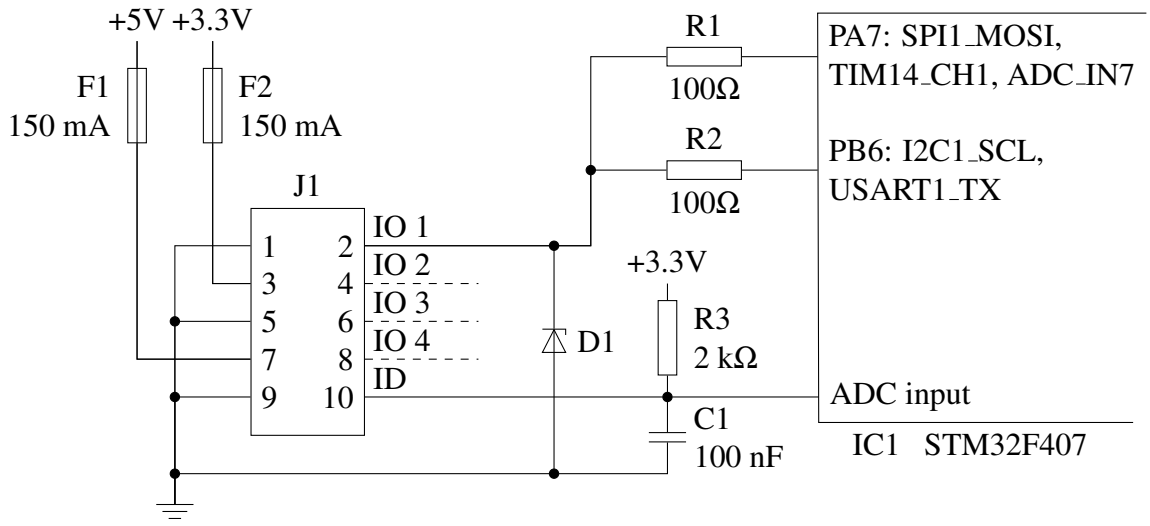


Figure 3.6: *Simplified electrical schematic of the external sensor connectors. The resistor arrangement for combining functions is shown for IO 1; the other three IO pins on the 10-pin connector are connected similarly.*

Through careful selection of pins, it was possible to form two sets of 4 data pins so that each set containing the exactly same functions on each pin. This way the two external sensor connectors perform identically. Therefore any sensor can be connected to either connector. By providing appropriate abstraction in the software API, the sensor drivers do not even have to know which port the sensor is connected to.

To give the final touch of usability, each sensor should be automatically identifiable. The simplest method of identifying the sensors is to simply have a pull-down resistor of a defined resistance on the pins. Combined with the R3 pull-up resistor in the device this forms a voltage divider. The analog input in the microcontroller can then be used to determine the voltage and identify the sensor. The capacitor C1 reduces the noise in the measurement. A single resistor with 1 % tolerance is enough to identify about 50 different kinds of sensors, and requires only a single pin on the connector.

Finally, the sensor connectors should be robust against normal use and occasional misconnections. Therefore each pin has its own electro-static discharge protection diode (D1 for IO 1) to protect against static electricity, and each power rail has a automatically resetting overcurrent fuse. The series resistors R1 and R2 included in the design provide further protection of the IO pins against overloading and short-circuits.

3.3.2 Extension board connector

The extension board connector allows insertion of another PCB inside the same device enclosure. It has especially been designed to be an efficient way to connect a FPGA as a co-processor to the main processor. Suitable tasks would be to capture signals from high-speed sensors, or equivalently provide high-speed data output.

The extension board connector provides access to the 16-bit external memory bus of the microcontroller. This bus is shared with the TFT display, and each device on the bus has a separate enable signal. The connector also provides the same +3.3 V and +5 V power supplies as are available on the main board. Finally, several general purpose IO signals and a synchronous serial port are provided in order to be able to load a configuration into an FPGA in runtime.

The design effort of an extension board is necessarily large, as the interface is not designed to be directly applicable for sensors. Whereas most sensor connections to the external sensor connectors consist only of a few discrete parts, an extension board will necessarily include an FPGA or a microcontroller. Therefore it is expected that most sensors will be connected through the sensor connectors.

However, the extension board connector provides an useful way to add even complex extra functionality in the future, without resorting to more intrusive hardware modifications. Because the connector is already designed in, even previously sold devices could be upgraded with an extension board.

3.4 Enclosure

The design of the enclosure posed some challenges in the project. Even though the scope of the thesis project did not necessarily include any enclosure at all, some solution was necessary in order to make the device usable and presentable for demos. For example, the optical fiber that connects into the color sensor would have been difficult to attach without an enclosure.

Furthermore, it was important to have some enclosure concept in mind when laying out the main board. For example, connectors had to be placed so that they could be brought easily to the sides of the enclosure, and the placement of the display also had to be reasonable.

In the prototype phase, low starting costs were essential for the enclosure design. It was very likely that several designs would have to be tried, so any starting costs such as mold making or CNC programming would quickly accumulate.

For the prototyping phase, it was therefore important to have a flexible technology that allows affordable testing of many test models, with fast turn-around time. For the pilot production runs, it was also necessary to minimize the amount of manual labor.

3.4.1 Options

One way to keep costs low would be to use a pre-made standard enclosure. However, the need for openings for the connectors and especially the display would still have required custom machining of the enclosure. Another issue is that very few standard enclosures would match the dimensions of the display well.

Most commercial electronic devices use an enclosure made of injection molded plastic. For this product, even the estimated production runs are too small to warrant the cost of mold making. Even the inexpensive prototyping services have costs starting at thousands of euros per each run [36].

Some of the other devices manufactured by OptoFidelity Oy in the past have used a custom aluminum enclosure that has been milled using CNC machinery. This is a reasonable option, and allows the production of high-quality enclosures in small runs. The per-unit cost is high compared to most other options, but not unreasonable for the price range of these products. However, there are still significant non-recurring engineering costs charged by the supplier, for tasks such as processing the model and programming the tools.

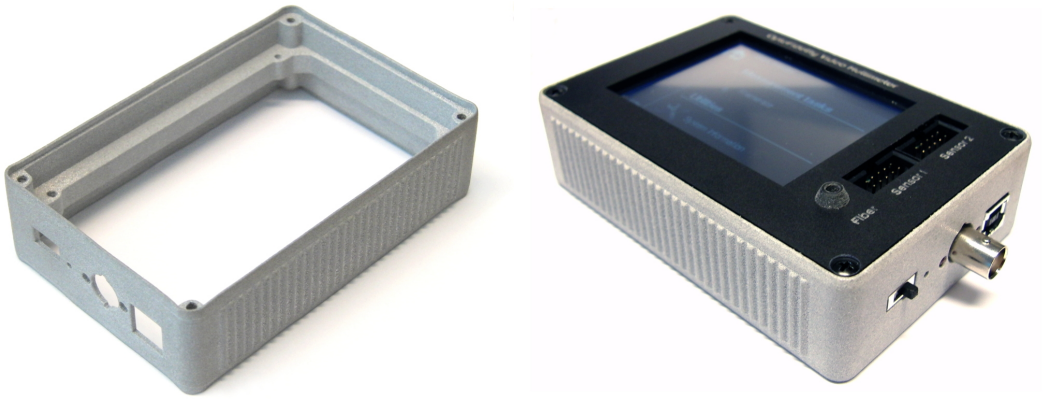
Recently, the price of 3D printing has also become affordable for this kind of applications. The prime benefit of 3D printing is that there are no setup costs, and that very complex shapes can be created. It is therefore a good fit for the prototyping run, as various enclosure designs can be tried out at low expense.

3.4.2 Implementation

The prototype enclosures were ordered from ShapeWays, a company based in Netherlands that provides 3D-printing services [37]. ShapeWays uses high-precision industrial machinery for the purpose, which provide a good accuracy and surface quality in the end product. The material used was *Polished Alumide*, which is a marketing name for polyamide mixed with fine aluminum powder, and then polished using glass beads after printing.

The 3D printed part and the complete enclosure are shown in Figure 3.7. The main board, connectors and display fit readily in the apertures designed into the model, reducing the amount of manual assembly work. The size of the device is approximately $12 \times 8 \times 3$ cm.

The front and back panels were ordered separately, laser-cut of acrylic sheet. This both reduces the amount of material that has to be printed, consequently reducing the price, and also makes the structure stronger. The laser cutting technology can also engrave the surface of the plastic, allowing the texts and markings to be added in the same step. The visibility of the engraved texts was not entirely satisfactory, but good enough for the pilot run.



a) 3D printed part b) Assembled device

Figure 3.7: *The 3D printed body and the assembled enclosure.*

Both the 3D printing process and the laser cutting are controlled by computer-based models. These were designed using the open source tools QCad [38] and OpenScad [39]. While they do not match the capabilities of commercial CAD packages, the amount of learning required is smaller and they allowed the quick production of models for this purpose.

4. SOFTWARE ARCHITECTURE

The high-level purpose of the device is straightforward: capture data from sensors, apply signal processing algorithms, show results or take an action. The design of the necessary filtering algorithms is the most difficult part of the core tasks. It is more of a signal processing problem than a software architecture problem.

However, to support these core tasks, a significant amount of low-level software is necessary. Because the hardware is custom, there is no premade operating system that would work without configuration. Also some device drivers will have to be written, because no operating system contains drivers for all of the wide variety of chips on the market.

4.1 Real-time operating system

Most important part of the support software is the operating system that will run on the device. The purpose of a RTOS (Real-Time Operating System) is to support multiple parallel tasks, without compromising the strict timing requirements of the tasks. Most RTOS also provide a HAL (Hardware Abstraction Layer), which contains device drivers and a programming interface for the various parts of the hardware.

4.1.1 Requirements

In the scope of this project, there are no resources to port or significantly extend the chosen RTOS. Therefore it is necessary that the RTOS already contains support for the STM32F4 processor. Most important subsystems should also be included or easy to integrate: USB driver, graphics toolkit and file system.

Furthermore, to be commercially useful, the operating system must have reasonable support and licensing model and reasonably secure future availability. This can mean either paid-for support package, or an open source community around the project. In the latter case, any features not implemented by the community can be implemented by the designer, only to the limit of available time.

Table 4.1: *Feature matrix of the software components included in each of the considered RTOS. Features marked as **separate** or **third-party** are sold separately by either the same company or by another one. Features marked as **no** do not have an existing implementation for the operating system.*

RTOS	Graphics	C library	File system	HAL	STM32F4 USB Driver
ChibiOS/RT	No	No	No	Yes	Yes
µC/OS-II	Separate	Separate	Separate	Yes	Third-party
NuttX	Yes	Yes	Yes	Yes	Incomplete
FreeRTOS	No	No	Third-party	No	Third-party

4.1.2 Options

Table 4.1 shows 4 candidate operating systems that were considered for the project. All of them could fullfill the requirements, either by themselves or when combined by other components. In some cases, however, this would require adapting separate libraries to work with the chosen RTOS, while in other cases an already integrated solution exists.

ChibiOS/RT [40] is an open-source RTOS, which contains basic thread facilities and a HAL with drivers for most STM32F4 peripherals [41]. It does not include a file system nor C standard library, but separate libraries can be used.

µC/OS-II [42] is a commercial RTOS developed by Micrium. There is no official STM32F4 support yet, but third-party ports exist [43]. Advanced file system and graphics modules are available.

NuttX [44] is an open-source RTOS that aims ”to be a tiny Linux-compatible OS for MCUs.” Consequently it supports many POSIX interfaces, such as pthreads. It has drivers for most STM32F4 peripherals, but some are not fully working or tested.

FreeRTOS [45] is an open-source RTOS with long history and extensively reviewed code base. The basic system includes only the kernel and thread functionality, but device drivers are available from third parties [43].

Some well-known operating systems were left out of comparison because of lack of hardware support, such as NutOS, eCos or VXWorks. These systems did not support the ARM Cortex-M4F processor at the time when the device was being designed.

Overall, NuttX contained the most features that were required by this project. Especially its integrated graphics subsystem, which supports multiple application threads and has a built-in window manager, contributed towards the decision. Furthermore, being open source and BSD licensed made it an inexpensive option that can be modified as necessary.

4.2 Programming language

The traditional de facto programming language for embedded systems has been C. However, modern microcontrollers are powerful enough to support higher level languages, which may offer faster software development and easier to understand code.

There are two main components in the system: the signal processing and the user interface. For the signal processing part, real-time requirements such as high performance and predictable latency are critical. For the user interface, performance is not as critical as the efficiency of software development.

4.2.1 Options

The Cortex-M4 processor is widely supported by many programming languages. Some of the most commonly used programming languages are:

C is supported by compilers from many vendors and is compiled to native machine code for the target. It is a relatively low-level language, allowing efficient use of the hardware but often requires more complex code for implementation than other languages.

C++ is a high-level object-oriented language, also supported by compilers from many vendors. Compared to C, it has higher level features but still allows everything that is possible through C and has equal performance.

C# is a high-level object-oriented language, which uses garbage collection to simplify development. It can be run on microcontrollers by the means of .NET Micro Framework [46]. Unlike the full .NET Framework, the Micro Framework only interprets the byte code and does not include just-in-time compilation. This leads to 2–10 times slower execution than the languages that are compiled to native code.

Lua [47] is a script language, which is widely used to implement the higher-level portions of applications and games. It is rarely used to implement complete programs, but is used as a method to allow easy customization of applications for various purposes. In the scope of this project, it could allow fast development of measurement tasks once the lower level components have been finished.

The NuttX operating system itself is implemented in C and C++. Therefore the use of any other language would require the development of a separate interface layer to provide access to the NuttX functions from the rest of the software. Choosing C++ as the development language was a reasonable way to maximize the performance while also providing high-level features for development.

4.3 Implementation

The software components implemented as part of this project can be divided into three categories: device drivers, signal processing filters, and user interface. The overview of the software architecture and the components implemented in this project are shown in Figure 4.1.

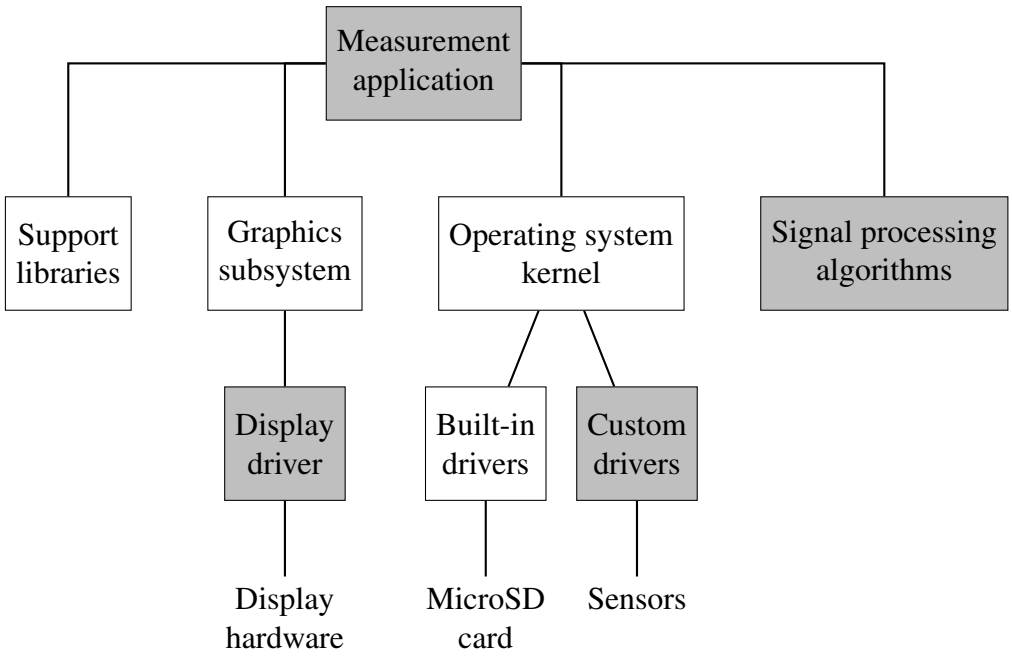


Figure 4.1: *Overview of the software architecture. The user interacts with the measurement application, which communicates with the hardware through the kernel and driver layers. The shaded parts were implemented in this thesis work, while the other parts are reused open source components.*

The complete system also contains the scheduler, device drivers and graphics libraries from the NuttX operating system, some of which were extended for this project but most of which are used as-is. Most changes to the NuttX core have been contributed back to the central project, which has the benefit of not needing to maintain them separately over the changes in future operating system versions.

4.3.1 Device drivers

The lowest level of the software implementation is the device drivers, which interface to hardware resources such as the color sensor or the synchronization port. These can be considered to form a part of the NuttX kernel, as applications communicate with them through standard kernel interfaces such as the device nodes in */dev*.

To meet the performance requirements, most of the device driver code runs in interrupts when new samples are available from the hardware. Also DMA (Direct Memory Access)

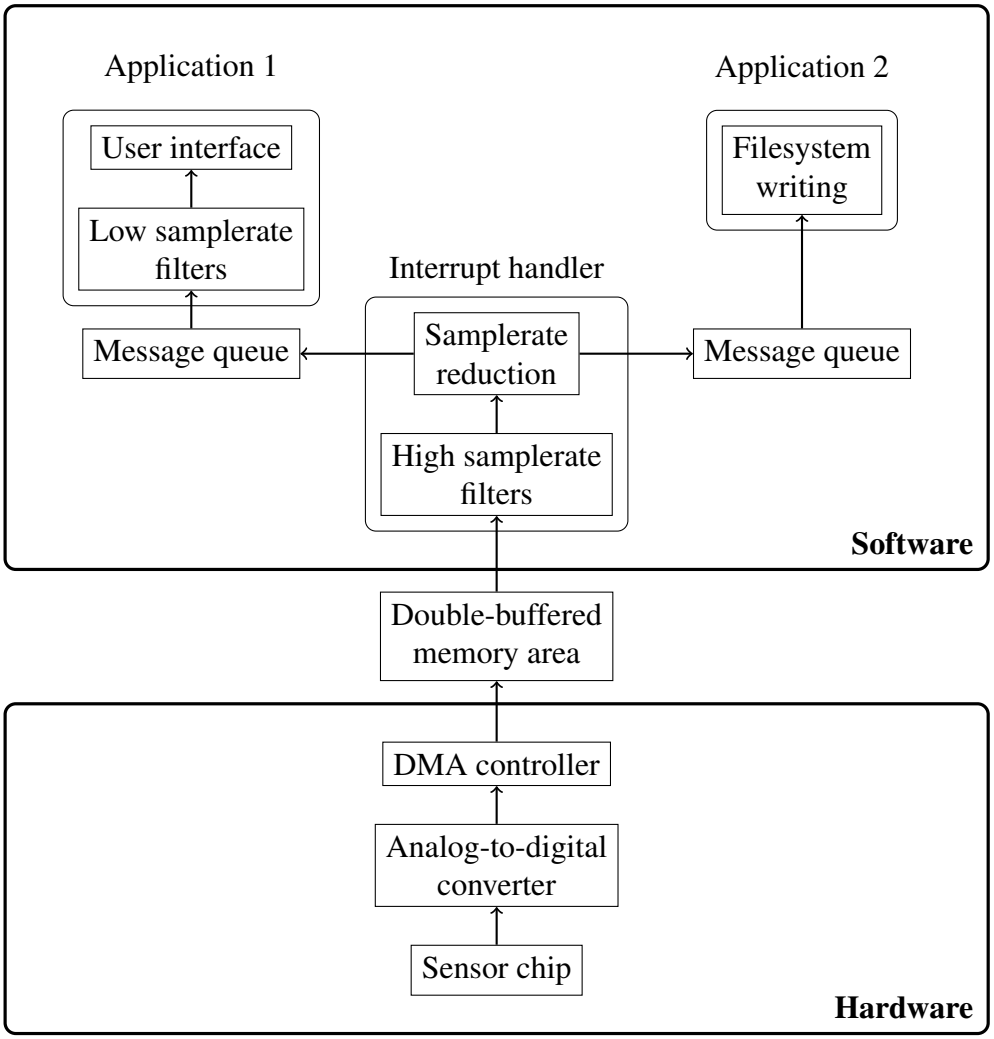


Figure 4.2: *Data flow in the system when two applications are reading the data from the color sensor device driver.*

techniques are used to offload the processing of the data to the hardware. When the samples are ready in a memory buffer, the interrupt passes them on to applications by means of message queues, which are implemented by the NuttX kernel.

Figure 4.2 shows the flow of the data throughout the hardware and software of the device. Hardware handles the most time critical parts of collecting the samples. When one of the memory buffers is full, the interrupt handler triggers, performs some of the more demanding filtering tasks, reduces sample rate of the signal and passes on the samples to memory queues. Each reading application has its own queue of incoming samples, which it can then read and process in whatever way is appropriate for the application's purpose.

The message queue mechanism provides thread safety, but it also has a negative performance impact when large amounts of data are being processed. Consequently, it is important that the device drivers do not have to to pass excessive amount of data to the applications. This is accomplished by running some of the signal processing algorithms

in the interrupt handlers, such as the backlight filtering. In order to function effectively, the backlight filter needs to process every input sample at the full 100 kHz sample rate, but for further analysis a smaller sample rate is sufficient.

A traditional approach to interrupt handlers is to keep them as short as possible, because a running interrupt handler blocks all other tasks on the system. However, modern CPU cores such as the Cortex-M4 have multiple pre-emptible interrupt priorities, so that a higher priority interrupt can interrupt another, longer running handler. This is beneficial in exactly this kind of applications, where there is a need to run long processing tasks in the interrupts. By setting a lower priority for the longer interrupt handlers, the shorter ones can pre-empt it. In this specific case, it allows the reduction of the sample rate as early as possible, without compromising the overall system interrupt latency.

4.3.2 Signal processing

Digital signal processing forms a core part of the software. It is necessary to both implement the currently required measurements, and to allow easy extensibility for future needs. Furthermore, efficiency, reliability and testability are important.

The signal processing tasks in this system consist of fairly basic algorithms: FIR (Finite Impulse Response) filters, sliding window variances and averages, derivatives and some custom non-linear filtering. It is reasonable that these basic algorithms should be implemented as reusable blocks, which can be combined in various ways to realize the measurement tasks.

The reusability was achieved by implementing each basic algorithm or filter block as a separate C++ class. The classes share a basic interface, which allows chaining of several instances together to form a filter chain. Input samples are given through a function call on the chain's first filter, and the samples then progress automatically through the chain. Output data is collected through a callback function on the last filter. An example of such a filter chain is shown in Figure 4.3.

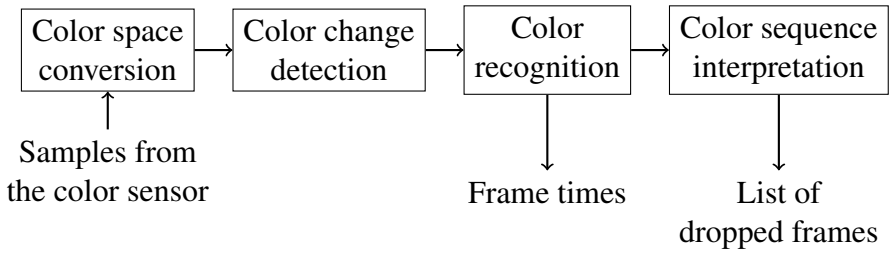


Figure 4.3: *Simplified diagram of a filter chain for detecting the frame times and dropped frames from a color marker. Each block in the diagram is implemented as a separate filter, which can be chained in different layouts for measurement tasks.*

For this thesis, the filter blocks implemented include such functions as backlight filtering, color space conversion, color change detection and color recognition. The algorithms used cannot be described here in detail due to their proprietary nature. An important goal was that the device would not require complex setup, but would instead automatically adapt to many kinds of display devices under measurement. This was accomplished through careful design of the algorithms, so that they will capture the fundamental characteristics of the signal and not be affected by variations between display types.

Because of the very large variations in the kinds of display devices in use, the algorithms used are necessarily somewhat heuristic in nature. Verifying the performance of the algorithms is a major task, as even small changes made to accommodate one kind of display could cause problems for another kind. In order to verify the performance of the filters, a regression test suite was needed.

Figure 4.4 shows two examples of the various kinds of display behaviors that the signal processing algorithms have to adapt to and tested for. The OLED display is characterized by much faster color changes, which are in fact so abrupt that the internal refresh cycles of the screen can be seen as glitches. The data shown in the graph has already been preprocessed by a color calibration algorithms, which separates the R, G and B channels and normalizes the range of color values to 0–255.

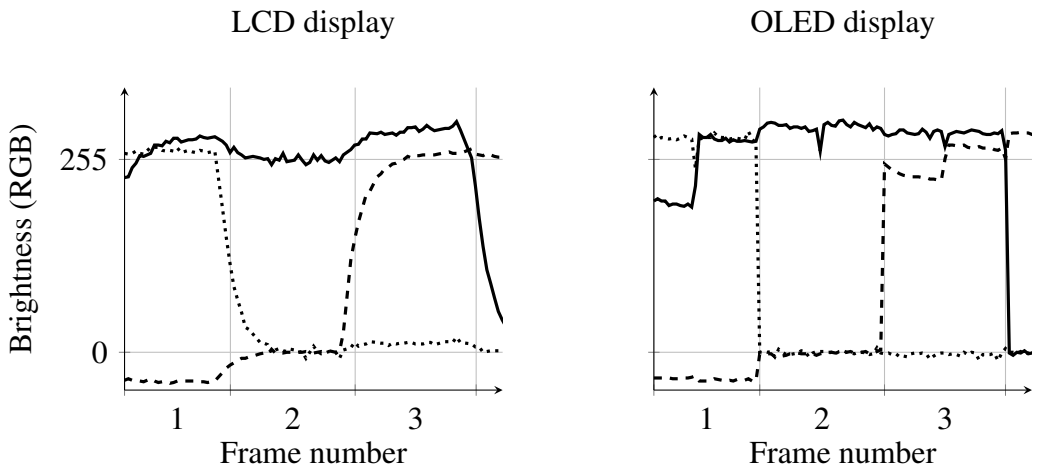


Figure 4.4: *Comparison of OLED and LCD display behaviors. Each graph shows the measured brightness of three color channels: red, green and blue.*

The very basic behavior of the filters can be verified using synthetic unit tests. For example, a color change detector can be tested by feeding a generated sequence of 100 samples of one color and then 100 samples of another color, and checking that the output shows a single transition at the correct time.

However, testing the adaptability to various display types cannot be done using synthetic data. Instead, throughout the project test data was collected from various kinds of devices and added to the test suite. Once the data is processed through the filters,

the output is checked visually using graphing tools. For example, graphing the detected color changes against the raw color data gives a graph that is feasible, if tedious, to check manually.

After the output has been manually checked once, the results are stored as a reference file. Any further runs will compare the output of this (filter, data file) pair against the reference file, and only report differences that exceed a predefined threshold. This allows automatic checking of all changes against the full test data set, which greatly reduces the possibility of regressions in filter performance.

Further development could be done in the area of comparing the reference files and actual filter output. Currently the checks report considerably often false positives, as there is only a single threshold defined for the whole test. For displays with slow color transition, like LCD in Figure 4.4, the uncertainty in the results is necessarily larger than for displays with very fast transition. Similarly, the glitches in OLED waveforms can result in false transitions which must be filtered out separately, a step which complicates the individual testing of the filter blocks.

As the signal processing subsystem is implemented as standard C++, it is very portable to various platforms. For example, the tests can be readily run on the development PC, which speeds up the process of running through all the test data files. Similarly, even if a future version of this device is designed based on a different hardware platform and operating system, the signal processing portion can be reused.

4.3.3 User interface

NuttX includes its own GUI and window manager libraries, called *NxWidgets* [48] and *NxWM* [49]. These are designed to fit the resources available on common microcontrollers, and are therefore a good fit for the hardware in this project. The programming language for the graphical side is C++, and the libraries use the familiar object-oriented approach to GUI programming.

The NXWidgets library contains user interface elements (widgets) for the most common functions, such as buttons, text boxes and labels. Each widget is a separate C++ class, and new widgets can be implemented by the user of the library. Unlike the GUI libraries typically used on PC's, NXWidgets does not include any automatic layout support: each widget has to be manually placed based on pixel coordinates. As long as the screen size remains constant, however, this is not a large problem.

The NxWM window manager provides facilities for starting applications, switching between them and closing them. The NxWM's basic user interface, shown in Figure 4.5, is the taskbar at the side of the screen, and the start window which acts as a list of applications. Each application is implemented as a C++ class, all of which implement a

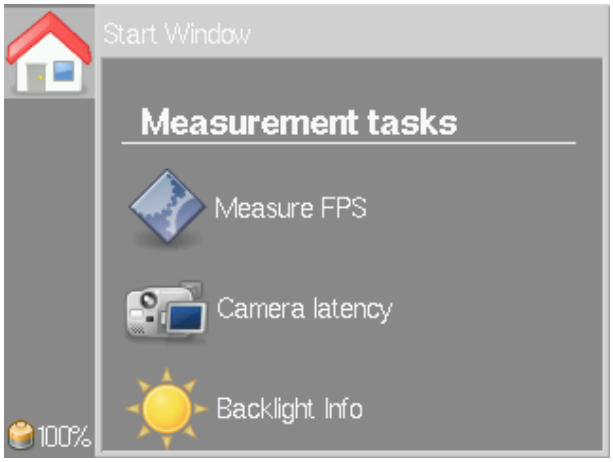


Figure 4.5: *The basic NxWM user interface, showing the taskbar on the left and the list of applications in the start window on the right.*

common interface *IApplication*. The application is responsible for creating its own windows, processing input events and painting the graphics when requested by the window manager.

Overall, the NXWidgets and NxWM combination provides a flexible graphics subsystem, which can run multiple applications simultaneously and each application can also have multiple threads itself. This is useful for example in the frame rate measurement application, shown in Figure 4.6, where the graph can be updated by one thread while the rest of the user interface is handled in another. For this to work, one aspect that must be taken care of is the threading model between the applications and user input events.

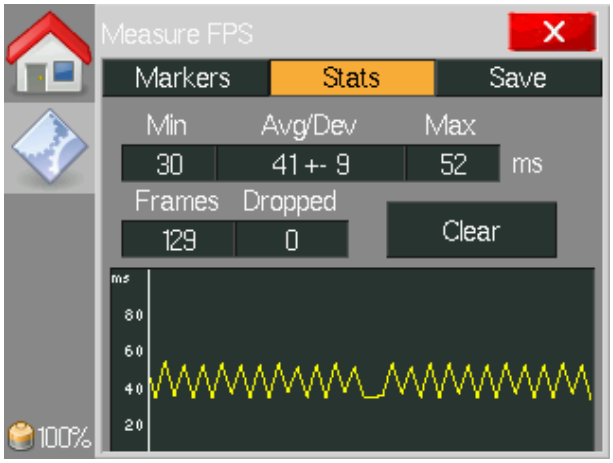


Figure 4.6: *User interface during frame rate measurement.*

Depending on which thread delivers the asynchronous input events from the user, there may be need to use interlocking between several threads to prevent corruption of the program state. As part of the development work relating to this thesis, the core of NxWM was modified to pass all asynchronous events through a single worker thread, as shown

in Figure 4.7 This change was also contributed to the upstream NuttX project, and it simplifies the development of multithreaded applications for this platform.

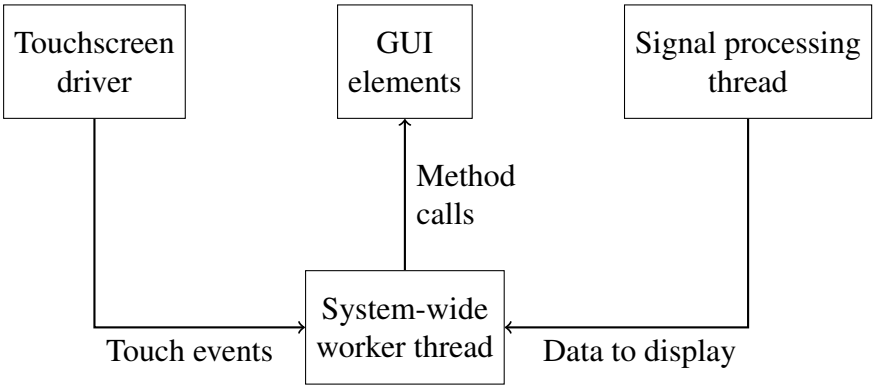


Figure 4.7: *Multithreading model used in the user interface. All events are synchronized by the use of a single worker thread, which makes it simpler to write reliable multithreaded code.*

By passing each event through the work queue, there is no need to perform interlocking as all GUI actions are delivered by a single thread. Applications can still also use additional threads when necessary, as long as they implement the interlocking themselves.

The applications developed for the device in this thesis are straightforward in architecture. As the signal processing subsystem handles most of the actual measurement task, the application only has to provide means to display and save the results. This is accomplished through a few custom controls, such as the scrolling frame time graph, and a single class per each application. The class instantiates the necessary controls, places them on the screen and runs its own thread to for the signal processing filters.

5. RESULTS

The project did succeed in producing a design for a device, which accurately performs frame rate measurements using the multi-state color marker. The time frame from the beginning of the project to the final hardware design was 7 months, and to the first official release of the software 9 months. All design work was done by the author, while the product manager, sales team and hardware team provided highly useful feedback.

The 5-month schedule set at the beginning of the project was exceeded slightly, and the amount of features implemented during the thesis work had to be significantly reduced. Most significantly, it was initially planned to develop a high-speed camera extension board for the device, but this would have been an excessive amount of work.

When considering the schedule in retrospect, it becomes clear that the target product should have been defined more clearly in the beginning. Even though the hardware choices were made early, there were little discussions about at which point the device should be fit for sale as a product. This didn't affect the actual work much, as it smoothly transitioned from initial design to productization. However, it did cause confusion in communicating about the project status, which could have been avoided if the goals were set more accurately.

Product development in small companies poses a few unique challenges. Due to the difficult to predict future market success, relatively small amount of resources can be allocated to a single project. The most important thing is to be able to prove the concept and probe the market potential with a small investment, and then continue development if the product sells well. In this case, the single person development team and extensible hardware were good choices for keeping costs low. However, having only a single developer involved in the initial design caused trouble when transferring the further development work to other groups. Having wider involvement through design and code reviews during the development could have paid back by making further development easier.

Overall, bringing a new product from the first designs to a sellable pilot product in only 9 months and 3 hardware revisions is a good achievement for the project. The rest of this Chapter evaluates the product as a device, i.e. whether the instrument that has been designed fulfills its purpose well.

5.1 Compared to previous version

The measurement accuracy of the previous version, Frame Rate Meter, has never been completely characterized. The general feel is that both the Frame Rate Meter and the Video Multimeter have a similar accuracy, down to the millisecond level. However, the Video Multimeter does not require manual adjustments of the brightness limits, and also handles dropped frames better.

The market potential of the Video Multimeter is much larger than that of the Frame Rate Meter. Because many new measurement functions can be implemented in software, the company can provide custom solutions for individual customers. The co-operation with customers also provides new use cases, some of which have wider demand on the market.

The hardware costs of the two versions are similar. In both cases, the cost of assembly dominates the price. The number of manual steps in the assembly has been reduced compared to the previous version, and the amount of electrical components on the PCB is also smaller. The 3D-printed enclosure is also quick to assemble.

Characterization has shown that the Video Multimeter can measure up to 150 FPS with 1 ms resolution. This improves over the 120 FPS limit of the Frame Rate Meter. The resolution of the Frame Rate Meter was 0.1 ms, which was possible due to the simpler signal processing requirements. However, 1 ms is a sufficient measurement resolution for the displays expected in the near future, and very likely exceeds the limits of human perception. Specialized measurement tasks could also use a simpler signal processing pipeline in order to measure at higher resolution.

5.2 Achieving of goals

The primary goal was to implement frame detection in a way that could solve the problems with dropped frames that had arisen in the previous revision. This was achieved by means of a multi-state color marker and a sensor capable of reading it.

Secondary goals were to expand the device to new kinds of measurement tasks. The extensibility designed into the hardware makes this possible. This has been demonstrated by implementing a few additional measurements, such as camera latency, and more can be added through software and new sensors.

Tertiary goal was to simplify the use of the device by automatically compensating for different display types, backlight brightnesses and other factors, without manual configuration. This has been achieved through a careful design of signal processing algorithms. The difficulty of this task was greatly underestimated in the beginning of the project, and about a month of the schedule slip is entirely due to problems in making the signal processing cope with display variations. However, in the end a single algorithm that accepts

any display type allows a much better result than having a separate algorithm for every situation.

There were also other secondary goals that were related to the planned camera extension board. However, these had to be dropped midway through the project, when the schedule began to slip. The hardware does allow the implementation of the camera extension later on, so these goals can be considered to be postponed for now.

5.3 Sales and customer feedback

Generally, the customers have been interested in the possibilities of the new model. Some example measurement reports have been prepared for customer's own devices, and these have been considered very useful. A few customers have also provided us with completely new use cases, which will highly benefit from the extensibility built into the device.

The sales department has been pleased about the easier demonstrability of the new device, compared to the previous models. As the device is self-contained and portable, the demonstration is easy to set up. Also the new measurement tasks, such as camera latency measurement, have allowed more interesting demonstrations and they also give the potential customers an idea about the flexibility of the device.

Early pilot customers had concerns about the accuracy of the results. These have been addressed both by fixing of several software problems, and by performing a full characterization of the repeatability of the measurement results. The characterization results have proven the high repeatability and accuracy of the measurements across several display types.

Some negative feedback has been received about the look and feel of the enclosure. This is definitely something that could be improved now that the main hardware development is complete. A custom milled aluminum case could be more fitting for the price range of these instruments.

The basic design choice of making a portable instrument with its own touchscreen has also been questioned. However, the explanation that the portability allows mobile measurements in e.g. moving vehicle, has been generally accepted. It is true that a PC-based user interface can in some cases provide a better usability. Nevertheless, even that becomes feasible, as an API to control the device through USB has been planned.

Finally, the feature of camera latency measurements has drawn interest inside the company itself. OptoFidelity Oy develops many kinds of camera systems and the latency of video transfer is important in many of them. The portability and design of the Video Multimeter have been ideal for these measurements, as setting up the measurement takes only seconds. Usually it requires just the turning on of the device and positioning the fiber on the screen.

6. SUMMARY

This thesis has described the theoretical background, hardware design and software architecture of the OptoFidelity Video Multimeter. The designed device functions as planned and is able to measure frame rate and dropped frames of video playback. Table 6.1 presents some general statistics about the project.

Table 6.1: *Basic information of the project and device*

Scope of project	Complete hardware and software design of a measurement instrument.
Purpose	Accurate measurement of timing characteristics of video playback.
Time frame	9 months from initial planning to first software release.
Work effort	
hardware design	5 weeks
low-level software	10 weeks
signal processing	10 weeks
user interface	5 weeks
documentation	5 weeks
Lines of code	
low-level drivers	2000 lines
signal processing	3000 lines
user interface	5000 lines
contributions to NuttX	2000 lines, 107 patches
Strengths	Accurate measurement, portable device, extensibility with new sensors and software.
Drawbacks	High design cost compared to sales volumes, demanding platform for software development.

Overall, the development progressed rapidly and completing the device in just 9 months can be considered fast. However, even so the accumulated development costs are high, which can be problematic for a device with small sales volumes. In some points of the project further insight to the market conditions could have lead to better choices; for example, some other hardware design could have simplified the software development. Even if the cost of the hardware would have increased, up to a point it would be compensated by smaller development costs.

Nevertheless, the initial market interest has been promising and the potential is high due to the possibility of extending the device to perform various measurement tasks. Perhaps the most critical remaining problem is the difficulty of further software development. Because a large part of the software has real-time demands, it cannot be very abstract and performance characteristics have to be carefully thought out when implementing functionality. This leads to a highly demanding environment for software development, which consequently raises the time required and cost of features. One way to address this would have been to use faster hardware, allowing the software to be less efficient but still meet the speed requirements.

In conclusion, the project succeeded in producing a design that meets the needs, and is extensible for future demands. Further development depends on market requirements and financial aspects.

REFERENCES

[1] Intel Corporation, “Quantify and optimize the user interactions with Android devices,” 2011.
http://software.intel.com/en-us/articles/quantify-and-optimize-the-user-interactions-with-android-devices.

[2] International Telecommunication Union, *Objective perceptual assessment of video quality: Full reference television*, 218 pages, 2004.
http://www.itu.int/ITU-T/studygroups/com09/docs/tutorial_opavc.pdf.

[3] OptoFidelity Ltd, “OptoFidelity – technology and expertise,” 2013.
http://www.optofidelity.com/company/technology-and-expertise.

[4] OptoFidelity Ltd, “OptoFidelity FRM120 Frame Rate Meter,” 2 pages, 2008.

[5] OptoFidelity Ltd, “OptoFidelity Video Multimeter,” 2 pages, 2013.
http://www.optofidelity.com/wp-content/uploads/2013/07/OF_VideoMultimeter_PB_screen.pdf.

[6] D. Hands, “A basic multimedia quality model,” *IEEE Transactions on Multimedia*, vol. 6, no. 6, pp. 806–816, 11 pages, 2004.
http://dx.doi.org/10.1109/TMM.2004.837233.

[7] K. Seshadrinathan and A. C. Bovik, “Motion-based perceptual quality assessment of video,” *Proc. SPIE*, vol. 7240, pp. 72400X–72400X–12, 13 pages, 2009.
http://dx.doi.org/10.1117/12.811817.

[8] Apple Inc, “Learn about the high-resolution Retina display,” 2010.
https://web.archive.org/web/20100625135600/http://www.apple.com/iphone/features/retina-display.html.

[9] PassMark Software, “Android devices - CPUMark rating,” 2014.
http://www.androidbenchmark.net/cpumark_chart.html.

[10] S. Winkler and P. Mohandas, “The evolution of video quality measurement: From PSNR to hybrid metrics,” *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 660–668, 9 pages, 2008.
http://dx.doi.org/10.1109/TBC.2008.2000733.

[11] J. Garrett-Glaser, “Diary of an x264 developer: Why so many H.264 encoders are bad,” 2009.
http://x264dev.multimedia.cx/archives/164.

[12] A. B. Watson and J. Albert J. Ahumada, "Model of human visual-motion sensing," *J. Opt. Soc. Am. A*, vol. 2, pp. 322–341, 21 pages, Feb 1985. http://dx.doi.org/10.1364/JOSAA.2.000322.

[13] M. Kalloniatis and C. Luu, "Psychophysics of vision," in *WEBVISION The Organization of the Retina and Visual System* (H. Kolb, R. Nelson, E. Fernandez, and B. Jones, eds.), University of Utah, 2013. http://webvision.med.utah.edu/book/part-viii-gabac-receptors/.

[14] Y. Kuroki, T. Nishi, S. Kobayashi, H. Oyaizu, and S. Yoshimura, "Improvement of motion image quality by high frame rate," *SID Symposium Digest of Technical Papers*, vol. 37, no. 1, pp. 14–17, 4 pages, 2006. http://dx.doi.org/10.1889/1.2433276.

[15] Q. Huynh-Thu and M. Ghanbari, "Impact of jitter and jerkiness on perceived video quality," in *Second International Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM-06)*, 6 pages, 2006. http://enpub.fulton.asu.edu/resp/vpqm/vpqm2006/papers06/308.pdf.

[16] Rohde & Schwarz, *Lip-Sync Measurement (AV Delay) for TV Displays – Application Note*, 12 pages, 2010. http://www.rohde-schwarz.de/file/7BM77_1E.pdf.

[17] OptoFidelity Oy, *AV100 Video Quality Analysis System*, 2013. http://www.optofidelity.com/products-and-services/test-automation/video-playback-performance/av100-video-quality-analysis-system.

[18] O. Boyaci, A. Forte, S. Baset, and H. Schulzrinne, "vDelay: A tool to measure capture-to-display latency and frame rate," in *Multimedia, 2009. ISM '09. 11th IEEE International Symposium on*, pp. 194–200, 7 pages, 2009. http://dx.doi.org/10.1109/ISM.2009.46.

[19] Spirent Communications, "Chromatic video quality measurement system." http://www.spirent.com/Service-Experience/Fit4Launch_Measurement_Systems/Chromatic.

[20] Pixel Instruments Corporation, "LipTracker lip sync analyzer," 6 pages, 2009. http://www.pixelinstruments.tv/pdf/Manuals/LipTracker%20Data%20Sheet%202009.pdf.

[21] Intel, “Intel Atom processor,” 2013.
http://www.intel.com/content/www/us/en/processors/atom/atom-processor.html.

[22] Advanced Micro Devices, Inc., “AMD Geode LX processor family,” 2013.
http://www.amd.com/la/products/embedded/processors/geode-lx/Pages/geode-lx-processor-family.aspx.

[23] BeagleBoard.org Foundation, “BeagleBone,” 2013.
http://beagleboard.org/Products/BeagleBone.

[24] Pandaboard.org, “PandaBoard technical specifications,” 2013.
http://pandaboard.org/node/300/#Panda.

[25] ARM Ltd., “Cortex-M series,” 2013.
http://www.arm.com/products/processors/cortex-m/.

[26] Advantech Co.,Ltd., “ARK-3403 – Intel Atom D510/D525 fanless embedded box PC,” 2013.
http://www.advantech.com/products/ARK-3403/mod_96172106-3409-45B3-83A1-AE004C960E48.aspx.

[27] STMicroelectronics, “Discovery kit for STM32F407/417 lines,” 2013.
http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252419.

[28] Altera Corporation, “DE0-Nano development and education board,” 2013.
http://www.altera.com/education/univ/materials/boards/de0-nano/unv-de0-nano-board.html.

[29] Atmel, “AVR32 architecture document,” 377 pages, 2011.
http://www.atmel.fi/Images/doc32000.pdf.

[30] Imagination Technologies Ltd., “MIPS M4K hard IP cores,” 2013.
http://www.imgtec.com/mips/mips-m4k-hardip.asp.

[31] NXP Semiconductors, *LPC4350/30/20/10 32-bit ARM Cortex-M4/M0 microcontroller*, 150 pages, 2012.
http://www.nxp.com/documents/data_sheet/LPC4350_30_20_10.pdf.

[32] STMicroelectronics, *DS8626: STM32F405xx, STM32F407xx datasheet*, 185 pages, 2012.
http://www.st.com/web/en/resource/technical/document/datasheet/DM00037051.pdf.

[33] Atmel, *SAM4S ARM Cortex-M4 Microcontrollers*, 2012.
http://www.atmel.com/products/microcontrollers/arm/sam4s.aspx.

[34] Digi-Key, *Price listings*, November 2012.
http://digikey.fi/.

[35] Displaytech Ltd., *INT035TFT-TS LCD Module*, 2011.
https://www.displaytech-us.com/3-5-inch-integrated-tft-driver-boards.

[36] Proto Labs Inc., "Protomold injection molding," 2012.
http://www.protomold.com/.

[37] Shapeways Inc., "3D printing services."
http://www.shapeways.com/.

[38] RibbonSoft, "QCad open source CAD system for everyone."
http://www.qcad.org/.

[39] M. Kintel and C. Wolf, "OpenSCAD – the programmer's solid 3D CAD modeller."
http://www.openscad.org/.

[40] G. D. Sirio, "ChibiOS/RT," 2013.
http://chibios.org/.

[41] G. D. Sirio, "ChibiOS/RT features matrix," 2013.
http://www.chibios.org/dokuwiki/doku.php?id=chibios:matrix.

[42] Micrium, "µC/OS-II – the real-time kernel," 2013.
http://micrium.com/rtos/ucosii/overview/.

[43] HCC Embedded, "STM32F4 middleware," 2013.
http://www.hcc-embedded.com/targetdevices/stmicro/stm32f4.

[44] G. Nutt, "NuttX real-time operating system," 2013.
http://nuttx.org/.

[45] Real Time Engineers Ltd., "FreeRTOS," 2013.
http://www.freertos.org/.

[46] Microsoft Corporation, ".NET Micro Framework," 2013.
http://www.netmf.com/.

[47] Lua.org, "Lua programming language," 2013.
http://www.lua.org/about.html.

[48] G. Nutt, “NxWidgets.”
http://nuttx.org/doku.php?id=documentation:nxwidgets.

[49] G. Nutt, “NuttX window manager.”
http://nuttx.org/doku.php?id=wiki:graphics:nxwm.